ADAPTIVE SYSTEM FOR DYNAMIC HANDLING OF CONCEPT DRIFT: DETECTION, MODELING, AND WEIGHTED ENSEMBLE PREDICTIONS

A THESIS SUBMITTED TO
THE GRADUATE SCHOOL OF INFORMATICS OF
THE MIDDLE EAST TECHNICAL UNIVERSITY
BY

BARIŞ ÖZCAN

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR THE DEGREE OF
MASTER OF SCIENCE
IN
THE DEPARTMENT OF DATA INFORMATICS

DECEMBER 2024

ADAPTIVE SYSTEM FOR DYNAMIC HANDLING OF CONCEPT DRIFT: DETECTION, MODELING, AND WEIGHTED ENSEMBLE PREDICTIONS

submitted by **BARIŞ ÖZCAN** in partial fulfillment of the requirements for the degree of **Master of Science in Data Informatics Department, Middle East Technical University** by,

Date: 27.12.2024

I hereby declare that all information in this document has been obtained and presented in accordance with academic rules and ethical conduct. I also declare that, as required by these rules and conduct, I have fully cited and referenced all material and results that are not original to this work.

Name, Surname: Barış Özcan

Signature :

ABSTRACT

ADAPTIVE SYSTEM FOR DYNAMIC HANDLING OF CONCEPT DRIFT: DETECTION, MODELING, AND WEIGHTED ENSEMBLE PREDICTIONS

Özcan, Barış

M.S., Department of Data Informatics

Supervisor: Prof. Dr. Banu Günel Kılıç

December 2024, 153 pages

Machine learning models depend on the quality and quantity of data used in the training process. This dependency necessitates continuous development in data collection methods and the optimization of existing data resources. A notable obstacle for using collected data effectively is concept drift, which is evolving the underlying patterns within data over time, causing a reduction in model performance and relevance. Although the phenomenon of concept drift is widely acknowledged, a standardized approach for its quantification and remedy remains elusive. Our research proposes a holistic approach to handle concept drift by developing a system that not only detects but also dynamically adapts to new data concepts. We introduce a system that examines the concepts understood by existing models and compare these with the concepts present in new datasets. Based on this comparison, the system decides whether to update existing models or to develop new models tailored to the newly identified concepts. This approach enables continuous model improvement and addresses the potential for concept drift within specific classes of a dataset. To establish a comprehensive system for resolving concept drift problem, our research studies various detection methods, including centroid differences and performance-based evaluations. We adapt these concept drift detection methods to the training process by developing different machine learning models for each concept. Also, our study explores various prediction strategies for effectively using ensemble models that consist of models specialized in specific concepts. Additionally, our research addresses the potential occurrence of class imbalances during the training processes by incorporating synthetic data generation techniques. In summary, this thesis aims to refine methods to detect concept drift and develop adaptive solutions for the concept drift problem in dynamic settings.

Keywords: Concept drift, Ensemble learning models, Machine learning, Adaptive solutions, Class imbalance

ÖZ

KONSEPT KAYMASIYLA DİNAMİK BAŞA ÇIKMAK İÇİN ADAPTİF SİSTEM: TESPİT, MODELLEME VE AĞIRLIKLI OYLAMA İLE TAHMİN

Özcan, Barış

Yüksek Lisans, Veri Bilişimi Bölümü

Tez Yöneticisi: Prof. Dr. Banu Günel Kılıç

Aralık 2024, 153 sayfa

Makine öğrenimi modelleri, eğitim sürecinde kullanılan verilerin kalitesine ve miktarına bağımlıdır. Bu bağımlılık, veri toplama yöntemlerinde ve mevcut veri kaynaklarının optimizasyonunda sürekli gelişmeyi zorunlu kılmaktadır. Toplanan verileri etkin bir şekilde kullanmakta karşılaşılan önemli bir engel; zaman içinde verilerdeki temel örüntülerin evrimleşmesi olan, modelin performansını ve alakasını düşüren konsept kaymasıdır. Konsept kayması yaygın olarak varlığı kabul edilen bir fenomen olmasına karşın; onun niceliklendirilmesi ve çözümü için standartlaştırılmış bir yaklaşım henüz geliştirilememiştir. Araştırmamız, konsept kaymasını yalnızca tespit etmekle kalmayan, aynı zamanda dinamik olarak uyum sağlayan bir sistem geliştirilmesini içeren bütünsel bir yaklaşım önermektedir. Geliştirdiğimiz sistem, mevcut modeller tarafından anlaşılan konseptleri değerlendirir ve bu konseptleri yeni veri kümelerindeki konseptlerle karşılaştırır. Sistem bu karşılaştırmaya dayanarak; mevcut modellerin güncellenmesine veya yeni tanımlanan konseptlere uygun yeni modellerin geliştirilmesine karar verir. Bu yaklaşım, sürekli model iyileştirmesine olanak tanır ve veri kümesinin belirli sınıfları içindeki olası konsept kaymasını ele alır. Konsept kayması problemini çözmek için kapsamlı bir sistem kurmak amacıyla centroid farkları ve performansa dayalı değerlendirmeler de dahil olmak üzere çeşitli konsept kayması tespiti yöntemleri incelenmektedir. Konsept kayması tespiti yöntemlerini her konsept için farklı bir model eğitmek için eğitim sürecine uyarlıyoruz. Ayrıca, çalışmamız, belirli konseptlerde uzmanlaşmış modellerden oluşan model topluluğu için etkili tahmin stratejilerini araştırmaktadır. Araştırmamız, eğitim süreçlerinde sınıf dengesizliklerinin oluşması sorununu, sentetik veri üretim tekniklerini dahil ederek ele almaktadır. Kısaca; bu tez, dinamik bir düzlemde, konsept kaymasını tespit etme yöntemlerini geliştirmeyi ve konsept kayması olayına uyum sağlamayı amaçlamaktadır.

Anahtar Kelimeler: Konsept kayması, Topluluk öğrenme modelleri, Makina öğrenmesi, Adaptif çözümler, Sınıf dengesizliği

Dedicated to my father, Nurhan Özcan, my greatest role model

ACKNOWLEDGMENTS

First and foremost, I would like to express my sincere gratitude to my advisor, Banu Günel Kılıç, for her guidance and support. I feel incredibly fortunate to have had the opportunity to work with her. Throughout my thesis studies, not only did she share her experience and knowledge in our regular meetings, but she also valued my thoughts and allowed me to guide the research in the direction I desired. Her approach has deepened my commitment to this research and my passion for academia.

During my master's studies, I gained valuable knowledge and competencies in the field of data science at the Informatics Institute. I would like to extend my gratitude to the professors from whom I took courses and who significantly contributed to my development in this field.

I would like to thank Guzey Uzun, whom I met during my master's studies and who quickly became one of the friends I spent the most time with, and Sinan Yaşar, who is not only a friend but also my and our university's boxing coach.

I am grateful to my family, who have always supported me in pursuing my master's education and have never withheld their encouragement.

TABLE OF CONTENTS

ABSTRACT iv

ÖZ vi

DEDICATION viii

ACKNOWLEDGMENTS ix

TABLE OF CONTENTS x

LIST OF TABLES xiv

LIST OF FIGURES xxi

LIST OF ABBREVIATIONS xxviii

CHAPTERS

1 INTRODUCTION 1

1.1 Motivation and Problem Definition 1

1.2 Research Aim and Objectives 4

1.3 Organization of the Thesis 4

2 LITERATURE REVIEW 5

2.1 Detecting and Explaining Domain-Oriented Concept Drift 5

2.2 Addressing Concept Drift for Incremental Learning 7

2.3 Ensemble Methods for Dealing with Concept Drift 7

2.4 Ensemble Methods for Detecting Concept Drift 8
2.5 Developing Robust Model That Is Not Affected By Concept Drift 9
2.6 Out of Distribution Detection 10
3 METHODOLOGY 13
3.1 Problem Definition 13
3.1.1 Types of Concept Drift Event 13
3.2 General structure of the system 15
3.3 Strategies For Detecting Concept Drift in the Training Process 15
3.3.1 Strategy Based On Comparison of Datasets 16
3.3.2 Strategy Based On Performance of the Models 20
3.4 Balance Datasets That Are Used For Training Models 23
3.5 Type of Learner Models 24
3.6 Strategy for Prediction 25
3.6.1 Assign Weights on Models Based on Centroid Difference Metric 25
3.6.2 Assign Weights on Models Based on Similarity Score 27
3.6.3 Assign Weights on Models Based on Centroid Difference and Diversity 30
3.6.4 Assign Weights on Models Based on The Jensen-Shannon Divergence Score . . 32
3.6.5 Assign Weights on Models Based on Statistical Test 36
3.6.6 Give One Vote Right Each to Models Trained on Datasets That Are Not In Different Concept 39
4 EXPERIMENTS AND RESULTS 41
4.1 Experiments with Synthetic Data 43
4.1.1 Sudden Drift Scenario 43

4.1.1.1 Sudden Drift Experiment 1 44
4.1.1.2 Sudden Drift Experiment 2 49
4.1.1.3 Sudden Drift Experiment 3 54
4.1.2 Gradual Drift Scenario 60
4.1.2.1 Gradual Drift Experiment 1 60
4.1.2.2 Gradual Drift Experiment 2 66
4.1.2.3 Gradual Drift Experiment 3 71
4.1.3 Incremental Drift 76
4.1.3.1 Incremental Drift Experiment 1 77
4.1.3.2 Incremental Drift Experiment 2 81
4.1.3.3 Incremental Drift Experiment 3 86
4.1.4 Reoccurring Drift 92
4.1.4.1 Reoccurring Drift Experiment 1 92
4.1.4.2 Reoccurring Drift Experiment 2 97
4.1.4.3 Reoccurring Drift Experiment 3 102
4.2 Experiments with Real-Word Data 108
4.2.1 Performance of the System According to Training Strategy 109
4.2.1.1 Analysis of Performance of the System That Uses Training Strategy Based on Comparison of Datasets According to Batch Number. 113
4.2.1.2 Analysis of Performance of the System That Uses Training Strategy Based on Accuracy According to Batch Number. 119
4.2.2 Performance of the System According to Prediction Strategy 124
5 DISCUSSION 139
5.1 Concept Drift Scenarios 139

5.1.1 Sudden Drift 139
5.1.2 Gradual Drift 141
5.1.3 Incremental Drift 142
5.1.4 Reoccurring Drift 143
5.2 Training Strategies 144
5.3 Prediction Strategies 145
5.4 Effect of Size of Data 147
5.5 Effect of Balancing Datasets 148
6 CONCLUSION AND FUTURE WORK 149
REFERENCES 151

LIST OF TABLES

Table 1 Summary of dataset of sudden drift experiment 1 where the sudden drift event occurs in an increasing manner across all three classes. 46

Table 2 Performance metrics of single model approach in sudden drift experiment 1 where the sudden drift event occurs in an increasing manner across all three classes. 47

Table 3 Performance of the system for test sets that are used in sudden drift experiment 1 where the sudden drift event occurs in an increasing manner across all three classes. 48

Table 4 Performance of the system that implemented training process with balanced datasets in sudden drift experiment 1 where the sudden drift event occurs in an increasing manner across all three classes. 48

Table 5 Performance metrics of the system that used accuracy based training strategy on test sets which is used in sudden drift experiment 1 where the sudden drift event occurs in an increasing manner across all three classes. 49

Table 6 Performance of the system that used accuracy based training strategy and implemented training process with balanced datasets on test sets which used in sudden drift experiment 1 where the sudden drift event occurs in an increasing manner across all three classes. 49

Table 7 Summary of dataset of sudden drift experiment 2 where the sudden drift event occurs as a decrease in classes A and B, and an increase in class C. 52

Table 8 Performance metrics of single model approach in sudden drift experiment 2 where the sudden drift event occurs as a decrease in classes A and B, and an increase in class C. . . 53

Table 9 Performance of the system for test sets that are used in sudden drift experiment 2 where the sudden drift event occurs as a decrease in classes A and B, and an increase in class C. 53

Table 10 Performance of the system that implemented training process with balanced datasets in sudden drift experiment 2 where the sudden drift event occurs as a decrease in classes A and B, and an increase in class C. 54

Table 11 Performance metrics of the system that used accuracy based training strategy on test sets which used in sudden drift experiment 2 where the sudden drift event occurs as a decrease in classes A and B, and an increase in class C. 54

Table 12 Performance of the system that used accuracy based training strategy and implemented training process with balanced datasets on test sets which used in sudden drift experiment 2 where the sudden drift event occurs as a decrease in classes A and B, and an increase in class C. 54

Table 13 Summary of dataset of sudden drift experiment 3 where the sudden drift event occurs as an increase in classes A and C, and a decrease in class B, and after the concept drift event, classes A and B converge to the more similar concept. 55

Table 14 Performance metrics of single model approach in sudden drift experiment 3 where the sudden drift event occurs as an increase in classes A and C, and a decrease in class B, and after the concept drift event, classes A and B converge to the more similar concept. . . . 57

Table 15 Performance of the system for test sets that are used in sudden drift experiment 3 where the sudden drift event occurs as an increase in classes A and C, and a decrease in class B, and after the concept drift event, classes A and B converge to the more similar concept. 58

Table 16 Performance of the system that implemented training process with balanced datasets in sudden drift experiment 3 where the sudden drift event occurs as an increase in classes A and C, and a decrease in class B, and after the concept drift event, classes A and B converge to the more similar concept. 59

Table 17 Performance metrics of the system that used accuracy based training strategy on test sets which used in sudden drift experiment 3 where the sudden drift event occurs as an increase in classes A and C, and a decrease in class B, and after the concept drift event, classes A and B converge to the more similar concept. 59

Table 18 Performance of the system that used accuracy based training strategy and implemented training process with balanced datasets on test sets which used in sudden drift experiment 3 where the sudden drift event occurs as an increase in classes A and C, and a decrease in class B, and after the concept drift event, classes A and B converge to the more similar concept. 59

Table 19 Summary of dataset of gradual drift experiment 1 where the gradual drift event occurs in an increasing manner across all three classes. 63

Table 20 Performance metrics of single model approach in gradual drift experiment 1 where the gradual drift event occurs in an increasing manner across all three classes. 64

Table 21 Performance of the system for test sets that are used in gradual drift experiment 1 where the gradual drift event occurs in an increasing manner across all three classes. 64

Table 22 Performance of the system that implemented training process with balanced datasets in gradual drift experiment 1 where the gradual drift event occurs in an increasing manner across all three classes. 64

Table 23 Performance metrics of the system that used accuracy based training strategy on test sets which used in gradual drift experiment 1 where the gradual drift event occurs in an increasing manner across all three classes. 65

Table 24 Performance of the system that used accuracy based training strategy and implemented training process with balanced datasets on test sets which used in gradual drift experiment 1 where the gradual drift event occurs in an increasing manner across all three classes. 65

Table 25 Summary of dataset of gradual drift experiment 2 where the gradual drift event occurs as an increase in classes A and B, and a decrease in class C. 68

Table 26 Performance metrics of single model approach in gradual drift experiment 2 where the gradual drift event occurs as an increase in classes A and B, and a decrease in class C. . 69

Table 27 Performance of the system for test sets that are used in gradual drift experiment 2 where the gradual drift event occurs as an increase in classes A and B, and a decrease in class C. 69

Table 28 Performance of the system that implemented training process with balanced datasets in gradual drift experiment 2 where the gradual drift event occurs as an increase in classes A and B, and a decrease in class C. 70

Table 29 Performance metrics of the system that used accuracy based training strategy on test sets which used in gradual drift experiment 2 where the gradual drift event occurs as an increase in classes A and B, and a decrease in class C. 70

Table 30 Performance of the system that used accuracy based training strategy and implemented training process with balanced datasets on test sets which used in gradual drift experiment 2 where the gradual drift event occurs as an increase in classes A and B, and a decrease in class C. 70

Table 31 Summary of dataset of gradual drift experiment 3 where the gradual drift event occurs as an increase in class A, and a decrease in classes B and C, and after the concept drift event, classes A and B converge to the similar concept. 73

Table 32 Performance metrics of single model approach in gradual drift experiment 3 where the gradual drift event occurs as an increase in class A, and a decrease in classes B and C, and after the concept drift event, classes A and B converge to the similar concept. 75

Table 33 Performance of the system for test sets that are used in gradual drift experiment 3 where the gradual drift event occurs as an increase in class A, and a decrease in classes B and C, and after the concept drift event, classes A and B converge to the similar concept. . . 75

Table 34 Performance of the system that implemented training process with balanced datasets in gradual drift experiment 3 where the gradual drift event occurs as an increase in class A, and a decrease in classes B and C, and after the concept drift event, classes A and B converge to the similar concept. 75

Table 35 Performance metrics of the system that used accuracy based training strategy on test sets which used in gradual drift experiment 3 where the gradual drift event occurs as an increase in class A, and a decrease in classes B and C, and after the concept drift event, classes A and B converge to the similar concept. 75

Table 36 Performance of the system that used accuracy based training strategy and implemented training process with balanced datasets on test sets which used in gradual drift experiment 3 where the gradual drift event occurs as an increase in class A, and a decrease in classes B and C, and after the concept drift event, classes A and B converge to the similar concept. 76

Table 37 Summary of dataset of incremental drift experiment 1 where the incremental drift event occurs in an increasing manner across all three classes. 79

Table 38 Performance metrics of single model approach in incremental drift experiment 1 where the incremental drift event occurs in an increasing manner across all three classes. . . 80

Table 39 Performance of the system for test sets that are used in incremental drift experiment 1 where the incremental drift event occurs in an increasing manner across all three classes. . 80

Table 40 Performance of the system that implemented training process with balanced datasets in incremental drift experiment 1 where the incremental drift event occurs in an increasing manner across all three classes. 80

Table 41 Performance metrics of the system that used accuracy based training strategy on test sets which used in incremental drift experiment 1 where the incremental drift event occurs in an increasing manner across all three classes. 81

Table 42 Performance of the system that used accuracy based training strategy and implemented training process with balanced datasets on test sets which used in incremental drift experiment 1 where the incremental drift event occurs in an increasing manner across all three classes. 81

Table 43 Summary of dataset of incremental drift experiment 2 where the incremental drift event occurs as an increase in class A, and a decrease in classes B and C. 84

Table 44 Performance metrics of single model approach in incremental drift experiment 2 where the incremental drift event occurs as an increase in class A, and a decrease in classes B and C. 85

Table 45 Performance of the system for test sets that are used in incremental drift experiment 2 where the incremental drift event occurs as an increase in class A, and a decrease in classes B and C. 85

Table 46 Performance of the system that implemented training process with balanced datasets in incremental drift experiment 2 where the incremental drift event occurs as an increase in class A, and a decrease in classes B and C. 85

Table 47 Performance metrics of the system that used accuracy based training strategy on test sets which used in incremental drift experiment 2 where the incremental drift event occurs as an increase in class A, and a decrease in classes B and C. 86

Table 48 Performance of the system that used accuracy based training strategy and implemented training process with balanced datasets on test sets which used in incremental drift experiment 2 where the incremental drift event occurs as an increase in class A, and a decrease in classes B and C. 86

Table 49 Summary of dataset of incremental drift experiment 3 where the Incremental drift event occurs as an increase in classes A and B, and a decrease in class C, and after the concept drift event, classes A and B converge to the same concept. 89

Table 50 Performance metrics of single model approach in incremental drift experiment 3 where the Incremental drift event occurs as an increase in classes A and B, and a decrease in class C, and after the concept drift event, classes A and B converge to the same concept. . 90

Table 51 Performance of the system for test sets that are used in incremental drift experiment 3 where the Incremental drift event occurs as an increase in classes A and B, and a decrease in class C, and after the concept drift event, classes A and B converge to the same concept. . 90

Table 52 Performance of the system that implemented training process with balanced datasets in incremental drift experiment 3 where the Incremental drift event occurs as an increase in classes A and B, and a decrease in class C, and after the concept drift event, classes A and B converge to the same concept. 91

Table 53 Performance metrics of the system that used accuracy based training strategy on test sets which used in incremental drift experiment 3 where the Incremental drift event occurs as an increase in classes A and B, and a decrease in class C, and after the concept drift event, classes A and B converge to the same concept. 91

Table 54 Performance of the system that used accuracy based training strategy and implemented training process with balanced datasets on test sets which used in incremental drift experiment 3 where the Incremental drift event occurs as an increase in classes A and B, and a decrease in class C, and after the concept drift event, classes A and B converge to the same concept. 91

Table 55 Summary of dataset of reoccurring drift experiment 1 where the reoccurring drift event occurs in an increasing manner across all three classes. 95

Table 56 Performance metrics of single model approach in reoccurring drift experiment 1 where the reoccurring drift event occurs in an increasing manner across all three classes. . . 96

Table 57 Performance of the system for test sets that are used in reoccurring drift experiment 1 where the reoccurring drift event occurs in an increasing manner across all three classes. . 96

Table 58 Performance of the system that implemented training process with balanced datasets in reoccurring drift experiment 1 where the reoccurring drift event occurs in an increasing manner across all three classes. 96

Table 59 Performance metrics of the system that used accuracy based training strategy on test sets which used in reoccurring drift experiment 1 where the reoccurring drift event occurs in an increasing manner across all three classes. 97

Table 60 Performance of the system that used accuracy based training strategy and implemented training process with balanced datasets on test sets which used in reoccurring drift experiment 1 where the reoccurring drift event occurs in an increasing manner across all three classes. 97

Table 61 Summary of dataset of reoccurring drift experiment 2 where the reoccurring drift event occurs as an increase in classes A and B, and a decrease in class C. 100

Table 62 Performance metrics of single model approach in reoccurring drift experiment 2 where the reoccurring drift event occurs as an increase in classes A and B, and a decrease in class C. 101

Table 63 Performance of the system for test sets that are used in reoccurring drift experiment 2 where the reoccurring drift event occurs as an increase in classes A and B, and a decrease in class C. 101

Table 64 Performance of the system that implemented training process with balanced datasets in reoccurring drift experiment 2 where the reoccurring drift event occurs as an increase in classes A and B, and a decrease in class C. 101

Table 65 Performance metrics of the system that used accuracy based training strategy on test sets which used in reoccurring drift experiment 2 where the reoccurring drift event occurs as an increase in classes A and B, and a decrease in class C. 102

Table 66 Performance of the system that used accuracy based training strategy and implemented training process with balanced datasets on test sets which used in reoccurring drift experiment 2 where the reoccurring drift event occurs as an increase in classes A and B, and a decrease in class C. 102

Table 67 Summary of dataset of reoccurring drift experiment 3 where the reoccurring drift event occurs as an increase in classes A and B, and a decrease in class C, and after the concept drift event, classes A and C converge to the same concept. 105

Table 68 Performance metrics of single model approach in reoccurring drift experiment 3 where the reoccurring drift event occurs as an increase in classes A and B, and a decrease in class C, and after the concept drift event, classes A and C converge to the same concept. . 106

Table 69 Performance of the system for test sets that are used in reoccurring drift experiment 3 where the reoccurring drift event occurs as an increase in classes A and B, and a decrease in class C, and after the concept drift event, classes A and C converge to the same concept. . 106

Table 70 Performance of the system that implemented training process with balanced datasets in reoccurring drift experiment 3 where the reoccurring drift event occurs as an increase in classes A and B, and a decrease in class C, and after the concept drift event, classes A and C converge to the same concept. 107

Table 71 Performance metrics of the system that used accuracy based training strategy on test sets which used in reoccurring drift experiment 3 where the reoccurring drift event occurs as an increase in classes A and B, and a decrease in class C, and after the concept drift event, classes A and C converge to the same concept. 107

Table 72 Performance of the system that used accuracy based training strategy and implemented training process with balanced datasets on test sets which used in reoccurring drift experiment 3 where the reoccurring drift event occurs as an increase in classes A and B, and a decrease in class C, and after the concept drift event, classes A and C converge to the same concept. 107

Table 73 Comparison of Performance of Different Training Approaches and Single Model Approach on Last Batch Data Based on Accuracy Metric 110

Table 74 Comparison of Performance of Different Training Approaches and Single Model Approach on Last Batch Data Based on Recall Metric 110

Table 75 Comparison of Performance of Different Training Approaches and Single Model Approach on Last Batch Data Based on Precision Metric 110

Table 76 Comparison of Performance of Different Training Approaches and Single Model Approach on Random Instances Dataset Based on Accuracy Metric 111

Table 77 Comparison of Performance of Different Training Approaches and Single Model Approach on Random Instances Dataset Based on Recall Metric 111

Table 78 Comparison of Performance of Different Training Approaches and Single Model Approach on Random Instances Dataset Based on Precision Metric 111

Table 79 Comparison of Tuned Version of the Systems and Single Model Approach Based on Test Set Accuracy Where Test Set Is Last Batch 112

Table 80 Comparison of Tuned Version of the Systems and Single Model Approach Based on Test Set Accuracy Where Test Set Consists Random Instances 113

Table 81 Comparison of Tuned Version of the Systems That Use Prediction Strategy Based on Similarity Score and Single Model Approach Based on Test Set Accuracy 127

Table 82 Comparison of Tuned Version of the Systems That Use Prediction Strategy Based on Centroid Difference and Diversity and Single Model Approach Based on Test Set Accuracy 128

Table 83 Comparison of Tuned Version of the Systems That Use Prediction Strategy Based on a Statistical Test and Single Model Approach Based on Test Set Accuracy 132

Table 84 Comparison of Tuned Version of the Systems That Use Prediction Strategy Based on Give One Vote Right Each to Models Trained on Datasets That Are Not In Different Concept and Single Model Approach Based on Test Set Accuracy 134

Table 85 Comparison of Tuned Version of the Systems That Use Prediction Strategy Based on The Jensen-Shannon Divergence Score and Single Model Approach Based on Test Set Accuracy 138

LIST OF FIGURES

Figure 1 The Method That Is to Solve Out of Distribution Problem Adapted From [20] ... 11

Figure 2 Main Concept Drift Scenarios in the Literature 14

Figure 3 General structure of adaptive system for dynamic handling of concept drift 16

Figure 4 Training Algorithm Based on Comparison of Concept of Corresponding Classes . 19

Figure 5 Algorithm for Implement Training Based on Performance of the Models 21

Figure 6 Voting Strategy Based On Centroid Differences 26

Figure 7 Strategy Based on Similarity Score 29

Figure 8 Simplified Representation of Algorithm for Assigning Weights on Models Based on Centroid Difference and Diversity 31

Figure 9 Prediction Strategy Which Assigns Weights on Models Based on The jensen-Shannon Divergence Score 35

Figure 10 Simplified Representation of Algorithm for Assign Weights on Models Based on Statistical Test 38

Figure 11 The Distribution of Data in Sudden Drift Experiment 1 Where The Sudden Drift Event Occurs In An Increasing Manner Across All Three Classes 44

Figure 12 The Distribution of Class A Data in Sudden Drift Experiment 1 Where The Sudden Drift Event Occurs In An Increasing Manner Across All Three Classes 45

Figure 13 The Distribution of Class B Data in Sudden Drift Experiment 1 Where The Sudden Drift Event Occurs In An Increasing Manner Across All Three Classes 45

Figure 14 The Distribution of Class C Data in Sudden Drift Experiment 1 Where The Sudden Drift Event Occurs In An Increasing Manner Across All Three Classes 46

Figure 15 The Distribution of Data in Sudden Drift Experiment 2 Where the Sudden Drift Event Occurs as an Increase in Classes A and B, and a Decrease in Class C 50

Figure 16 The Distribution of Class A Data in Sudden Drift Experiment 2 Where the Sudden Drift Event Occurs as a Decrease in Classes A and B, and an Increase in Class C 50

Figure 17 The Distribution of Class B Data in Sudden Drift Experiment 2 Where the Sudden Drift Event Occurs as a Decrease in Classes A and B, and an Increase in Class C 51

Figure 18 The Distribution of Class C Data in Sudden Drift Experiment 2 Where the Sudden Drift Event Occurs as a Decrease in Classes A and B, and an Increase in Class C 51

Figure 19 The Distribution of Data in Sudden Drift Experiment 3 Where the Sudden Drift Event Occurs as an Increase in Classes A and C, and a Decrease in Class B, and After the Concept Drift Event, Classes A and B Converge to the More Similar Concept 55

Figure 20 The Distribution of Class A Data in Sudden Drift Experiment 3 Where the Sudden Drift Event Occurs as an Increase in Classes A and C, and a Decrease in Class B, and After the Concept Drift Event, Classes A and B Converge to the More Similar Concept . . 56

Figure 21 The Distribution of Class B Data in Sudden Drift Experiment 3 Where the Sudden Drift Event Occurs as an Increase in Classes A and C, and a Decrease in Class B, and After the Concept Drift Event, Classes A and B Converge to the More Similar Concept . . 57

Figure 22 The Distribution of Class C Data in Sudden Drift Experiment 3 Where the Sudden Drift Event Occurs as an Increase in Classes A and C, and a Decrease in Class B, and After the Concept Drift Event, Classes A and B Converge to the More Similar Concept . . 58

Figure 23 The Distribution of Data in Gradual Drift Experiment 1 Where The Gradual Drift Event Occurs As Increasing Manner Across All Three Classes 61

Figure 24 The Distribution of Class A Data in Gradual Drift Experiment 1 Where The Gradual Drift Event Occurs As Increasing Manner Across All Three Classes 61

Figure 25 The Distribution of Class B Data in Gradual Drift Experiment 1 Where The Gradual Drift Event Occurs As Increasing Manner Across All Three Classes 62

Figure 26 The Distribution of Class C Data in Gradual Drift Experiment 1 Where The Gradual Drift Event Occurs As Increasing Manner Across All Three Classes 62

Figure 27 The Distribution of Data in Gradual Drift Experiment 2 Where the Gradual Drift Event Occurs as an Increase in Classes A and B, and a Decrease in Class C 66

Figure 28 The Distribution of Class A Data in Gradual Drift Experiment 2 Where the Gradual Drift Event Occurs as an Increase in Classes A and B, and a Decrease in Class C 66

Figure 29 The Distribution of Class B Data in Gradual Drift Experiment 2 Where the Gradual Drift Event Occurs as an Increase in Classes A and B, and a Decrease in Class C 67

Figure 30 The Distribution of Class C Data in Gradual Drift Experiment 2 Where the Gradual Drift Event Occurs as an Increase in Classes A and B, and a Decrease in Class C 67

Figure 31 The Distribution of Data in Gradual Drift Experiment 3 Where the Gradual Drift Event Occurs as an Increase in Class A, and a Decrease in Classes B and C, and After the Concept Drift Event, Classes A and B Converge to the Similar Concept 71

Figure 32 The Distribution of Class A Data in Gradual Drift Experiment 3 Where the Gradual Drift Event Occurs as an Increase in Class A, and a Decrease in Classes B and C, and After the Concept Drift Event, Classes A and B Converge to the Similar Concept 72

Figure 33 The Distribution of Class B Data in Gradual Drift Experiment 3 Where the Gradual Drift Event Occurs as an Increase in Class A, and a Decrease in Classes B and C, and After the Concept Drift Event, Classes A and B Converge to the Similar Concept 72

Figure 34 The Distribution of Class C Data in Gradual Drift Experiment 3 Where the Gradual Drift Event Occurs as an Increase in Class A, and a Decrease in Classes B and C, and After the Concept Drift Event, Classes A and B Converge to the Similar Concept 73

Figure 35 The Distribution of Data in Incremental Drift Experiment 1 Where The Incremental Drift Event Occurs As Increasing Manner Across All Three Classes 77

Figure 36 The Distribution of Class A Data in Incremental Drift Experiment 1 Where The Incremental Drift Event Occurs As Increasing Manner Across All Three Classes 77

Figure 37 The Distribution of Class B Data in Incremental Drift Experiment 1 Where The Incremental Drift Event Occurs As Increasing Manner Across All Three Classes 78

Figure 38 The Distribution of Class C Data in Incremental Drift Experiment 1 Where The Incremental Drift Event Occurs As Increasing Manner Across All Three Classes 78

Figure 39 The Distribution of Data in Incremental Drift Experiment 2 Where the Incremental Drift Event Occurs as an Increase in Class A, and a Decrease in Classes B and C 82

Figure 40 The Distribution of Class A Data in Incremental Drift Experiment 2 Where the Incremental Drift Event Occurs as an Increase in Class A, and a Decrease in Classes B and C 82

Figure 41 The Distribution of Class B Data in Incremental Drift Experiment 2 Where the Incremental Drift Event Occurs as an Increase in Class A, and a Decrease in Classes B and C 83

Figure 42 The Distribution of Class C Data in Incremental Drift Experiment 2 Where the Incremental Drift Event Occurs as an Increase in Class A, and a Decrease in Classes B and C 83

Figure 43 The Distribution of Data in Incremental Drift Experiment 3 Where the Incremental Drift Event Occurs as an Increase in Classes A and B, and a Decrease in Class C, and After the Concept Drift Event, Classes A and B Converge to the Same Concept 87

Figure 44 The Distribution of Class A Data in Incremental Drift Experiment 3 Where the Incremental Drift Event Occurs as an Increase in Classes A and B, and a Decrease in Class C, and After the Concept Drift Event, Classes A and B Converge to the Same Concept 87

Figure 45 The Distribution of Class B Data in Incremental Drift Experiment 3 Where the Incremental Drift Event Occurs as an Increase in Classes A and B, and a Decrease in Class C, and After the Concept Drift Event, Classes A and B Converge to the Same Concept 88

Figure 46 The Distribution of Class C Data in Incremental Drift Experiment 3 Where the Incremental Drift Event Occurs as an Increase in Classes A and B, and a Decrease in Class C, and After the Concept Drift Event, Classes A and B Converge to the Same Concept 88

Figure 47 The Distribution of Data in Reoccurring Drift Experiment 1 Where The Reoccurring Drift Event Occurs As Increasing Manner Across All Three Classes 93

Figure 48 The Distribution of Class A Data in Reoccurring Drift Experiment 1 Where The Reoccurring Drift Event Occurs As Increasing Manner Across All Three Classes 93

Figure 49 The Distribution of Class B Data in Reoccurring Drift Experiment 1 Where The Reoccurring Drift Event Occurs As Increasing Manner Across All Three Classes 94

Figure 50 The Distribution of Class C Data in Reoccurring Drift Experiment 1 Where The Reoccurring Drift Event Occurs As Increasing Manner Across All Three Classes 94

Figure 51 The Distribution of Data in Reoccurring Drift Experiment 2 Where the Reoccurring Drift Event Occurs as an Increase in Classes A and B, and a Decrease in Class C 98

Figure 52 The Distribution of Class A Data in Reoccurring Drift Experiment 2 Where the Reoccurring Drift Event Occurs as an Increase in Classes A and B, and a Decrease in Class C 98

Figure 53 The Distribution of Class B Data in Reoccurring Drift Experiment 2 Where the Reoccurring Drift Event Occurs as an Increase in Classes A and B, and a Decrease in Class C 99

Figure 54 The Distribution of Class C Data in Reoccurring Drift Experiment 2 Where the Reoccurring Drift Event Occurs as an Increase in Classes A and B, and a Decrease in Class C 99

Figure 55 The Distribution of Data in Reoccurring Drift Experiment 3 Where the Reoccurring Drift Event Occurs as an Increase in Classes A and B, and a Decrease in Class C, and After the Concept Drift Event, Classes A and C Converge to the Same Concept 103

Figure 56 The Distribution of Class A Data in Reoccurring Drift Experiment 3 Where the Reoccurring Drift Event Occurs as an Increase in Classes A and B, and a Decrease in Class C, and After the Concept Drift Event, Classes A and C Converge to the Same Concept 103

Figure 57 The Distribution of Class B Data in Reoccurring Drift Experiment 3 Where the Reoccurring Drift Event Occurs as an Increase in Classes A and B, and a Decrease in Class C, and After the Concept Drift Event, Classes A and C Converge to the Same Concept 104

Figure 58 The Distribution of Class C Data in Reoccurring Drift Experiment 3 Where the Reoccurring Drift Event Occurs as an Increase in Classes A and B, and a Decrease in Class C, and After the Concept Drift Event, Classes A and C Converge to the Same Concept 104

Figure 59 The Performance of The System That Uses Strategy Based On Comparison of Datasets With Threshold As MAD on The Validation Set Which is Formed By Last Batch Method According to The Accuracy Metric Across Different Batch Numbers 114

Figure 60 The Performance of The System That Uses Strategy Based On Comparison of Datasets With Threshold As MAD on The Validation Set Which is Formed By Last Batch Method According to The Recall Metric Across Different Batch Numbers 114

Figure 61 The Performance of The System That Uses Strategy Based On Comparison of Datasets With Threshold As MAD on The Validation Set Which is Formed By Last Batch Method According to The Precision Metric Across Different Batch Numbers 115

Figure 62 The Performance of The System That Uses Strategy Based On Comparison of Datasets With Threshold As MAD on The Validation Set Which Contains Random Instances According to The Accuracy Metric Across Different Batch Numbers 116

Figure 63 The Performance of The System That Uses Strategy Based On Comparison of Datasets With Threshold As MAD on The Validation Set Which Contains Random Instances According to The Recall Metric Across Different Batch Numbers 116

Figure 64 The Performance of The System That Uses Strategy Based On Comparison of Datasets With Threshold As MAD on The Validation Set Which Contains Random Instances According to The Precision Metric Across Different Batch Numbers 117

Figure 65 The Performance of The System That Uses Strategy Based On Comparison of Datasets With Threshold As 2 MAD on The Validation Set According to The Accuracy Metric Across Different Batch Numbers 118

Figure 66 The Performance of The System That Uses Strategy Based On Comparison of Datasets With Threshold As 2 MAD on The Validation Set According to The Recall Metric Across Different Batch Numbers 118

Figure 67 The Performance of The System That Uses Strategy Based On Comparison of Datasets With Threshold As 2 MAD on The Validation Set According to The Precision Metric Across Different Batch Numbers 119

Figure 68 The Performance of The System That Uses Training Strategy Based On Accuracy on The Validation Set Which is Formed By Last Batch Method According to The Accuracy Metric Across Different Batch Numbers 120

Figure 69 The Performance of The System That Uses Training Strategy Based On Accuracy on The Validation Set Which is Formed By Last Batch Method According to The Recall Metric Across Different Batch Numbers 121

Figure 70 The Performance of The System That Uses Training Strategy Based On Accuracy on The Validation Set Which is Formed By Last Batch According to The Precision Metric Across Different Batch Numbers 121

Figure 71 The Performance of The System That Uses Training Strategy Based On Accuracy on The Validation Set Which Contains Random Instances According to The Accuracy Metric Across Different Batch Numbers 122

Figure 72 The Performance of The System That Uses Training Strategy Based On Accuracy on The Validation Set Which Contains Random Instances According to The Recall Metric Across Different Batch Numbers 123

Figure 73 The Performance of The System That Uses Training Strategy Based On Accuracy on The Validation Set Which Contains Random Instances According to The Precision Metric Across Different Batch Numbers 123

Figure 74 The Performance of The System That Uses Prediction Strategy Based on Similarity Score on The Validation Set Which is Last Batch According to The Accuracy Metric Across Different Batch Numbers 124

Figure 75 The Performance of The System That Uses Prediction Strategy Based on Similarity Score on The Validation Set Which is Last Batch According to The Precision Metric Across Different Batch Numbers 125

Figure 76 The Performance of The System That Uses Prediction Strategy Based on Similarity Score on The Validation Set Which Contains Random Instances According to The Accuracy Metric Across Different Batch Numbers 126

Figure 77 The Performance of The System That Uses Prediction Strategy Based on Similarity Score on The Validation Set Which Contains Random Instances According to The Precision Metric Across Different Batch Numbers 126

Figure 78 The Performance of The System That Uses the Diversity-Rewarding Prediction Strategy on The Validation Set Which is Last Batch According to The Accuracy Metric Across Different Batch Numbers 127

Figure 79 The Performance of The System That Uses the Diversity-Rewarding Prediction Strategy on The Validation Set Which is Last Batch According to The Precision Metric Across Different Batch Numbers 128

Figure 80 The Performance of The System That Use Prediction Strategy Based on Centroid Difference and Diversity on The Validation Set Which Consist Random Instances According to The Accuracy Metric Across Different Batch Numbers 129

Figure 81 The Performance of The System That Uses Prediction Strategy Based on Centroid Difference and Diversity on The Validation Set Which Consist Random Instances According to The Precision Metric Across Different Batch Numbers 129

Figure 82 The Performance of The System That Uses Prediction Strategy Based on Statistical Test on The Validation Set Which is Last Batch According to The Accuracy Metric Across Different Batch Numbers 130

Figure 83 The Performance of The System That Uses Prediction Strategy Based on Statistical Test on The Validation Set Which is Last Batch According to The Precision Metric Across Different Batch Numbers 131

Figure 84 The Performance of The System That Uses Prediction Strategy Based on Statistical Test on The Validation Set Which Consist Random Instances According to The Accuracy Metric Across Different Batch Numbers 131

Figure 85 The Performance of The System That Uses Prediction Strategy Based on Statistical Test on The Validation Set Which Consist Random Instances According to The Precision Metric Across Different Batch Numbers 132

Figure 86 The Performance of The System That Uses Prediction Strategy Based on Voting Right Constraints on The Validation Set Which is Last Batch According to The Accuracy Metric Across Different Batch Numbers 133

Figure 87 The Performance of The System That Uses Prediction Strategy Based on Voting Right Constraints on The Validation Set Which is Last Batch According to The Precision Metric Across Different Batch Numbers 133

Figure 88 The Performance of The System That Uses Prediction Strategy Based on Voting Right Constraints on The Validation Set Which Consist Random Instances According to The Accuracy Metric Across Different Batch Numbers 134

Figure 89 The Performance of The System That Uses Prediction Strategy Based on Voting Right Constraints on The Validation Set Which Consist Random Instances According to The Precision Metric Across Different Batch Numbers 135

Figure 90 The Performance of The System That Uses Prediction Strategy Based on The Jensen-Shannon Divergence Score on The Validation Set Which is Last Batch According to The Accuracy Metric Across Different Batch Numbers 136

Figure 91 The Performance of The System That Uses Prediction Strategy Based on The Jensen-Shannon Divergence Score on The Validation Set Which is Last Batch According to The Precision Metric Across Different Batch Numbers 136

Figure 92 The Performance of The System That Uses Prediction Strategy Based on The Jensen-Shannon Divergence Score on The Validation Set Which Consist Random Instances According to The Accuracy Metric Across Different Batch Numbers 137

Figure 93 The Performance of The System That Uses Prediction Strategy Based on The Jensen-Shannon Divergence Score on The Validation Set Which Consist Random Instances According to The Precision Metric Across Different Batch Numbers 137

LIST OF ABBREVIATIONS

Bagging	Bootstrap Aggregating
DDD	Diversity for Dealing with Drifts
DDE	Drift Detection Ensemble
DDM	Drift Detection Method
DDM-ORF	Drift Detection Method-Online Random Forest
DNN	Deep Neural Network
EWMA	Exponentially Weighted Moving Average
FN	False Negative
FP	False Positive
JDS	The Jensen-Shannon Divergence
logit	The inverse of the standard logistic function
MAD	Median Absolute Deviation
model	Machine Learning Model
NC	Nearest Centroid
NCM	Nearest Class Mean
NIDS	Network Intrusion Detection Systems
ODIN	Out-of-Distribution Detector for Neural Networks
OOD	Out-of-Distribution
ORF	Online Random Forest
OS-ELM	Online Sequential Extreme Learning Machine
SMOTE	Synthetic Minority Oversampling Technique
TN	True Negative
TP	True Positive

CHAPTER 1

INTRODUCTION

1.1 Motivation and Problem Definition

Machine learning models are fundamentally dependent on data for training, where the amount and quality of data significantly influence their performance. Due to dependence of machine learning models on amount of data, the need for more data, and make gathered data usable increases. Problems such as the limitation of resources available to gather data, the inadequacy of existing technological tools to collect data, and legal restrictions on data collection have encouraged both the development of methods to obtain data and the development of studies on how to use the available data resources more effectively. For the effective use of the available data, it is important to ensure that the content of the data is suitable for the purpose of use of the model, as well as its quality and correctness. This suitability means that the data must contain relevant patterns and examples tailored to the specific problems the model is designed to solve. When the underlying patterns in the data evolve—a phenomenon known as concept drift— suitability of the data for the designed model is deteriorates [1].

Concept drift is widely recognized as a real phenomenon, where underlying data distributions change over time. However, defining and quantifying concept drift by mathematical measurements remains a subject of ongoing debate among researchers. While the existence of concept drift is generally acknowledged due to its significant impact on the performance of predictive models, the methods for measuring it vary, and there is no single consensus on a standardized mathematical approach. There are techniques developed to identify the phenomenon of concept drift. For instance, Yang et al. propose a method that calculates the Euclidean distance between the centroid of the test sample and the centroid of the new sample, and if the difference in centroids exceeds a threshold value, it detects concept drift since the concept of the new sample drift from that of the test sample [2]. They determine the threshold value by multiplying the Median Absolute Deviation (MAD) value of the test sample with a value manually determined by the user.

Concept drift is the phenomenon of changing statistical properties in a domain [3]. Many of the existing studies focus on detecting concept drift to adapting their model for new concept by analyzing the concept drift [1]. These studies use an approach that is fractional. Firstly, they determine whether there is concept drift. Then, they adapt their learners for new concept by analyzing new concept. However, besides this popular approach for handling concept drift, there are also studies that recommend developing new models for new concepts [4].

In machine learning, data drift and concept drift both refer to changes that can occur over time in the data or the relationships that your model is learning. Although these terms are commonly used

interchangeably, there are studies that separate these notions. Data drift particularly refers to changes in the input data distribution over time [5]. On the other hand, concept drift specifically refers to changes in the underlying relationship between the features and the target variable. Our research addresses both concept drift and data drift phenomena, under the broader title of concept drift, whose effects and definitions in the literature cannot be entirely distinguished from each other.

The main motivation behind this study is to address the gap created by the prevailing trend in the literature, which focuses predominantly on detecting concept drifts and defers the analysis of their resolution to subsequent discussions. Thus, the research aims not only to detect concept drift but also to develop a comprehensive method that simultaneously resolves it, thereby presenting a holistic approach. To develop a method capable of comprehensively resolving the issue of concept drift, it is necessary to both detect concept drift and simultaneously develop an approach that will contend with it.

This study proposes a holistic approach in the field of concept drift by comparing new data with the datasets on which existing learners have been trained to test whether they share the same concept. If they do not share the same concept, the study suggests developing different learners tailored to the distinct concepts. This dual approach ensures that the system is not only aware of changes in data patterns but is also prepared to adapt and respond to these changes effectively.

Our designed system continuously monitor incoming data streams to detect any drifts from the established patterns. This involves setting up mechanisms to identify drift in the data distribution that signal a concept drift. When a concept drift is detected, our system is capable of creating new models tailored to the new data patterns. Furthermore, if concept drift has not occurred, the system will use the incoming data to enhance the existing models, thereby increasing the data on which the models are trained. Even the new data does not contain a separate concept, it is valuable for understanding new or previously unnoticed details and features within the current concept. Consequently, the existing models will better generalize the concepts they pertain to, and their performance will improve.

This study explores various methods for detecting concept drift [2][6][7][8][9]. For instance, in addition to approaches based on differences in centroids between datasets, the performance of developed models on new datasets has also been employed to identify concept drift. Our study's approach to detecting concept drift based on centroid differences involves setting a threshold according to the distribution in the datasets used for training the models. If the difference in centroids between the new data and the current dataset exceeds the breadth of the distribution of the existing dataset, our system assumes that the new dataset differs from the compared dataset. According to Yang et al., the Median Absolute Deviation (MAD) is a valuable metric for quantifying the spread of a dataset's distribution and provides a practical threshold value that indicates how low the centroid difference between datasets must be to consider them as sharing the same concept [2]. In our study, we have adopted this assumption according to our experimental results and study's objectives. On the other hand, our approach based on model performance considers that if the performance of the model decreases on a new dataset, this may indicate a difference in the concept between the dataset used for training the model and the new incoming dataset. In determining the extent of performance degradation in the learner that would imply its failure in adapting to the concept of a new dataset, we utilized the performance of a hypothetical model. This model would be trained using both the dataset on which the learner was originally trained and the new incoming dataset as a unified training set. The performance of this hypothetical model established a threshold that enabled us to identify under-performance in the learner's predictions when applied to the new dataset. If the learner under-performs according to this criteria, it

can be assumed that the new dataset represents a different concept. There will be a need to train a new model specifically for this different concept. Thus, our system can perform incremental learning even in the presence of concept drift.

Our study does not test for concept drift across the entire dataset, but rather examines whether the concept of classes within the new dataset has changed. For example, if some classes in the new dataset share the same concept as the datasets used to train current models, the data from these classes are merged with the existing model's datasets. New models are only trained for classes in the new dataset that do not have existing models trained on datasets with the same concept. Thus, if drift occurs only within specific classes, new models are trained specifically for these classes, while classes without change are used to update existing models.

Both the imbalance among classes within real-world datasets and the presence of differing concepts for some classes, while others remain unchanged, contribute to class imbalance in the datasets used to train models. The scope of our research covers this issue. To mitigate these challenges, our study employs synthetic data generation methods. We use SMOTE (Synthetic Minority Oversampling Technique) to generate synthetic data based on existing data in our work.

In this study, the training phase is completed by training a different learner for each distinct concept. After the training phase, various strategies have been tested to determine how the system will predict test datasets. The purpose of prediction strategies is to evaluate the capacity of the generated models to understand the datasets that are predicted and to enhance the influence of models with a high capacity for understanding the data in the ultimate prediction of the ensemble system. For this purpose, prediction strategies rely on metrics developed to measure the model's capacity to understand the concept of the dataset to be predicted. Assuming that the classes of the test dataset are unknown, measurements are derived to determine how closely the concepts of the datasets used to train the existing models align with the test dataset. Weights are then assigned to each learner based on these measurements, and weighted voting mechanism is employed. The similarity of concepts between datasets is measured by comparing the entire datasets with each other, rather than evaluating classes separately. This approach is due to the assumption that the classes of the test set are unknown. In this context, formulations derived from the differences between the centroids of the datasets, statistical tests such as the Mann-Whitney U test, and measurements like the Jensen-Shannon Divergence have been used to assign weights to learners. For the weight assignment mechanism, strategies involving granting each model a certain voting right have been utilized. In this strategy, each model is assigned a weight based on the aforementioned measurements, depending on how closely the dataset it was trained on aligns with the test set. In addition to this voting strategy, other strategies were also explored, such as granting a single vote to models trained on datasets about which there is insufficient information to reject they belong to same concept from the test set.

Our approach can be used in various domains such as security, energy system management, or other data-intensive fields where concept drift is a critical challenge. Security models rely heavily on patterns of malicious activity to detect attacks. However, attackers constantly modify their techniques to evade detection tools, causing concept drift. Our system has the capability to dynamically adapt to different concepts. Instead of relying solely on static detection methods that flag a data distribution change and leave the adaptation to a separate process, our system actively updates or expands the set of learned concepts. In the security domain, where threats evolve rapidly and quick decision-making is critical, this adaptability is essential. Similarly, in energy systems, machine learning models are employed to predict energy consumption patterns, optimize grid operations, and manage renewable energy sources

efficiently. However, environmental changes, fluctuations in energy demand, and changes in supply patterns introduce concept drift, which can reduce the predictive accuracy. Our adaptive system can dynamically respond to these changes, helping maintain stable and efficient energy grid operations even under evolving conditions. Beyond these areas, any data-intensive fields, such as retail, e-commerce, finance, or healthcare, can benefit from our approach.

In summary, this study aims to present a comprehensive approach to handling the issue of concept drift, rather than deferring its resolution to subsequent analyses conducted after detecting concept drift, as is often seen in the literature. Our study proposes training a separate model for each different concept and utilizes majority voting strategies to make predictions on a test set after the models have been trained. Consequently, the approach we have developed has the capacity to predict datasets across different concepts, providing a holistic and automated solution to the problem of concept drift.

1.2 Research Aim and Objectives

The primary objective of this research is to design a system that can automatically adapt to concept drift event, providing dynamic solutions to the problems caused by concept drift. Accordingly, the research addresses the following objectives:

1. Determination of the method for detecting concept drift phenomenon.
2. Method for utilizing the available dataset in the most optimal manner where concept drift occurs only within specific classes of a dataset.
3. Solution of the possible class imbalance problem in datasets used for training models.
4. Determination of voting strategy for effectively using developed models that are designed to generalize different concepts for predicting test data.

1.3 Organization of the Thesis

The thesis is organized into five chapters. The following chapter provides a literature review on addressing the concept drift problem. Chapter 3 describes the study's methodology. Chapter 4 reports on experiments that were implemented using generated data reflecting various concept drift scenarios, as well as a real-world dataset, and evaluates the results of these experiments. Chapter 5 discusses the contributions of this study and the limitations of the proposed method. Chapter 6 outlines the key findings and proposes future works.

CHAPTER 2

LITERATURE REVIEW

There are studies that address concept drift problem which our study aims to solve. In this chapter, we explain these studies to provide information about the domain we study, and show our study's contribution for research area that aims to solve the concept drift problem.

2.1 Detecting and Explaining Domain-Oriented Concept Drift

In this section, studies focused on detecting concept drift and explaining it after its identification will be discussed. These studies implement works on detecting and explaining concept drift, typically defer any solution to later stages that are not included in these studies [10]. The methods proposed in these studies are generally developed more for the detection of concept drift rather than for its explanation, which is often relegated to conventional and straightforward statistical analyses [11]. This type of approach is the most commonly seen in the literature.

In the literature, there are studies on concept drift that focus on types of concept drift occurring in specific domains and on addressing the concept drift problem in their respective fields. Since these studies focus on solving problems within certain domains, they typically limit their explanations of concept drift to what is sufficient for detecting concept drift and mitigating its effects in their area of focus instead of addressing the general issues of the concept drift event.

The field of security is one of the key areas where efforts are ongoing to address the problem of concept drift. The evolution of data over time within security applications can create security vulnerabilities. When concept drift occurs, a security application may fail to understand the new concept, potentially allowing malicious software to infiltrate and mislabeling benign software as adversarial attacks. Therefore, solving the problem of concept drift is a critical challenge in the security domain.

Yang et al. propose a method that examines the relationship between the incoming data and the training data to detect concept drift for security purposes [2]. They employ a contrastive learning to map data samples into a low-dimensional space and learn a distance function to measure dissimilarity between samples. By using this map, they calculate the centroid for each class based on the training samples, and the Euclidean distance between the testing sample and each class's centroid. They determine whether the testing sample is an outlier for each class by comparing their centroid distance and Median Absolute Deviation (MAD) value. MAD provides a measure of the variability in the dataset that is less sensitive to outliers compared to the standard deviation. Thus, MAD provides class-specific distance thresholds to identify outliers. If centroid difference exceeds the threshold that is dependent on MAD

value, they assume that it is a drifting sample. The second part of their study, which focuses on explaining the concept drift, concentrates on understanding the drift after it has been detected. For this purpose, researchers have adapted the perturbation method, commonly used in data science for assessing variable importance, to address their specific challenges. They perturb features by replacing them with corresponding values from a reference training sample that is closest to the centroid. By this way, they identify the features whose substitution with the reference values has the most significant influence on the distance to the centroid. The authors claim that this explanation provides knowledge to update model for handling the concept drift. This study, which is quite effective at detecting concept drift, focuses solely on identifying which variables have changed the most in the explanation phase. They leave the solution of concept drift to whom utilizing its detection and explanation methods by claiming that the finding of their method can be used for solving the concept drift problem.

Kuppo et al. claim that detecting the concept drift occurrences is the key element to robust machine learning models against rapidly evolving threats against security applications [6]. Therefore, they developed a method that detects concept drift by comparing new coming data with corresponding data that exist in their system. Firstly, the embedding network model, trained to keep similar examples close and dissimilar ones apart, transforms high-dimensional input data into a lower-dimensional output space. The resulting low-dimensional representations are given into the nearest class mean (NCM) classifier as input, which labels them based on proximity to the average value of a class in the dataset used to train the NCM model. An abnormally large distance indicates a change in the underlying data distribution, suggesting a potential concept drift. It means that if an example is not sufficiently close to the mean of any existing classes, it can be interpreted as an indication of concept drift. The criterion for sufficient closeness is determined by whether the distance exceeds a predefined threshold value. The authors mention that the threshold value can vary depending on the dataset's characteristics, so they do not provide a formula or explanation for determining the threshold.

Other domain-oriented studies that focus on concept drift events are in the energy domain. Energy production and consumption have a non-stationary nature, as they are highly susceptible to seasonal effects, climatic changes, and global economic and political events. This inherent nature of the energy domain serves as a fundamental motivation for studies addressing concept drift.

Fenza et al. introduce an anomaly detection framework tailored for smart grids [12]. Smart meters, which are integral to modern energy systems, generate near real-time consumption data, enabling efficient anomaly detection for energy optimization and fraud prevention. However, anomaly detection systems often suffer from high false positive rates due to their inability to adapt to changing consumption patterns, known as concept drift. Therefore, they combined clustering and Long Short-Term Memory (LSTM) neural networks to address these issues. Initially, K-means clustering is applied to historical time series data to extract distinct energy consumption profiles. These profiles serve as training centroids for the LSTM network which is tasked with predicting future consumption trends. Anomalies are identified by analyzing deviations between predicted and actual consumption values, using standard deviation thresholds derived from recent prediction errors. Thus, their system can distinguish between genuine anomalies and normal behavioral changes caused by evolving user habits. Although they have developed an approach that clusters user behavior and makes predictions based on the specific characteristics of these behaviors using deep learning models, their work does not have a structure that automatically adapts concept drift. Their study is limited to detect concept drift events occurring in the Smart Grid domain.

2.2 Addressing Concept Drift for Incremental Learning

Incremental learning domain is one of the key areas that is related to the concept drift problem [13]. Incremental learning, which is also known as online learning or continual learning, implement training for machine learning model continuously as new data becomes available [14]. In an incremental learning setup, models are continually updated by using new data. The update process makes them sensitive to shifts in patterns and distributions. These changes can cause previously learned knowledge to become obsolete or incorrect, and leads to decreased accuracy and reliability. Significant changes, such as the concept drift, can lead to major issues in an incremental learning setup. Therefore, addressing the concept drift while implementing incremental learning holds a prominent place in the literature.

Yang et al. focuses on detecting the concept drift in non-stationary environments using a two-fold approach [15]. Their approach is based on the combination of the adaptive nature of online sequential extreme learning machines (OS-ELMs) and with a drift detection mechanism to response to the changing data dynamics. In the first stage, they use OS-ELMs to continuously update and adapt models based on incoming data streams. The OS-ELM provides the dynamic updating of model parameters in real-time without retraining from scratch, thus efficiently handling data variability over time. In the second stage, they introduce a mechanism which quantifies the extent of model changes due to new data for determining whether a concept drift event has occurred. For determining the concept drift, after the model is updated, they measure the dissimilarity between the updated model and the model before the update. If the dissimilarity exceeds the dynamic threshold, they assume that concept drift occurs. The dynamic threshold is adjusted based on performance metrics calculated over the validation set. They claim that detection of the concept drift provide sources to update OS-ELMs to adjust parameters such as learning rates or weights for better understanding of new data patters. In this study, their two-fold approach is tested on both synthetic datasets and real-world data for energy production forecasting in wind plants.

Jemili et al. introduces the Drift Detection Method Online Random Forest (DDM-ORF) model for intrusion detection that is a combination of the Drift Detection Method (DDM) for detecting concept drift and Online Random Forest (ORF) for incremental learning [16]. The approach addresses the challenges of dynamic and nonstationary data. The DDM component ensures real-time drift detection, while ORF enables continuous model updates without retraining from scratch. DDM continuously monitors the error rate of the classification model, comparing it with predefined thresholds. When deviations exceed acceptable limits, it signals the presence of drift. Once drift is identified, the Online Random Forest mechanism ensures incremental learning by dynamically updating decision trees without requiring a full retraining of the model. In short, the DDM-ORF model detects concept drift and facilitates incremental learning to ensure real-time adaptability in intrusion detection systems.

2.3 Ensemble Methods for Dealing with Concept Drift

An approach used in the literature to address concept drift is ensemble voting. In this approach, multiple models are trained to predict, and the collective outcome from this ensemble of models is accepted as the ultimate result. Different strategies like weighted voting, hard voting, bagging etc. can be used to determine the final outcome of the ensemble. The fundamental motivation of this approach is to gain the capacity to understand different concepts by training various models.

Minku et al. develop Diversity for Dealing with Drifts (DDD) method that utilizes varying diversity levels within an ensemble of machine learning models to maintain high generalization performance on both old and new concepts [7]. Before a concept drift is detected in the Diversity for Dealing with Drifts (DDD) framework, there are two ensembles, low diversity and high diversity. The former refers to the individual models within the ensemble which tend to make similar predictions to each other and the latter refers to those that make different predictions from each other. Once the concept drift is detected, the system initiates several adaptive measures. New low and high diversity ensembles are created to learn the new concept from scratch, allowing them to adapt without the influence of previous data. Also, the existing high diversity ensemble shifts to a low diversity learning strategy to facilitate faster convergence to the new concept. However, they do not propose a concept drift detection method to use in this stage of their system. They mention some detection algorithms from the literature which may be used. Their system's predictions are then made using a weighted majority vote strategy. The weights for these votes are determined based on the previous accuracy of each ensemble since the last detected drift, ensuring that the most accurate models have greater influence on decision-making. This means that their system may not perform well when the concept of the tested data differs, because the weight for voting given to the models is dependent on their performance on previous test sets. They used both artificial and real-world datasets which are KDD Cup 1999 network intrusion detection data, and PAKDD 2009 credit card data, and Australian electricity market data to test their system.

2.4 Ensemble Methods for Detecting Concept Drift

In the literature, an approach used to detect concept drift involves the use of detector ensembles. In these approaches, multiple detectors are created and organized as an ensemble [17]. Whether concept drift has occurred is not determined based on the result of a single detector model, but rather on the collective outcome of the ensemble models. The collective outcome is determined by using strategies like weighted voting, qualified majority voting, and relative majority voting. The fundamental difference between these studies and those in the field of 'Ensemble Methods for Dealing with Concept Drift' is that the purpose of forming an ensemble is not to create models specialized in different concepts, but rather to increase the precision or recall value of concept drift detection.

Maciel et al. introduce the Drift Detection Ensemble (DDE) which is a lightweight classifier ensemble that integrates signals from three different concept drift detectors [8]. Concept drift is detected according to warnings of concept drift detectors which can be chosen by users for custom setups. While the authors note that users can specify the detectors, they have recommended six different detectors that can be selected as three detectors of their DDE system. The strategies that are used by these detectors can be summarized as follows. The Drift Detection Method (DDM) tracks the classifier's error rate to signal drifts when significant increases occur. The Early Drift Detection Method (EDDM) builds on DDM by monitoring the distance between consecutive errors, allowing for earlier drift detection. Adaptive Windowing (ADWIN) dynamically adjusts its analysis window to reflect the most current data, detecting drifts through statistical differences in sub-window averages. EWMA for Concept Drift Detection (ECDD) uses an exponentially weighted moving average to spot substantial shifts in data means. Paired Learners (PL) employ two models, one stable and one reactive, to detect drifts by comparing their performance, where the reactive model quickly adapts to new data. Lastly, Hoeffding's Bound-based Drift Detection Methods (HDDM) utilize statistical bounds to detect changes, with variations like HDDMA and HDDMW applying moving and weighted moving averages, respectively. The user specifies how many warnings of detectors need to identify concept drift for it to be consid-

ered detected using the "sens" (sensitivity) parameter. For instance, if "sens" is specified as 2, the DDE system requires two detectors to signal concept drift. After detecting a concept drift by system, DDE resets its detectors and signals a drift event to the base classifier, which can then take appropriate actions such as retraining.

2.5 Developing Robust Model That Is Not Affected By Concept Drift

In this section, we discuss studies aimed at creating models that are minimally affected by concept drift, rather than detecting or resolving it. These studies focus on designing systems that not harmed by the variations in data distributions, and provide stable performance against the changes in the data.

Andresini et al. assert that while Deep Neural Networks (DNN) models are highly beneficial tools for network intrusion detection systems (NIDS), they fail to fully use their potential due to issues encountered in situations of concept drift [18]. To address this issue, they developed a method which is semi-supervised, a Nearest Centroid Neighbor classifier, named as INSOMNIA. The INSOMNIA method utilizes a continuously updated model to adapt to the time-varying distributions in network traffic characteristics. The primary goal of INSOMNIA is to mitigate the effects of concept drift and ensure the network security system can rapidly adapt to changing conditions. The model operates by monitoring changes within network traffic and updating itself in response to these changes. Rather than directly detecting concept drift, it is designed to counteract the effects caused by such conditions. DNN model that is used in INSOMNIA learns an intrusion detection model using labeled historical network traffic data. Additionally, a label predictor model is trained to support a semi-supervised learning approach. This model is typically a Nearest Centroid Neighbor (NC) classifier and is used to generate approximate labels for incoming data points. According to the authors, it is crucial to train the DNN and NC models independently because if the prediction model (DNN) updates itself based on its own predictions, the model could repeatedly learn from its mistakes, leading to bias. The labels provided by the NC model enable the INSOMNIA system to update the DNN model with unlabeled data as well. Since the DNN model continues to be trained with both newly arriving labeled and unlabeled data, it remains unaffected by changes in the distributions of network traffic over time. Thus, a robust DNN model against concept drift can be created.

He et al. claim that when concept drift occurrs, regardless of implementation of retraining, degradation in performance is inevitable [19]. Therefore, they propose developing an incremental learning framework that is robust against concept drift and catastrophic forgetting instead of detecting and solving concept drift. They introduce a dynamic method for updating the exemplar set using features from new observations of old classes. They implement training process with exemplar set which contains only samples that are most representative of the class mean in the feature space. In their approach, when a new dataset arrives, if the features of the classes have changed, samples that demonstrate the common characteristics of both the new dataset and the data accumulated up to that point can be selected to update the exemplar set. This aims to ensure that the model does not predict based on a specific concept, but rather learns patterns according to the average of all concepts. They assert that implementing the training process with dataset which generalizes all concepts makes the model robust against concept drift.

2.6 Out of Distribution Detection

The field of Out of Distribution (OOD) Detection is categorized under different labels in the literature. However, it shares many similarities with studies on concept drift, both in terms of the problems it addresses and the methods used. While concept drift refers to the change in the underlying data distribution over time, OOD refers to identifying data points that fall outside the distribution on which the model was trained [20]. Although there are nuances in their formal definitions, in many cases they refer to the same situation because the evolution of incoming data towards a different distribution from the dataset the model was trained on means that the incoming data fall outside the distribution from the training dataset. Therefore, the intersections emphasize techniques developed in one field can be adapted to solve the challenges in the other.

The field of Out of Distribution (OOD) Detection studies generally focus on deep learning models in the literature although not necessarily required to be defined as OOD [21][22]. Also, the studies in this domain are mostly related to applications of image classification tasks.

Hu et al. address the challenges of maintaining and enhancing deep neural networks (DNNs) when data distribution shifts [9]. They state Out of Distribution (OOD) data as a problem because the prediction of a DNN model may be erroneous and unreliable if the data follow a different distribution while a DNN model can mainly correctly predict the data that follow the same distribution as the training data. In their study, they use the OOD detection method to detect the data that are from the mutated version derived from a dataset. To minimize error that is caused by OOD data, they proposed a system. Their system begins with the preparation of data, where four different datasets which are a training set for initially training the DNN, an ID candidate set that follows the same distribution as the training data, an ID test set to evaluate the initial performance of the model, and an out-of distribution (OOD) candidate set.

As illustrated in Figure1, the DNN is trained using the training set. From a candidate set, data is selected based on specific selection metrics like Random Selection, Multiple-boundary Clustering and Prioritization (MCP), Cross Entropy Based Sampling (CES), Distance-based Surprise Adequacy (DSA), DeepGini, and Entropy. Additionally, the Jensen-Shannon divergence (JSD) is used to measure the distribution differences between the original training set and the mutated test sets, ensuring that the selection metrics are capturing significant distribution shifts. The DNN is then retrained using the selected set. The selected data can be used as either selected data from candidate set or the combination of selected data from candidate set and the original training set. Finally, the retrained DNN is tested using both an in-distribution (ID) test set and a general test set to evaluate its accuracy. They claim that retraining the DNN model with the selected set solves the issues caused by OOD events because the selection metrics used for choosing the selected set and the check for generalization with Jensen-Shannon Divergence (JSD) ensure that the selected subset generalizes both ID and OOD data.

Berend et al. focus on the event that the Deep Learning (DL) models typically perform well when new inputs resemble the training data but struggle with OOD inputs, which can lead to incorrect behavior or decisions [23]. The authors assess performance of OOD detection methods such as softmax-based methods, Mahalanobis distance, and ODIN (Out-of-Distribution Detector for Neural Networks). Softmax-based methods typically rely on the confidence scores of softmax outputs. If the model's prediction for an input has low confidence (i.e., the softmax probability is low), the predicted data is considered as OOD. Mahalanobis distance measures the statistical distance between the feature repre-

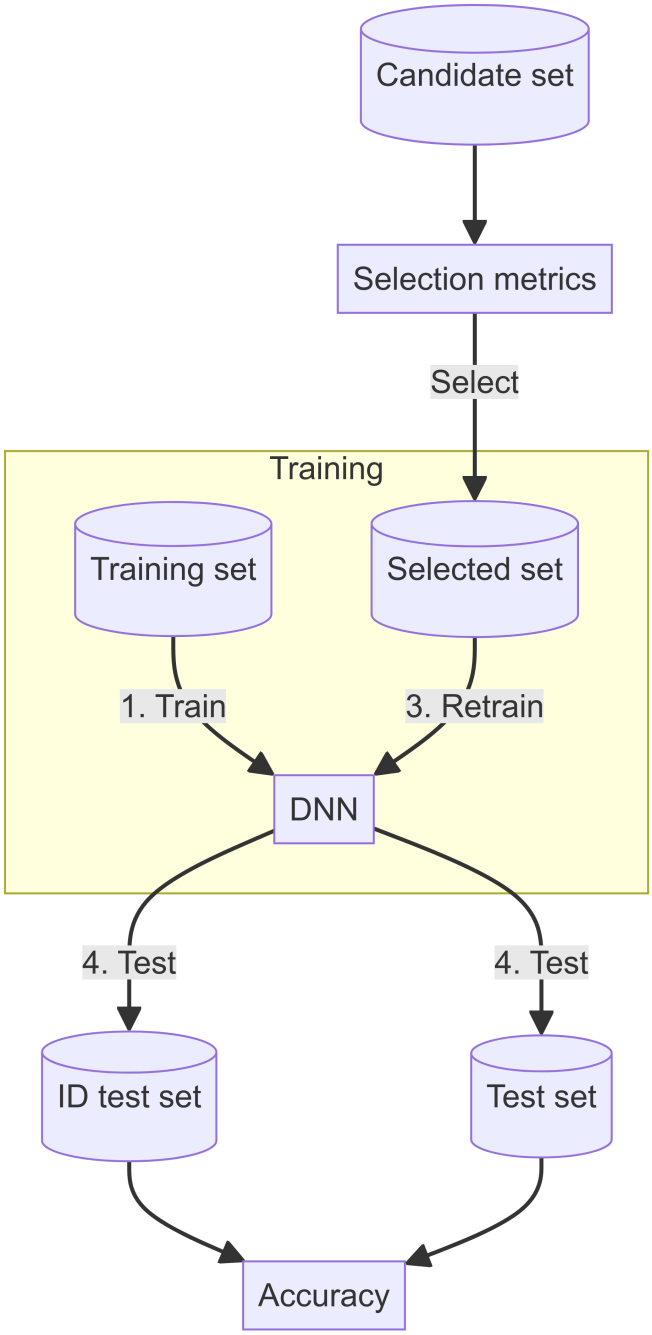


Figure 1: The Method That Is to Solve Out of Distribution Problem Adapted From [20]

sentation of data tested and the class means in the feature space of the trained model. A larger distance suggests that the data is OOD. ODIN apply temperature scaling to soften the softmax output and introduce small, gradient-based perturbations to the input data. Thus, ODIN reduces model confidence for OOD samples while increasing confidence for in-distribution data. According to their findings, these techniques can detect the ID data as they recognize ID data patterns, but performance of these models on detecting OOD differs. Softmax-based methods may not always be reliable for OOD detection,

especially if the model has been trained to be overly confident. Mahalanobis Distance method tends to be more sensitive to samples that are statistically far from the training data distribution, so it is a good option where high precision rates are required in detecting OOD samples. ODIN generally outperforms for distinguishing between in-distribution and OOD inputs more effectively. Additionally, they inspect mutation operators such as translation, scale, shear, rotation, contrast, brightness, blur, and noise. Their inspection shows that mutation operators, like blur and scale, are more likely to generate OOD data because these operators significantly alter the original characteristics of the images by urging them outside the distribution of the training data. Also, they tried these mutations on different type of data such as benign data which the model classifies correctly and problematic data (error test case) which the model classifies incorrectly. Their experiments show that problematic data generated by the mutation operators were more likely to be considered OOD compared to benign data. These findings imply that the model's performance may be considered as a metric to measure OOD event. Furthermore, the authors analyze which testing criteria, such as Mutation Operators, Coverage Criteria, Outlier Exposure, TPR99, Neuron Coverage (NC), k-Multisection Neuron Coverage (KMNC), Neuron Boundary Coverage (NBC), Strong Neuron Activation Coverage (SNAC), Top-k Neuron Coverage (TKNC), and Fast Approximate Nearest Neighbor (FANN), better highlight the strengths and weaknesses of a model. In their studies, the authors assert that Neuron Coverage (NC), k-Multisection Neuron Coverage (KMNC), and Top-k Neuron Coverage (TKNC) testing criteria tend to generate more "In-Distribution" (ID) data, the Fast Approximate Nearest Neighbor (FANN) testing criterion tends to keep the model closer to its training data, and the Neuron Boundary Coverage (NBC) and Strong Neuron Activation Coverage (SNAC) testing criteria tend to generate more Out-of-Distribution data. The methods that are studied in that paper have the potential to be adapted to the concept drift problem on image classifier models.

CHAPTER 3

METHODOLOGY

3.1 Problem Definition

Machine learning models are intricately dependent on the quality and volume of data utilized during their training phase. The performance of these models is profoundly influenced by the quantity and quality of the data. Therefore, the effective utilization of the available data is a significant issue. Effective use of the data is achieved by addressing the inherent problems within the data and ensuring compatibility with the machine learning models.

One of the challenges that hinder the proper utilization of acquired data is the problem of concept drift. Concept drift refers to the phenomenon where the patterns and general distribution of the data is changed. Machine learning models assume that the patterns they learned from the training data will hold in the future. When concept drift occurs, these patterns no longer represent the current situation, leading to degraded performance of the model. Therefore, detecting concept drift, and maintaining our system to address this issue is important to effectively use the acquired data.

To address the concept drift issue, it is essential not only to detect this phenomenon of concept drift but also to develop a method suitable for solving the problem. In this context, an effective way to solve the concept drift problem is to train distinct models capable of generalizing the identified concepts, rather than creating a single model that attempts to handle different concepts.

3.1.1 Types of Concept Drift Event

Concept drift can occur in various forms, and the way it happens has significant effects on machine learning models. To address this, specific types of concept drift have been identified and categorized. Furthermore, concept shift is a broad term used to describe changes in data over time. For instance, it is quite common to use the term concept shift to refer to changes in the distribution of data as well [24][25] .

Data drift specifically refers to changes in the input data distribution over time, whereas concept drift describes changes in the underlying relationship between the features and the target variable [5]. Also, there are other understandings claim that while real drift is referred to as concept drift, virtual drift is called as data drift [26]. While these two types of drift are conceptually distinct, they often interact or co-occur in real-world applications, making it difficult to draw a clear line between them. For instance, a shift in the data distribution (data drift) can precipitate alterations in how features map to the target

(concept drift) and vice versa. As a result, we address both phenomena together under the broader title of concept drift, which allows us to more comprehensively capture and address the various ways in which models may degrade over time.

In the literature, there are four main concept drift events which are sudden drift, gradual drift, incremental drift, and reoccurring drift [27][28][29][30].

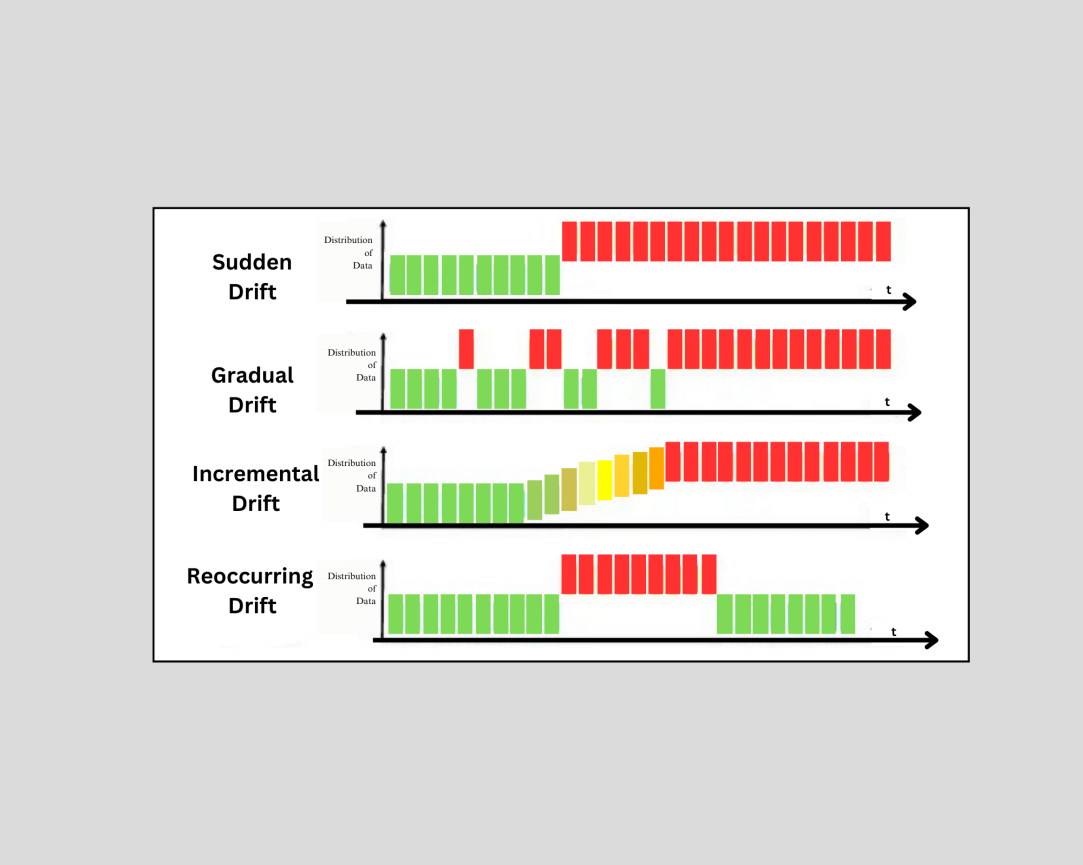


Figure 2: Main Concept Drift Scenarios in the Literature

As illustrated in Figure 2, sudden drift occurs when the underlying distribution of the data changes abruptly. This type of concept drift is characterized as quick change from one concept to another. Gradual drift is a type of concept drift where the transition to a new concept is not permanently achieved on the first attempt. After moving to the new concept, there are returns to the old concept, and it is only after these oscillations that the change to the new concept becomes permanent. During this period, instances of both old and new concepts are present in the data stream, increasing in frequency until the new concept completely replaces the old one. Incremental drift involves a series of small, progressive changes that accumulate over time, leading to a significant shift in the data distribution. While two states are clearly defined in gradual drift, incremental drift refers continuous evolution of the data. Reoccurring drift happens when old concepts which is previously seen in the data stream return after some time. This could be due to cyclic or seasonal patterns inherent in the domain. Reoccurring drift involves predictable, cyclical returns of previous concepts different from slow, continuous transition between old and new concepts over time characteristics of gradual drift.

Although the concept drift phenomenon is categorized into specific scenarios, there is no universally agreed-upon mathematical definition in the literature. Various paradigms have been developed to define and detect concept drift. While concept drift is commonly addressed in the literature through four main scenarios, real-world datasets do not always follow these scenarios. In real word concept drift may manifest itself as degenerate forms of these four scenarios or as interactions between them.

3.2 General structure of the system

The main idea behind our system is developing different learners for each different detected concept, instead of adapting a single model for generalizing all concepts. Thus, the goal is to create a system that directly addresses the problem without deferral, as opposed to the single model approaches that suggest diagnosing the concept drift through data analysis and then propose solutions. This approach aims to provide immediate resolution following the detection and understanding of the concept drift.

As illustrated in Figure 3, if there are trained models within the system, the arrival of a new dataset prompts a check to determine if new dataset's concept aligns with the concepts understood by the existing models, thereby identifying whether concept drift has occurred. Accordingly, if concept drift is detected, a new model is trained using the incoming dataset, and like the other models, it is associated with the dataset it was trained on and recorded. If the new dataset has a concept that any of the previously trained models can generalize, it is merged with the dataset of the corresponding model, and the model is updated using the augmented dataset. This process is repeated with each new dataset arrival. If a new dataset arrives and there are no existing trained models, a new model is trained for this dataset, which is considered a new concept not previously registered in the system, and it is recorded in association with its training dataset.

3.3 Strategies For Detecting Concept Drift in the Training Process

To detect concept drift, two primary strategies are employed: testing whether the dataset used to train the models and the newly arrived dataset belong to the same concept, and determining whether there is a decline in the performance of the trained models on the new dataset. Additionally, various approaches have been utilized to determine whether the newly arrived dataset and the dataset used to train the models belong to the same concept. These approaches will be described in the subsections of the main strategies.

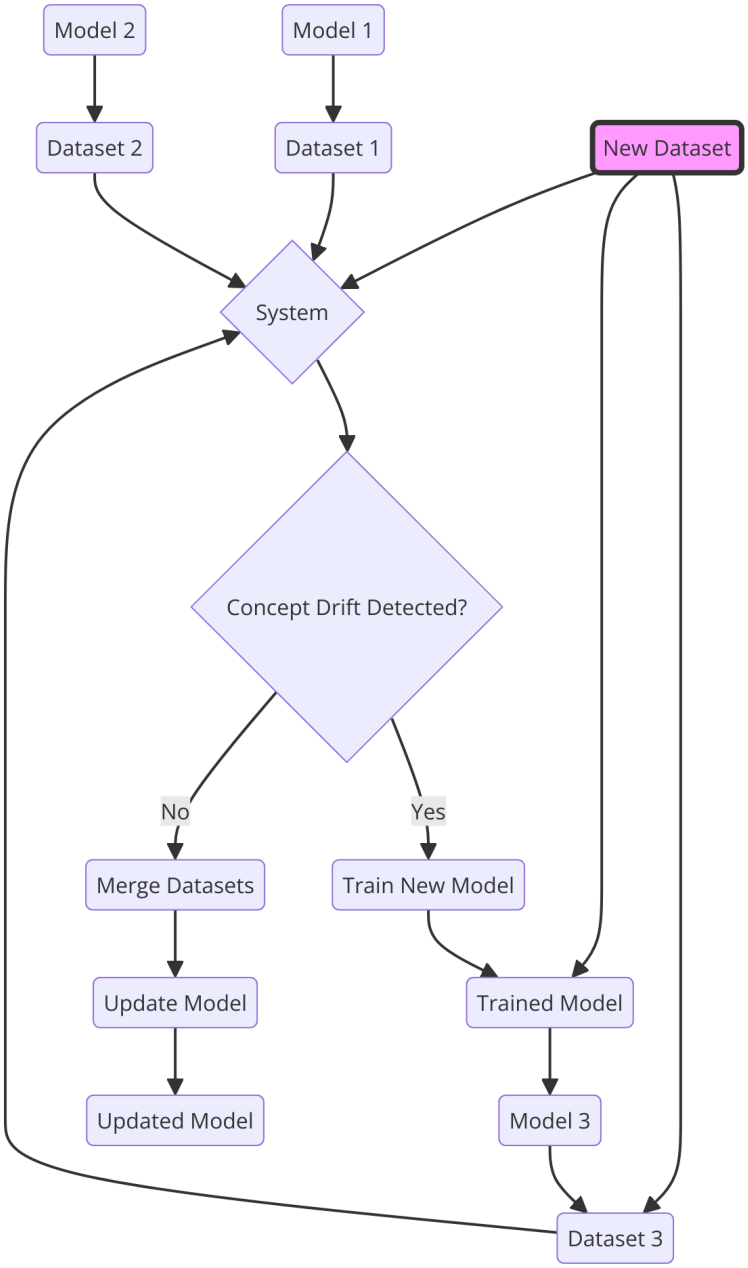


Figure 3: General structure of adaptive system for dynamic handling of concept drift

3.3.1 Strategy Based On Comparison of Datasets

The strategy based on comparing datasets use the centroid metric for detecting concept drift.

Let $X = \{X_1, X_2, \ldots, X_{n_X}\}$ and $Y = \{Y_1, Y_2, \ldots, Y_{n_Y}\}$ be two samples with sizes n_X and n_Y respectively. The centroids (means) of these samples are defined as:

$$\mu_X = \frac{1}{n_X} \sum_{i=1}^{n_X} X_i \tag{1}$$

and

$$\mu_Y = \frac{1}{n_Y} \sum_{i=1}^{n_Y} Y_i \tag{2}$$

The centroid difference between the two samples is given by:

$$\Delta\mu = \mu_X - \mu_Y \tag{3}$$

Rather than using the entire datasets, methods based on comparison utilize individual classes within the datasets. For example, if only certain classes have different concept in the new dataset compared to the datasets on which the existing models were trained, only data in these classes are considered as new concepts, and different models are trained specifically for these classes. Classes that retain the same concept as any previously saved data are merged with the corresponding dataset containing data from the same class and concept, and the model trained on this merged dataset is updated using the newly combined data. Thanks to this approach, classes that belong to the same concept are utilized to train the same model, aiming for more effective use of the available dataset. Instead of comparing the entire dataset, comparing each class with other datasets containing the same class makes the approach more flexible and adaptable to the specific dataset in use.

After calculating the centroid difference for each class, a critical point is to determine what this centroid difference signifies. Yang et al. suggest establishing the boundaries of the concept in the compared dataset using a threshold, which they recommend setting based on the MAD value [2]. In our study, two different threshold values have been defined based on the Median Absolute Deviation (MAD) value.

1. The threshold T is defined based on the Median Absolute Deviation (MAD) of the recorded dataset:
$$T = \text{MAD}(D_{closest}) \tag{4}$$
2. The threshold T is defined as twice the Median Absolute Deviation (MAD) of the recorded dataset:
$$T = 2 \times \text{MAD}(D_{closest}) \tag{5}$$

Experiments have been conducted using methods that employ these two threshold values. If the centroid difference between the compared data is less than the threshold value, the datasets are considered to belong to the same concept.

Given a new dataset D_{new}, we first calculate the class distribution before processing:

$$\text{class_counts_before} = \{\text{class} : \text{count}\} \tag{6}$$

For each class $c \in D_{new}$, we find the closest dataset $D_{closest}$ from existing datasets D_i based on the centroid difference:

$$\text{Centroid Difference}(D_{i,c}, D_{new,c}) = \left\| \mu_{D_{i,c}} - \mu_{D_{new,c}} \right\| \tag{7}$$

where μ denotes the centroid of the class in the respective dataset.

If the dataset represents a different concept (i.e., the centroid difference is greater than the threshold), a new model is trained:

$$\Delta\mu_{D_{new}} > T \implies \text{Train a new model} \tag{8}$$

If the minimum centroid difference Centroid Difference$(D_{closest})$ is below a threshold T, we merge the new class data with the closest existing dataset:

$$\text{merged_dataset} = D_{closest} \cup D_{new,c} \tag{9}$$

After merging, we update the classifiers using the newly merged datasets:

$$\text{update_classifiers}(\text{merged_dataset}) \tag{10}$$

If there is any remaining data in D_{new} after merging:

- If the remaining data contains more than one class, add it as a new dataset.
- If it contains only one class, save this class with other classes that reach to the system together, but do not delete these classes from other datasets.

Ensure the function handles edge case such as:

- Datasets with only one class.

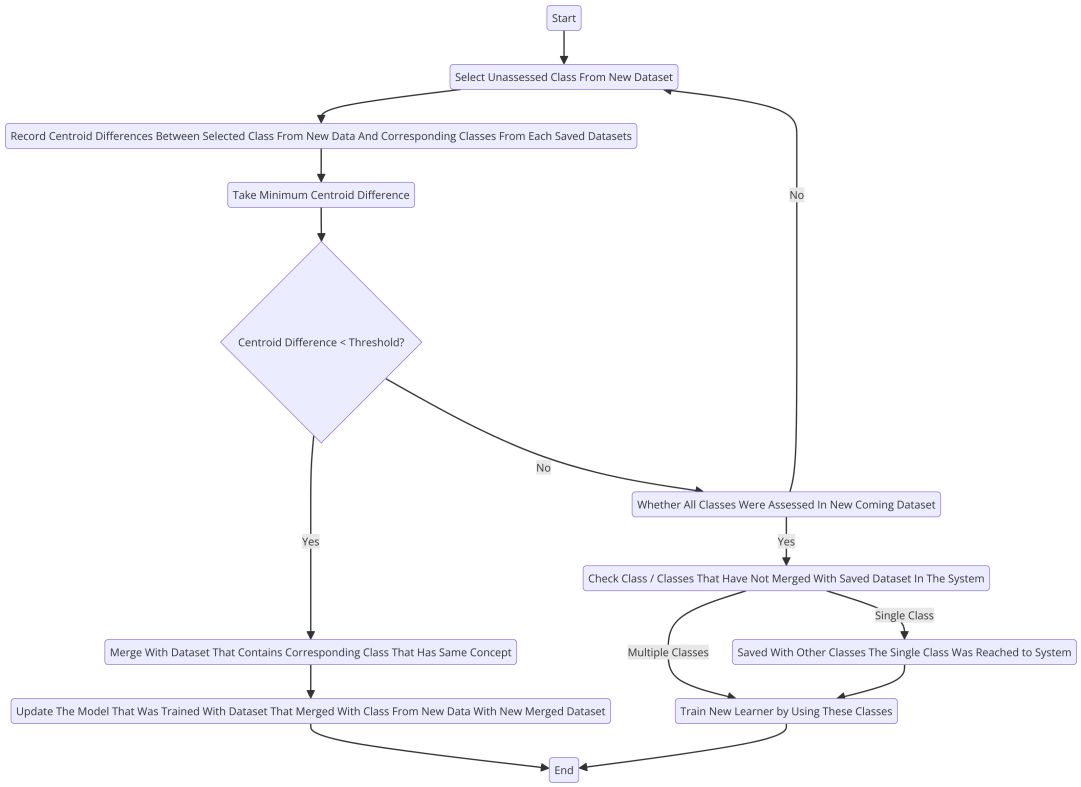


Figure 4: Training Algorithm Based on Comparison of Concept of Corresponding Classes

As illustrated in Figure 4, algorithm of strategy based on comparison of the classes in data is started with choosing a class that was not previously reviewed by system from new coming dataset. Then, system calculate centroid difference between chosen class from new coming dataset and corresponding classes from each saved datasets that were used for training a model. The system identifies the minimum calculated centroid value and the registered dataset containing the corresponding class for which this centroid value was calculated. The threshold value, reflecting the concept boundaries of the registered data with the minimum centroid difference, is calculated. In this strategy, the threshold value is determined using formulas dependent on the Median Absolute Deviation (MAD) metric. If minimum centroid difference value is below the threshold value, it is assumed that the data from the class is examining from the new coming dataset that belongs to the same concept as a corresponding class in the saved datasets. The dataset containing the class with the smallest centroid difference with data that is examined from new coming dataset, which is believed to enable the model to generalize well, is merged with the dataset that the model was trained on. Subsequently, the model is updated using the newly merged dataset. If the minimum calculated centroid difference exceeds the threshold value, it is assumed that the data under examination does not belong to the same concept as any registered data. In such cases, the system checks whether all classes in the new dataset have been examined. If not all classes have been reviewed, the process continues until all are assessed. After examining all classes, if the new dataset contains multiple classes that differ in concept from all registered datasets, the data from these classes are recorded as a separate dataset, and a new model is created by using this dataset. If there is only one class in the new dataset that differs in concept, a viable model cannot be trained using a dataset containing only one class. Therefore, it is recorded along with other classes in

the new dataset, and a new model is developed. These classes are not deleted from the other datasets they are merged with.

The training phase described above involves the system creating a separate model for each concept, with the expectation that these models will generalize their respective concepts effectively. If desired, the user can address the class imbalance in the registered datasets either during the training phase or after it concludes, as detailed in the Section 3.4. In addition, should the user acquire new data suitable for training, they may initiate another training phase.

After completion of the training process by using the strategy described above, the system becomes ready to make predictions for datasets containing different concepts. How predictions are made using the models trained by the system is explained in the Section 3.6.

3.3.2 Strategy Based On Performance of the Models

The strategy based on performance of the models use the accuracy metric for detecting concept drift. The accuracy metric is defined as the ratio of correctly predicted instances to the total number of instances. Mathematically, it can be represented as:

$$\text{Accuracy} = \frac{TP + TN}{TP + TN + FP + FN} \tag{11}$$

where:

- TP (True Positives) is the number of correctly predicted positive instances.
- TN (True Negatives) is the number of correctly predicted negative instances.
- FP (False Positives) is the number of incorrectly predicted positive instances.
- FN (False Negatives) is the number of incorrectly predicted negative instances.

In this strategy, whether the concept of the incoming dataset can be understood by any of the existing models is determined by whether their performance, based on the accuracy metric is described above, exceeds a predefined threshold value. The threshold value is also determined using the accuracy metric. It means that the threshold is defined based on the performance that would be achieved if all available data were used to train a single model. If the data arrives in different times, meaning not all data intended for the training phase is available from the start, then the performance of a hypothetical single model, which would be constructed using all data available up to that point, is assessed as the threshold when a new dataset arrives. On the other hand, if all data intended for training is available but is sent to the system as parts, the performance of a single model built using the entire dataset is used as the threshold.

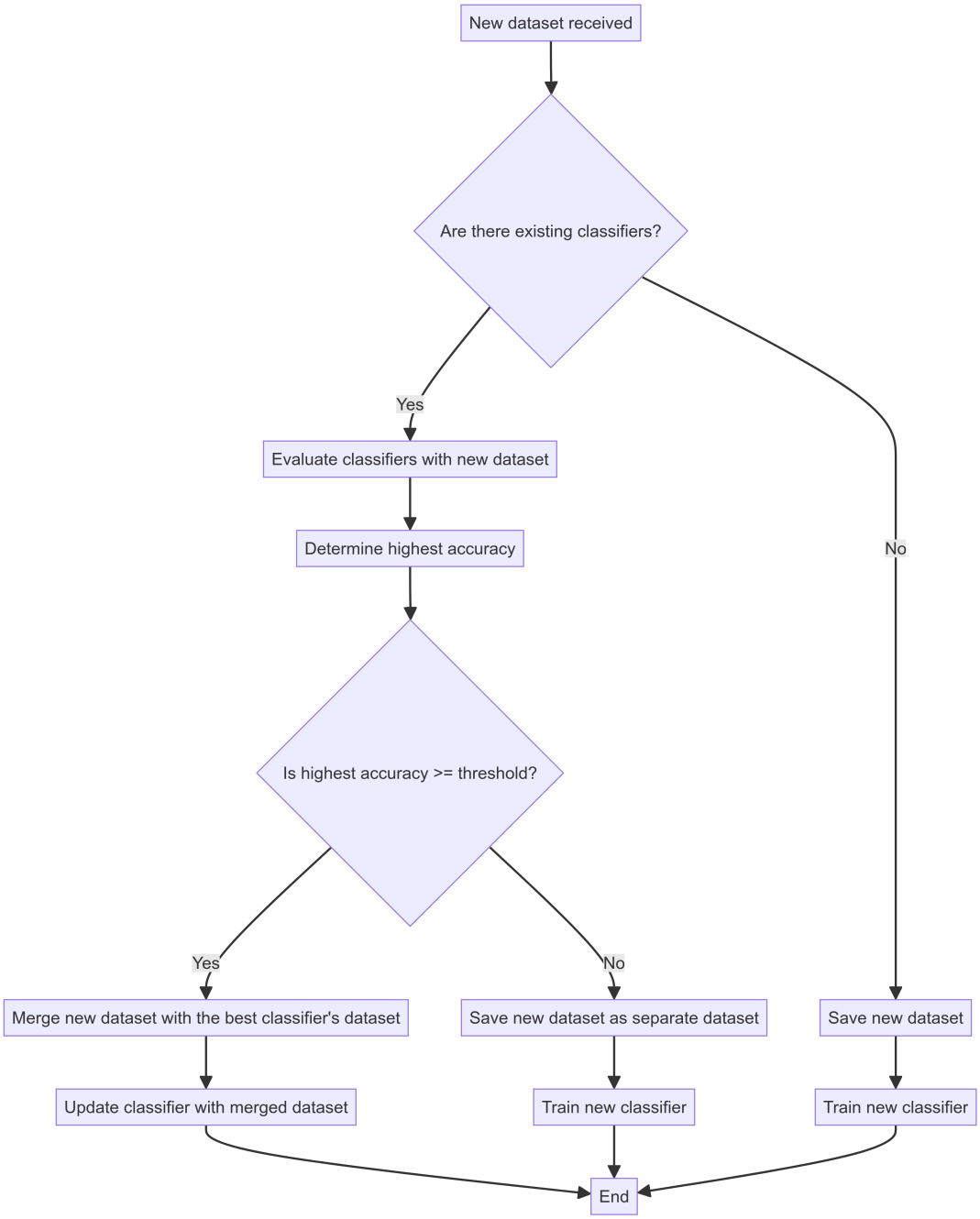


Figure 5: Algorithm for Implement Training Based on Performance of the Models

As illustrated in Figure 5, when a new dataset arrives, the performance of the existing trained models on this dataset is evaluated based on the accuracy metric. If the performance of the best-performing model exceeds the threshold value, the new dataset is merged with the dataset on which that model was trained, and the model is updated. If the performance is below the threshold value, the new dataset is recorded separately, and a model is created using this dataset.

When new data comes to the system, the following steps are followed:

1. **Initial Check and Classifier Training:**

If there are no existing classifiers, the new dataset is added to the collection of datasets, and a classifier is trained on this dataset:

$$\text{If classifiers} = \emptyset, \text{ then:}$$

$$\text{DatasetCollection} \leftarrow \text{DatasetCollection} \cup \{\text{new_dataset}\} \tag{12}$$

$$\text{classifiers}[\text{len(DatasetCollection)} - 1] \leftarrow \text{train_classifier(new_dataset)} \tag{13}$$

2. **Evaluation of Existing Classifiers:**

For each existing classifier, its accuracy on the new dataset is evaluated:

$$\text{For each classifier } C_i:$$

$$\text{accuracy}_i = \text{accuracy_score}(C_i(X_{\text{new_dataset}}), y_{\text{new_dataset}}) \tag{14}$$

3. **Selection of the Best Classifier:**

The classifier with the highest accuracy is identified:

$$\text{max_accuracy} = \max(\text{accuracies}) \tag{15}$$

$$\text{best_classifier_index} = \arg\max(\text{accuracies}) \tag{16}$$

4. **Integration Based on Accuracy Threshold:**

If the highest accuracy is greater than or equal to a predefined threshold, the new dataset is merged with the dataset corresponding to the best classifier, and the classifier is updated:

$$\text{If max_accuracy} \geq \text{threshold, then:}$$

$$\text{DatasetCollection[best_classifier_index]} \leftarrow \text{concat(batches[best_classifier_index], new_dataset)} \tag{17}$$

$$\text{update_classifier(best_classifier_index, DatasetCollection[best_classifier_index])} \tag{18}$$

If the highest accuracy is less than the threshold, the new dataset is saved as a separate dataset, and a new classifier is trained on it:

$$\text{DatasetCollection} \leftarrow \text{DatasetCollection} \cup \{\text{new_batch}\} \tag{19}$$

$$\text{classifiers}[\text{len(DatasetCollection)} - 1] \leftarrow \text{train_classifier(new_dataset)} \tag{20}$$

This process ensures effective integration of new dataset into the system by either enhancing existing classifiers or creating new ones based on an accuracy threshold.

After some models are created, the system becomes ready to work for making predictions. Also, the user of the system always can continue the training process with new data.

Upon the completion of the training process as explained above, the system is prepared to generate predictions for datasets. The methodology employed for making predictions using the models that are trained by the system is described in the Section 3.6.

3.4 Balance Datasets That Are Used For Training Models

Class imbalance occurs when there is a significant variation in the number of samples across different classes [31]. Both the class imbalance found in the datasets used for training and the strategy of examining the concept of each class individually have led to the issue of class imbalance in the datasets where models are trained. To overcome this problem, synthetic data can be generated to rectify the class imbalance in the datasets [32][33][34].

The system is capable of automatically generating synthetic data to address class imbalance. On the other side, the stage at which synthetic data is incorporated into the training process is left to the discretion of the user. This provides flexibility to the user, allowing adjustments based on the characteristics of the data used and the subjective conditions of the training being conducted.

When the user wishes to eliminate class imbalances in training datasets, the system generates synthetic data based on the existing data using the Synthetic Minority Over-Sampling Technique (SMOTE), in response to the command issued. This approach effectively addresses class imbalances.

SMOTE is a statistical technique designed to increase the number of items in the dataset in a balanced way [35]. To generate data in balanced manner by SMOTE, system identifies the minority class that needs oversampling to reach a balance between the classes. For each sample in the minority class, system finds its k-nearest neighbors. Then, it selects one of these neighbors and generates a synthetic instance somewhere on the line joining the instance under consideration and its selected neighbor. The values of the synthetic data are calculated as a convex combination of the values of the original instance and its chosen neighbor. This means that for each feature in the data, the synthetic instance's feature value is generated by interpolating between the values of that feature for the two instances. The system repeats this process until the class distribution is balanced.

The synthetic sample generation process in SMOTE can be mathematically represented by the following formula. Suppose x_i is a sample from the minority class, and x_{nn} is one of its nearest neighbors, also from the minority class. A synthetic sample x_{new} is generated as follows:

$$x_{new} = x_i + \lambda \times (x_{nn} - x_i) \tag{21}$$

where:

- x_i is the feature vector of the original sample.
- x_{nn} is the feature vector of the nearest neighbor.
- λ is a random number between 0 and 1.

This formula ensures that the synthetic sample x_{new} lies on the line segment between x_i and x_{nn}, effectively creating a sample that is a combination of the two, with the specific combination determined

randomly by λ. This approach helps in creating diversity within the minority class samples in the dataset, enhancing the training process for models that might otherwise be biased towards the majority class.

The primary advantage of SMOTE is that by creating synthetic samples rather than simply duplicating existing samples, it helps the model generalize better, thus avoiding over-fitting which can occur with simple oversampling [36]. However, it can also lead to increased noise, as synthetic instances might combine attributes of the minority class in ways that are not representative.

3.5 Type of Learner Models

In this study, the logistic regression model, a classification-based machine learning model that does not require complex hyper-parameter tuning, was chosen for the experimental phase. This choice was made because the real-life dataset that we used for experiment were suitable for classification and we want to minimize other variables like hyper-parameters that could affect the validity of the method being tested.

The logistic regression model can be represented as follows:

$$P(y = 1|X) = \frac{1}{1 + e^{-(\beta_0 + \beta_1 X_1 + \beta_2 X_2 + \cdots + \beta_p X_p)}} \tag{22}$$

where:

- $P(y = 1|X)$ is the probability that the dependent variable y equals 1 given the predictors X.
- β_0 is the intercept term.
- $\beta_1, \beta_2, \ldots, \beta_p$ are the coefficients of the predictor variables $X_1, X_2, \ldots, X_p$.

The odds of the dependent variable equaling 1 are given by:

$$\text{Odds} = \frac{P(y = 1|X)}{1 - P(y = 1|X)} \tag{23}$$

The log-odds (logit) is the natural logarithm of the odds:

$$\text{Logit}(P(y = 1|X)) = \log\left(\frac{P(y = 1|X)}{1 - P(y = 1|X)}\right) = \beta_0 + \beta_1 X_1 + \beta_2 X_2 + \cdots + \beta_p X_p \tag{24}$$

The cost function for logistic regression, also known as the log-loss function, is given by:

$$J(\beta) = -\frac{1}{m} \sum_{i=1}^{m} \left[y^{(i)} \log(h_\beta(x^{(i)})) + (1 - y^{(i)}) \log(1 - h_\beta(x^{(i)})) \right] \tag{25}$$

where:

- m is the number of training examples.
- $y^{(i)}$ is the actual label of the i-th training example.
- $h_\beta(x^{(i)})$ is the predicted probability for the i-th training example.

The best set of logistic regression model's parameters that minimize the loss function can be found by using the cost function for logistic regression, as mentioned above. As logistic regression does not use many hyper-parameters, there are not parameters that cannot be defined objectively optimal. Thanks to this nature of the logistic regression model, our research can focus on the success of the proposed method instead of concerning the choice of the hyper-parameters.

Depending on the characteristics of the data used and the intended purpose, the system can be operated using different models different from the logistic regression. Logistic regression was chosen for this study because of the reasons mentioned above.

3.6 Strategy for Prediction

Different strategies have been developed for making predictions using the models trained by the system. The aim is to ensure that the developed models can be accurately used for generalizing datasets of varying concepts through appropriate strategy.

For this purpose, a strategy has been developed where each model is assigned a weight proportional to how closely the concept of the dataset it will predict aligns with the dataset it was trained on, and voting rights are granted to models based on these weights. Additionally, strategies such as granting a single vote to models trained with datasets for which there is no evidence of being in a different concept from the dataset to be predicted have also been explored.

One of the key aspects of the developed strategies is determining how to calculate a metric that indicates the similarity of the concepts of the datasets under examination. For this purpose, various mathematical formulas have been employed. These formulas will be explained in the relevant subsections.

3.6.1 Assign Weights on Models Based on Centroid Difference Metric

In this approach, each model is granted voting rights based on the centroid difference between the dataset it was trained on and the dataset it will predict. A larger centroid difference, indicating a significant concept difference, means that the voting power for each model is inversely proportional to the product of the centroid differences. Additionally, to avoid issues when the centroid value is zero, one is added to the denominator in the formula to calculate weight that is assigned to a model.

The formula for calculating weight of a model is seen below:

$$weight = \frac{1}{1+d} \tag{26}$$

where d denotes the centroid differences between dataset which is used for training the model and dataset that the model predict.

Each model makes its predictions, and the models cast votes according to their voting rights. The votes from the models are then counted, and the prediction receiving the most votes is accepted as the system's final prediction. For example, consider a scenario where there are three models registered in the system, with respective voting rights of 3, 1, and 1. If the first model predicts the label 'a' and the other two models predict the label 'b', the system's final prediction would be the label 'a', despite more models voting for 'b'. This outcome is due to the total number of voting rights, which favor the label predicted by the first model.

Using centroid difference as a metric to measure concept variation is an adaptation of the 'comparison of datasets' training strategy to the prediction strategy 3.3.1. In this training strategy, whether the classes in the newly arrived dataset and the corresponding classes in the datasets used to train the models share the same concept is determined by whether the centroid difference exceeds a threshold. In the prediction strategy, centroid difference is used as an indicator of concept difference. In both approaches, the centroid difference is evaluated in relation to concept variation. When assigning weights, unlike during the training phase, centroid differences are calculated based on the entire dataset rather than on a class basis. This is because the classes of the dataset to be predicted are unknown [37].

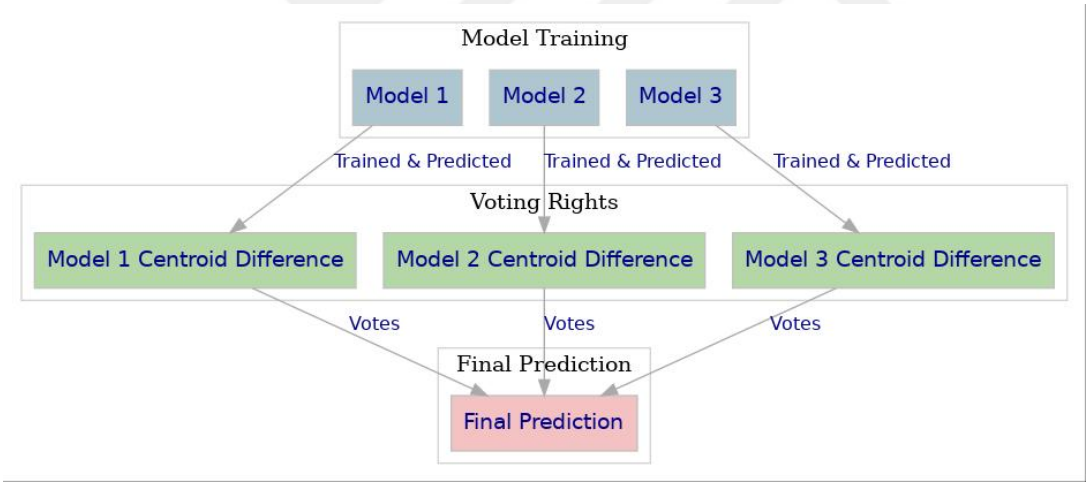


Figure 6: Voting Strategy Based On Centroid Differences

In Figure 6, a scenario with three models is given. In this scenario, to determine the voting rights of the three models, the centroid difference between the datasets used to train the models and the datasets they are intended to predict is first calculated. Based on this, specific voting rights are assigned to each model. Then, models cast their votes, and the votes are counted. According to weighed votes, the prediction that has most votes is defined as the ultimate prediction of the system.

Mathematical formulations for this process is defined below:

1. **Prediction and Centroid Difference Calculation**:

Use the classifier C_i to predict the labels $\hat{y}_{\text{test}}$ for the test data X_{test}:

$$\hat{y}_{\text{test}} = C_i(X_{\text{test}}) \tag{27}$$

Calculate the centroid difference between the dataset that was used for training the model X_i and the test data X_{test}:

$$d_i = \|\mu_{X_i} - \mu_{X_{\text{test}}}\| \tag{28}$$

2. **Weight Calculation**:

Calculate the weight w_i based on the centroid difference:

$$w_i = \frac{1}{1 + d_i} \tag{29}$$

3. **Weighted Voting**:

Compute the weighted votes for each class c and select the class with the highest weighted vote. Let y_{pred} be the predicted label for each sample j in the test data:

$$\text{weighted_votes}[y_{\text{pred}_j}] += \frac{w_i}{\sum_k w_k} \tag{30}$$

Select the class with the highest weighted vote:

$$y_{\text{final}} = \begin{cases} \text{random.choice}\left(\{c \in C \mid \text{votes}[c] = \max_c \text{votes}[c]\}\right) & \text{if there is a tie,} \\ \arg\max_c \text{votes}[c] & \text{otherwise.} \end{cases} \tag{31}$$

3.6.2 Assign Weights on Models Based on Similarity Score

The strategy which is assigning weights on the models based on the similarity score is quite similar to strategy of assigning weights on models based on the centroid difference metric 3.6.1. However, in this strategy, we use the similarity score instead of directly using the centroid difference. Similarity score is calculated by a formula which is based on centroid difference. However, similarity score uses the centroid difference metric in a smoother manner. This approach aims to reduce abrupt fluctuations during the prediction process.

The similarity score is formulated below:

$$\text{Similarity Score} = e^{-\lambda \times d} \tag{32}$$

where d denotes the centroid difference between examined datasets, and λ refers to a constant.

The formula is based on the "Centroid Distance" value between two objects or elements. This distance measures how far apart two objects are in space.

Lambda (λ) is used as a tuning parameter in the formula and regulates the effect of centroid distance on the similarity score. The higher the value of λ, the greater the impact of the distance increase on

the similarity score. That is, when λ is high, even a small distance can greatly reduce the score. The analyzer who use the system should determine the lambda value based on the desired impact of the centroid difference.

The formula calculates the score obtained by taking the exponent of the negative λ times the calculated distance and calculating the e number according to this exponent. e is a constant number in mathematics (approximately 2.71828). Exponentiation here ensures that the similarity score drops rapidly as distance increases, indicating that more distant objects are less similar to each other.

As illustrated in Figure 7, the algorithm initially creates a space to record the predictions and voting rights of the registered models. Then, it iterates through all models, ensuring that each model makes a prediction and calculating the centroid differences between the dataset each model was trained on and the dataset to be predicted. A voting right value for each model is computed using the similarity score formula. The predictions and voting rights of each model are recorded. This process is repeated until it has been completed for all models. Once there are no models left for which this process has not been performed, the system transitions to calculating the final results. Voting is conducted based on the voting rights of the models, and the outcome is accepted as the system's prediction.

The mathematical explanations of these processes are described below:

1. **Prediction and Centroid Difference Calculation**:

Use the classifier C_i to predict the labels $\hat{y}_{\text{test}}$ for the test data X_{test}:

$$\hat{y}_{\text{test}} = C_i(X_{\text{test}}) \tag{33}$$

Calculate the centroid difference between the dataset that was used for training the model X_i and the test data X_{test}:

$$d_i = \|\mu_{X_i} - \mu_{X_{\text{test}}}\| \tag{34}$$

2. **Weight Calculation**:

Calculate the weight w_i based on the centroid difference using the similarity score formula:

$$w_i = e^{-\lambda \times d_i} \tag{35}$$

where λ is a constant that determines the rate of decay for the similarity score.

3. **Weighted Voting**:

Compute the weighted votes for each class c and select the class with the highest weighted vote. Let y_{pred} be the predicted label for each sample j in the test data:

$$\text{weighted_votes}[y_{\text{pred}_j}] += \frac{w_i}{\sum_k w_k} \tag{36}$$

Select the class with the highest weighted vote:

$$y_{\text{final}} = \begin{cases} \text{random.choice}(\{c \in C \mid \text{votes}[c] = \max_c \text{votes}[c]\}) & \text{if there is a tie,} \\ \arg\max_c \text{votes}[c] & \text{otherwise.} \end{cases} \tag{37}$$

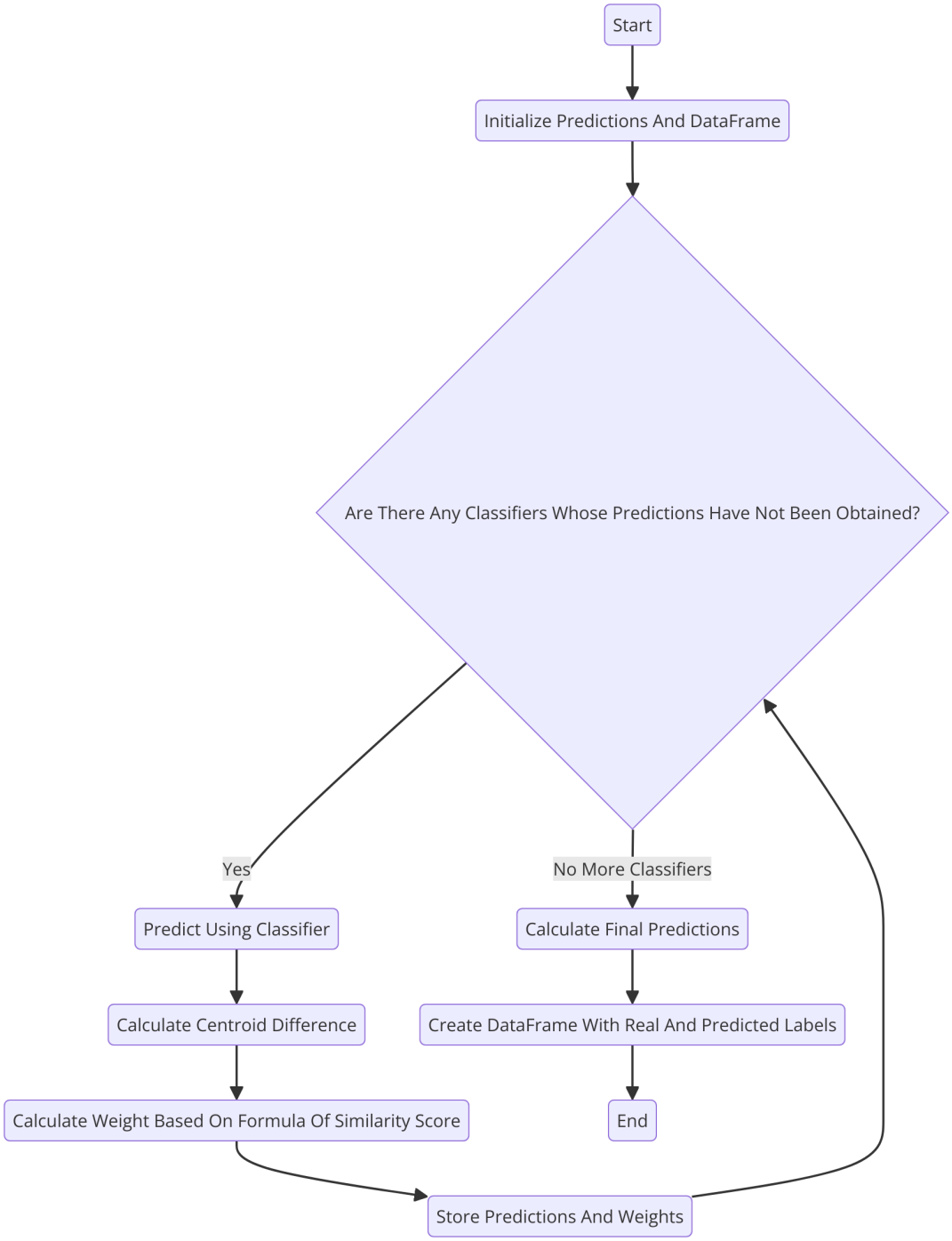


Figure 7: Strategy Based on Similarity Score

3.6.3 Assign Weights on Models Based on Centroid Difference and Diversity

This approach represents an updated version of the centroid difference approach, wherein models trained with more diverse datasets are rewarded. In this strategy, in addition to models trained on datasets that exhibit a smaller centroid difference from the predicted dataset are allocated more voting rights, models trained on datasets containing a wider variety of data and broader concept boundaries are also granted increased voting rights.

To achieve this objective, the threshold determination method used during the training phase has been adapted to be purpose-specific. It was assumed that for two datasets to be considered as sharing the same concept, the centroid difference between them needed to be less than a threshold value calculated based on the MAD value of the registered dataset on the subsection 3.6.1. This allowed for a wider centroid difference interval for datasets with larger MAD values, under the assumption that these datasets, which contain more diversity, have broader conceptual boundaries. This approach has been adapted for the prediction method, where models trained with datasets having larger MAD values are allocated more voting rights.

The formula in the subsection 3.6.1 has been modified to award diversity on datasets that was used for training the models as follows:

$$weight = \frac{m_d}{1+d} \tag{38}$$

where d refers centroid difference between dataset that the model was trained and dataset that is predicted, and m_d denotes MAD value of dataset that the model was trained,

and where the Median Absolute Deviation (MAD) m_d for a dataset X is calculated by:

$$\mathrm{m}_d = \mathrm{median}(|X_j - \mathrm{median}(X)|) \tag{39}$$

The centroid difference between the datasets is calculated by:

$$d = \|\mu_{X_1} - \mu_{X_2}\| \tag{40}$$

In the formula designed to determine the weight assigned to models, the Median Absolute Deviation (MAD) value is included in the numerator to reward models trained on datasets with greater diversity. Conversely, the denominator includes the centroid difference value between datasets, penalizing models with a high centroid difference from the dataset to be predicted and those not reset. Additionally, the reason for adding one to the centroid difference value in the denominator is to prevent system errors when the centroid difference is zero.

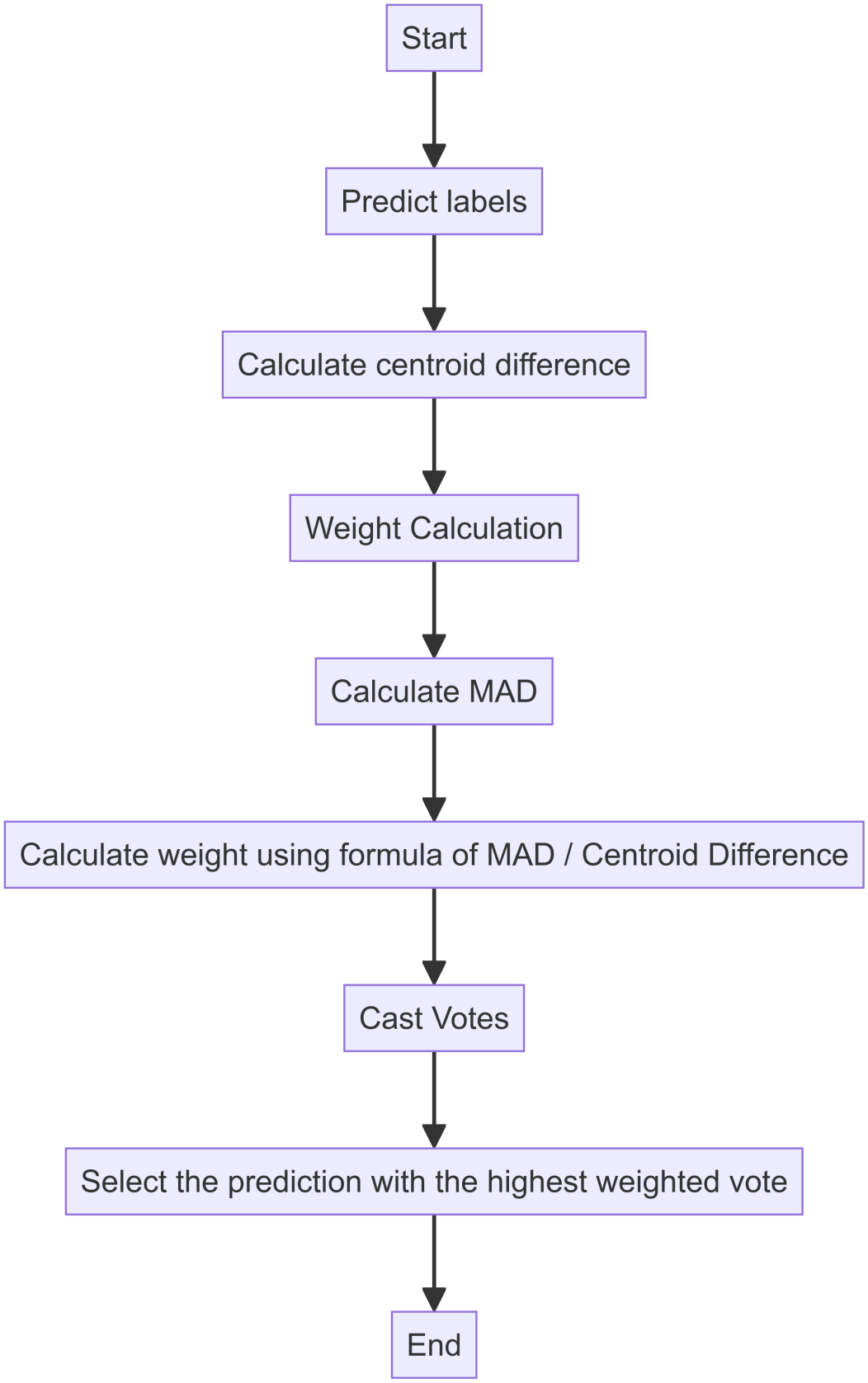


Figure 8: Simplified Representation of Algorithm for Assigning Weights on Models Based on Centroid Difference and Diversity

As shown in simplified representation of the process at Figure 8, the process starts by predicting labels for the test data using a trained classifier. Then, the centroid difference between the training dataset and the test dataset is calculated to measure how different the datasets are. Following this, the Median Absolute Deviation (MAD) of the training dataset is computed. Using the MAD and the centroid difference, weights are calculated for models. These weights are then used to cast votes for each model's predictions. Finally, the prediction with the highest weighted vote is selected as the final prediction, concluding the process.

The mathematical explanations of the process is given below:

1. **Prediction and Centroid Difference Calculation**:

Use the classifier C_i to predict the labels $\hat{y}_{\text{test}}$ for the test data X_{test}:

$$\hat{y}_{\text{test}} = C_i(X_{\text{test}}) \tag{41}$$

Calculate the centroid difference between the dataset that the model trained X_i and the test data X_{test}:

$$d_{i,\text{test}} = \|\mu_{X_i} - \mu_{X_{\text{test}}}\| \tag{42}$$

2. **Weight Calculation**:

Calculate the Median Absolute Deviation (MAD) for the dataset that the model trained X_i:

$$\text{MAD}_i = \text{median}(|X_{ij} - \text{median}(X_i)|) \tag{43}$$

Calculate the weight w_i based on the MAD of the dataset that was used for training the model and centroid difference between the dataset that was used for training the model and dataset that the model predicts:

$$w_i = \frac{\text{MAD}_i}{1 + d_{i,\text{test}}} \tag{44}$$

3. **Weighted Voting**:

Compute the weighted votes for each class c and select the class with the highest weighted vote. Let y_{pred} be the predicted label for each sample j in the test data:

$$\text{weighted_votes}[y_{\text{pred}_j}] += \frac{w_i}{\sum_k w_k} \tag{45}$$

Select the class with the highest weighted vote:

$$y_{\text{final}} = \begin{cases} \text{random.choice}(\{c \in C \mid \text{votes}[c] = \max_c \text{votes}[c]\}) & \text{if there is a tie,} \\ \arg\max_c \text{votes}[c] & \text{otherwise.} \end{cases} \tag{46}$$

3.6.4 Assign Weights on Models Based on The Jensen-Shannon Divergence Score

The method to address out-of-Distribution (OOD) problem that is described in Section 2.6 can be derived to use in measuring closeness of concept of the datasets. Hu et al. used Jensen-Shannon

Divergence score (JSD) to quantify the distribution differences between the original training set and the various test sets [9]. Thanks to Jensen-Shannon Divergence score, they validate that whether the selected data effectively represent the distribution shifts. The logic behind their method can be adapted to implement the voting strategy based on measuring differences between the training set and the test set.

In our approach, the Jensen-Shannon Divergence (JSD) method is utilized to determine how closely datasets are in related concepts. The goal of using the JSD method is to derive a metric that indicates the conceptual proximity of the datasets. Then, a weight based on how closely the dataset a model was trained on aligns with the dataset to be predicted is produced, and these weights are assigned to each model. This approach aims to allocate more voting rights to models that are better able to understand the concept of the dataset to be predicted, as was the case in previous prediction strategies. After specific voting rights are determined for each model, the models will cast votes, and the predictions receiving the most weighted votes will be determined as the system's final prediction.

The Jensen–Shannon divergence method is used for measuring the similarity between two probability distributions [38]. In this study, it is assumed that a metric measuring the proximity of probabilistic distributions can be used to gauge how closely related the concepts of two datasets might be, based on the assumption that such a metric provides information about the distributions of the datasets.

The Jensen-Shannon divergence metric $D_{JS}(P\|Q)$ that is measuring the similarity between two probability distributions P and Q is defined below:

1. **Average Distribution**:

Calculate the average distribution M:

$$M = \frac{1}{2}(P + Q) \tag{47}$$

2. **Kullback-Leibler Divergence**:

Calculate the Kullback-Leibler divergence for each distribution relative to M:

$$D_{KL}(P\|M) = \sum_i P(i) \log \frac{P(i)}{M(i)} \tag{48}$$

$$D_{KL}(Q\|M) = \sum_i Q(i) \log \frac{Q(i)}{M(i)} \tag{49}$$

3. **Jensen-Shannon Divergence**:

The Jensen-Shannon divergence is the average of these Kullback-Leibler divergences:

$$D_{JS}(P\|Q) = \frac{1}{2} D_{KL}(P\|M) + \frac{1}{2} D_{KL}(Q\|M) \tag{50}$$

The result is a number that ranges from 0 to 1, where:

- 0 indicates that the two distributions are identical.

- 1 suggests that the distributions have no overlap.

The metric provided by the Jensen-Shannon Divergence (JSD) method indicates closer conceptual similarity as it approaches zero, and suggests a more different concept as it approaches one. Therefore, the weights assigned to the models are determined by subtracting the JSD result from one:

$$Weight = 1 - D_{JS}(P' \| Q') \tag{51}$$

As illustrated in Figure 9, each model makes predictions for the test set. Subsequently, the JSD score, which indicates the proximity between the dataset used for training the model and the dataset to be predicted, is calculated. The weight assigned to each model is determined by the value of 1 minus the JSD score of the dataset on which the model was trained. After each model casts votes proportionally to their weights, the prediction receiving the highest total weighted votes is presented as the output of the entire system.

The mathematical description of the process is defined below:

1. **Extract Features and Labels**:

For each classifier C_i and its corresponding dataset X_i, label vector y_i:

$$X_i, y_i = \text{extract_data_labels}(B_i) \tag{52}$$

2. **Models Predict**:

Use the classifier C_i to predict the labels $\hat{y}_{\text{test}}$ for the test data X_{test}:

$$\hat{y}_{\text{test}} = C_i(X_{\text{test}}) \tag{53}$$

3. **Weight Calculation Using Jensen-Shannon Divergence**:

Calculate the Jensen-Shannon Divergence (JSD) between the dataset that the model trained X_i and the test data X_{test}:

First, calculate the histograms P and Q for X_i and X_{test}, respectively:

$$P = \text{hist}(X_i) \tag{54}$$

$$Q = \text{hist}(X_{\text{test}}) \tag{55}$$

Normalize the histograms:

$$P' = \frac{P}{\sum P} \tag{56}$$

$$Q' = \frac{Q}{\sum Q} \tag{57}$$

Calculate the average distribution M:

$$M = \frac{1}{2}(P' + Q') \tag{58}$$

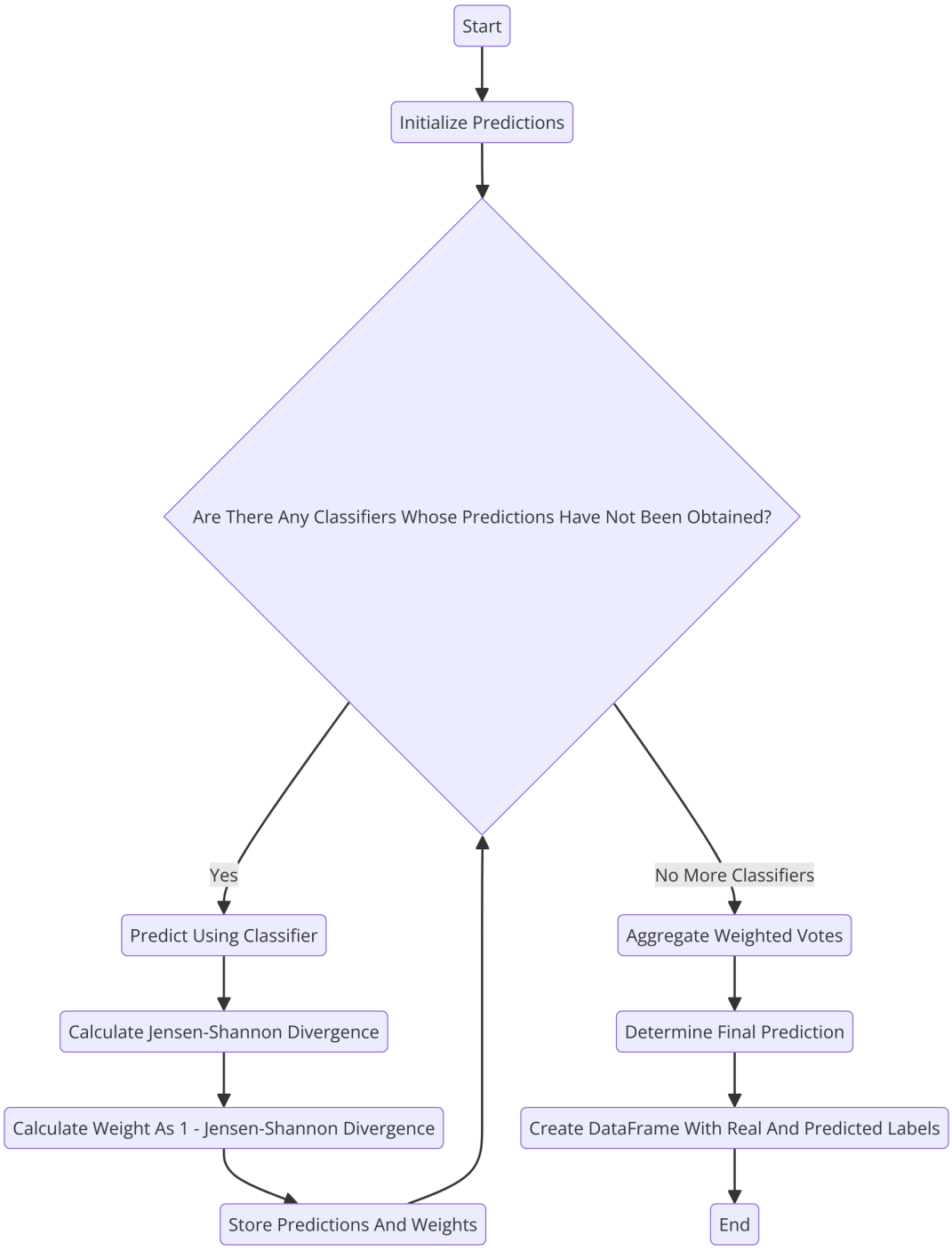


Figure 9: Prediction Strategy Which Assigns Weights on Models Based on The jensen-Shannon Divergence Score

Calculate the Kullback-Leibler divergence for each distribution relative to M:

$$D_{KL}(P' \| M) = \sum_i P'(i) \log \frac{P'(i)}{M(i)} \tag{59}$$

$$D_{KL}(Q' \| M) = \sum_i Q'(i) \log \frac{Q'(i)}{M(i)} \tag{60}$$

The Jensen-Shannon divergence is:

$$D_{JS}(P' \| Q') = \frac{1}{2} D_{KL}(P' \| M) + \frac{1}{2} D_{KL}(Q' \| M) \tag{61}$$

Calculate the weight w_i based on the JSD:

$$w_i = 1 - D_{JS}(P' \| Q') \tag{62}$$

4. **Weighted Voting**:

Compute the weighted votes for each class c and select the class with the highest weighted vote. Let y_{pred} be the predicted label for each sample j in the test data:

$$\text{weighted_votes}[y_{\text{pred}_j}] += \frac{w_i}{\sum_k w_k} \tag{63}$$

Select the class with the highest weighted vote:

$$y_{\text{final}} = \begin{cases} \text{random.choice}(\{c \in C \mid \text{votes}[c] = \max_c \text{votes}[c]\}) & \text{if there is a tie,} \\ \arg\max_c \text{votes}[c] & \text{otherwise.} \end{cases} \tag{64}$$

3.6.5 Assign Weights on Models Based on Statistical Test

This approach is based on measuring the proximity of datasets using statistical tests. For this purpose, the Mann-Whitney U test is employed as the statistical test.

The Mann-Whitney U test is a non-parametric test that is used to determine whether there is a significant difference between the distributions of two independent groups [39]. The steps and formulas of Mann-Whitney U test are defined as follows:

Let:

- n_1: sample size of group 1
- n_2: sample size of group 2
- R_1: sum of ranks for group 1
- R_2: sum of ranks for group 2

The U statistic is calculated for each group as follows:

$$U_1 = n_1 n_2 + \frac{n_1(n_1 + 1)}{2} - R_1 \tag{65}$$

$$U_2 = n_1 n_2 + \frac{n_2(n_2+1)}{2} - R_2 \tag{66}$$

The Mann-Whitney U statistic is the minimum of U_1 and U_2:

$$U = \min(U_1, U_2) \tag{67}$$

Under the null hypothesis, the expected value ($E[U]$) and standard deviation (σ_U) of the U statistic are given by:

$$E[U] = \frac{n_1 n_2}{2} \tag{68}$$

$$\sigma_U = \sqrt{\frac{n_1 n_2 (n_1 + n_2 + 1)}{12}} \tag{69}$$

For large sample sizes, the U statistic can be approximated by a normal distribution. The Z-score is calculated as:

$$Z = \frac{U - E[U]}{\sigma_U} \tag{70}$$

The p-value is determined from the Z-score using the standard normal distribution:

- For a two-tailed test, the p-value is:

$$\text{p-value} = 2 \times (1 - \Phi(|Z|)) \tag{71}$$

  where Φ is the cumulative distribution function (CDF) of the standard normal distribution.

- For a one-tailed test, the p-value is:

$$\text{p-value} = 1 - \Phi(Z) \quad \text{(if the test is upper-tailed)} \tag{72}$$

$$\text{p-value} = \Phi(Z) \quad \text{(if the test is lower-tailed)} \tag{73}$$

The hypotheses for the Mann-Whitney U test are:

- H_0: The distributions of both groups are equal.
- H_1: The distributions of both groups are not equal.

The p-value represents the probability of obtaining test results at least as extreme as the results actually observed. In the context of the our case, it tells us the probability that the observed difference in ranks between the two groups could have occurred by random chance if the null hypothesis were true. If the p-value is less than a threshold determined by the chosen confidence interval range, we have sufficient evidence to reject the null hypothesis, which states that the distributions of two datasets are equal. On the other side, if the p-value exceeds the threshold defined by the chosen confidence interval, we do not have enough evidence to reject the null hypothesis that the distributions of two datasets are equal, and thus we accept that the two datasets are identical. According to this p-value approach, a decrease in the p-value reduces the likelihood that the examined datasets share the same distribution.

Similarly, an increase in the p-value enhances the probability that the examined datasets are from the same distribution. In this context, the p-value can be used as a metric to indicate whether the examined datasets are within the same concept.

Based on the logic described above, the p-value of the Mann-Whitney U Test on the dataset used to train the models and the dataset that is predicted can be used to determine the voting rights allocated to the models. In this study, voting rights are allocated to the models inversely proportional to the p-value. Thus, models trained with datasets that share a similar concept to the dataset to be predicted are granted more voting rights.

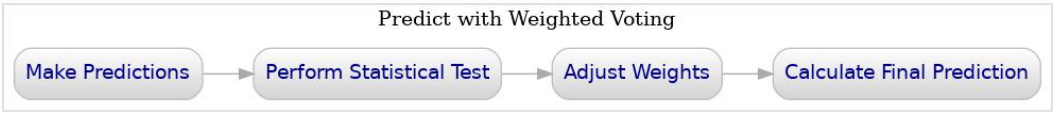


Figure 10: Simplified Representation of Algorithm for Assign Weights on Models Based on Statistical Test

As illustrated in Figure 10, each trained model makes a prediction for the dataset that is predicted by system. Subsequently, a statistical test is applied between the datasets used for training the models and the dataset to be predicted. Based on the results of these tests, weights that indicate each model's voting rights is assigned, and the models cast their votes according to their weights. The prediction receiving the most weighed votes is presented as the system's ultimate prediction.

The process of using the Mann-Whitney U test to compute weights for models and perform weighted voting is detailed below:

1. **Mann-Whitney U Test**:

Perform the Mann-Whitney U test between the dataset that was used for training a model X_i and the test data X_{test} to compute the p-values:

$$U_{\text{stat}}, p_{\text{value}} = \text{mannwhitneyu}(X_i, X_{\text{test}}) \tag{74}$$

2. **Weight Calculation**:

Determine the weight w_i of model_i based on the reciprocal of the p-values:

$$w_i = \frac{1}{p_{\text{i,test}}} \tag{75}$$

3. **Weighted Voting**:

Compute the weighted votes for each class and select the class with the highest weighted vote. Let y_{pred} be the predicted label for each sample j in the test data:

$$\text{weighted_votes}[y_{\text{pred}_j}] += \frac{w_i}{\sum_k w_k} \tag{76}$$

Select the class with the highest weighted vote as ultimate prediction of the system:

$$y_{\text{final}} = \begin{cases} \text{random.choice}(\{c \in C \mid \text{votes}[c] = \max_c \text{votes}[c]\}) & \text{if there is a tie,} \\ \arg\max_c \text{votes}[c] & \text{otherwise.} \end{cases} \tag{77}$$

3.6.6 Give One Vote Right Each to Models Trained on Datasets That Are Not In Different Concept

This approach is a modification of the previous approach which is described in Section 3.6.5, adapted to align with the nature of statistical tests.

As a statistical test, the Mann-Whitney U test is employed in accordance with its nature by performing hypothesis testing after calculating the p-value. Models trained with datasets that are determined to have the same distribution as the dataset to be predicted are identified through this test. This approach aligns more with the traditional use of statistical tests in the literature than simply using the p-value as a metric of proximity.

The formulas for calculating the z-score of the Mann-Whitney U test are defined in Section 3.6.5. The formulas for processes after calculating z-score are defined below:

Based on confidence interval and z-score, reject or accept hypothesis zero.

The hypotheses for the Mann-Whitney U test are:

- H_0: The distributions of both groups are equal.
- H_1: The distributions of both groups are not equal.

Following the implementation of the formulas mentioned above, it is determined whether the dataset on which the model was trained and the dataset it will predict share the same distribution. Accordingly, if the training dataset and the prediction dataset are in the same distribution, the model is assumed as capable of understanding the concept of the data to be predicted, and voting rights are granted to the model. Otherwise, no voting rights are assigned to the model.

After models eligible for voting rights have been determined, these models cast their votes. Unlike previous approaches, all models eligible to vote have an equal single vote right; hence, weighted voting is not conducted. The prediction receiving the majority of the votes is accepted as the final outcome of the system The process can be described mathematically as follows:

1. **Mann-Whitney U Test**:

Implement the Mann-Whitney U test on the feature values of X_i and the test data X_{test} to compute the p-values:

$$u_{\text{stat}}, p_{\text{value}} = \text{mannwhitneyu}(X_i, X_{\text{test}}) \tag{78}$$

2. **Determine Models That Have Voting Right**:

Determine the weight w based on the p-values. If any p-value p is less than the significance level α, set the weight to 0; otherwise, set it to 1:

$$w_i = \begin{cases} 0 & \text{if } p < \alpha \\ 1 & \text{otherwise} \end{cases} \tag{79}$$

3. **Voting**:

Only models that have a weight of 1, have voting right. These models cast a vote. Compute the votes for each class and select the class with the highest vote count. Let y_{pred} be the predicted label for each sample j in the test data:

$$\text{if } w_i = 1 \text{ then votes}[y_{\text{pred}_j}] += 1 \tag{80}$$

Select the prediction with the highest vote count as the final prediction of the system. If two or more predictions receive the maximum number of votes, randomly select one of them:

$$y_{\text{final}} = \begin{cases} \text{random.choice}(\{c \in C \mid \text{votes}[c] = \max_c \text{votes}[c]\}) & \text{if there is a tie,} \\ \arg\max_c \text{votes}[c] & \text{otherwise.} \end{cases} \tag{81}$$

CHAPTER 4

EXPERIMENTS AND RESULTS

In this chapter, the experiments are conducted and their results are shared. The experiments are divided into two sections: those conducted with synthetic data and those conducted with real-world datasets. Within these sections, the subsections further divide the experiments according to the concept drift scenarios and specific characteristics of the concept drift scenarios.

The experiments conducted with synthetic data aim to mimic typical concept drift scenarios that are commonly discussed in the literature and to observe the performance of our adaptive system when these scenarios occur. On the other hand, the experiments conducted with real-world datasets do not aim to follow a specific scenario. Instead, experiment that use real-word datasets aim to test the adaptive system under conditions that suit the complex and intricate structure of real-life data, which scenarios may not fully capture.

Accuracy, precision, and recall metrics are used to evaluate the experiment results. The micro-averaged versions of precision and recall metrics have been preferred in experiments with synthetic data since the focus is not on a specific positive class. In our synthetic data experiment design where experiments are designed without a specific focus on a positive class, using micro-averaged metrics ensures that all classes are treated equally. Also, by focusing on the sum total of true positives, false positives, and false negatives across all classes, micro-averaged precision and recall provide a comprehensive view of the model's error dynamics. Notably, Farhadpour et al. highlight that micro-averaged precision, micro-averaged recall, and overall accuracy always have equivalent mathematical results even though their definitions differ [40]. While this equivalence implies redundancy when used for detailed analysis, the application of micro-averaged metrics is still valuable in scenarios where a holistic assessment of performance across all classes is required. The mathematical definitions of the metrics used in expressing the results of the experiments are defined below:

Accuracy:

$$\text{Accuracy} = \frac{TP + TN}{TP + TN + FP + FN} \tag{1}$$

Accuracy metric measures the overall correctness of the model, defined as the ratio of correct predictions (both true positives and true negatives) to the total number of cases examined. It gives an idea of which percentage of the model is correct across all classes.

Precision (Micro):

$$\text{Precision (micro)} = \frac{\sum_{i=1}^{k} TP_i}{\sum_{i=1}^{k} (TP_i + FP_i)} \tag{2}$$

Precision evaluates the correctness of positive predictions. Micro-average precision calculates the total number of true positives across all classes and divides this by the total number of elements labeled as belonging to the positive class (both true positives and false positives). This metric reflects the proportion of actual positives among all instances predicted as positive.

Recall (Micro):

$$\text{Recall (micro)} = \frac{\sum_{i=1}^{k} TP_i}{\sum_{i=1}^{k} (TP_i + FN_i)} \tag{3}$$

Recall evaluates the model's capability to identify all relevant instances within a dataset. Micro-average recall aggregates the contributions of all classes to compute the average recall. It calculates the total true positives across all classes and divides this by the sum of true positives and false negatives across all classes. Recall (micro) gives an idea about the model's capacity to capture all positive instances without missing them.

In experiments conducted with a real-world dataset, the weighted versions of recall and precision metrics were preferred instead of their micro-averaged counterparts to evaluate the system's performance. The reason for this choice is the presence of class imbalances in real-world datasets, unlike synthetic data. Micro scores do not differentiate between classes and only measure overall performance. In contrast, weighted averaging provides a fairer evaluation at the class level by considering each class's recall and precision scores proportionally to their contribution. Therefore, using weighted recall and precision is a better option when evaluating the real-world data set where class imbalances are likely to exist.

$$\text{Weighted Precision} = \frac{\sum_{i=1}^{N} w_i \cdot \text{Precision}_i}{\sum_{i=1}^{N} w_i} \tag{4}$$

$$w_i = \frac{\text{Number of samples in class } i}{\text{Total number of samples}} \tag{5}$$

$$\text{Precision}_i = \frac{TP_i}{TP_i + FP_i} \tag{6}$$

Weighted precision is a variation of the precision metric in classification tasks that accounts for the size of each class in a dataset. Unlike traditional precision, which calculates the ratio of true positive results to the sum of true positives and false positives, weighed precision assigns a weight to each class or outcome, reflecting its relative significance in the specific context.

$$\text{Weighted Recall} = \frac{\sum_{i=1}^{N} w_i \cdot \text{Recall}_i}{\sum_{i=1}^{N} w_i} \tag{7}$$

$$w_i = \frac{\text{Number of samples in class } i}{\text{Total number of samples}} \tag{8}$$

$$\text{Recall}_i = \frac{TP_i}{TP_i + FN_i} \tag{9}$$

Weighted recall is a variation of recall that accounts for prevalence of different classes in a multi-class classification problem. It is calculated as the sum of the recall values for each class, weighted by the proportion of instances of each class in the dataset. The weighted recall metric is especially useful

when dealing with imbalanced datasets because it ensures that the overall recall score proportionally represents the performance across classes, mitigating the bias introduced by class imbalance.

4.1 Experiments with Synthetic Data

In this section, the experiments are conducted with synthetic data and their results are introduced. The synthetic data has been generated to simulate concept drift scenarios. The characteristics of the synthetic data that are produced are determined according to the experiment scenario, and the features of the synthetic data used for the experiments will also be explained in the sections discussing the relevant experiments.

While generating synthetic data, a fixed size was not specified to demonstrate that our system can operate with datasets of varying sizes. In each experiment, the synthetic data was generated to include a random number of instances.

In Section 3.1.1, the main concept drift scenarios commonly discussed in the literature were explained. In experiments conducted with synthetic data, prevalent concept drift scenarios from the literature were simulated, and the system's performance was observed under these conditions. The performance of our system is reported in related subsections.

4.1.1 Sudden Drift Scenario

In this subsection, experiments simulate the sudden drift scenario and their results are shared.

"Sudden drift," also known as "abrupt drift," refers to a quick and significant change in the underlying distribution of a data stream or dataset that affects the performance of a predictive model.

To fight with sudden drift, the training methods discussed in Section 3.3.1 and Section 3.3.2 were tried, and the prediction approach described in Section 3.6.1 was used. The training methods utilized in the experiments will be detailed along with the characteristics of each experiment.

In sudden drift experiments, instead of generating a random number of instances for each batch, a random number of data was created and then evenly divided into batches, each containing an equal number of instances. The number of batches was also determined randomly.

The sudden drift scenario was tested with different nuances in different experiments. In the first experiment, an increasing concept change occurred across all classes, but the extent of the increase varied among different classes. In the second experiment, while some classes showed increasing characteristics, others exhibited decreasing traits. In the final experiment, variables of some classes increased, variables of some classes decreased, and additionally, some classes began to show the conceptual characteristics of each other.

4.1.1.1 Sudden Drift Experiment 1

In Experiment 1, a sudden change occurred across all classes, simulating a sudden drift scenario where this abrupt change manifested as an increase. Figure11, Figure12, Figure13, and Figure14 illustrate the concept drift event in the dataset through boxplots, showing the distribution of the data across different batches. Figure11 presents the overall data distribution, while Figure12, Figure13, and Figure14 focus on the distributions within Class A, Class B, and Class C, respectively, highlighting how the patterns and general distribution evolve before, and after the drift.

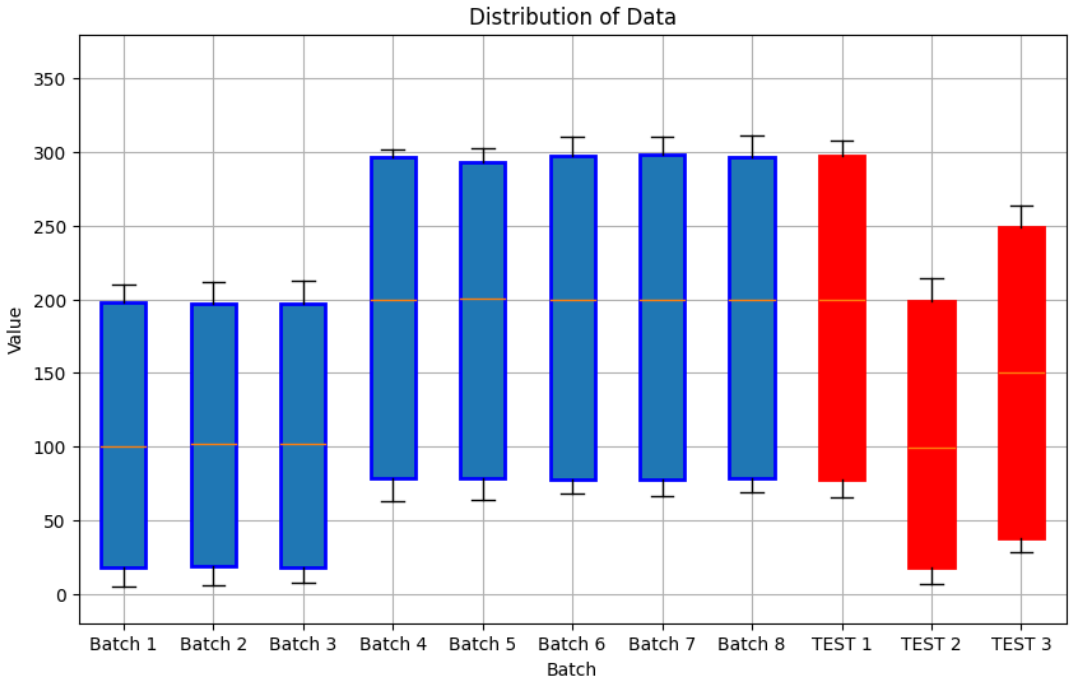


Figure 11: The Distribution of Data in Sudden Drift Experiment 1 Where The Sudden Drift Event Occurs In An Increasing Manner Across All Three Classes

The Table1 numerically presents characteristics such as the mean and standard deviation of the classes contained in the batches of the dataset.

The data generated as 8 batches for training process and 3 batches for testing, each containing samples distributed across three distinct classes: A, B, and C. Each sample within a dataset includes three variables.

For Class A, features in the first dataset are generated with a mean (mu) of 15 and a standard deviation (sd) of 3. In the second dataset, the mean increases to 16 with a standard deviation of 4. The third dataset reverts to a mean of 15 and a standard deviation of 3. From the fourth to the eighth dataset, the mean significantly increases to around 75 (with a slight variation to 76 in the seventh dataset) and the standard deviation ranges from 3 to 5. For Class B, features in the first dataset have a mean of 100 and a standard deviation of 4. The mean slightly increases to 102 in the second dataset, maintaining the same standard deviation. In third dataset mean and standard deviation remain same. In the subsequent datasets (fourth to eighth), the mean stabilizes around 200 (199 in the seventh dataset), with

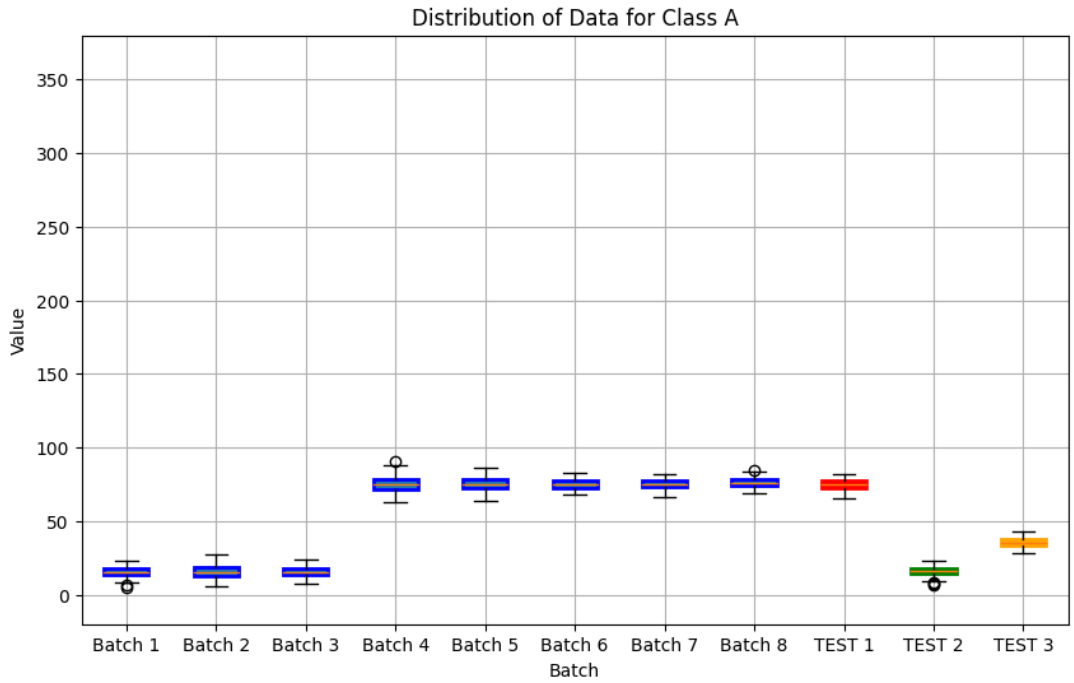


Figure 12: The Distribution of Class A Data in Sudden Drift Experiment 1 Where The Sudden Drift Event Occurs In An Increasing Manner Across All Three Classes

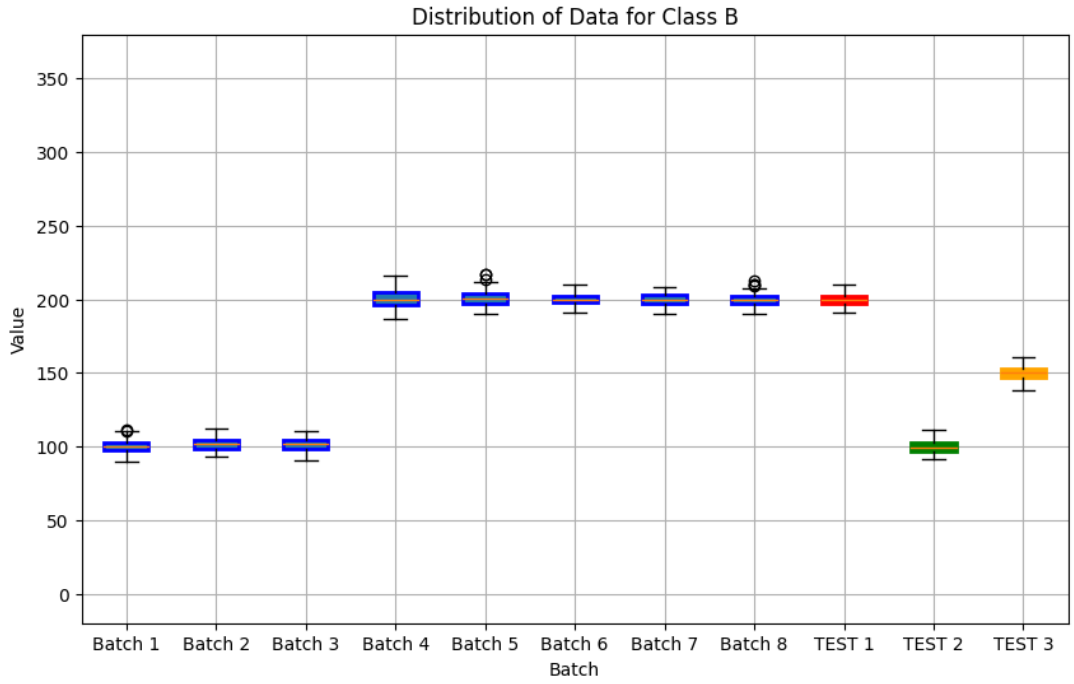


Figure 13: The Distribution of Class B Data in Sudden Drift Experiment 1 Where The Sudden Drift Event Occurs In An Increasing Manner Across All Three Classes

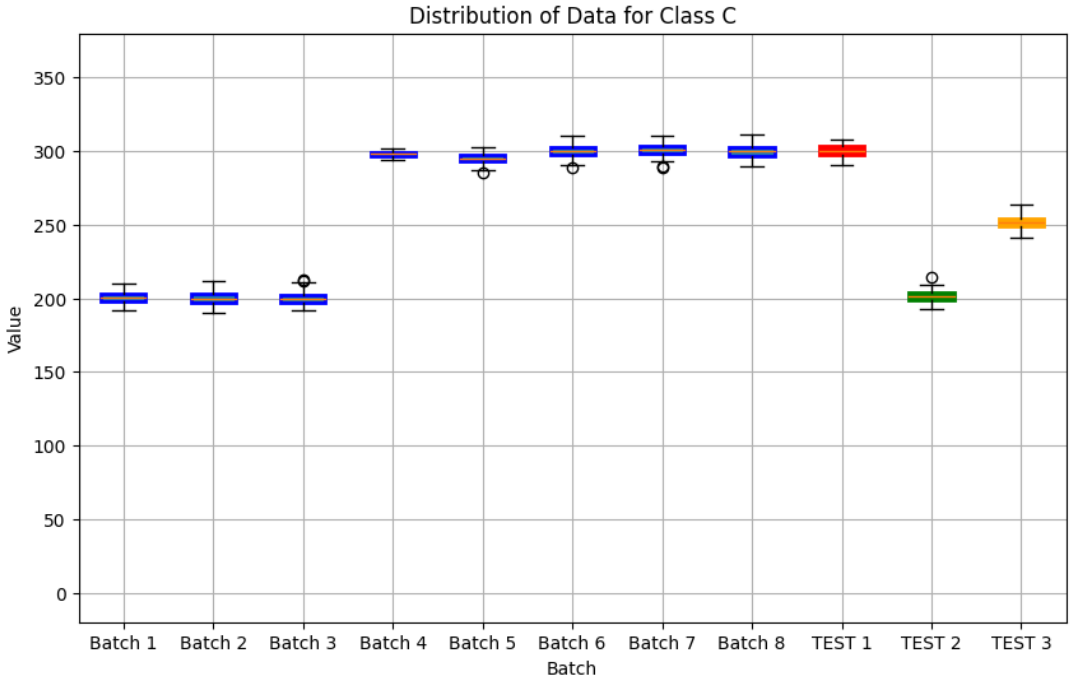


Figure 14: The Distribution of Class C Data in Sudden Drift Experiment 1 Where The Sudden Drift Event Occurs In An Increasing Manner Across All Three Classes

Table 1: Summary of dataset of sudden drift experiment 1 where the sudden drift event occurs in an increasing manner across all three classes.

Batch/Test Set	**Samples**	**Class A**		**Class B**		**Class C**		**Type**
		Mean (μ)	SD (σ)	Mean (μ)	SD (σ)	Mean (μ)	SD (σ)	
Batch 1	150	15	3	100	4	200	4	Training
Batch 2	150	16	4	102	4	200	4	Training
Batch 3	150	15	3	102	4	200	4	Training
Batch 4	150	75	3	200	5	300	3	Training
Batch 5	150	75	4	200	5	295	4	Training
Batch 6	150	75	3	200	4	300	3	Training
Batch 7	150	76	5	199	6	300	3	Training
Batch 8	150	75	3	200	4	300	3	Training
Test 1	150	75	3	200	4	300	4	Test
Test 2	150	16	3	100	4	201	4	Test
Test 3	150	36	3	150	4	251	4	Test

the standard deviation fluctuating between 4 and 6. Class C's features in the first three datasets are generated with a mean of 200 and a standard deviation of 4. In the fourth dataset, the mean increases significantly to 300, with a standard deviation of 3. In the fifth dataset, the mean decreases slightly to 295, with a standard deviation of 4. From the sixth to the eighth dataset, the mean again stabilizes around 300, with the standard deviation consistently between 3 and 4.

As seen in figures 11, 12, 13,14, the concept drift occurred with the fourth batch and is characterized as a sudden drift.

The test datasets, similarly structured, provide features for classes A, B, and C with varying means and standard deviations. In Test 1 dataset, Class A has a mean of 75 and a standard deviation of 3, Class B has a mean of 200 and a standard deviation of 4, and Class C has a mean of 300 and a standard deviation of 4. For Test 2, Class A has a mean of 16 and a standard deviation of 3, Class B has a mean of 100 and a standard deviation of 4, and Class C has a mean of 201 and a standard deviation of 4. In Test 3, Class A features have a mean of 36 and a standard deviation of 3, Class B features have a mean of 150 and a standard deviation of 4, and Class C features have a mean of 251 and a standard deviation of 4. For Test 1 data, it can be said that all classes are in the same concept as the data after the concept drift occurred. For Test 2 data, it can be stated that all classes are in the same concept as the data before the concept drift occurred. Test 3 data can be described as being in a concept that falls between the two different concepts in the training datasets, though it does not appear in the training set itself.

These test sets are used to measure the system's ability to predict data from the concept before the sudden drift occurred, the concept after the sudden drift, and a concept that lies between these two concepts. This assessment helps determine how effectively the system can handle transitions and variations in data concepts.

To compare with result of the system, a single model is trained by using 8 training batch. In order to make an objective comparison with the results of the system, the model used by the system explained in Section 3.5 was used while producing a single model. The results of the single model approach are given in Table2:

Table 2: Performance metrics of single model approach in sudden drift experiment 1 where the sudden drift event occurs in an increasing manner across all three classes.

	Test 1	Test 2	Test 3
Accuracy	0.9822	0.7133	1.0000
Recall	0.9822	0.7133	1.0000
Precision	0.9831	0.8459	1.0000

The training method discussed in Section 3.3.1 was used for training process and method discussed in Section 3.6.1 for predicting test sets. After the training process was completed, in addition to making predictions, all the saved training data were balanced using the method specified in the Section 3.4, and the models were trained and predictions made using these balanced datasets. This approach ensures that the models are not biased towards any particular class in the dataset. The result of these implementations are shown in Table3 and Table4:

Table 3: Performance of the system for test sets that are used in sudden drift experiment 1 where the sudden drift event occurs in an increasing manner across all three classes.

Metric	**Test 1**	**Test 2**	**Test 3**
Accuracy	0.6667	1.0000	0.8689
Recall	0.6667	1.0000	0.8689
Precision	0.5000	1.0000	0.9059

Table 4: Performance of the system that implemented training process with balanced datasets in sudden drift experiment 1 where the sudden drift event occurs in an increasing manner across all three classes.

Metric	**Test 1 (Balanced Datasets)**	**Test 2 (Balanced Datasets)**	**Test 3 (Balanced Datasets)**
Accuracy	0.6667	1.0000	0.8533
Recall	0.6667	1.0000	0.8533
Precision	0.5000	1.0000	0.8981

When comparing the results of the single model approach and our proposed system by looking at Table2 and Table3, the system has been much more successful in understanding the concept that was existed before the concept drift. On the other hand, it has performed worse than the single model approach in predicting the dataset that belongs to the concept formed after the concept drift. Although our system successfully predicted the test set that lies between the two concepts, it has shown poorer performance compared to the single model approach.

As seen in Table4, after balancing the training datasets, the system, which includes models created using the balanced datasets, displayed the same performance in predicting the Test 1 dataset, which is in the same concept as after the concept drift, as the system that includes models created using unbalanced datasets. Additionally, for the Test 2 dataset, which is in the same concept as before the concept drift, the system with balanced datasets also showed top performance, similar to the system with unbalanced datasets. However, after the datasets were balanced, there is a small setback in the performance of predicting the Test 3 dataset, which has a concept different from those in the training set.

The training method discussed in Section 3.3.2 was used for training process and method discussed in Section 3.6.1 for predicting test sets. After the training process was completed, in addition to making predictions, all the saved training data were balanced using the method specified in the Section 3.4, and the models were trained and predictions made using these balanced datasets. This approach ensures that the models are not biased towards any particular class in the dataset. The result are shown in Table5 and Table6:

Table 5: Performance metrics of the system that used accuracy based training strategy on test sets which is used in sudden drift experiment 1 where the sudden drift event occurs in an increasing manner across all three classes.

Metric	**Test 1**	**Test 2**	**Test 3**
Accuracy	1.0000	1.0000	0.8667
Recall	1.0000	1.0000	0.8667
Precision	1.0000	1.0000	0.9048

Table 6: Performance of the system that used accuracy based training strategy and implemented training process with balanced datasets on test sets which used in sudden drift experiment 1 where the sudden drift event occurs in an increasing manner across all three classes.

Metric	**Test 1 (Balanced Datasets)**	**Test 2 (Balanced Datasets)**	**Test 3 (Balanced Datasets)**
Accuracy	1.0000	1.0000	0.8667
Recall	1.0000	1.0000	0.8667
Precision	1.0000	1.0000	0.9048

Results of accuracy based training strategy are shown in Table5. It can be stated that the system operating with an accuracy-based training strategy is more successful in predicting the concept that was emerged after concept drift, compared to both the single model approach and the system using a centroid difference-focused training strategy. Also, accuracy-based training strategy is more successful than the single model approach to understand existing concept before concept drift. Accuracy-based training strategy is slight better than the system using a centroid difference-focused training strategy to understand concept that is lying between the concepts present during the training phase. On the other hand, it can be said that the single model approach is more successful in predicting the test set that represents a concept lying between the concepts present during the training phase. As seen in Table6, when class imbalances in the synthetic training dataset were addressed, there was no increase or decrease in the performance of the system using the accuracy-based strategy.

4.1.1.2 Sudden Drift Experiment 2

In Experiment 2, the sudden drift occurred with some classes experiencing a sudden decrease and others a sudden increase. Figure15, Figure16, Figure17, and Figure18 provide a visual representation of the concept drift event in the dataset using boxplots. Figure15 displays the overall data distribution, while Figure16, Figure17, and Figure18 focus specifically on the distributions within Class A, Class B, and Class C, respectively. These visualizations reveal how the patterns and general distribution of the data shift across batches before, and after the drift event.

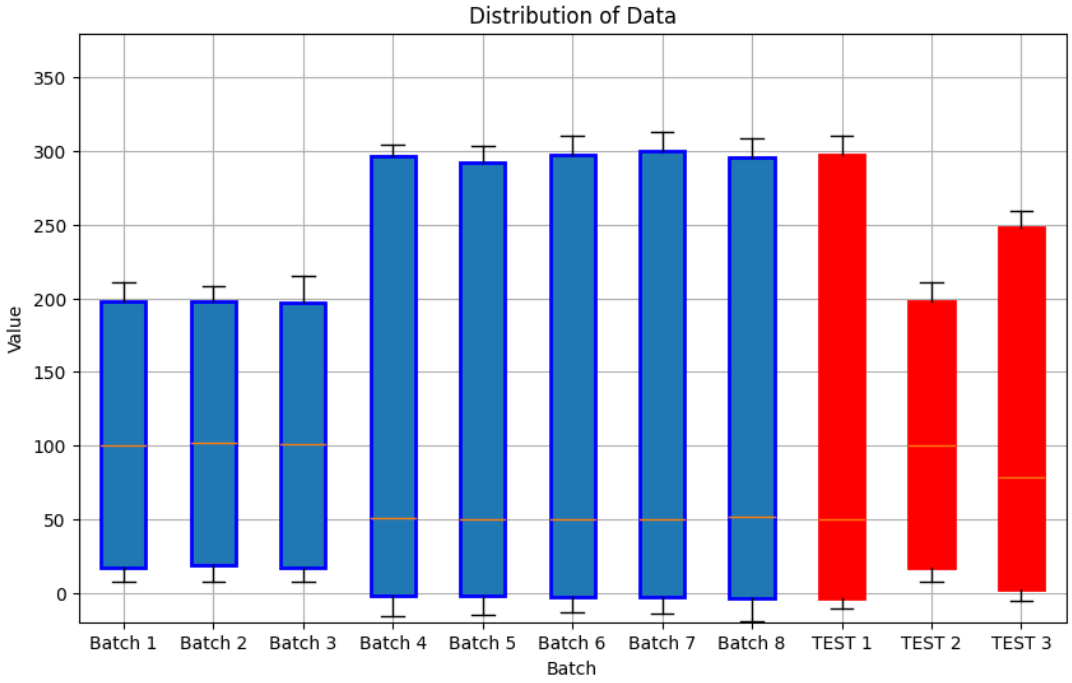


Figure 15: The Distribution of Data in Sudden Drift Experiment 2 Where the Sudden Drift Event Occurs as an Increase in Classes A and B, and a Decrease in Class C

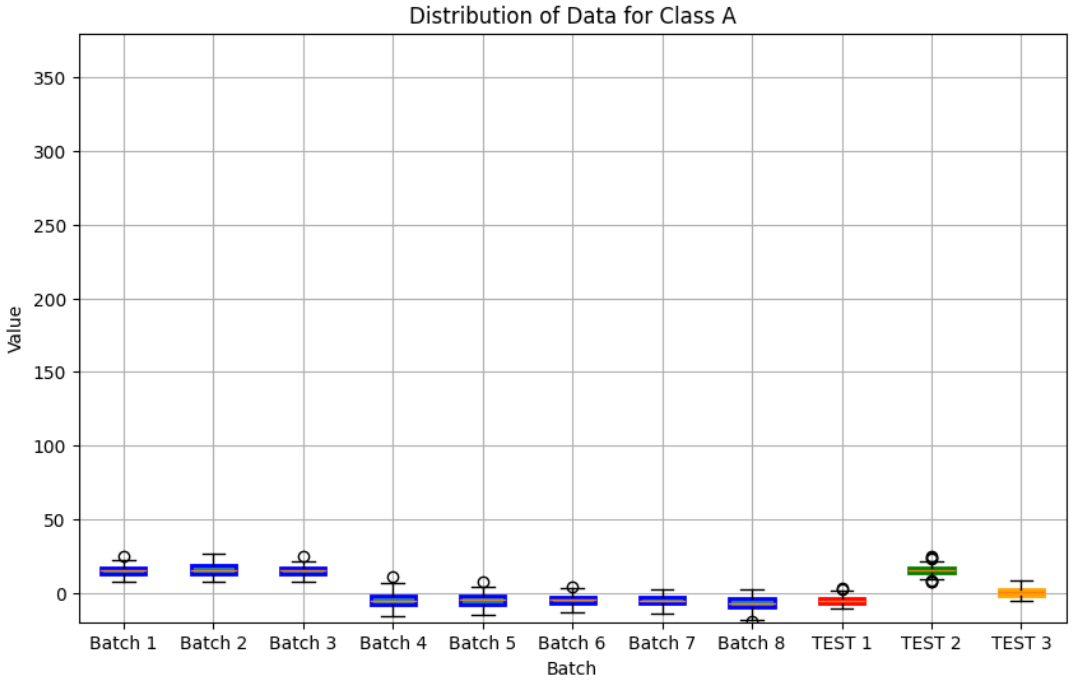


Figure 16: The Distribution of Class A Data in Sudden Drift Experiment 2 Where the Sudden Drift Event Occurs as a Decrease in Classes A and B, and an Increase in Class C

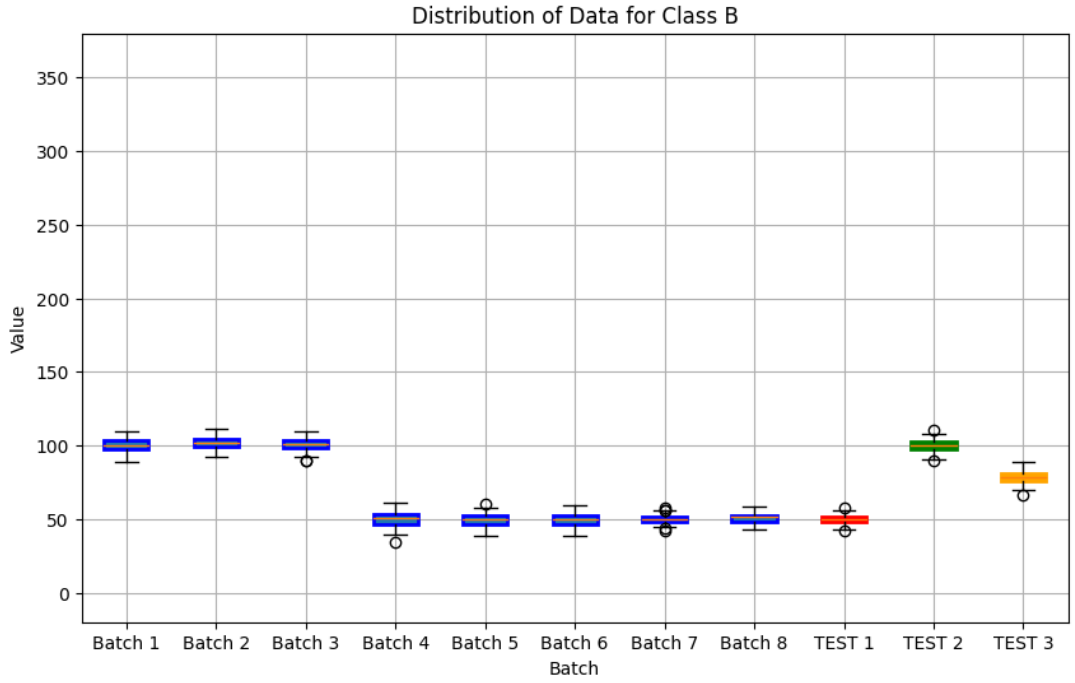


Figure 17: The Distribution of Class B Data in Sudden Drift Experiment 2 Where the Sudden Drift Event Occurs as a Decrease in Classes A and B, and an Increase in Class C

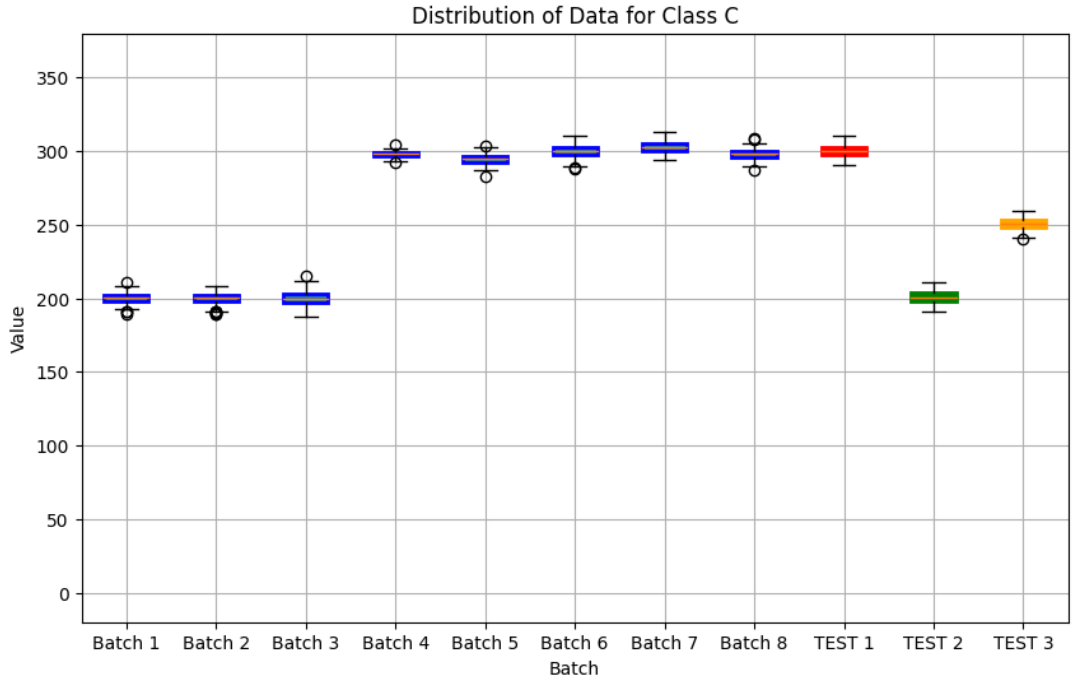


Figure 18: The Distribution of Class C Data in Sudden Drift Experiment 2 Where the Sudden Drift Event Occurs as a Decrease in Classes A and B, and an Increase in Class C

The Table7 numerically presents characteristics such as the mean and standard deviation of the classes contained in the batches of the dataset.

Table 7: Summary of dataset of sudden drift experiment 2 where the sudden drift event occurs as a decrease in classes A and B, and an increase in class C.

Batch/Test Set	**Samples**	**Class A**		**Class B**		**Class C**		**Type**
		Mean (μ)	SD (σ)	Mean (μ)	SD (σ)	Mean (μ)	SD (σ)	
Batch 1	150	15	3	100	4	200	4	Training
Batch 2	150	16	4	102	4	200	4	Training
Batch 3	150	15	3	102	4	200	4	Training
Batch 4	150	-5	3	50	3	295	3	Training
Batch 5	150	-5	4	50	4	290	4	Training
Batch 6	150	-5	3	50	3	300	4	Training
Batch 7	150	-7	5	51	4	305	4	Training
Batch 8	150	-5	3	50	3	295	3	Training
Test 1	150	-5	3	50	3	300	4	Test
Test 2	150	16	3	100	4	201	4	Test
Test 3	150	0	3	79	4	251	4	Test

The dataset generated consists of multiple datasets, each containing samples distributed across three distinct classes: A, B, and C. Each instance within a dataset includes three features. The dataset is designed to contain a consistent number of samples per class as 150 across multiple datasets. For Class A, features in the first dataset are generated with a mean (mu) of 15 and a standard deviation (sd) of 3. In the second dataset, the mean increases to 16 with a standard deviation of 4. The third dataset reverts to a mean of 15 and a standard deviation of 3. From the fourth to the eighth dataset, the mean significantly decreases to around -5 (with a slight variation to -7 in the seventh dataset) and the standard deviation ranges from 3 to 5. For Class B, features in the first dataset have a mean of 100 and a standard deviation of 4. The mean slightly increases to 102 in the second dataset, maintaining the same standard deviation. In the subsequent datasets (fourth to eighth), the mean stabilizes around 50 (51 in the seventh dataset), with the standard deviation fluctuating between 3 and 4. Class C's features in the first three datasets are generated with a mean of 200 and a standard deviation of 4. In the fourth dataset, the mean increases to 295, with a standard deviation of 3. From the fifth to the eighth dataset, the mean differs around 300, with the standard deviation consistently between 3 and 4.

As shown in 15, the sudden change for the entire dataset occurred with the fourth incoming dataset, manifesting as both increases and decreases. This situation can be interpreted as the concept formed after the concept drift containing instances with more diverse characteristics. To better understand the characteristics of the concept drift, an analysis should also be conducted on a class-by-class basis.

In Figure16, a sudden decrease can be observed in Class A with the fourth dataset. It is seen that in Class B, a more significant sudden decrease occurred with the fourth dataset compared to the change in Class A in Figure17. On the other hand, a sudden increase involving concept drift has been observed in Class C with the fourth dataset as seen in Figure18,

The test datasets, similarly structured, provide features for classes A, B, and C with varying means and standard deviations. In Test 1, Class A features have a mean of -5 and a standard deviation of 3,

Class B features have a mean of 50 and a standard deviation of 3, and Class C features have a mean of 300 and a standard deviation of 4. For Test 2, Class A has a mean of 16 and a standard deviation of 3, Class B has a mean of 100 and a standard deviation of 4, and Class C has a mean of 201 and a standard deviation of 4. In Test 3, Class A features have a mean of 0 and a standard deviation of 3, Class B features have a mean of 79 and a standard deviation of 4, and Class C features have a mean of 251 and a standard deviation of 4. It can be said that Test 1 dataset represents concept after concept drift occurred. Test 2 dataset represents concept that exists before concept drift occurred. Test 3 has concept which does not exists in training datasets.

To compare with result of the system, a single model is trained by using 8 training batch. In order to make an objective comparison with the results of the system, the model used by the system explained in Section 3.5 was used while producing a single model. The results of the single model approach are given in Table8:

Table 8: Performance metrics of single model approach in sudden drift experiment 2 where the sudden drift event occurs as a decrease in classes A and B, and an increase in class C.

	Test 1	Test 2	Test 3
Accuracy	1.0	1.0	1.0
Recall	1.0	1.0	1.0
Precision	1.0	1.0	1.0

The training strategy defined in Section 3.3.1 was used for training process and method discussed in Section 3.6.1 for predicting test sets. After the training process was completed, in addition to making predictions, all the collected training data were balanced using the method specified in the Section 3.4, and the models were trained and predictions made using these balanced datasets. The results of these implementation are shown in Table9 and Table10:

Table 9: Performance of the system for test sets that are used in sudden drift experiment 2 where the sudden drift event occurs as a decrease in classes A and B, and an increase in class C.

Metric	**Test 1**	**Test 2**	**Test 3**
Accuracy	1.0000	1.0000	1.0000
Recall	1.0000	1.0000	1.0000
Precision	1.0000	1.0000	1.0000

Both our system and the single model approach successfully predicted all the test datasets in experiment 2.

The training method discussed in Section 3.3.2 was used for training process and method discussed in Section 3.6.1 for predicting test sets. After the training process was completed, in addition to making predictions, all the saved training data were balanced using the method specified in the Section 3.4, and the models were trained and predictions made using these balanced datasets. This approach

Table 10: Performance of the system that implemented training process with balanced datasets in sudden drift experiment 2 where the sudden drift event occurs as a decrease in classes A and B, and an increase in class C.

Metric	Test 1 (Balanced Datasets)	Test 2 (Balanced Datasets)	Test 3 (Balanced Datasets)
Accuracy	1.0000	1.0000	1.0000
Recall	1.0000	1.0000	1.0000
Precision	1.0000	1.0000	1.0000

ensures that the models are not biased towards any particular class in the dataset. The results of these implementations are shown in Table11 and Table12:

Table 11: Performance metrics of the system that used accuracy based training strategy on test sets which used in sudden drift experiment 2 where the sudden drift event occurs as a decrease in classes A and B, and an increase in class C.

Metric	Test 1	Test 2	Test 3
Accuracy	1.0000	1.0000	1.0000
Recall	1.0000	1.0000	1.0000
Precision	1.0000	1.0000	1.0000

Table 12: Performance of the system that used accuracy based training strategy and implemented training process with balanced datasets on test sets which used in sudden drift experiment 2 where the sudden drift event occurs as a decrease in classes A and B, and an increase in class C.

Metric	Test 1 (Balanced Datasets)	Test 2 (Balanced Datasets)	Test 3 (Balanced Datasets)
Accuracy	1.0000	1.0000	1.0000
Recall	1.0000	1.0000	1.0000
Precision	1.0000	1.0000	1.0000

According to the results in Table11and Table12, the system built with an accuracy-centered training strategy also performed with complete accuracy.

4.1.1.3 Sudden Drift Experiment 3

In Experiment 3, while some classes increase, others decrease. Adding to the challenge, when two classes undergo concept drift, they tend to approach each other's concepts. This is expected to make distinguishing between classes more difficult. Figure 19 illustrates the overall distribution of the data, while Figure 20, Figure 21, and Figure 22 provide focused views of the distributions within Class A, Class B, and Class C, respectively. Under these conditions, the performance of our system will be evaluated.

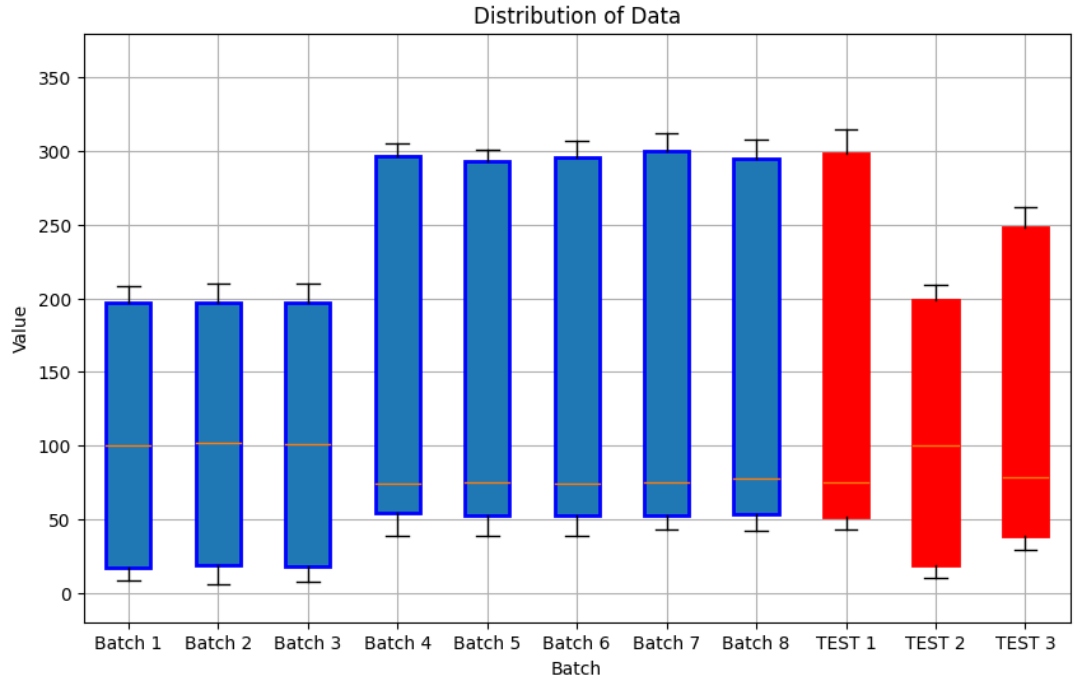


Figure 19: The Distribution of Data in Sudden Drift Experiment 3 Where the Sudden Drift Event Occurs as an Increase in Classes A and C, and a Decrease in Class B, and After the Concept Drift Event, Classes A and B Converge to the More Similar Concept

The Table13 numerically presents characteristics such as the mean and standard deviation of the classes contained in the batches of the dataset.

Table 13: Summary of dataset of sudden drift experiment 3 where the sudden drift event occurs as an increase in classes A and C, and a decrease in class B, and after the concept drift event, classes A and B converge to the more similar concept.

Batch/Test Set	**Samples**	**Class A**		**Class B**		**Class C**		**Type**
		Mean (μ)	SD (σ)	Mean (μ)	SD (σ)	Mean (μ)	SD (σ)	
Batch 1	150	15	3	100	4	200	4	Training
Batch 2	150	16	4	102	4	200	4	Training
Batch 3	150	15	3	102	4	200	4	Training
Batch 4	150	75	3	50	3	295	3	Training
Batch 5	150	75	4	50	4	290	3	Training
Batch 6	150	75	3	50	3	300	4	Training
Batch 7	150	77	5	51	4	305	3	Training
Batch 8	150	75	3	50	3	300	4	Training
Test 1	150	75	3	50	3	300	4	Test
Test 2	150	16	3	100	4	201	4	Test
Test 3	150	36	3	79	4	251	4	Test

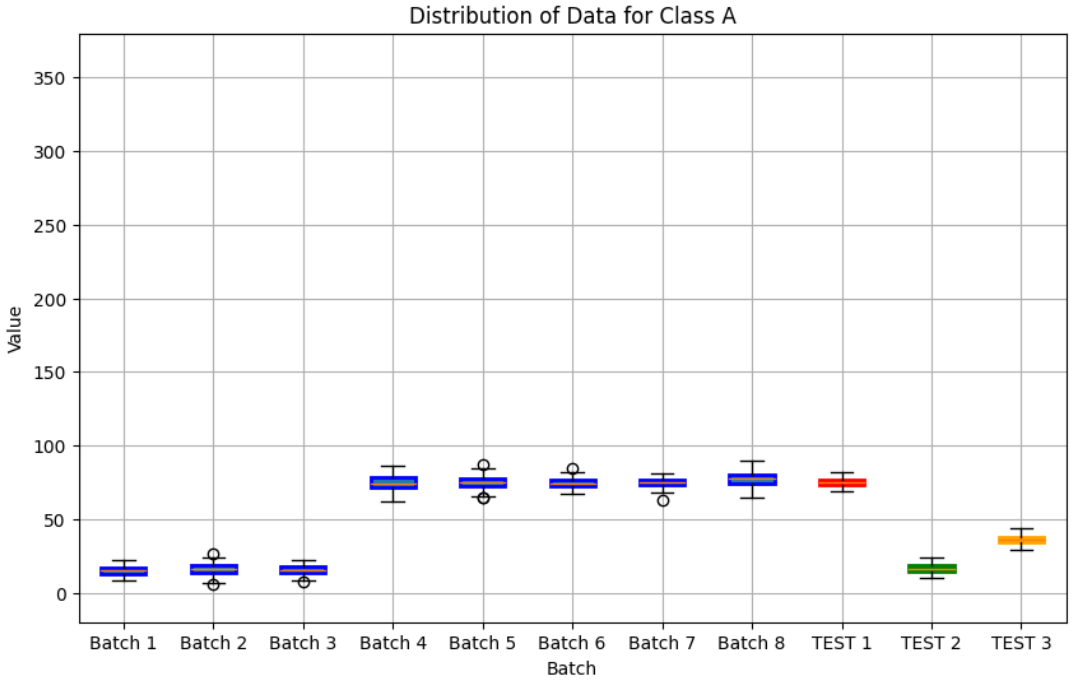


Figure 20: The Distribution of Class A Data in Sudden Drift Experiment 3 Where the Sudden Drift Event Occurs as an Increase in Classes A and C, and a Decrease in Class B, and After the Concept Drift Event, Classes A and B Converge to the More Similar Concept

The dataset generated comprises multiple datasets, each containing samples distributed across three distinct classes: A, B, and C. For Class A, features in the first dataset are generated with a mean of 15 and a standard deviation of 3. In the second dataset, the mean increases to 16 and the standard deviation to 4. The third dataset reverts to a mean of 15 and a standard deviation of 3. From the fourth to the eighth dataset, the mean significantly rises to approximately 75, with a slight variation to 77 in the seventh dataset, and the standard deviation ranges from 3 to 5. For Class B, the features in the first dataset have a mean of 100 and a standard deviation of 4. The mean slightly increases to 102 in the second dataset, with the same standard deviation. In subsequent datasets (fourth to eighth), the mean stabilizes around 50, with a slight increase to 51 in the seventh dataset, and the standard deviation fluctuates between 3 and 5. Class C's features in the first three datasets are generated with a mean of 200 and a standard deviation of 4. In the fourth dataset, the mean increases to 295, with a standard deviation of 3. From the fifth to the eighth dataset, the mean again stabilizes around 300, with the standard deviation consistently between 3 and 4. As seen in Figure20 and Figure21, After the concept drift, the new concepts of Classes A and B converge significantly, becoming quite similar to each other.

The test datasets, similarly structured, provide features for classes A, B, and C with varying means and standard deviations. For Test 1, Class A has a mean of 75 and a standard deviation of 3, Class B has a mean of 50 and a standard deviation of 3, and Class C has a mean of 300 and a standard deviation of 4. In Test 2, Class A features have a mean of 16 and a standard deviation of 3, Class B features have a mean of 100 and a standard deviation of 4, and Class C features have a mean of 201 and a standard deviation of 4. For Test 3, Class A features have a mean of 36 and a standard deviation of 3, Class B

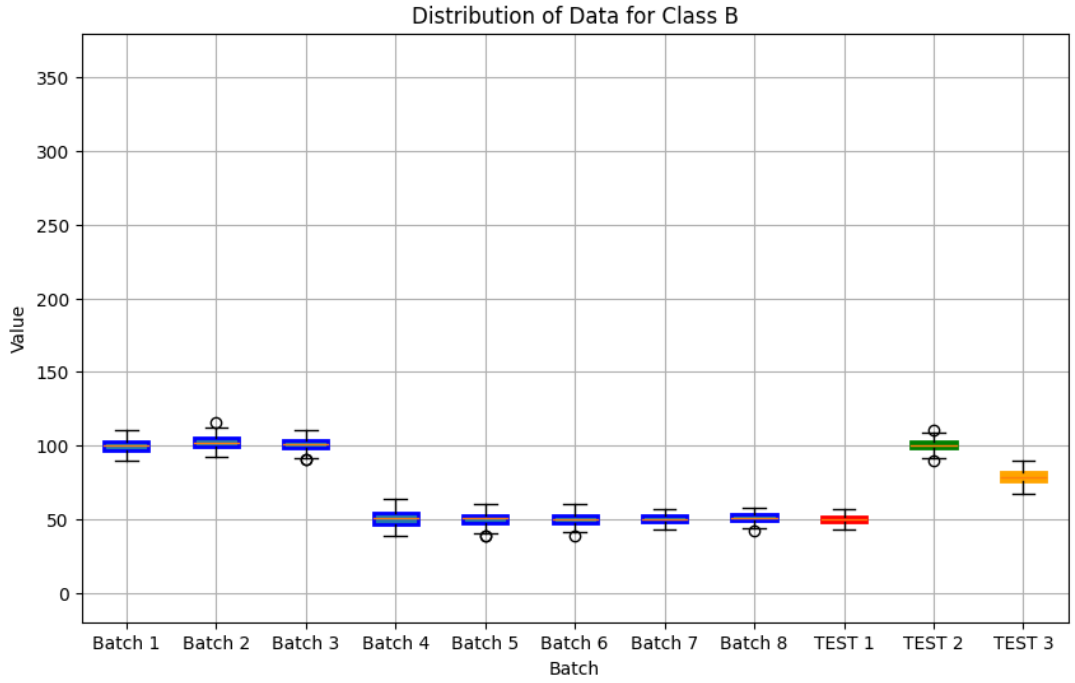


Figure 21: The Distribution of Class B Data in Sudden Drift Experiment 3 Where the Sudden Drift Event Occurs as an Increase in Classes A and C, and a Decrease in Class B, and After the Concept Drift Event, Classes A and B Converge to the More Similar Concept

features have a mean of 79 and a standard deviation of 4, and Class C features have a mean of 251 and a standard deviation of 4.

To compare with the results of the system, a single model was trained using eight training batches. For an objective comparison, the model employed by the system, as detailed in Section 3.5, was utilized for producing a single model. The results of the single model approach are presented in Table14:

Table 14: Performance metrics of single model approach in sudden drift experiment 3 where the sudden drift event occurs as an increase in classes A and C, and a decrease in class B, and after the concept drift event, classes A and B converge to the more similar concept.

	Test 1	Test 2	Test 3
Accuracy	0.3333	1.0	1.0
Recall	0.3333	1.0	1.0
Precision	0.3333	1.0	1.0

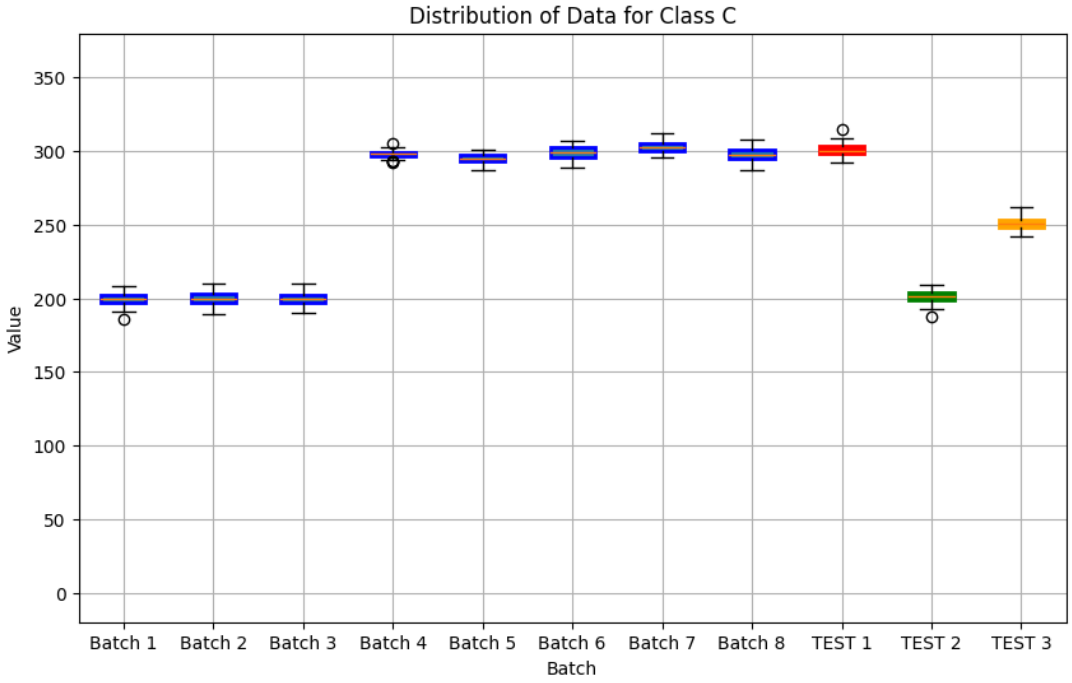


Figure 22: The Distribution of Class C Data in Sudden Drift Experiment 3 Where the Sudden Drift Event Occurs as an Increase in Classes A and C, and a Decrease in Class B, and After the Concept Drift Event, Classes A and B Converge to the More Similar Concept

The training strategy outlined in Section 3.3.1 was applied during the training process, and the method described in Section 3.6.1 was utilized for predicting test sets. Upon completion of the training process, in addition to making predictions, all the collected training data were balanced using the method specified in Section 3.4 The models were trained and predictions were made using these balanced datasets. The results of these implementations are shown below:

Table 15: Performance of the system for test sets that are used in sudden drift experiment 3 where the sudden drift event occurs as an increase in classes A and C, and a decrease in class B, and after the concept drift event, classes A and B converge to the more similar concept.

Metric	**Test 1**	**Test 2**	**Test 3**
Accuracy	0.6667	1.0000	0.6667
Recall	0.6667	1.0000	0.6667
Precision	0.6667	1.0000	0.6667

The system has been observed to perform better than the single model approach on the test data representing the concepts that emerged after the concept drift as seen in Table15. Additionally, the system successfully predicted all instances in the Test 2 dataset, which represents the concepts before the concept drift. However, for the Test 3 dataset, which represents a concept not present in the training data, the system performed worse than the single model approach.

The balanced dataset version of the system that uses training strategy that is described in Section 3.3.1 is given Table16.

Table 16: Performance of the system that implemented training process with balanced datasets in sudden drift experiment 3 where the sudden drift event occurs as an increase in classes A and C, and a decrease in class B, and after the concept drift event, classes A and B converge to the more similar concept.

Metric	**Test 1 (Balanced Datasets)**	**Test 2 (Balanced Datasets)**	**Test 3 (Balanced Datasets)**
Accuracy	0.6667	1.0000	0.3333
Recall	0.6667	1.0000	0.3333
Precision	0.6667	1.0000	0.3333

The system, consisting of models trained with balanced training data, produced the same results as the unbalanced system for Test 1 and Test 2 datasets, but its performance was lower for Test 3.

The training method discussed in Section 3.3.2 was used for training process and method discussed in Section 3.6.1 for predicting test sets. After the training process was completed, in addition to making predictions, all the saved training data were balanced using the method specified in the Section 3.4, and the models were trained and predictions made using these balanced datasets. This approach ensures that the models are not biased towards any particular class in the dataset. The result of these implementation is shown below:

Table 17: Performance metrics of the system that used accuracy based training strategy on test sets which used in sudden drift experiment 3 where the sudden drift event occurs as an increase in classes A and C, and a decrease in class B, and after the concept drift event, classes A and B converge to the more similar concept.

Metric	**Test 1**	**Test 2**	**Test 3**
Accuracy	1.0000	1.0000	1.0000
Recall	1.0000	1.0000	1.0000
Precision	1.0000	1.0000	1.0000

Table 18: Performance of the system that used accuracy based training strategy and implemented training process with balanced datasets on test sets which used in sudden drift experiment 3 where the sudden drift event occurs as an increase in classes A and C, and a decrease in class B, and after the concept drift event, classes A and B converge to the more similar concept.

Metric	**Test 1 (Balanced Datasets)**	**Test 2 (Balanced Datasets)**	**Test 3 (Balanced Datasets)**
Accuracy	1.0000	1.0000	1.0000
Recall	1.0000	1.0000	1.0000
Precision	1.0000	1.0000	1.0000

As shown in Table17, the system using an accuracy-based training strategy outperformed both the single model approach and the previous training strategy, correctly predicting all the test data.

The application of the balancing process to the training datasets did not reduce performance, as all predictions were still made correctly as seen in Table18 .

4.1.2 Gradual Drift Scenario

In this subsection, the gradual drift scenarios are simulated to evaluate performance of our system against this scenario and results of experiments are shared.

Gradual drift describes a scenario where this shift in the distribution of data unfolds slowly over time, rather than suddenly permanent change of the concept. In a gradual drift scenario, the concept of the data may revert to its old concept after changing. During the process of permanently transitioning to a new concept, the data oscillates between the new and old concepts.

Against gradual drift event, the training methods described in Section 3.3.1 and Section 3.3.2 are implemented in the experiments of this section, and the prediction approach defined in Section 3.6.1 are used. The methods that are used in the experiments will be specified along with the characteristics of each experiment.

In gradual drift experiments, a random number of instances was generated for each batch. The number of batches was also determined randomly. After the size of the test set was randomly determined, all test sets were generated with the same size because bias due to the test set was not desired during comparisons.

The gradual drift scenarios were applied with different nuances in different experiments. In the first experiment, an increasing concept change occurred across all classes, but the extent of the increase varied among different classes. Also, new concept of some of the classes have approached the old concepts of another class in first experiment design. In the second experiment, while features of some classes were increased, others exhibited decreasing behaviors. In the final experiment, some classes increased, some decreased, and additionally, some classes began to show the conceptual properties of each other.

4.1.2.1 Gradual Drift Experiment 1

Experiment 1 simulated a gradual drift scenario in all classes. This gradual change manifested as an increase across all classes. Additionally, after the concept drift, some classes began to exhibit characteristics similar to the pre-drift concepts of other classes. Figure24displays the overall data distribution, while Figure25, Figure25, and Figure26 focus specifically on the distributions within Class A, Class B, and Class C, respectively.

The Table19 numerically presents characteristics such as the mean and standard deviation of the classes contained in the batches of the dataset.

The training dataset consists of 11 distinct batches that contain two different concepts. Batch 1 comprises 250 samples, featuring three attributes: class A has a mean of -3 and a standard deviation of 6,

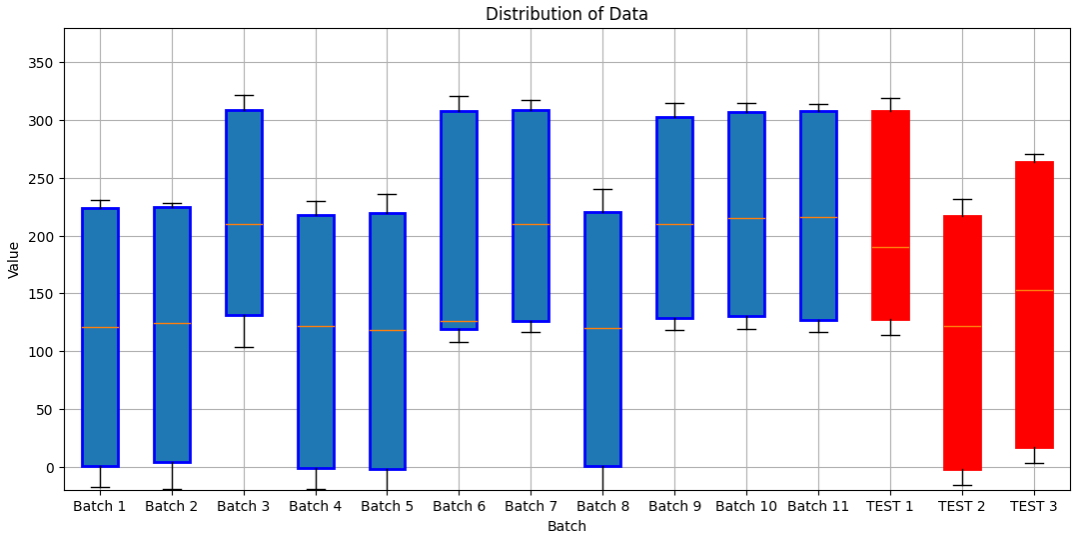


Figure 23: The Distribution of Data in Gradual Drift Experiment 1 Where The Gradual Drift Event Occurs As Increasing Manner Across All Three Classes

class B has a mean of 121 and a standard deviation of 3, and class C has a mean of 225 and a standard deviation of 2. Batch 2 includes 259 samples, where class A has a mean of 0 and a standard deviation of 7, class B has a mean of 124 and a standard deviation of 2, and class C has a mean of 225 and a standard deviation of 1. Batch 3 contains 90 samples, with class A having a mean of 125 and a standard deviation of 8, class B having a mean of 210 and a standard deviation of 4, and class C having a mean of 312 and a standard deviation of 4. Batch 4 consists of 160 samples, with class A parameters being a mean of -5 and a standard deviation of 4, class B having a mean of 122 and a standard deviation of 4,

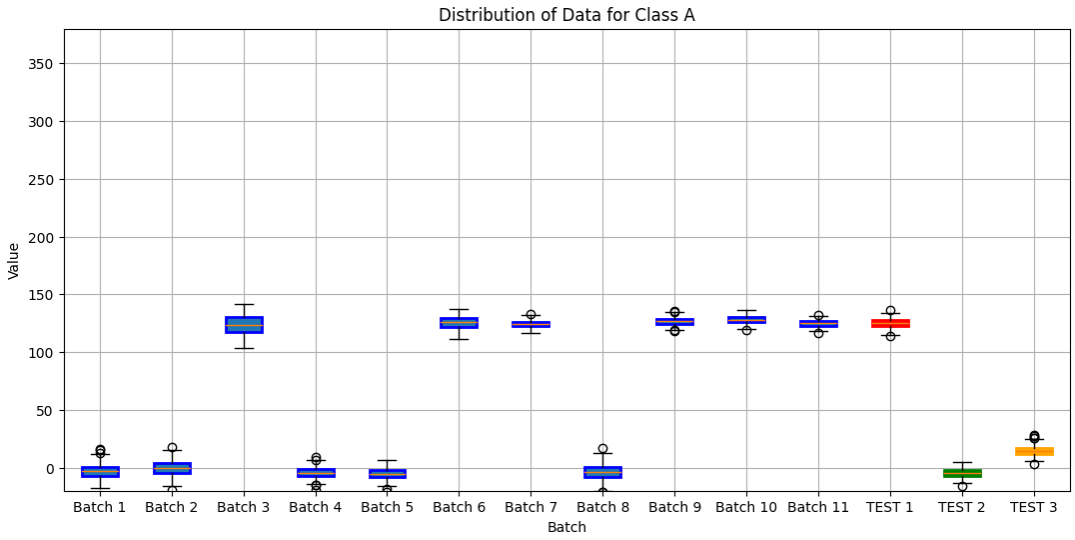


Figure 24: The Distribution of Class A Data in Gradual Drift Experiment 1 Where The Gradual Drift Event Occurs As Increasing Manner Across All Three Classes

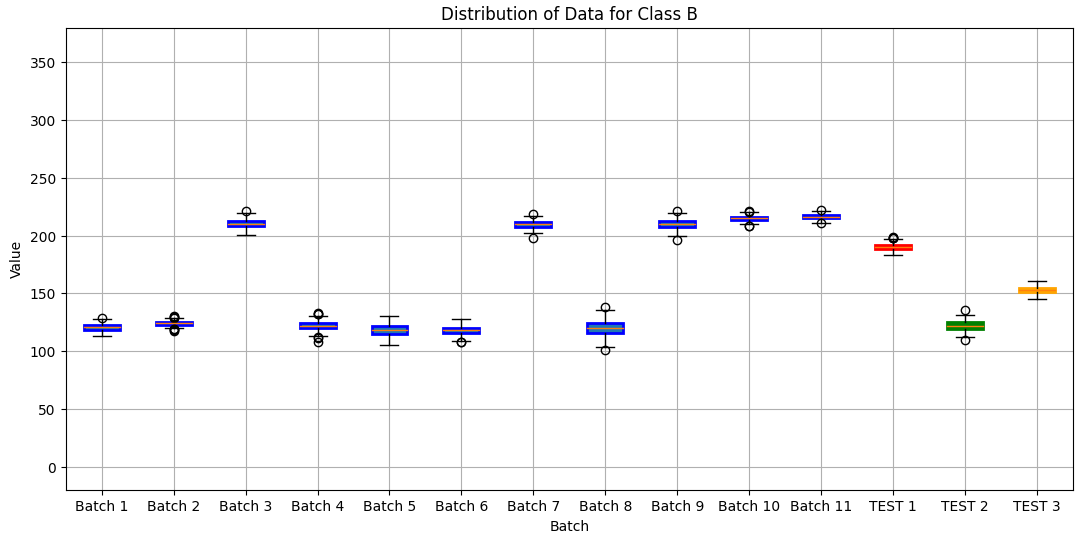


Figure 25: The Distribution of Class B Data in Gradual Drift Experiment 1 Where The Gradual Drift Event Occurs As Increasing Manner Across All Three Classes

and class C having a mean of 220 and a standard deviation of 4. Batch 5, with 258 samples, displays class A with a mean of -5 and a standard deviation of 5, class B with a mean of 118 and a standard deviation of 5, and class C with a mean of 223 and a standard deviation of 5. Batch 6, which includes 249 samples, has class A with a mean of 126 and a standard deviation of 5, class B with a mean of 118 and a standard deviation of 4, and class C with a mean of 311 and a standard deviation of 4. Batch 7 consists of 150 samples, including class A with a mean of 125 and a standard deviation of 4, class B with a mean of 210 and a standard deviation of 3, and class C with a mean of 311 and a standard

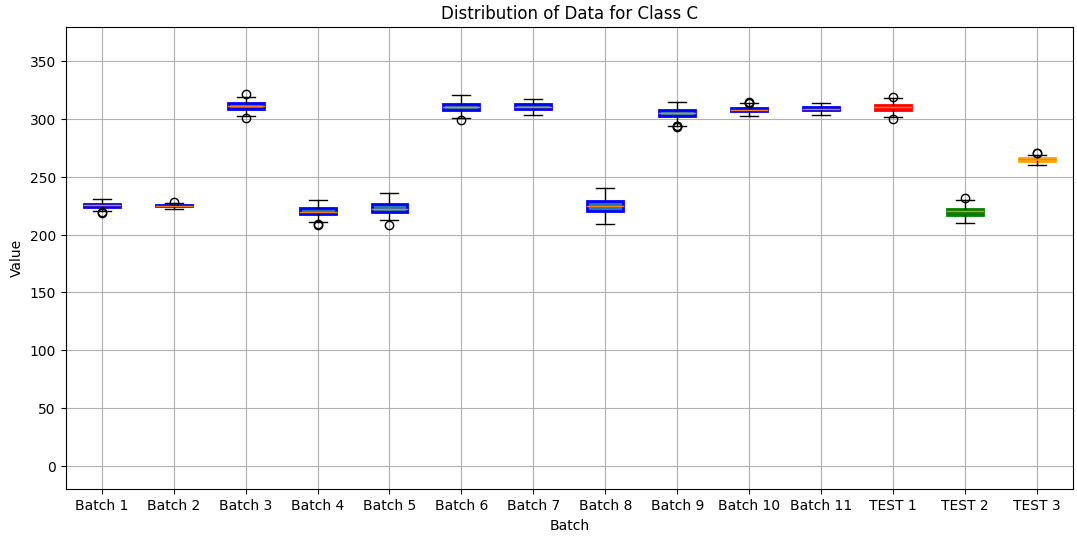


Figure 26: The Distribution of Class C Data in Gradual Drift Experiment 1 Where The Gradual Drift Event Occurs As Increasing Manner Across All Three Classes

Table 19: Summary of dataset of gradual drift experiment 1 where the gradual drift event occurs in an increasing manner across all three classes.

Batch/Test Set	**Samples**	**Class A**		**Class B**		**Class C**		**Type**
		Mean (μ)	SD (σ)	Mean (μ)	SD (σ)	Mean (μ)	SD (σ)	
Batch 1	250	-3	6	121	3	225	2	Training
Batch 2	259	0	7	124	2	225	1	Training
Batch 3	90	125	8	210	4	312	4	Training
Batch 4	160	-5	4	122	4	220	4	Training
Batch 5	258	-5	5	118	5	223	5	Training
Batch 6	249	126	5	118	4	311	4	Training
Batch 7	150	125	4	210	3	311	3	Training
Batch 8	353	-4	7	120	6	225	6	Training
Batch 9	450	127	3	210	4	305	4	Training
Batch 10	548	128	3	215	2	308	2	Training
Batch 11	150	125	3	216	2	309	2	Training
Test Set 1	250	125	4	190	3	310	3	Test
Test Set 2	250	-5	4	122	4	220	4	Test
Test Set 3	250	15	4	153	3	265	2	Test

deviation of 3. Batch 8, the largest with 353 samples, has class A with a mean of -4 and a standard deviation of 7, class B with a mean of 120 and a standard deviation of 6, and class C with a mean of 225 and a standard deviation of 6. Batch 9 comprises 450 samples, with class A parameters being a mean of 127 and a standard deviation of 3, class B having a mean of 210 and a standard deviation of 4, and class C having a mean of 305 and a standard deviation of 4. Batch 10, consisting of 548 samples, includes class A with a mean of 128 and a standard deviation of 3, class B with a mean of 215 and a standard deviation of 2, and class C with a mean of 308 and a standard deviation of 2. Finally, Batch 11, with 150 samples, features class A with a mean of 125 and a standard deviation of 3, class B with a mean of 216 and a standard deviation of 2, and class C with a mean of 309 and a standard deviation of 2.

As illustrated in Figure24, and Figure25, after the concept drift occurred, the new concept of Class A shows characteristics similar to the pre-drift concept of Class B. How this affects the results of our system and the single model approach will provide important insights.

In addition to training data, there are three test sets used to evaluate the model's performance. Test set 1, with 250 samples, class A has a mean of 125 and a standard deviation of 4, class B with a mean of 190 and a standard deviation of 3, and class C with a mean of 310 and a standard deviation of 3. Test set 2, also with 250 samples, has class A parameters of a mean of -5 and a standard deviation of 4, class B with a mean of 122 and a standard deviation of 4, and class C with a mean of 220 and a standard deviation of 4. Test set 3, comprising 250 samples, presents class A with a mean of 15 and a standard deviation of 4, class B with a mean of 153 and a standard deviation of 3, and class C with a mean of 265 and a standard deviation of 2.

Test 1 dataset represents the concepts of the classes after the concept drift has occurred. Test 2 dataset contains data corresponding to the concepts that the classes had before the concept drift. Test 3 dataset

does not align with the two different concepts present in the training datasets and show characteristics of a concept that falls between these two concepts that are exists in training data.

To compare with the system result, a single model was trained by dataset consist of 11 training groups. To make an objective comparison with the system results, the model used by the system described in Section 3.5 was used to generate a single model. The results of the single model approach are given in Table20:

Table 20: Performance metrics of single model approach in gradual drift experiment 1 where the gradual drift event occurs in an increasing manner across all three classes.

	Test 1	Test 2	Test 3
Accuracy	0.7880	0.5827	1.0000
Recall	0.7880	0.5827	1.0000
Precision	0.7880	0.5827	1.0000

For the training process, the training method described in Section 3.3.1 was used, and for predicting the test sets, the method described in Section 3.6.1 was implemented. After results of the system that has these properties are recorded, all the recorded training data were balanced using the method described in Section 3.4 and the models were trained and predictions were made on these balanced datasets. The output of this application is shown below:

Table 21: Performance of the system for test sets that are used in gradual drift experiment 1 where the gradual drift event occurs in an increasing manner across all three classes.

Metric	**Test 1**	**Test 2**	**Test 3**
Accuracy	0.6667	1.0000	0.6520
Recall	0.6667	1.0000	0.6520
Precision	0.6667	1.0000	0.6520

Table 22: Performance of the system that implemented training process with balanced datasets in gradual drift experiment 1 where the gradual drift event occurs in an increasing manner across all three classes.

Metric	**Test 1 (Balanced Datasets)**	**Test 2 (Balanced Datasets)**	**Test 3 (Balanced Datasets)**
Accuracy	0.6667	1.0000	1.0000
Recall	0.6667	1.0000	1.0000
Precision	0.6667	1.0000	1.0000

By looking the tables which are Table20, Table21, and Table22, it can be seen that the single model approach is more successful in predicting a test dataset that represents the concept formed after the concept drift. Although there is approximately a 12 percent difference in performance, this gap is

not sufficient to indicate that the system is ineffective. On the other hand, our system is completely successful in predicting the Test 2 dataset, which represents the concept before the concept drift in the training dataset. Despite our system's success, the single model approach showed rather weak performance. For predicting the Test 3 dataset, which resembles a concept between the two concepts present in the training dataset, the single model approach was completely successful. Conversely, while our system initially showed weak performance before addressing class imbalance in the training datasets, it achieved the same performance as the single model approach and accurately predicted all of the Test 3 dataset after balancing with synthetic data.

For the training process, the training method described in the Section 3.3.2 was applied, and the method described in the Section 3.6.1 was used to predict the test sets. After getting results of this implementation, all stored training data were balanced using the method described in the Section 3.4, and the models were trained and predictions were made on these balanced datasets. The result of this application is shown below:

Table 23: Performance metrics of the system that used accuracy based training strategy on test sets which used in gradual drift experiment 1 where the gradual drift event occurs in an increasing manner across all three classes.

Metric	**Test 1**	**Test 2**	**Test 3**
Accuracy	1.0000	1.0000	0.6667
Recall	1.0000	1.0000	0.6667
Precision	1.0000	1.0000	0.6667

Table 24: Performance of the system that used accuracy based training strategy and implemented training process with balanced datasets on test sets which used in gradual drift experiment 1 where the gradual drift event occurs in an increasing manner across all three classes.

Metric	**Test 1 (Balanced Datasets)**	**Test 2 (Balanced Datasets)**	**Test 3 (Balanced Datasets)**
Accuracy	1.0000	1.0000	0.6667
Recall	1.0000	1.0000	0.6667
Precision	1.0000	1.0000	0.6667

In the Table23 and Table24, it can be seen that the system using the accuracy-based approach (Section 3.3.2) predicted the Test 1 dataset, which represents the concept formed after a concept drift, better than the system using the approach that was mentioned in Section 3.3.1. With the unbalanced training datasets, the accuracy-based approach performed slightly better than the approach using the method described in Section 3.3.1. However, unlike the system using the training strategy from Section 3.3.1, the performance of the accuracy-based approach did not improve after the training datasets were balanced.

4.1.2.2 Gradual Drift Experiment 2

In this experiment, the concept drift event occurs as a gradual increase for some classes and a gradual decrease for others. Figure27, illustrates the concept drift event in the dataset through boxplots, showing the distribution of the data across different batches.

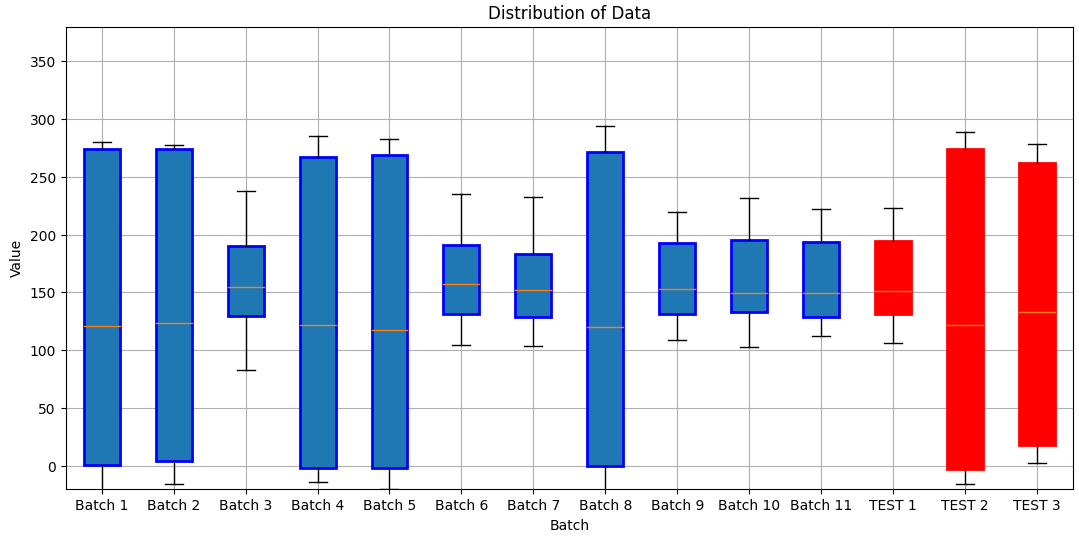


Figure 27: The Distribution of Data in Gradual Drift Experiment 2 Where the Gradual Drift Event Occurs as an Increase in Classes A and B, and a Decrease in Class C

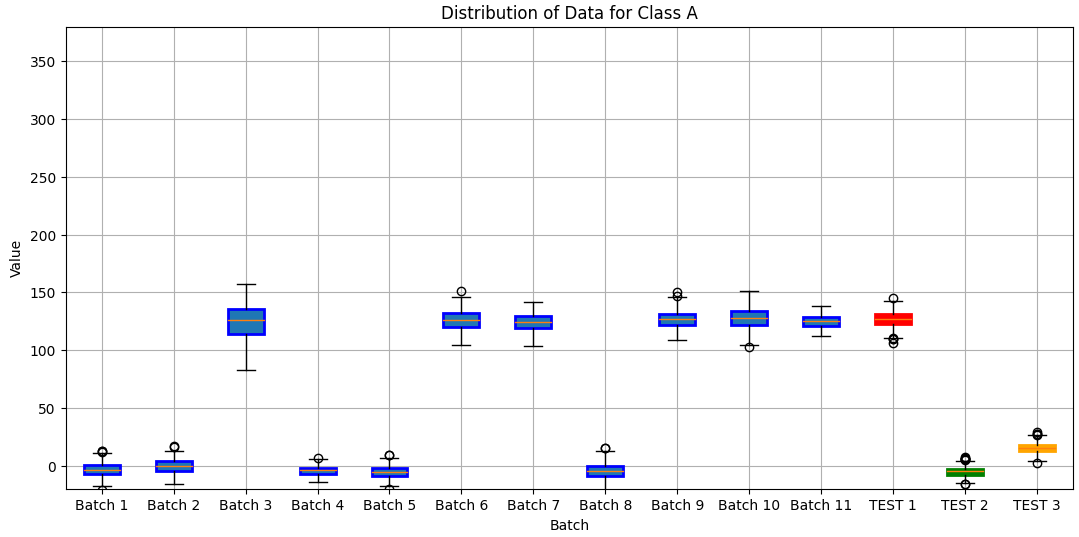


Figure 28: The Distribution of Class A Data in Gradual Drift Experiment 2 Where the Gradual Drift Event Occurs as an Increase in Classes A and B, and a Decrease in Class C

The Table25 numerically presents characteristics such as the mean and standard deviation of the classes contained in the batches of the dataset.

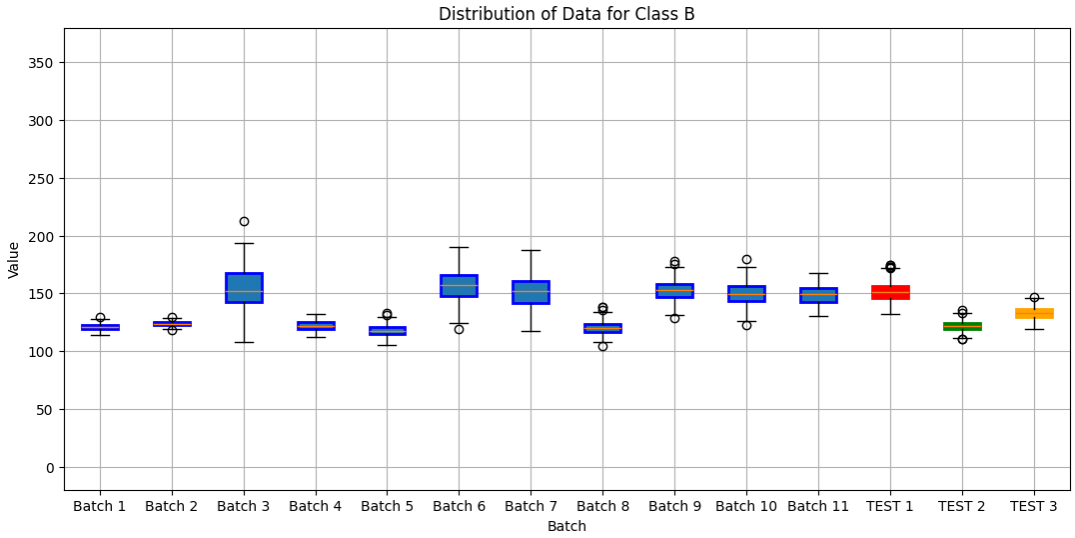


Figure 29: The Distribution of Class B Data in Gradual Drift Experiment 2 Where the Gradual Drift Event Occurs as an Increase in Classes A and B, and a Decrease in Class C

The dataset which is for training purposes consists of 11 distinct batches, each containing three features, and various test sets designed to evaluate the model's performance. In Batch 1, with 250 samples, class A has a mean of -3 and a standard deviation of 6, class B has a mean of 121 and a standard deviation of 3, and class C has a mean of 275 and a standard deviation of 2. Batch 2, comprising 259 samples, shows class A with a mean of 0 and a standard deviation of 7, class B with a mean of 124 and a standard deviation of 2, and class C with a mean of 275 and a standard deviation of 1. Batch 3 includes 90 samples with class A having a mean of 125 and a standard deviation of 18, class B a mean

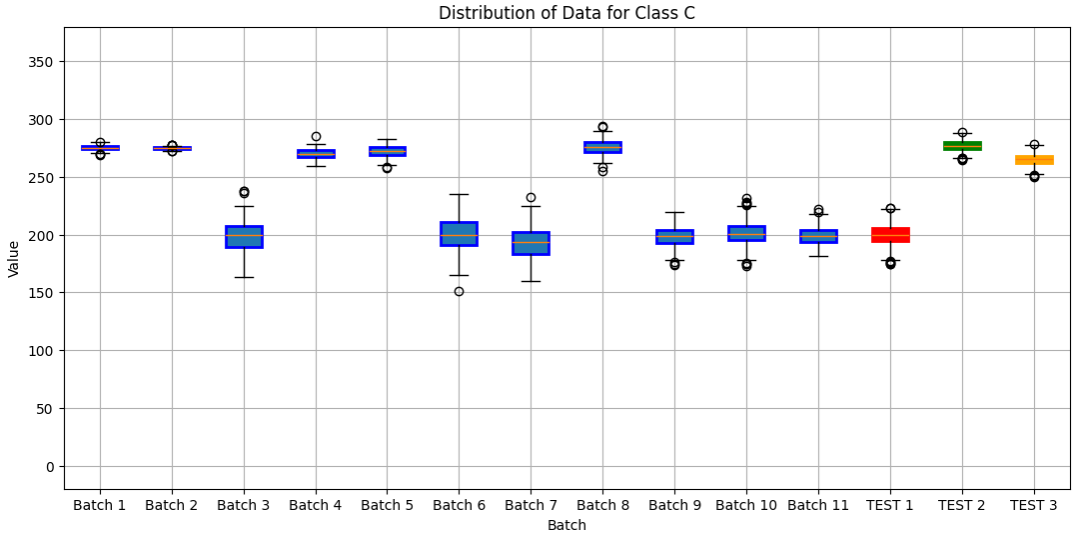


Figure 30: The Distribution of Class C Data in Gradual Drift Experiment 2 Where the Gradual Drift Event Occurs as an Increase in Classes A and B, and a Decrease in Class C

Table 25: Summary of dataset of gradual drift experiment 2 where the gradual drift event occurs as an increase in classes A and B, and a decrease in class C.

Batch/Test Set	**Samples**	**Class A**		**Class B**		**Class C**		**Type**
		Mean (μ)	SD (σ)	Mean (μ)	SD (σ)	Mean (μ)	SD (σ)	
Batch 1	250	-3	6	121	3	275	2	Training
Batch 2	259	0	7	124	2	275	1	Training
Batch 3	90	125	18	155	18	200	16	Training
Batch 4	160	-5	4	122	4	270	4	Training
Batch 5	258	-5	5	118	5	273	5	Training
Batch 6	249	126	9	158	14	201	13	Training
Batch 7	150	125	7	150	13	195	13	Training
Batch 8	353	-4	7	120	6	275	6	Training
Batch 9	450	127	7	153	9	199	8	Training
Batch 10	548	128	8	150	9	201	9	Training
Batch 11	150	125	6	150	8	200	8	Training
Test Set 1	1050	127	6	151	7	200	8	Test
Test Set 2	1050	-5	4	122	4	277	4	Test
Test Set 3	1050	15	4	133	5	265	5	Test

of 155 and a standard deviation of 18, and class C a mean of 200 and a standard deviation of 16. Batch 4, consisting of 160 samples, features class A with a mean of -5 and a standard deviation of 4, class B a mean of 122 and a standard deviation of 4, and class C a mean of 270 and a standard deviation of 4. Batch 5 contains 258 samples with class A showing a mean of -5 and a standard deviation of 5, class B a mean of 118 and a standard deviation of 5, and class C a mean of 273 and a standard deviation of 5. Batch 6 has 249 samples with class A having a mean of 126 and a standard deviation of 9, class B a mean of 158 and a standard deviation of 14, and class C a mean of 201 and a standard deviation of 13. Batch 7 includes 150 samples where class A has a mean of 125 and a standard deviation of 7, class B a mean of 150 and a standard deviation of 13, and class C a mean of 195 and a standard deviation of 13. Batch 8, the largest with 353 samples, features class A with a mean of -4 and a standard deviation of 7, class B a mean of 120 and a standard deviation of 6, and class C a mean of 275 and a standard deviation of 6. Batch 9 comprises 450 samples with class A showing a mean of 127 and a standard deviation of 7, class B a mean of 153 and a standard deviation of 9, and class C a mean of 199 and a standard deviation of 8. Batch 10, with 548 samples, features class A with a mean of 128 and a standard deviation of 8, class B a mean of 150 and a standard deviation of 9, and class C a mean of 201 and a standard deviation of 9. Finally, Batch 11 includes 150 samples, presenting class A with a mean of 125 and a standard deviation of 6, class B a mean of 150 and a standard deviation of 8, and class C a mean of 200 and a standard deviation of 8.

Examining the figures that are Figure28, Figure29, and Figure30,show the changes in the distributions of the classes, it can be seen that the concept changes in all classes occurred as a gradual drift starting with the fourth batch. The concept drift occurred as a gradual increase in Classes A and B, while it manifested as a gradual decrease in Class C. Additionally, the severity of the gradual drift event varied among the classes. For instance, the increase in Class B was relatively small, whereas in Class A, it

was significantly larger compared to Class B. The gradual decrease in Class C, although not as severe as the gradual increase in Class A, is more significant than the increase in Class B.

Three test sets have been generated to evaluate the model's performance. Test set 1, with 1050 samples, includes class A with a mean of 127 and a standard deviation of 6, class B a mean of 151 and a standard deviation of 7, and class C a mean of 200 and a standard deviation of 8. Test set 2, also with 1050 samples, shows class A with a mean of -5 and a standard deviation of 4, class B a mean of 122 and a standard deviation of 4, and class C a mean of 277 and a standard deviation of 4. Test set 3 comprises 1050 samples and features class A with a mean of 15 and a standard deviation of 4, class B a mean of 133 and a standard deviation of 5, and class C a mean of 265 and a standard deviation of 5. The Test 1 dataset exhibits the characteristics of the concept formed after the concept drift occurred, while the Test 2 dataset reflects the characteristics of the concept before the concept drift. The Test 3 dataset does not align with either concept but is closer to the pre-drift concept than to the post-drift concept.

To evaluate the success of our system, a single model was trained using an 11-batch training dataset. The single model uses the same type of model as described in our system in the Section 3.5. The results of the single model are given in Table26:

Table 26: Performance metrics of single model approach in gradual drift experiment 2 where the gradual drift event occurs as an increase in classes A and B, and a decrease in class C.

	Test 1	Test 2	Test 3
Accuracy	0.7825	0.6844	0.9987
Recall	0.7825	0.6844	0.9987
Precision	0.7825	0.6844	0.9987

The system created using the training strategy described in the Section 3.3.1, combined with the prediction strategy outlined in Section 3.6.1, has been used to predict the test datasets that are defined above. The performance of our system is shown in Table27.

Table 27: Performance of the system for test sets that are used in gradual drift experiment 2 where the gradual drift event occurs as an increase in classes A and B, and a decrease in class C.

Metric	**Test 1**	**Test 2**	**Test 3**
Accuracy	0.9997	1.0000	1.0000
Recall	0.9997	1.0000	1.0000
Precision	0.9997	1.0000	1.0000

The training datasets used by our system were balanced using the method described in Section 3.4, and the system created with these balanced datasets was used to predict the test sets. The performance of the system created with the balanced datasets is shown in Table28.

By comparing Table26 and Table27, it can be observed that our system outperformed the single model in all test scenarios. Although our system significantly outperformed the single model approach on the Test 1 and Test 2 datasets, the single model approach managed to achieve a performance close to that of our system on the Test 3 dataset. As seen in Table28, in an experimental setup where the training

Table 28: Performance of the system that implemented training process with balanced datasets in gradual drift experiment 2 where the gradual drift event occurs as an increase in classes A and B, and a decrease in class C.

Metric	Test 1 (Balanced Datasets)	Test 2 (Balanced Datasets)	Test 3 (Balanced Datasets)
Accuracy	0.9997	0.6667	0.6825
Recall	0.9997	0.6667	0.6825
Precision	0.9997	0.6667	0.6825

datasets experienced gradual drift, with some classes increasing and others decreasing, a performance drop was observed in certain concepts.

Additionally, a system was created using the training strategy described in Section 3.3.2, and predictions were made using the method outlined in Section 3.6.1. The results are shown in Table29:

Table 29: Performance metrics of the system that used accuracy based training strategy on test sets which used in gradual drift experiment 2 where the gradual drift event occurs as an increase in classes A and B, and a decrease in class C.

Metric	Test 1	Test 2	Test 3
Accuracy	1.0000	1.0000	1.0000
Recall	1.0000	1.0000	1.0000
Precision	1.0000	1.0000	1.0000

After balancing the training datasets, predictions were made on the test sets using the same training and prediction strategies. The results are as given in Table30:

Table 30: Performance of the system that used accuracy based training strategy and implemented training process with balanced datasets on test sets which used in gradual drift experiment 2 where the gradual drift event occurs as an increase in classes A and B, and a decrease in class C.

Metric	Test 1 (Balanced Datasets)	Test 2 (Balanced Datasets)	Test 3 (Balanced Datasets)
Accuracy	1.0000	1.0000	1.0000
Recall	1.0000	1.0000	1.0000
Precision	1.0000	1.0000	1.0000

The accuracy-based approach and the method described in Section 3.3.1 performed similarly on the Test 2 and Test 3 datasets. Although the accuracy-based approach showed slightly better performance on the Test 1 dataset, the performances were quite close. On the other hand, when the datasets were balanced, the accuracy-based approach did not experience a performance loss, unlike the training strategy described in Section 3.3.1.

4.1.2.3 Gradual Drift Experiment 3

In Experiment 3, the occurrence of gradual drift in all classes in the dataset was simulated, with some classes experiencing an increase and others a decrease. Unlike other gradual drift experiments, one of the main challenges in this experiment is that the new concepts of two classes, which underwent an increase and a decrease after the concept drift, became the same or similar.

Figure31, Figure32, Figure33, and Figure34 provide a visual representation of the concept drift event in the dataset using boxplots. Figure20displays the overall data distribution, while Figure20, Figure21, and Figure22 focus specifically on the distributions within Class A, Class B, and Class C, respectively. These visualizations reveal how the patterns and general distribution of the data shift across batches before, and after the drift event.

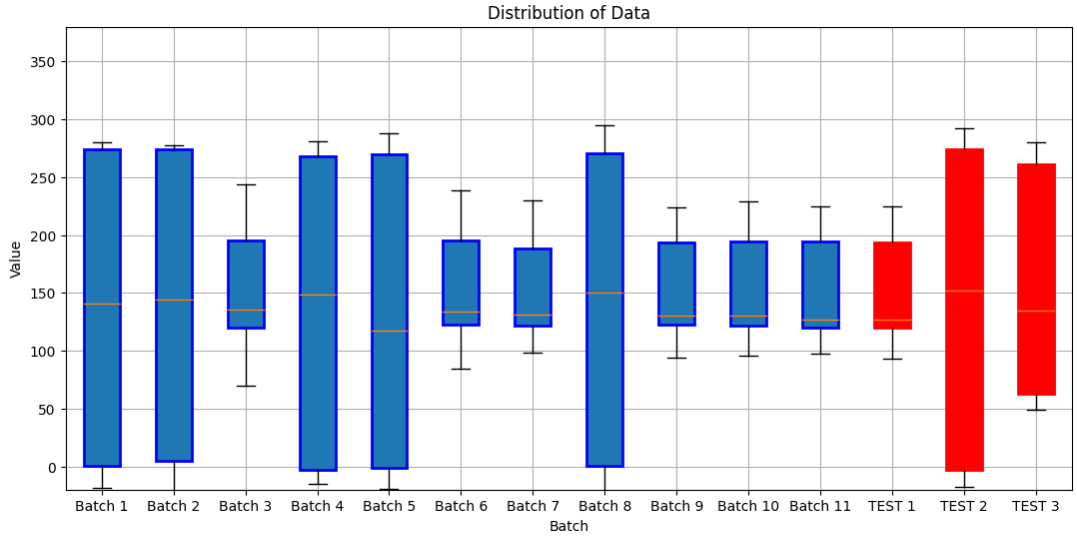


Figure 31: The Distribution of Data in Gradual Drift Experiment 3 Where the Gradual Drift Event Occurs as an Increase in Class A, and a Decrease in Classes B and C, and After the Concept Drift Event, Classes A and B Converge to the Similar Concept

The Table31 numerically presents characteristics such as the mean and standard deviation of the classes contained in the batches of the dataset.

The training dataset consists of 11 batches, each containing three features and various class distributions. Batch 1 includes 250 samples with class A having a mean of -3 and a standard deviation of 6, class B with a mean of 141 and a standard deviation of 3, and class C with a mean of 275 and a standard deviation of 2. Batch 2 has 259 samples, where class A has a mean of 0 and a standard deviation of 7, class B a mean of 144 and a standard deviation of 2, and class C a mean of 275 and a standard deviation of 1. Batch 3, with 90 samples, presents class A having a mean of 125 and a standard deviation of 18, class B a mean of 125 and a standard deviation of 16, and class C a mean of 200 and a standard deviation of 19. Batch 4 comprises 160 samples, featuring class A with a mean of -5 and a standard deviation of 4, class B a mean of 149 and a standard deviation of 4, and class C a mean of 270 and a standard deviation of 4. Batch 5 consists of 258 samples where class A has a mean of -5 and a standard deviation of 5, class B a mean of 118 and a standard deviation of 5, and class C

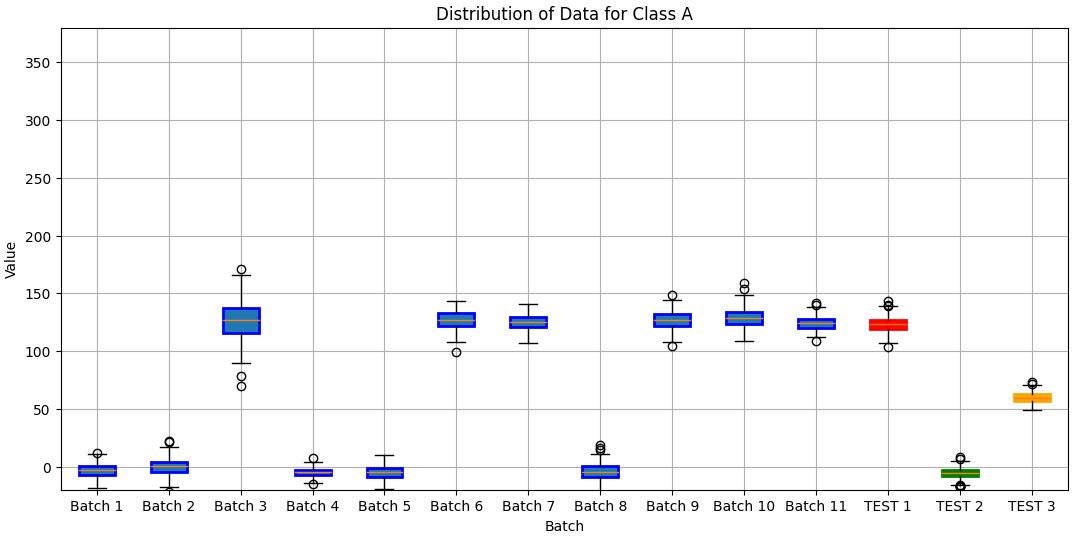


Figure 32: The Distribution of Class A Data in Gradual Drift Experiment 3 Where the Gradual Drift Event Occurs as an Increase in Class A, and a Decrease in Classes B and C, and After the Concept Drift Event, Classes A and B Converge to the Similar Concept

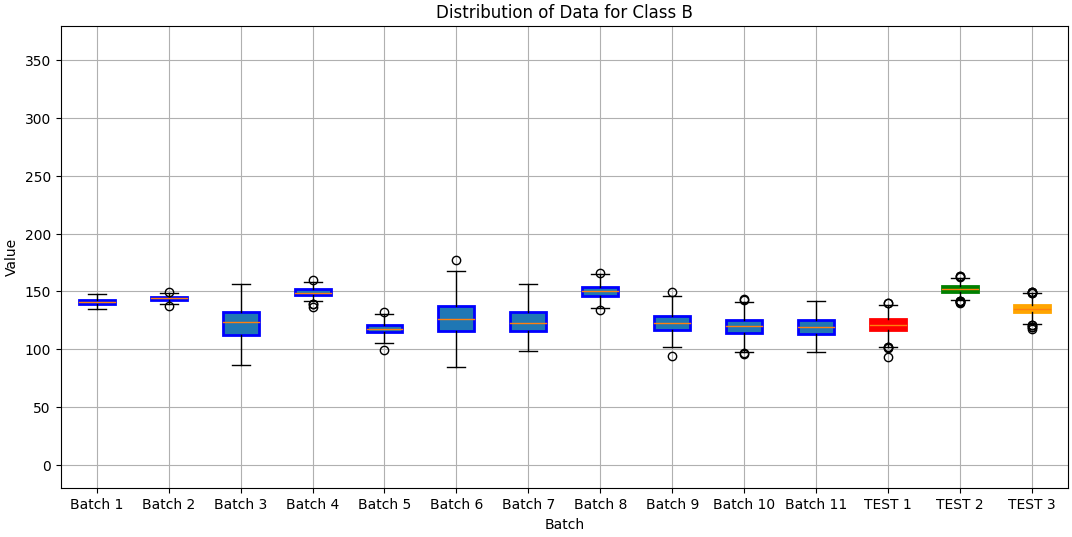


Figure 33: The Distribution of Class B Data in Gradual Drift Experiment 3 Where the Gradual Drift Event Occurs as an Increase in Class A, and a Decrease in Classes B and C, and After the Concept Drift Event, Classes A and B Converge to the Similar Concept

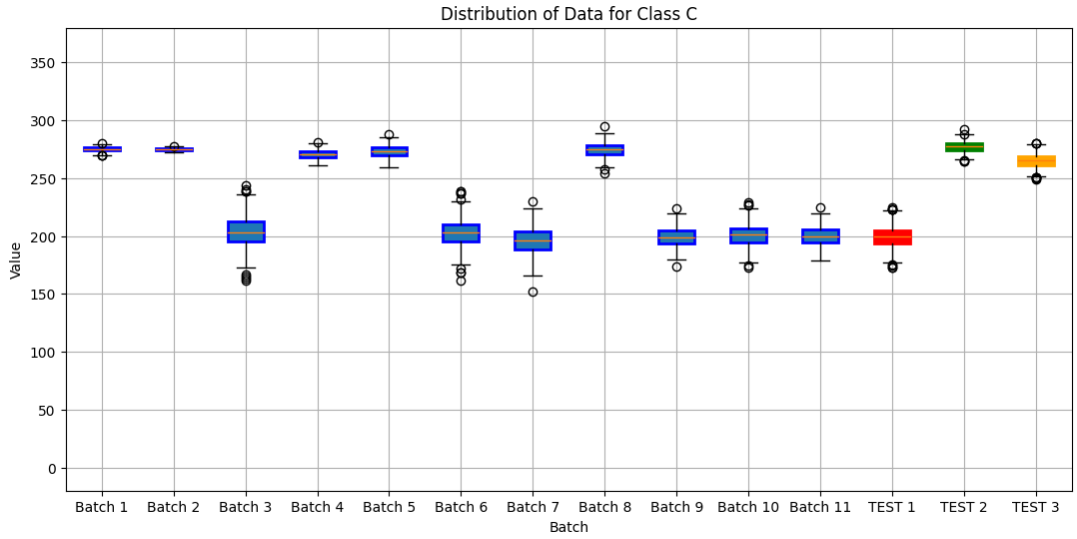


Figure 34: The Distribution of Class C Data in Gradual Drift Experiment 3 Where the Gradual Drift Event Occurs as an Increase in Class A, and a Decrease in Classes B and C, and After the Concept Drift Event, Classes A and B Converge to the Similar Concept

Table 31: Summary of dataset of gradual drift experiment 3 where the gradual drift event occurs as an increase in class A, and a decrease in classes B and C, and after the concept drift event, classes A and B converge to the similar concept.

Batch/Test Set	**Samples**	**Class A**		**Class B**		**Class C**		**Type**
		Mean (μ)	SD (σ)	Mean (μ)	SD (σ)	Mean (μ)	SD (σ)	
Batch 1	250	-3	6	141	3	275	2	Training
Batch 2	259	0	7	144	2	275	1	Training
Batch 3	90	125	18	125	16	200	19	Training
Batch 4	160	-5	4	149	4	270	4	Training
Batch 5	258	-5	5	118	5	273	5	Training
Batch 6	249	126	9	125	14	201	13	Training
Batch 7	150	125	7	124	13	195	13	Training
Batch 8	353	-4	7	150	6	275	6	Training
Batch 9	450	127	7	123	9	199	8	Training
Batch 10	548	128	8	120	9	201	9	Training
Batch 11	150	125	6	120	8	200	8	Training
Test Set 1	1050	123	6	121	7	200	8	Test
Test Set 2	1050	-5	4	152	4	277	4	Test
Test Set 3	1050	60	4	135	5	265	5	Test

a mean of 273 and a standard deviation of 5. Batch 6, containing 249 samples, includes class A with a mean of 126 and a standard deviation of 9, class B a mean of 125 and a standard deviation of 14, and class C a mean of 201 and a standard deviation of 13. Batch 7, with 150 samples, shows class A having a mean of 125 and a standard deviation of 7, class B a mean of 124 and a standard deviation of 13, and class C a mean of 195 and a standard deviation of 13. Batch 8, the largest batch with 353 samples, presents class A with a mean of -4 and a standard deviation of 7, class B a mean of 150 and a standard deviation of 6, and class C a mean of 275 and a standard deviation of 6. Batch 9 comprises 450 samples where class A has a mean of 127 and a standard deviation of 7, class B a mean of 123 and a standard deviation of 9, and class C a mean of 199 and a standard deviation of 8. Batch 10, consisting of 548 samples, includes class A with a mean of 128 and a standard deviation of 8, class B a mean of 120 and a standard deviation of 9, and class C a mean of 201 and a standard deviation of 9. Finally, Batch 11, with 150 samples, features class A with a mean of 125 and a standard deviation of 6, class B a mean of 120 and a standard deviation of 8, and class C a mean of 200 and a standard deviation of 8.

As illustrated in Figure31, the dataset undergoes concept drift in Batch 3. However, in Batch 4 and Batch 5, it reverts to the pre-drift concept. It then returns to the post-drift concept again, but two batches later, it reverts to the old concept once more, before finally undergoing another concept drift. After this final concept drift event, the new concept becomes permanent. This is a clear process of gradual drift.

As seen in Figure32 and Figure33, the concepts of Classes A and B became identical after the concept drift. In this experiment, in addition to the gradual drift event, measuring our system's performance when two different classes converge to the same concept will be one of the main events.

The test sets are designed to evaluate the model's performance. Test set 1 includes 1050 samples with class A having a mean of 123 and a standard deviation of 6, class B a mean of 121 and a standard deviation of 7, and class C a mean of 200 and a standard deviation of 8. Test set 2 also includes 1050 samples, with class A having a mean of -5 and a standard deviation of 4, class B a mean of 152 and a standard deviation of 4, and class C a mean of 277 and a standard deviation of 4. Test set 3, containing 1050 samples, has class A with a mean of 60 and a standard deviation of 4, class B a mean of 135 and a standard deviation of 5, and class C a mean of 265 and a standard deviation of 5. For Test 1 data, it can be considered that are in the same concept as the data before the concept drift occurred. For Test 2 data, it can be stated that are in the same concept as the data after the concept drift occurred. Test 3 data can be described as being in a concept that falls between the two different concepts in the training datasets, but it does not exist in the training set.

To evaluate the performance of our system, a single machine learning model was created using the entire training dataset from this experiment. The learner type used for the single model was selected to match the learners used by our system, as described in Section 3.5, to ensure a more objective comparison. The results of the single model approach are provided in Table32:

The results of our system, which uses the training strategy described in Section 3.3.1 and the prediction strategy described in Section 3.6.1, are shown in Table33:

The results of the system, which uses the same methods and is trained with the training datasets balanced using the method described in Section 3.4, are given in Table34:

Our system successfully predicted all instances in the Test 2 and Test 3 datasets correctly. Although it did not achieve the same level of success with the Test 1 dataset, its performance was still better

Table 32: Performance metrics of single model approach in gradual drift experiment 3 where the gradual drift event occurs as an increase in class A, and a decrease in classes B and C, and after the concept drift event, classes A and B converge to the similar concept.

	Test 1	Test 2	Test 3
Accuracy	0.6025	1.0	1.0
Recall	0.6025	1.0	1.0
Precision	0.6025	1.0	1.0

Table 33: Performance of the system for test sets that are used in gradual drift experiment 3 where the gradual drift event occurs as an increase in class A, and a decrease in classes B and C, and after the concept drift event, classes A and B converge to the similar concept.

Metric	**Test 1**	**Test 2**	**Test 3**
Accuracy	0.6667	1.0000	1.0000
Recall	0.6667	1.0000	1.0000
Precision	0.6667	1.0000	1.0000

Table 34: Performance of the system that implemented training process with balanced datasets in gradual drift experiment 3 where the gradual drift event occurs as an increase in class A, and a decrease in classes B and C, and after the concept drift event, classes A and B converge to the similar concept.

Metric	**Test 1 (Balanced Datasets)**	**Test 2 (Balanced Datasets)**	**Test 3 (Balanced Datasets)**
Accuracy	0.6667	1.0000	1.0000
Recall	0.6667	1.0000	1.0000
Precision	0.6667	1.0000	1.0000

than that of the single model approach. After balancing the training datasets, there was no increase or decrease in the performance of our system.

The results of our system, which uses the training strategy described in Section 3.3.2 and the prediction strategy described in Section 3.6.1, are seen in Table35.

Table 35: Performance metrics of the system that used accuracy based training strategy on test sets which used in gradual drift experiment 3 where the gradual drift event occurs as an increase in class A, and a decrease in classes B and C, and after the concept drift event, classes A and B converge to the similar concept.

Metric	**Test 1**	**Test 2**	**Test 3**
Accuracy	0.7194	0.3333	0.3333
Recall	0.7194	0.3333	0.3333
Precision	0.7194	0.3333	0.3333

The performance of the system using the same methods, with the training datasets balanced as described in Section 3.4, is as in Table36:

Table 36: Performance of the system that used accuracy based training strategy and implemented training process with balanced datasets on test sets which used in gradual drift experiment 3 where the gradual drift event occurs as an increase in class A, and a decrease in classes B and C, and after the concept drift event, classes A and B converge to the similar concept.

Metric	**Test 1 (Balanced Datasets)**	**Test 2 (Balanced Datasets)**	**Test 3 (Balanced Datasets)**
Accuracy	0.7194	0.3333	0.3333
Recall	0.7194	0.3333	0.3333
Precision	0.7194	0.3333	0.3333

Based on the results from Table35, the accuracy-based approach provided better predictions for the Test 1 dataset, which reflects the concept after the concept drift, compared to both the single model approach and the system using the training strategy described in Section 1. On the other hand, it performed quite poorly in predicting the Test 2 and Test 3 datasets. After balancing, there was no increase or decrease in its performance according to Table36.

4.1.3 Incremental Drift

In this subsection, experiments simulate the incremental drift scenario and their results are given.

Incremental drift describes a phenomenon where the underlying distribution of data progressively shifts over time. For incremental drift scenarios, the changes are generally subtle and accumulate slowly, but accumulated small changes have the potential to lead to significant concept shifts.

To fight with incremental drift, the training methods discussed in Section 3.3.1 and Section 3.3.2 were used, and the prediction approach described in Section 3.6.1 was experimented. The training methods utilized in the experiments will be also specified along with the characteristics of each experiment.

For incremental drift experiments, the number of instances in each batch was randomly determined. The number of batches was also randomly determined. After randomly determining the size of the test set, all test sets were generated with the same size to avoid bias arising from variations in test set size during comparisons of measurements.

The incremental drift scenario was tested with different nuances in different experiments. In the first experiment, an increasing concept change occurred across all classes, but the acceleration of the increase varied among different classes. Also, each class also has its own rate of increase, which varies between batches. In the second experiment, while some classes reflect increasing, others exhibited decreasing behavior. In the third experiment, some decreased, some classes increased, and additionally, some classes began to show the conceptual characteristics of each other, indicating a shift in their defining features.

4.1.3.1 Incremental Drift Experiment 1

In Experiment 1, incremental drift occurred across all classes where this progressive change manifested as an increase. Figure35, Figure36, Figure37, and Figure38 provide a visual representation of the concept drift event by using boxplots. Figure36displays the overall data distribution, while Figure36, Figure37, and Figure38 focus specifically on the distributions within Class A, Class B, and Class C, respectively. These visualizations reveal how the patterns and general distribution of the data shift across batches before, during, and after the concept drift event.

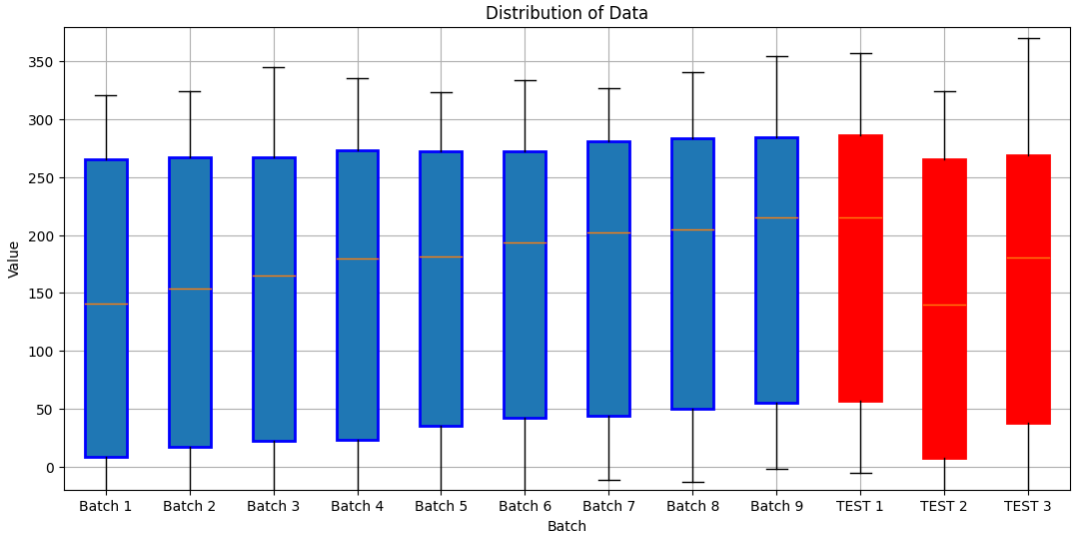


Figure 35: The Distribution of Data in Incremental Drift Experiment 1 Where The Incremental Drift Event Occurs As Increasing Manner Across All Three Classes

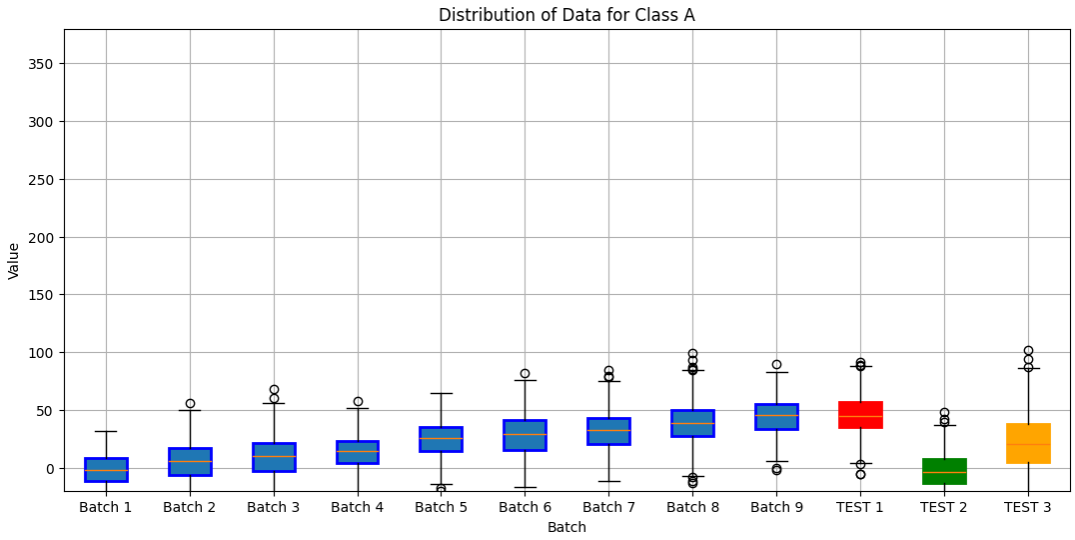


Figure 36: The Distribution of Class A Data in Incremental Drift Experiment 1 Where The Incremental Drift Event Occurs As Increasing Manner Across All Three Classes

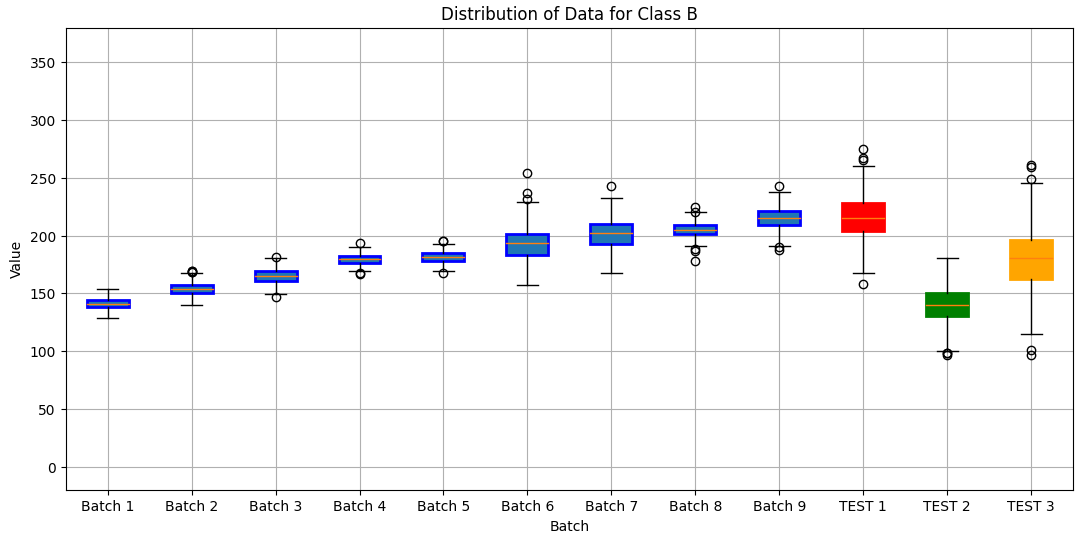


Figure 37: The Distribution of Class B Data in Incremental Drift Experiment 1 Where The Incremental Drift Event Occurs As Increasing Manner Across All Three Classes

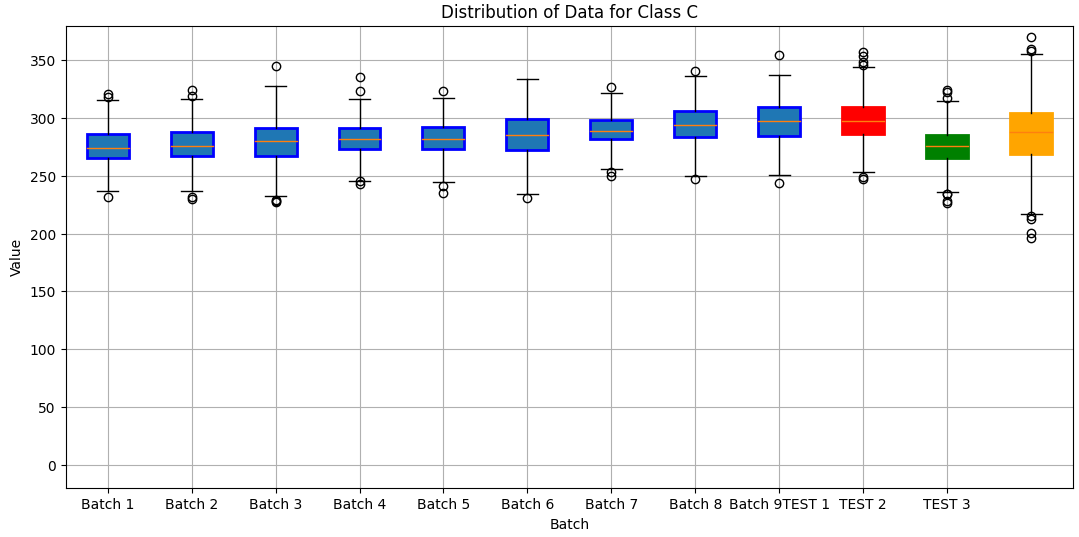


Figure 38: The Distribution of Class C Data in Incremental Drift Experiment 1 Where The Incremental Drift Event Occurs As Increasing Manner Across All Three Classes

Table37 displays statistical properties, including the mean and standard deviation, of the classes within the dataset batches.

The training dataset comprises nine distinct batches, each containing three features and various class distributions. Batch 1 includes 250 samples with class A having a mean of -3 and a standard deviation of 16, class B with a mean of 141 and a standard deviation of 5, and class C with a mean of 275 and a standard deviation of 15. Batch 2 contains 769 samples, where class A has a mean of 5 and a standard

Table 37: Summary of dataset of incremental drift experiment 1 where the incremental drift event occurs in an increasing manner across all three classes.

Batch/Test Set	**Samples**	**Class A**		**Class B**		**Class C**		**Type**
		Mean (μ)	SD (σ)	Mean (μ)	SD (σ)	Mean (μ)	SD (σ)	
Batch 1	250	-3	16	141	5	275	15	Training
Batch 2	769	5	17	154	5	277	15	Training
Batch 3	590	9	18	165	6	280	19	Training
Batch 4	560	15	14	179	4	282	14	Training
Batch 5	458	23	15	182	5	283	15	Training
Batch 6	349	30	19	195	14	285	19	Training
Batch 7	350	34	17	202	13	290	13	Training
Batch 8	953	38	17	205	6	295	16	Training
Batch 9	250	45	17	215	9	298	18	Training
Test Set 1	1050	45	16	215	17	298	18	Test
Test Set 2	1050	-3	16	141	15	275	15	Test
Test Set 3	1050	21	26	180	25	287	25	Test

deviation of 17, class B a mean of 154 and a standard deviation of 5, and class C a mean of 277 and a standard deviation of 15. Batch 3, with 590 samples, presents class A having a mean of 9 and a standard deviation of 18, class B a mean of 165 and a standard deviation of 6, and class C a mean of 280 and a standard deviation of 19. Batch 4 consists of 560 samples, featuring class A with a mean of 15 and a standard deviation of 14, class B a mean of 179 and a standard deviation of 4, and class C a mean of 282 and a standard deviation of 14. Batch 5 includes 458 samples where class A has a mean of 23 and a standard deviation of 15, class B a mean of 182 and a standard deviation of 5, and class C a mean of 283 and a standard deviation of 15. Batch 6, containing 349 samples, features class A with a mean of 30 and a standard deviation of 19, class B a mean of 195 and a standard deviation of 14, and class C a mean of 285 and a standard deviation of 19. Batch 7, with 350 samples, shows class A having a mean of 34 and a standard deviation of 17, class B a mean of 202 and a standard deviation of 13, and class C a mean of 290 and a standard deviation of 13. Batch 8, the largest batch with 953 samples, presents class A with a mean of 38 and a standard deviation of 17, class B a mean of 205 and a standard deviation of 6, and class C a mean of 295 and a standard deviation of 16. Batch 9, comprising 250 samples, includes class A having a mean of 45 and a standard deviation of 17, class B a mean of 215 and a standard deviation of 9, and class C a mean of 298 and a standard deviation of 18. As seen in the figures which are Figure35, Figure36,Figure37, and Figure38, the concept drift did not occur at a single point but rather through small, gradual changes over time. This reflects the fundamental characteristic of an incremental drift scenario.

The test sets are designed to evaluate the model's performance. Test set 1 includes 1050 samples with class A having a mean of 45 and a standard deviation of 16, class B a mean of 215 and a standard deviation of 17, and class C a mean of 298 and a standard deviation of 18. Test set 2, also with 1050 samples, shows class A having a mean of -3 and a standard deviation of 16, class B a mean of 141 and a standard deviation of 15, and class C a mean of 275 and a standard deviation of 15. Test set 3, containing 1050 samples, features class A with a mean of 21 and a standard deviation of 26, class B a mean of 180 and a standard deviation of 25, and class C a mean of 287 and a standard deviation of

25. In this context, the Test 1 dataset represents the concept formed by the accumulated small changes leading to a significant concept drift. Based on the characteristics of the data, it can be said that the Test 2 dataset represents the concept before any changes began. The Test 3 dataset can be interpreted as representing a concept during the process of these small changes occurring.

To compare the results of our system, a single model was trained using all the data from the training phase. The model is of the same type used by our system, as described in Section 3.5. The performance of the single model approach is provided in Table38:

Table 38: Performance metrics of single model approach in incremental drift experiment 1 where the incremental drift event occurs in an increasing manner across all three classes.

	Test 1	Test 2	Test 3
Accuracy	0.9978	0.9775	0.9997
Recall	0.9978	0.9775	0.9997
Precision	0.9978	0.9775	0.9997

The model trained using the training strategy from Section 3.3.1 made predictions using the prediction strategy from Section 3.6.1. It demonstrated the following performance as seen in Table39:

Table 39: Performance of the system for test sets that are used in incremental drift experiment 1 where the incremental drift event occurs in an increasing manner across all three classes.

Metric	**Test 1**	**Test 2**	**Test 3**
Accuracy	0.9981	1.0000	0.9997
Recall	0.9981	1.0000	0.9997
Precision	0.9981	1.0000	0.9997

The training datasets for the system using the prediction and training strategies mentioned above were balanced using the method described in Section 3.4. Its performance is seen in Table40:

Table 40: Performance of the system that implemented training process with balanced datasets in incremental drift experiment 1 where the incremental drift event occurs in an increasing manner across all three classes.

Metric	**Test 1 (Balanced Datasets)**	**Test 2 (Balanced Datasets)**	**Test 3 (Balanced Datasets)**
Accuracy	0.9981	1.0000	0.9997
Recall	0.9981	1.0000	0.9997
Precision	0.9981	1.0000	0.9997

Based on the results, although the difference is not very large, our system has been more successful than the single model approach in predicting Test 2. The performance for Test 1 and Test 3 was the same. Balancing the training datasets had no impact on performance.

The model trained using the training strategy from Section 3.3.2 made predictions using the prediction strategy from Section 3.6.1. It demonstrated the following performance as seen in Table41:

Table 41: Performance metrics of the system that used accuracy based training strategy on test sets which used in incremental drift experiment 1 where the incremental drift event occurs in an increasing manner across all three classes.

Metric	**Test 1**	**Test 2**	**Test 3**
Accuracy	0.9981	1.0000	1.0000
Recall	0.9981	1.0000	1.0000
Precision	0.9981	1.0000	1.0000

The training datasets for the system using the prediction and training strategies described above were balanced using the method described in Section 3.4. Its performance is presented in Table 42 as follows:

Table 42: Performance of the system that used accuracy based training strategy and implemented training process with balanced datasets on test sets which used in incremental drift experiment 1 where the incremental drift event occurs in an increasing manner across all three classes.

Metric	**Test 1 (Balanced Datasets)**	**Test 2 (Balanced Datasets)**	**Test 3 (Balanced Datasets)**
Accuracy	0.9981	1.0000	1.0000
Recall	0.9981	1.0000	1.0000
Precision	0.9981	1.0000	1.0000

The accuracy-based approach produced the same results as the training approach described in Section 3.3.1 for Test 1 and Test 2. On the other hand, the accuracy-based approach performed slightly better on Test 3. Using the balancing method on the training datasets did not change the performance of the system which uses training method was mentioned in Section 3.3.2.

4.1.3.2 Incremental Drift Experiment 2

In this experiment, an incremental drift scenario will be simulated in which the concept characteristics of some classes increase and those of other classes decrease. The concept drift event in the dataset is visualized in Figure 39, Figure 40, Figure 41, and Figure 42 using boxplots. Figure 39 provides an overview of the overall data distribution, while Figure 40, Figure 41, and Figure 42 delve into the specific distributions of Class A, Class B, and Class C, respectively, showcasing how data patterns change across batches during and after the drift event.

Table43 summarizes the numerical attributes, such as mean and standard deviation, of the classes across the dataset batches.

The training part of dataset comprises nine distinct batches, each containing three features and various class distributions. Batch 1 includes 250 samples, with class A having a mean of -3 and a standard

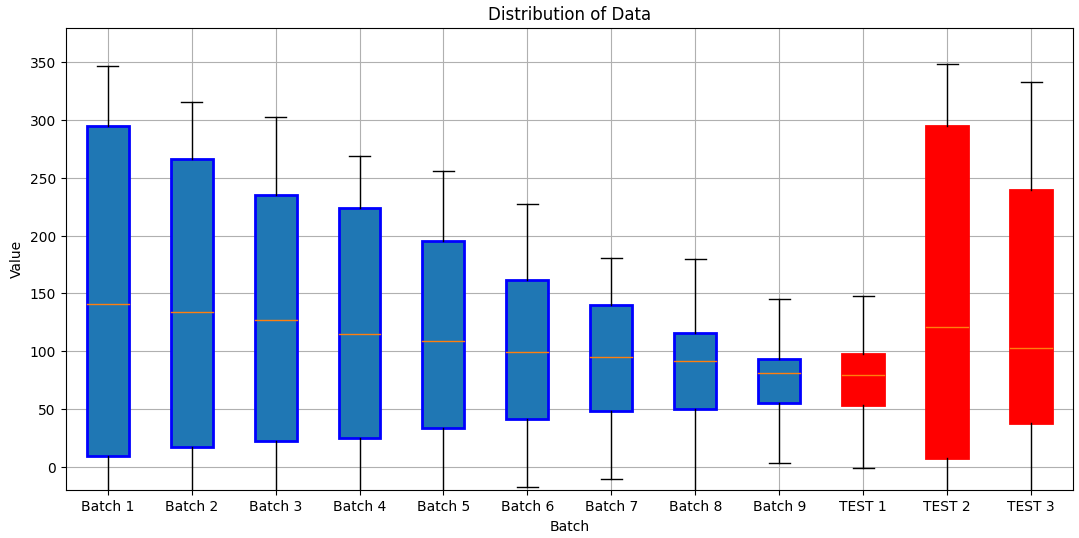


Figure 39: The Distribution of Data in Incremental Drift Experiment 2 Where the Incremental Drift Event Occurs as an Increase in Class A, and a Decrease in Classes B and C

deviation of 16, class B with a mean of 141 and a standard deviation of 5, and class C with a mean of 305 and a standard deviation of 15. Batch 2 contains 769 samples, where class A has a mean of 5 and a standard deviation of 17, class B a mean of 134 and a standard deviation of 5, and class C a mean of 277 and a standard deviation of 15. Batch 3, with 590 samples, presents class A having a mean of 9 and a standard deviation of 18, class B a mean of 127 and a standard deviation of 6, and class C a mean of 250 and a standard deviation of 19. Batch 4 consists of 560 samples, featuring class A with a mean of 15 and a standard deviation of 14, class B a mean of 115 and a standard deviation of 4, and

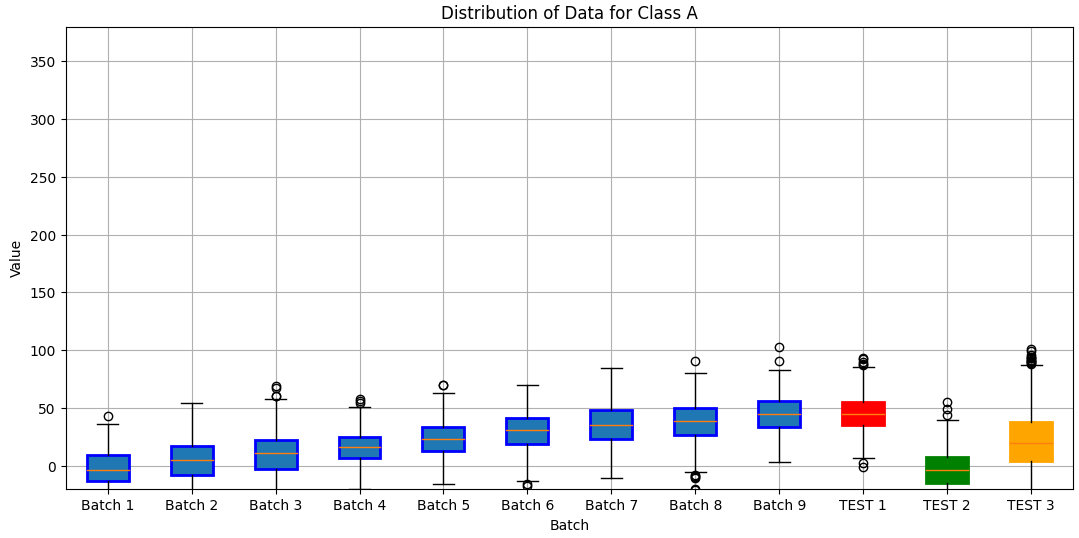


Figure 40: The Distribution of Class A Data in Incremental Drift Experiment 2 Where the Incremental Drift Event Occurs as an Increase in Class A, and a Decrease in Classes B and C

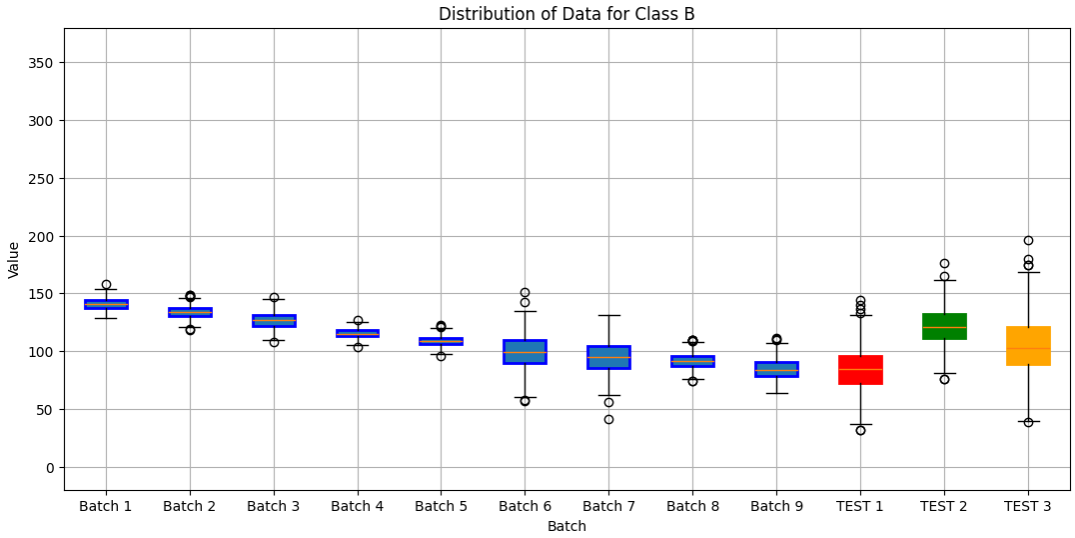


Figure 41: The Distribution of Class B Data in Incremental Drift Experiment 2 Where the Incremental Drift Event Occurs as an Increase in Class A, and a Decrease in Classes B and C

class C a mean of 232 and a standard deviation of 14. Batch 5 includes 458 samples where class A has a mean of 23 and a standard deviation of 15, class B a mean of 109 and a standard deviation of 5, and class C a mean of 206 and a standard deviation of 15. Batch 6, containing 349 samples, features class A with a mean of 30 and a standard deviation of 19, class B a mean of 100 and a standard deviation of 14, and class C a mean of 175 and a standard deviation of 19. Batch 7, with 350 samples, shows class A having a mean of 34 and a standard deviation of 17, class B a mean of 95 and a standard deviation of 13, and class C a mean of 150 and a standard deviation of 13. Batch 8, the largest batch with 953

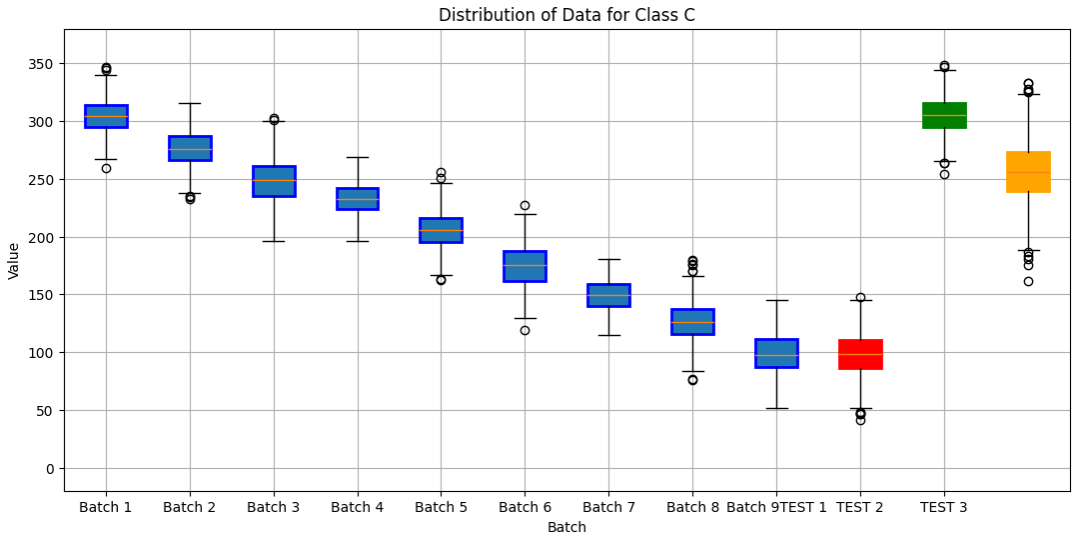


Figure 42: The Distribution of Class C Data in Incremental Drift Experiment 2 Where the Incremental Drift Event Occurs as an Increase in Class A, and a Decrease in Classes B and C

Table 43: Summary of dataset of incremental drift experiment 2 where the incremental drift event occurs as an increase in class A, and a decrease in classes B and C.

Batch/Test Set	**Samples**	**Class A**		**Class B**		**Class C**		**Type**
		Mean (μ)	SD (σ)	Mean (μ)	SD (σ)	Mean (μ)	SD (σ)	
Batch 1	250	-3	16	141	5	275	15	Training
Batch 2	769	5	17	154	5	277	15	Training
Batch 3	590	9	18	165	6	280	19	Training
Batch 4	560	15	14	179	4	282	14	Training
Batch 5	458	23	15	182	5	283	15	Training
Batch 6	349	30	19	195	14	285	19	Training
Batch 7	350	34	17	202	13	290	13	Training
Batch 8	953	38	17	205	6	295	16	Training
Batch 9	250	45	17	215	9	298	18	Training
Test Set 1	1050	45	16	215	17	298	18	Test
Test Set 2	1050	-3	16	141	15	275	15	Test
Test Set 3	1050	21	26	180	25	287	25	Test

samples, presents class A with a mean of 38 and a standard deviation of 17, class B a mean of 92 and a standard deviation of 6, and class C a mean of 126 and a standard deviation of 16. Batch 9, comprising 250 samples, includes class A having a mean of 45 and a standard deviation of 17, class B with a mean of 84 and a standard deviation of 9, and class C with a mean of 99 and a standard deviation of 18. As seen in Figure39, concept drift was not occurred in certain point. The concept drift event was happened as process that consists stages of change.

Test set 1 includes 1050 samples, with class A having a mean of 45 and a standard deviation of 16, class B with a mean of 84 and a standard deviation of 17, and class C with a mean of 99 and a standard deviation of 18. Test set 2, also with 1050 samples, shows class A having a mean of -3 and a standard deviation of 16, class B with a mean of 121 and a standard deviation of 15, and class C with a mean of 305 and a standard deviation of 15. Test set 3, containing 1050 samples, features class A with a mean of 21 and a standard deviation of 26, class B with a mean of 105 and a standard deviation of 25, and class C with a mean of 257 and a standard deviation of 25. Test 1 dataset reflects the concepts of the classes after the concept drift has occurred. Test 2 dataset contains data corresponding to the concepts that the classes had before the concept drift. Test 3 dataset does not align with the two different concepts present in the training datasets, but it shows characteristics of a concept that falls between these two concepts that are exists in training data.

To compare with the system result, a single model was trained by dataset consist of 9 training groups. To make an objective comparison with the system results, the model used by the system described in Section 3.5 was used to generate a single model. The performance of the single model approach is seen in Table44:

Table 44: Performance metrics of single model approach in incremental drift experiment 2 where the incremental drift event occurs as an increase in class A, and a decrease in classes B and C.

	Test 1	Test 2	Test 3
Accuracy	0.6425	0.9930	0.9952
Recall	0.6425	0.9930	0.9952
Precision	0.6425	0.9930	0.9952

The training strategy defined in Section 3.3.1 was used for training process and strategy discussed in Section 3.6.1 for predicting test sets. After the training process was completed, in addition to making predictions, all the collected training data were balanced using the method specified in the Section 3.4, and the models were trained and predictions made using these balanced datasets. The result of these implementation is given in Table45 and Table46:

Table 45: Performance of the system for test sets that are used in incremental drift experiment 2 where the incremental drift event occurs as an increase in class A, and a decrease in classes B and C.

Metric	**Test 1**	**Test 2**	**Test 3**
Accuracy	0.7708	0.9994	0.9981
Recall	0.7708	0.9994	0.9981
Precision	0.7708	0.9994	0.9981

Table 46: Performance of the system that implemented training process with balanced datasets in incremental drift experiment 2 where the incremental drift event occurs as an increase in class A, and a decrease in classes B and C.

Metric	**Test 1 (Balanced Datasets)**	**Test 2 (Balanced Datasets)**	**Test 3 (Balanced Datasets)**
Accuracy	0.7708	0.9990	0.9981
Recall	0.7708	0.9990	0.9981
Precision	0.7708	0.9990	0.9981

Based on the results, although the performances on Test 2 and Test 3 are close, our system has achieved better results than the single model approach on all test sets. The performance on Test 1 is significantly better compared to the single model approach. Balancing the training datasets in our system led to a very slight performance loss on the Test 2 dataset, but it did not create a significant change.

For the training process, the training method described in the Section 3.3.2 was applied, and the method described in the Section 3.6.1 was used to predict the test sets. After getting results of this implementation, all stored training data were balanced using the method described in the Section 3.4, and the models were trained and predictions were made on these balanced datasets. The result of this application is shown in Table47 and Table48:

Table 47: Performance metrics of the system that used accuracy based training strategy on test sets which used in incremental drift experiment 2 where the incremental drift event occurs as an increase in class A, and a decrease in classes B and C.

Metric	Test 1	Test 2	Test 3
Accuracy	0.8184	1.0000	0.9997
Recall	0.8184	1.0000	0.9997
Precision	0.8184	1.0000	0.9997

Table 48: Performance of the system that used accuracy based training strategy and implemented training process with balanced datasets on test sets which used in incremental drift experiment 2 where the incremental drift event occurs as an increase in class A, and a decrease in classes B and C.

Metric	Test 1 (Balanced Datasets)	Test 2 (Balanced Datasets)	Test 3 (Balanced Datasets)
Accuracy	0.8184	1.0000	0.9997
Recall	0.8184	1.0000	0.9997
Precision	0.8184	1.0000	0.9997

Based on the results, the system using the accuracy-based training strategy has achieved better results on all test sets compared to both the single model approach and the system using the training strategy described in Section 3.3.1. While the performance improvement on Test 2 and Test 3 is minimal, the performance on Test 1 is significantly better compared to the others. Balancing the training datasets for the system using the accuracy-based training strategy did not have any impact on its performance.

4.1.3.3 Incremental Drift Experiment 3

In this experiment, in addition to the characteristics of some classes increasing while those of other classes decrease, the concepts of the classes become similar after the concept drift occurs. Figure43, Figure44, Figure45, and Figure46 visually represent the dataset's concept drift event by using boxplots. While Figure43 illustrates the overall data distribution, Figure44, Figure45, and Figure46 specifically display the distributions for Class A, Class B, and Class C, respectively, highlighting shifts in data patterns before and after the drift event.

Table49 summarizes the numerical attributes, such as mean and standard deviation, of the classes across batches of the dataset.

The training dataset consists nine distinct batches, each containing three features and various class distributions. In Batch 1, which includes 250 samples, class A has a mean of -8 and a standard deviation of 16, class B has a mean of 141 and a standard deviation of 5, and class C has a mean of 305 and a standard deviation of 15. Batch 2 contains 769 samples, where class A has a mean of 5 and a standard deviation of 17, class B has a mean of 134 and a standard deviation of 5, and class C has a mean of 277 and a standard deviation of 15. Batch 3 comprises 590 samples, presenting class A with a mean of 19 and a standard deviation of 18, class B with a mean of 127 and a standard deviation of 6, and class C with a mean of 250 and a standard deviation of 19. Batch 4 includes 560 samples,

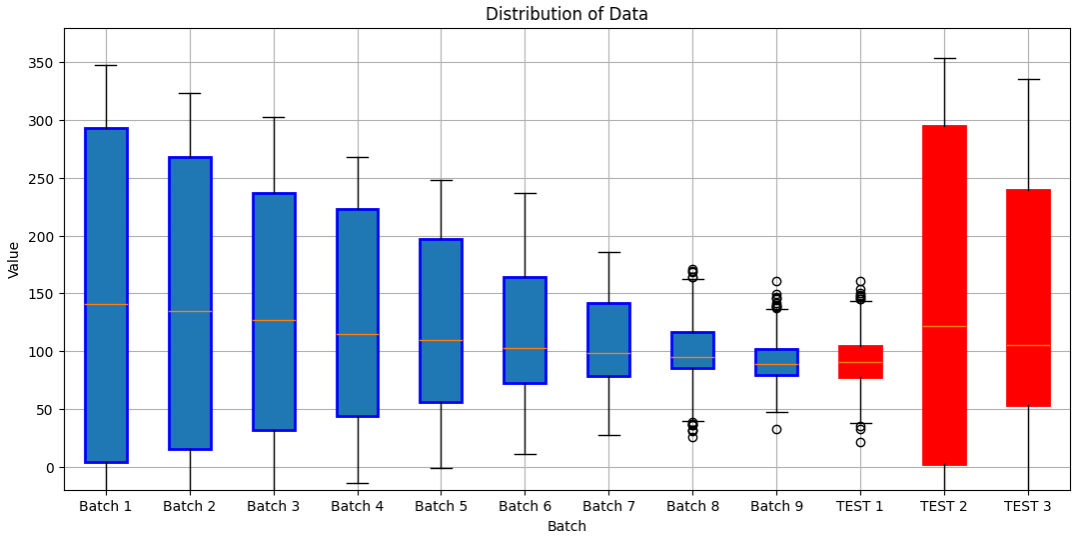


Figure 43: The Distribution of Data in Incremental Drift Experiment 3 Where the Incremental Drift Event Occurs as an Increase in Classes A and B, and a Decrease in Class C, and After the Concept Drift Event, Classes A and B Converge to the Same Concept

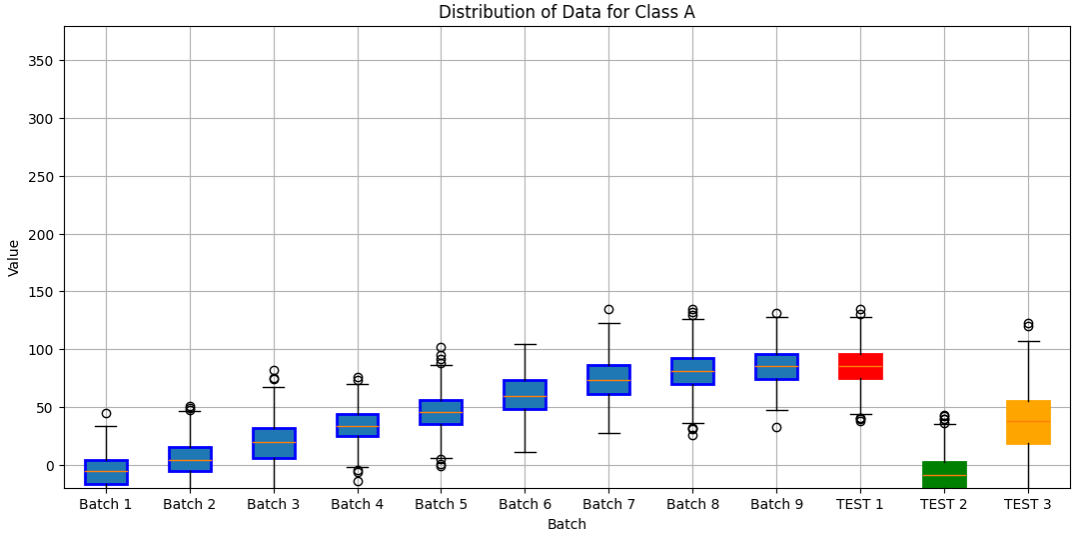


Figure 44: The Distribution of Class A Data in Incremental Drift Experiment 3 Where the Incremental Drift Event Occurs as an Increase in Classes A and B, and a Decrease in Class C, and After the Concept Drift Event, Classes A and B Converge to the Same Concept

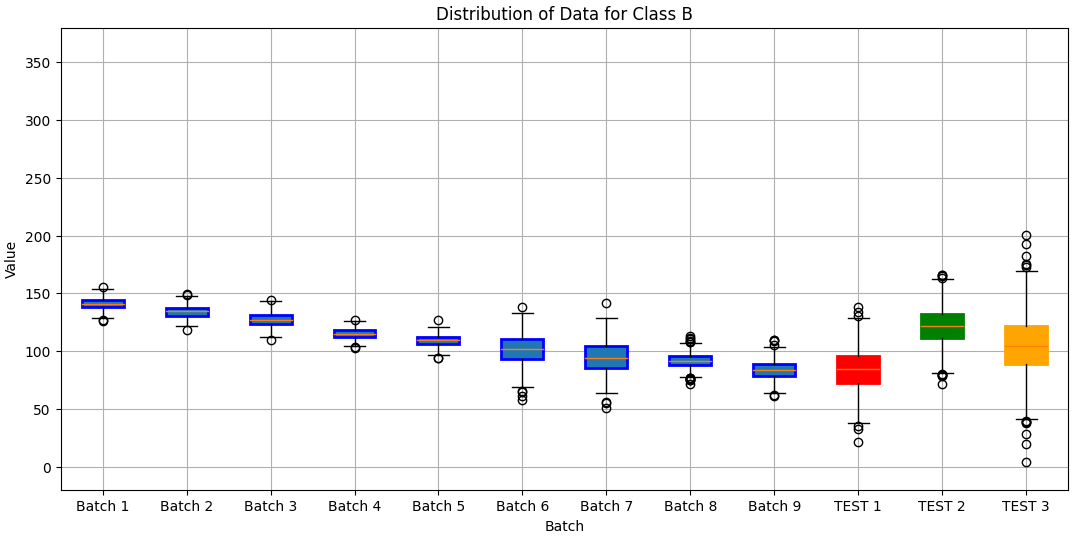


Figure 45: The Distribution of Class B Data in Incremental Drift Experiment 3 Where the Incremental Drift Event Occurs as an Increase in Classes A and B, and a Decrease in Class C, and After the Concept Drift Event, Classes A and B Converge to the Same Concept

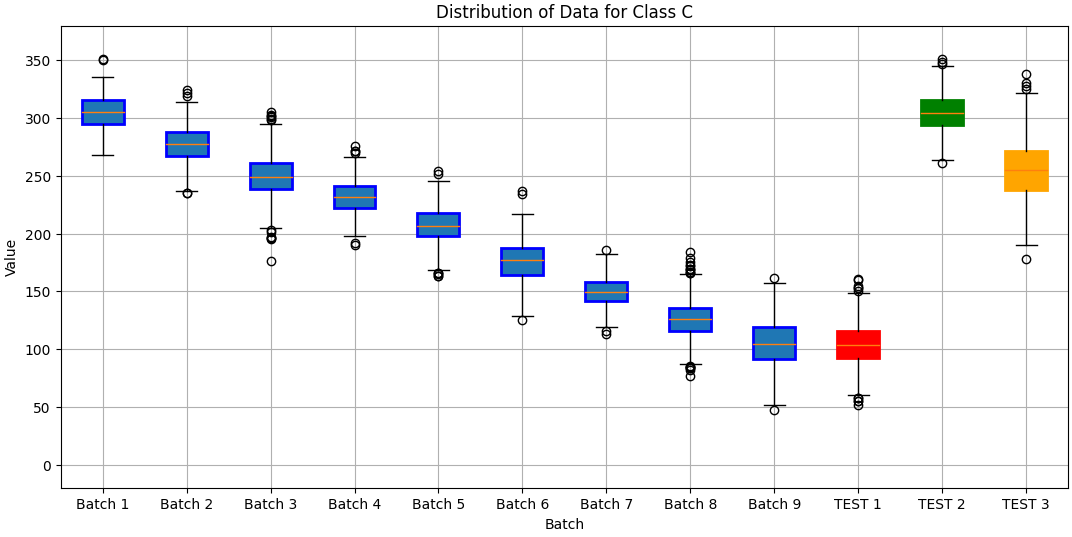


Figure 46: The Distribution of Class C Data in Incremental Drift Experiment 3 Where the Incremental Drift Event Occurs as an Increase in Classes A and B, and a Decrease in Class C, and After the Concept Drift Event, Classes A and B Converge to the Same Concept

Table 49: Summary of dataset of incremental drift experiment 3 where the Incremental drift event occurs as an increase in classes A and B, and a decrease in class C, and after the concept drift event, classes A and B converge to the same concept.

Batch/Test Set	**Samples**	**Class A**		**Class B**		**Class C**		**Type**
		Mean (μ)	SD (σ)	Mean (μ)	SD (σ)	Mean (μ)	SD (σ)	
Batch 1	250	-8	16	141	5	305	15	Training
Batch 2	769	5	17	134	5	277	15	Training
Batch 3	590	19	18	127	6	250	19	Training
Batch 4	560	35	14	115	4	232	14	Training
Batch 5	458	46	15	109	5	206	15	Training
Batch 6	349	60	19	100	14	175	19	Training
Batch 7	350	71	17	95	13	150	13	Training
Batch 8	953	80	17	92	6	126	16	Training
Batch 9	250	85	17	84	9	105	18	Training
Test Set 1	1050	85	16	84	17	105	18	Test
Test Set 2	1050	-8	16	121	15	305	15	Test
Test Set 3	1050	36	26	105	25	257	25	Test

featuring class A with a mean of 35 and a standard deviation of 14, class B with a mean of 115 and a standard deviation of 4, and class C with a mean of 232 and a standard deviation of 14. Batch 5 consists of 458 samples, where class A has a mean of 46 and a standard deviation of 15, class B with a mean of 109 and a standard deviation of 5, and class C with a mean of 206 and a standard deviation of 15. Batch 6 contains 349 samples, featuring class A with a mean of 60 and a standard deviation of 19, class B with a mean of 100 and a standard deviation of 14, and class C with a mean of 175 and a standard deviation of 19. Batch 7, with 350 samples, shows class A having a mean of 71 and a standard deviation of 17, class B with a mean of 95 and a standard deviation of 13, and class C with a mean of 150 and a standard deviation of 13. Batch 8, the largest batch with 953 samples, presents class A with a mean of 80 and a standard deviation of 17, class B with a mean of 92 and a standard deviation of 6, and class C with a mean of 126 and a standard deviation of 16. Batch 9, comprising 250 samples, includes class A having a mean of 85 and a standard deviation of 17, class B with a mean of 84 and a standard deviation of 9, and class C with a mean of 105 and a standard deviation of 18.

As illustrated in Figure44 and Figure45, Class A is incrementally increasing while Class B is incrementally decreasing. As a result of this concept drift, they gradually approach each other's concepts and eventually exhibit the same concept characteristics. Class C is also incrementally decreasing and approaches Classes A and B but does not reach the same concept according to Figure46.

Test set 1 includes 1050 samples, with class A having a mean of 85 and a standard deviation of 16, class B with a mean of 84 and a standard deviation of 17, and class C with a mean of 105 and a standard deviation of 18. Test set 2, also with 1050 samples, shows class A having a mean of -8 and a standard deviation of 16, class B with a mean of 121 and a standard deviation of 15, and class C with a mean of 305 and a standard deviation of 15. Test set 3, containing 1050 samples, features class A with a mean of 36 and a standard deviation of 26, class B with a mean of 105 and a standard deviation of 25, and class C with a mean of 257 and a standard deviation of 25. For Test 1 data, it can be assumed that are

in the same concept as the data before the concept drift occurred. For Test 2 data, it can be considered that are in the same concept as the data after the concept drift occurred. Test 3 data can be described as being in a concept that falls between the two different concepts in the training datasets, but it does not directly exist in the training set.

To evaluate the success of our system, a single model was trained using an 9-batch training dataset. The single model uses the same type of model as described in our system in the Section 3.5. The results of the single model are given in Table50:

Table 50: Performance metrics of single model approach in incremental drift experiment 3 where the Incremental drift event occurs as an increase in classes A and B, and a decrease in class C, and after the concept drift event, classes A and B converge to the same concept.

	Test 1	Test 2	Test 3
Accuracy	0.3197	0.9873	0.9622
Recall	0.3197	0.9873	0.9622
Precision	0.3197	0.9873	0.9622

The system created using the training strategy described in the Section 3.3.1, combined with the prediction strategy that was described in Section 3.6.1, has been used to predict the test datasets that are defined above. The performance of our system is shown in Table51:

Table 51: Performance of the system for test sets that are used in incremental drift experiment 3 where the Incremental drift event occurs as an increase in classes A and B, and a decrease in class C, and after the concept drift event, classes A and B converge to the same concept.

Metric	**Test 1**	**Test 2**	**Test 3**
Accuracy	0.4419	1.0000	0.9781
Recall	0.4419	1.0000	0.9781
Precision	0.4419	1.0000	0.9781

Our system outperformed the single model approach across all test datasets as seen in Table51. Both the single model approach and our system performed well on the Test 2 and Test 3 datasets. However, neither the system nor the single model approach achieved desired performance levels on the Test 1 dataset. Nevertheless, our system achieved significantly better results on the Test 1 dataset compared to the single model approach.

The training datasets used by our system were balanced using the method mentioned in Section 3.4, and the system developed with these balanced datasets was used to predict the test sets. The performance of the system created with the balanced datasets is shown below:

Table 52: Performance of the system that implemented training process with balanced datasets in incremental drift experiment 3 where the Incremental drift event occurs as an increase in classes A and B, and a decrease in class C, and after the concept drift event, classes A and B converge to the same concept.

Metric	Test 1 (Balanced Datasets)	Test 2 (Balanced Datasets)	Test 3 (Balanced Datasets)
Accuracy	0.4419	1.0000	0.9781
Recall	0.4419	1.0000	0.9781
Precision	0.4419	1.0000	0.9781

Balancing the training datasets had no impact on the performance of our system using the training strategy from Section 3.3.1 in this experiment as seen in Table52.

Additionally, a system was created using the training strategy described in Section 3.3.2, and predictions were made using the method outlined in Section 3.6.1. The results are shown below:

Table 53: Performance metrics of the system that used accuracy based training strategy on test sets which used in incremental drift experiment 3 where the Incremental drift event occurs as an increase in classes A and B, and a decrease in class C, and after the concept drift event, classes A and B converge to the same concept.

Metric	Test 1	Test 2	Test 3
Accuracy	0.5816	1.0000	0.9870
Recall	0.5816	1.0000	0.9870
Precision	0.5816	1.0000	0.9870

The system using the accuracy-based training strategy outperformed both our system using the training strategy from Section 2 and the single model approach across all three test datasets as seen in Table53. Additionally, while its performance on the Test 1 dataset, which had previously not shown good results with other approaches, is promising, it is still not sufficient.

After balancing the training datasets, predictions were made on the test sets using the same training and prediction strategies. The results are as follows:

Table 54: Performance of the system that used accuracy based training strategy and implemented training process with balanced datasets on test sets which used in incremental drift experiment 3 where the Incremental drift event occurs as an increase in classes A and B, and a decrease in class C, and after the concept drift event, classes A and B converge to the same concept.

Metric	Test 1 (Balanced Datasets)	Test 2 (Balanced Datasets)	Test 3 (Balanced Datasets)
Accuracy	0.5816	1.0000	0.9870
Recall	0.5816	1.0000	0.9870
Precision	0.5816	1.0000	0.9870

Balancing the training datasets had no impact on the performance of our system using the training strategy from Section 3.3.2 in this experiment as seen in Table54.

4.1.4 Reoccurring Drift

Reoccurring drift, is also known as recurring concept drift, as a sub category of concept drift, refers to the phenomenon where the statistical properties of the variables, change over time and then later return to their original state. This type of drift can be particularly challenging because it suggests a cyclical or periodic change in the concepts or patterns within the data.

In a typical reoccurring drift scenario, there is a initial data where certain patterns and relationships between input variables and the target variable are established. Over time, these relationships may change due to factors like seasonal effects, user behavior changes, or market dynamics, leading to a degradation in performance of the model which learned initial data because it was trained under different initial conditions. However, these original conditions can reoccur, causing the patterns and relationships to revert to their initial state, making the previously trained model relevant again. This cycle of change requiring the model to adapt dynamically. Our study aims to operate a system that can adapt automatically this nature of reoccurring drift.

In recurring drift experiments, the number of batches was randomly determined, and a random number of instances was generated for each batch. After the size of the test set was randomly determined, all test sets were generated with the same size because bias due to the test set is not desired to fair evaluation.

In the first experiment, all classes will undergo concept drift such that their concept characteristics increase, and then they will revert to their original concepts. In the second experiment, while some classes will experience concept drift with increasing concept characteristics, other classes will have decreasing concept characteristics. In accordance with the reoccurring drift scenario, the classes will also return to their initial concepts in the second experiment. In the third experiment, in addition to the challenge of some classes experiencing an increase in concept characteristics and others a decrease, the classes will start to exhibit the same concept characteristics after the concept drift. The specific characteristics of each experiment will be detailed in the sections related to the respective experiments.

4.1.4.1 Reoccurring Drift Experiment 1

In this experiment design, the variables defining the concepts of all classes undergo concept drift in an increasing manner. After the concept drift occurs, the concepts of the classes revert back to their initial concepts. This situation is sufficient event for classifying the observed concept drift as a reoccurring drift. Through boxplots, Figure47, Figure48, Figure49, and Figure50 represent the concept drift phenomenon in the dataset by highlighting distribution changes across batches. Figure 47 displays the overall data distribution, while Figure48, Figure49, and Figure50 focus on Class A, Class B, and Class C, respectively.

The dataset comprises ten distinct batches, each containing three features and various class distributions. Table55 displays statistical properties, including the mean and standard deviation, of the classes within the dataset batches.

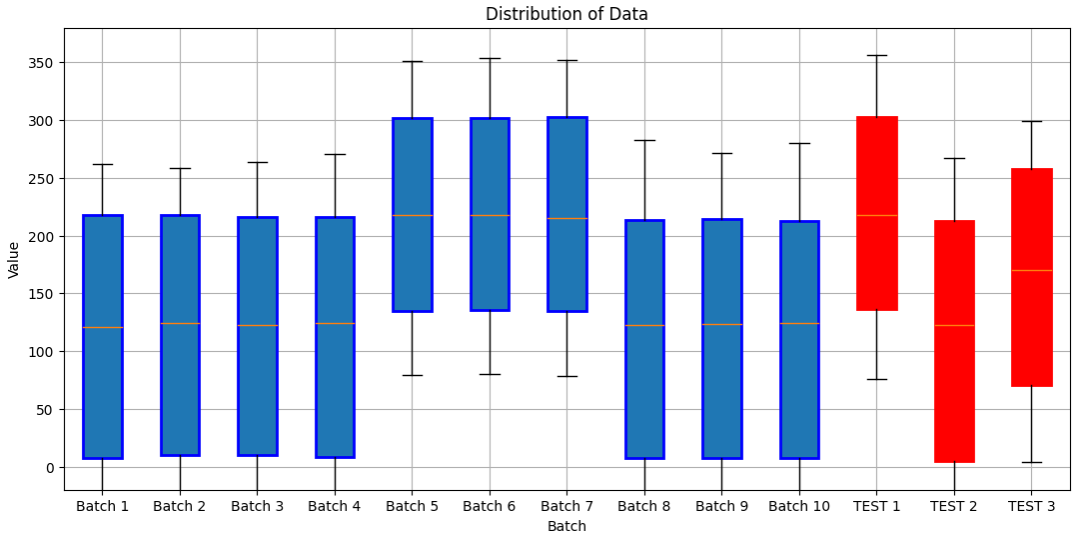


Figure 47: The Distribution of Data in Reoccurring Drift Experiment 1 Where The Reoccurring Drift Event Occurs As Increasing Manner Across All Three Classes

In Batch 1, which includes 1250 samples, class A has a mean of -3 and a standard deviation of 16, class B has a mean of 121 and a standard deviation of 13, and class C has a mean of 225 and a standard deviation of 12. Batch 2 contains 1259 samples, where class A has a mean of 0 and a standard deviation of 17, class B has a mean of 124 and a standard deviation of 12, and class C has a mean of 225 and a standard deviation of 11. Batch 3 comprises 990 samples, presenting class A with a mean of -1 and a standard deviation of 17, class B with a mean of 123 and a standard deviation of 12, and class C with a mean of 224 and a standard deviation of 11. Batch 4 includes 1160 samples, featuring class A with a

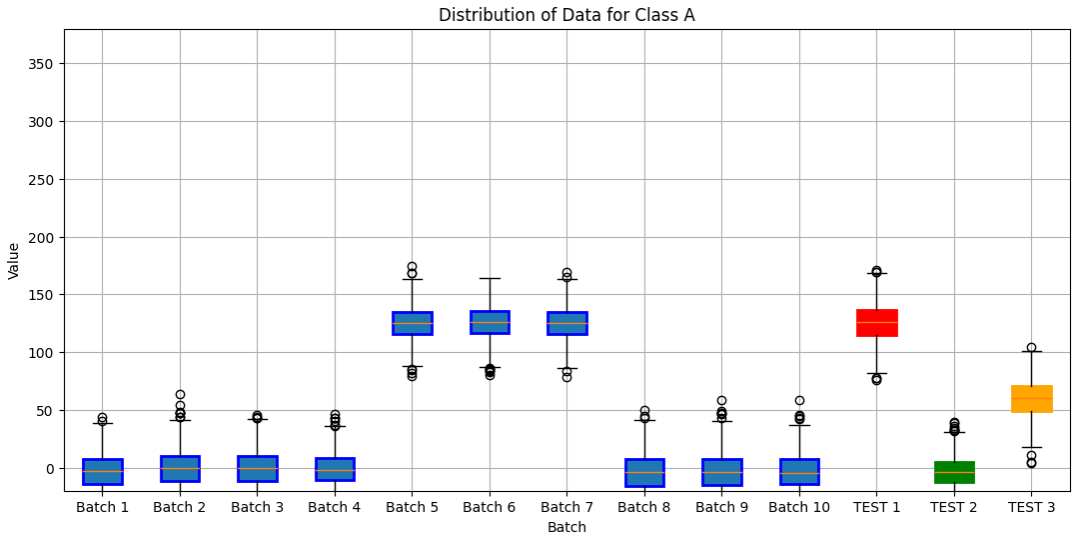


Figure 48: The Distribution of Class A Data in Reoccurring Drift Experiment 1 Where The Reoccurring Drift Event Occurs As Increasing Manner Across All Three Classes

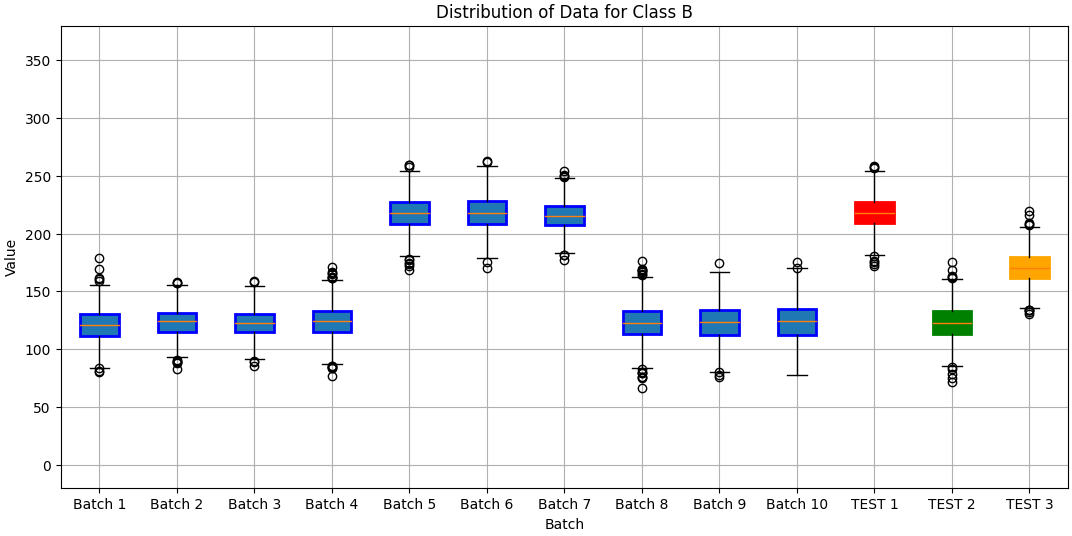


Figure 49: The Distribution of Class B Data in Reoccurring Drift Experiment 1 Where The Reoccurring Drift Event Occurs As Increasing Manner Across All Three Classes

mean of -2 and a standard deviation of 14, class B with a mean of 124 and a standard deviation of 14, and class C with a mean of 225 and a standard deviation of 14. Batch 5 consists of 1258 samples, where class A has a mean of 125 and a standard deviation of 15, class B with a mean of 218 and a standard deviation of 14, and class C with a mean of 311 and a standard deviation of 14. Batch 6 contains 1249 samples, featuring class A with a mean of 126 and a standard deviation of 15, class B with a mean of 218 and a standard deviation of 15, and class C with a mean of 311 and a standard deviation of 14. Batch 7, with 950 samples, shows class A having a mean of 125 and a standard deviation of 14, class

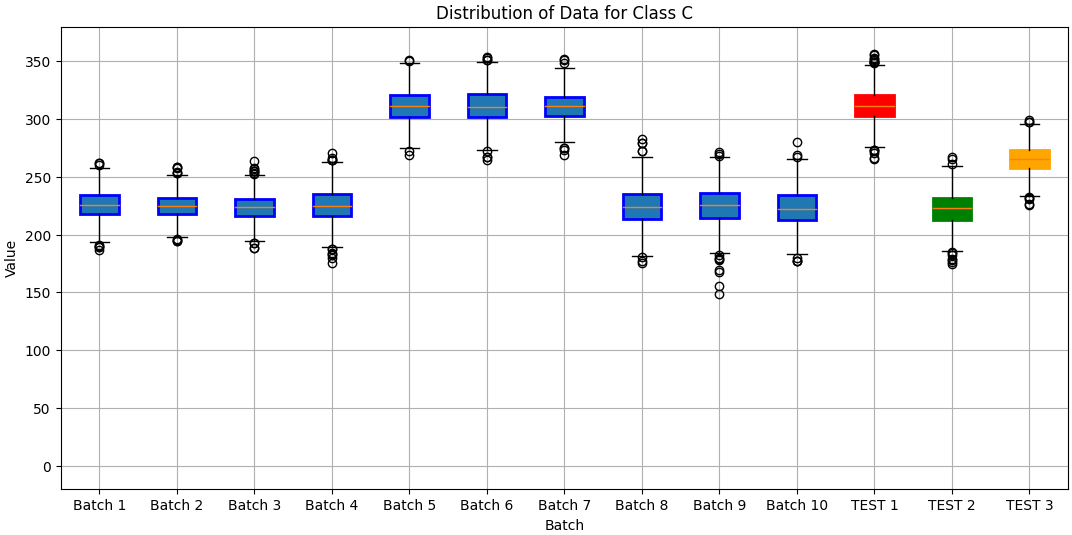


Figure 50: The Distribution of Class C Data in Reoccurring Drift Experiment 1 Where The Reoccurring Drift Event Occurs As Increasing Manner Across All Three Classes

Table 55: Summary of dataset of reoccurring drift experiment 1 where the reoccurring drift event occurs in an increasing manner across all three classes.

Batch/Test Set	**Samples**	**Class A**		**Class B**		**Class C**		**Type**
		Mean (μ)	SD (σ)	Mean (μ)	SD (σ)	Mean (μ)	SD (σ)	
Batch 1	1250	-3	16	121	13	225	12	Training
Batch 2	1259	0	17	124	12	225	11	Training
Batch 3	990	-1	17	123	12	224	11	Training
Batch 4	1160	-2	14	124	14	225	14	Training
Batch 5	1258	125	15	218	14	311	14	Training
Batch 6	1249	126	15	218	15	311	14	Training
Batch 7	950	125	14	216	13	312	13	Training
Batch 8	1353	-4	17	123	16	224	16	Training
Batch 9	1450	-3	17	123	16	224	16	Training
Batch 10	548	-4	17	124	16	224	16	Training
Test Set 1	1250	125	15	218	14	311	14	Test
Test Set 2	1250	-3	14	123	14	222	14	Test
Test Set 3	1250	60	16	170	13	265	12	Test

B with a mean of 216 and a standard deviation of 13, and class C with a mean of 312 and a standard deviation of 13. Batch 8, the largest batch with 1353 samples, presents class A with a mean of -4 and a standard deviation of 17, class B with a mean of 123 and a standard deviation of 16, and class C with a mean of 224 and a standard deviation of 16. Batch 9 comprises 1450 samples, where class A has a mean of -3 and a standard deviation of 17, class B with a mean of 123 and a standard deviation of 16, and class C with a mean of 224 and a standard deviation of 16. Batch 10 includes 548 samples, featuring class A with a mean of -4 and a standard deviation of 17, class B with a mean of 124 and a standard deviation of 16, and class C with a mean of 224 and a standard deviation of 16.

The test sets are designed to evaluate the model's performance. Test set 1 includes 1250 samples, with class A having a mean of 125 and a standard deviation of 15, class B with a mean of 218 and a standard deviation of 14, and class C with a mean of 311 and a standard deviation of 14. Test 1 reflects the concept which was observed after a drift event occurred. Test set 2, also with 1250 samples, shows class A having a mean of -3 and a standard deviation of 14, class B with a mean of 123 and a standard deviation of 14, and class C with a mean of 222 and a standard deviation of 14. The same performance metrics are evaluated. Test 2 reflects the initial concept of the classes in the training set. Test set 3, containing 1250 samples, features class A with a mean of 60 and a standard deviation of 16, class B with a mean of 170 and a standard deviation of 13, and class C with a mean of 265 and a standard deviation of 12. Test 3 does not fit any concept that is seen in training set. Its concept can classified between two concepts that is seen in the training set.

To facilitate a rigorous comparison with the results obtained from the system, a single model was constructed using a dataset comprising ten distinct training batch. This model is a logistic regression which is used by our system as mentioned in Section 3.5. The performance of this single model approach is documented in Table56:

Table 56: Performance metrics of single model approach in reoccurring drift experiment 1 where the reoccurring drift event occurs in an increasing manner across all three classes.

	Test 1	Test 2	Test 3
Accuracy	0.4469	0.9387	1.0000
Recall	0.4469	0.9387	1.0000
Precision	0.4469	0.9387	1.0000

The system created using the training strategy from Section 3.3.1 has predicted the test datasets using the prediction strategy from Section 3.6.1. The performance metrics are seen in Table57:

Table 57: Performance of the system for test sets that are used in reoccurring drift experiment 1 where the reoccurring drift event occurs in an increasing manner across all three classes.

Metric	**Test 1**	**Test 2**	**Test 3**
Accuracy	1.0000	0.3333	0.7131
Recall	1.0000	0.3333	0.7131
Precision	1.0000	0.3333	0.7131

Based on the results, while the single model approach performed poorly on Test 1, our system correctly predicted all instances. However, the performance of our system on Test 2 was quite poor. Although its performance on the Test 3 dataset was better compared to Test 2, it was still inferior to the single model approach.

The system using the training strategy from Section 3.3.1 had its training datasets balanced using the method described in Section 3.4 after each batch was added to the training datasets. The models were then trained with these balanced datasets, giving the system its final form. The system predicted by using method is mentioned in Section 3.6.1. The results of the system are provided in Table58:

Table 58: Performance of the system that implemented training process with balanced datasets in reoccurring drift experiment 1 where the reoccurring drift event occurs in an increasing manner across all three classes.

Metric	**Test 1 (Balanced Datasets)**	**Test 2 (Balanced Datasets)**	**Test 3 (Balanced Datasets)**
Accuracy	1.0000	0.9973	0.7429
Recall	1.0000	0.9973	0.7429
Precision	1.0000	0.9973	0.7429

Balancing the training datasets resolved the performance insufficiency on Test 2. Previously performing very poorly, our system's performance approached that of the single model approach after balancing. Although the balancing method also improved performance on the Test 3 dataset, it still did not reach the performance level of the single model approach. The system, which already performed

very well on the Test 1 dataset, sustained its high performance even after the training datasets were balanced.

The system created using the training strategy from Section 3.3.2 has predicted the test datasets using the prediction strategy from Section 3.6.1. The performance metrics are given in Table59:

Table 59: Performance metrics of the system that used accuracy based training strategy on test sets which used in reoccurring drift experiment 1 where the reoccurring drift event occurs in an increasing manner across all three classes.

Metric	**Test 1**	**Test 2**	**Test 3**
Accuracy	1.0000	1.0000	0.7371
Recall	1.0000	1.0000	0.7371
Precision	1.0000	1.0000	0.7371

As seen in Table59, while the system using the accuracy based training approach matched the success of the our other system on the Test 1 dataset, it did not fail on the Test 2 dataset as the other system did. The system using the accuracy-based training strategy also outperformed the single model approach on the Test 2 dataset. However, despite performing better on the Test 3 dataset compared to the system using the other training strategy, the accuracy-based system still fell short of the performance of the single model approach.

The system using the training strategy from Section 3.3.2 had its training datasets balanced using the method described in Section 3.4 after each batch was added to the training datasets. The models were then trained with these balanced datasets, giving the system its final form. The system predicted by using method is mentioned in Section 3.6.1. The results of the system are as follows:

Table 60: Performance of the system that used accuracy based training strategy and implemented training process with balanced datasets on test sets which used in reoccurring drift experiment 1 where the reoccurring drift event occurs in an increasing manner across all three classes.

Metric	**Test 1 (Balanced Datasets)**	**Test 2 (Balanced Datasets)**	**Test 3 (Balanced Datasets)**
Accuracy	1.0000	1.0000	0.7371
Recall	1.0000	1.0000	0.7371
Precision	1.0000	1.0000	0.7371

Applying the balancing method to the training datasets of the system using the accuracy-based training strategy had neither a positive nor a negative impact on its performance as seen in Table60.

4.1.4.2 Reoccurring Drift Experiment 2

In this experimental design, while the features defining the concepts of some classes will increase, for other classes, this change will manifest as a decrease. Figure51 provides an overview of the overall

data distribution. Figure52, Figure53, and Figure54 delve into the specific distributions of Class A, Class B, and Class C, respectively.

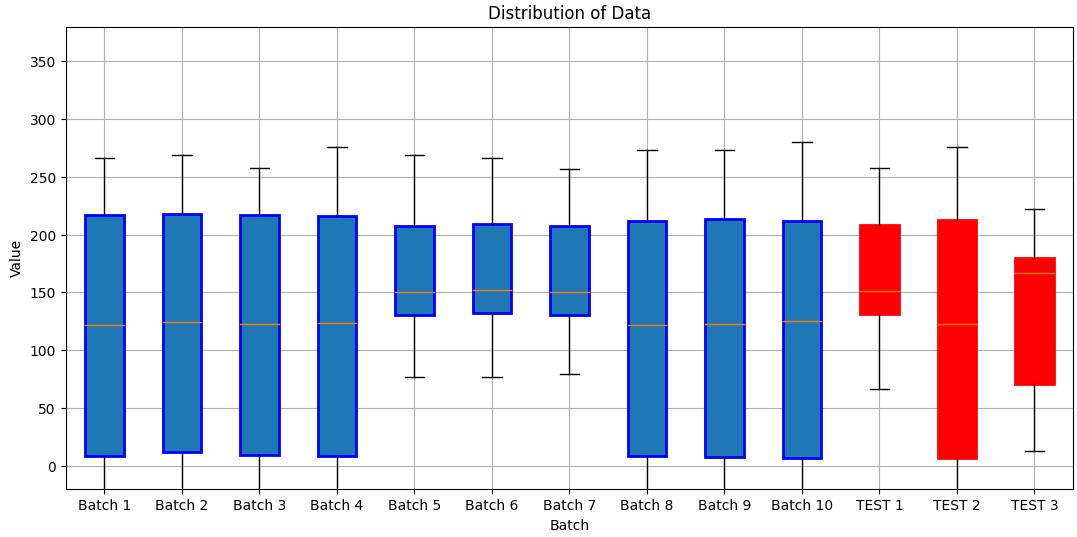


Figure 51: The Distribution of Data in Reoccurring Drift Experiment 2 Where the Reoccurring Drift Event Occurs as an Increase in Classes A and B, and a Decrease in Class C

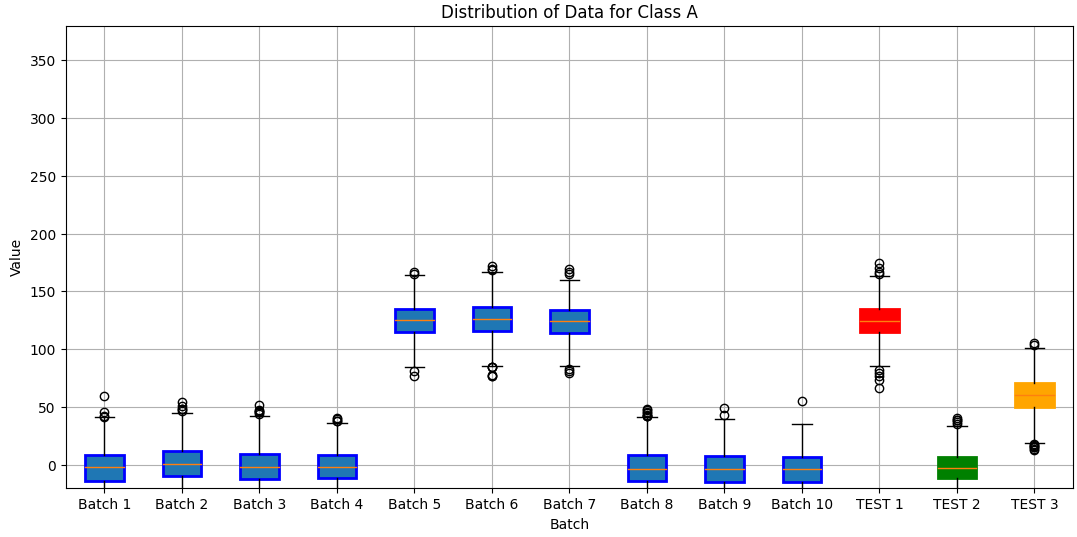


Figure 52: The Distribution of Class A Data in Reoccurring Drift Experiment 2 Where the Reoccurring Drift Event Occurs as an Increase in Classes A and B, and a Decrease in Class C

Table 61 provides numerical details, including the mean and standard deviation, for the classes present within the dataset's batches.

The training dataset comprises ten distinct batches, each containing three features. In Batch 1, which includes 1250 samples, class A has a mean of -3 and a standard deviation of 16, class B has a mean of 121 and a standard deviation of 13, and class C has a mean of 225 and a standard deviation of 12.

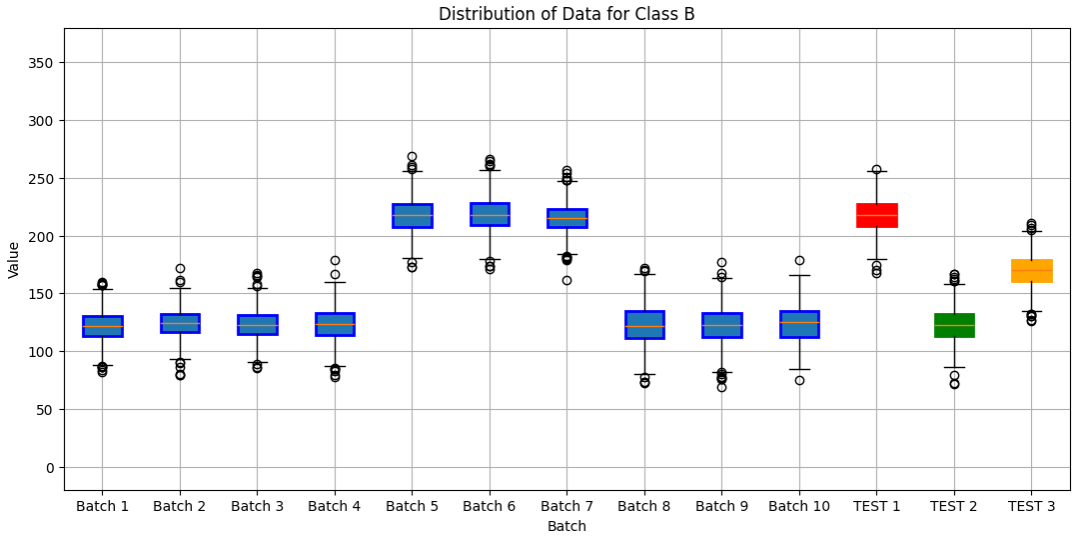


Figure 53: The Distribution of Class B Data in Reoccurring Drift Experiment 2 Where the Reoccurring Drift Event Occurs as an Increase in Classes A and B, and a Decrease in Class C

Batch 2 contains 1249 samples, where class A has a mean of 0 and a standard deviation of 17, class B has a mean of 124 and a standard deviation of 12, and class C has a mean of 225 and a standard deviation of 11. Batch 3 comprises 1490 samples, presenting class A with a mean of -1 and a standard deviation of 17, class B with a mean of 123 and a standard deviation of 12, and class C with a mean of 224 and a standard deviation of 11. Batch 4 includes 1264 samples, featuring class A with a mean of -2 and a standard deviation of 14, class B with a mean of 124 and a standard deviation of 14, and class C with a mean of 225 and a standard deviation of 14. Batch 5 consists of 1258 samples, where

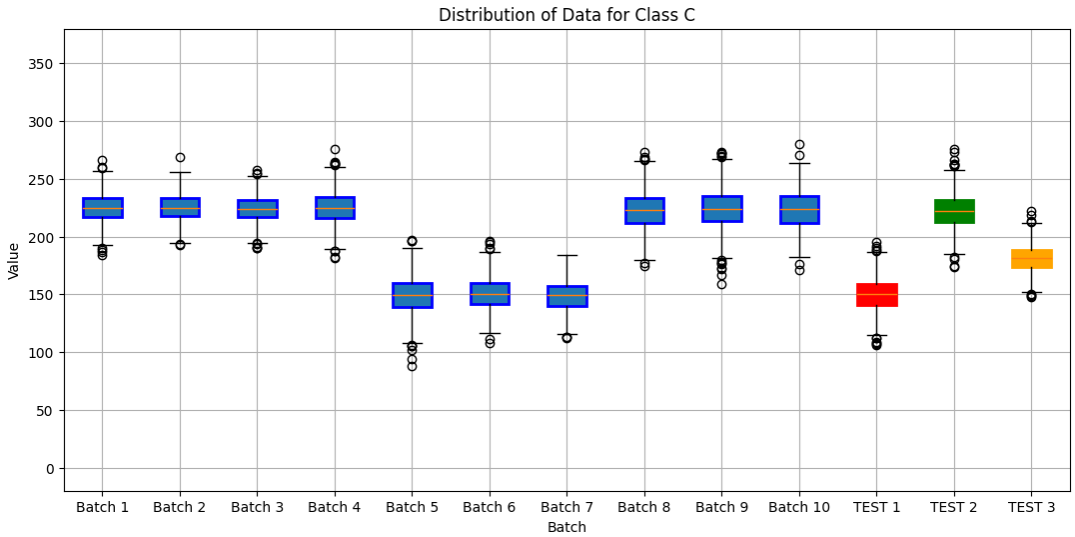


Figure 54: The Distribution of Class C Data in Reoccurring Drift Experiment 2 Where the Reoccurring Drift Event Occurs as an Increase in Classes A and B, and a Decrease in Class C

Table 61: Summary of dataset of reoccurring drift experiment 2 where the reoccurring drift event occurs as an increase in classes A and B, and a decrease in class C.

Batch/Test Set	**Samples**	**Class A**		**Class B**		**Class C**		**Type**
		Mean (μ)	SD (σ)	Mean (μ)	SD (σ)	Mean (μ)	SD (σ)	
Batch 1	1250	-3	16	121	13	225	12	Training
Batch 2	1249	0	17	124	12	225	11	Training
Batch 3	1490	-1	17	123	12	224	11	Training
Batch 4	1264	-2	14	124	14	225	14	Training
Batch 5	1258	125	15	218	14	150	16	Training
Batch 6	1249	126	15	218	15	151	14	Training
Batch 7	1050	125	14	216	13	149	13	Training
Batch 8	1303	-4	17	123	16	224	16	Training
Batch 9	1450	-3	17	123	16	224	16	Training
Batch 10	548	-4	17	124	16	224	16	Training
Test Set 1	1250	125	15	218	14	150	14	Test
Test Set 2	1250	-3	14	123	14	222	14	Test
Test Set 3	1250	60	16	170	13	181	12	Test

class A has a mean of 125 and a standard deviation of 15, class B with a mean of 218 and a standard deviation of 14, and class C with a mean of 150 and a standard deviation of 16. Batch 6 contains 1249 samples, featuring class A with a mean of 126 and a standard deviation of 15, class B with a mean of 218 and a standard deviation of 15, and class C with a mean of 151 and a standard deviation of 14. Batch 7, with 1050 samples, shows class A having a mean of 125 and a standard deviation of 14, class B with a mean of 216 and a standard deviation of 13, and class C with a mean of 149 and a standard deviation of 13. Batch 8, comprising 1303 samples, presents class A with a mean of -4 and a standard deviation of 17, class B with a mean of 123 and a standard deviation of 16, and class C with a mean of 224 and a standard deviation of 16. Batch 9 contains 1450 samples, where class A has a mean of -3 and a standard deviation of 17, class B with a mean of 123 and a standard deviation of 16, and class C with a mean of 224 and a standard deviation of 16. Batch 10 includes 548 samples, featuring class A with a mean of -4 and a standard deviation of 17, class B with a mean of 124 and a standard deviation of 16, and class C with a mean of 224 and a standard deviation of 16.

As seen in figures which are Figure48, Figure49, and Figure50, changes in concept of the classes is occurred on the fifth batch, but concepts of the classes revert back to their initial concepts in eight batch. This situation is sufficient proof for accepting the observed concept drift as a reoccurring drift.

Test set 1 includes 1250 samples, with class A having a mean of 125 and a standard deviation of 15, class B with a mean of 218 and a standard deviation of 14, and class C with a mean of 150 and a standard deviation of 14. Test set 2, also with 1250 samples, shows class A having a mean of -3 and a standard deviation of 14, class B with a mean of 123 and a standard deviation of 14, and class C with a mean of 222 and a standard deviation of 14. Test set 3, containing 1250 samples, features class A with a mean of 60 and a standard deviation of 16, class B with a mean of 170 and a standard deviation of 13, and class C with a mean of 181 and a standard deviation of 12. For Test 1 data, it can be considered that are in the same concept as the data after the concept drift occurred. It can be assumed that are

in the same concept as the data before the concept drift occurred for Test 2 data. Test 3 data can be described as being in a concept that falls between the two different concepts in the training datasets, but it does not directly exist in the training set.

To evaluate the success of our system, a single model was trained using an 10-batch training dataset. The single model uses the same type of model as described in our system in the Section 3.5. The results of the single model are given in Table62:

Table 62: Performance metrics of single model approach in reoccurring drift experiment 2 where the reoccurring drift event occurs as an increase in classes A and B, and a decrease in class C.

	Test 1	Test 2	Test 3
Accuracy	0.0104	0.9851	0.8421
Recall	0.0104	0.9851	0.8421
Precision	0.0104	0.9851	0.8421

The system created using the training strategy from Section 3.3.1 has predicted the test datasets using the prediction strategy from Section 3.6.1. The performance metrics are provided in Table63:

Table 63: Performance of the system for test sets that are used in reoccurring drift experiment 2 where the reoccurring drift event occurs as an increase in classes A and B, and a decrease in class C.

Metric	**Test 1**	**Test 2**	**Test 3**
Accuracy	0.9283	0.3333	0.5629
Recall	0.9283	0.3333	0.5629
Precision	0.9283	0.3333	0.5629

When compare Table62 and Table63, even our system performed significantly better than the single model approach on the Test 1 dataset, it was not successful on the Test 2 and Test 3 datasets.

The training datasets of the system using the training strategy from Section 3.3.1 balanced using the method described in Section 3.4 after each batch was added to the training datasets. The models were then trained with these balanced datasets, giving the system its final form. The system predicted by using method is mentioned in Section 3.6.1. The results of the system are seen in Table64:

Table 64: Performance of the system that implemented training process with balanced datasets in reoccurring drift experiment 2 where the reoccurring drift event occurs as an increase in classes A and B, and a decrease in class C.

Metric	**Test 1 (Balanced Datasets)**	**Test 2 (Balanced Datasets)**	**Test 3 (Balanced Datasets)**
Accuracy	0.9283	1.0000	0.5627
Recall	0.9283	1.0000	0.5627
Precision	0.9283	1.0000	0.5627

Balancing the training datasets improved our system's performance on the Test 2 dataset. However, despite balancing the training datasets, the poor performance on the Test 3 dataset persisted. Balancing the training datasets had no positive or negative impact on the performance on the Test 1 dataset.

The system created using the training strategy from Section 3.3.2 has predicted the test datasets using the prediction strategy from Section 3.6.1. The performance metrics are given in Table65:

Table 65: Performance metrics of the system that used accuracy based training strategy on test sets which used in reoccurring drift experiment 2 where the reoccurring drift event occurs as an increase in classes A and B, and a decrease in class C.

Metric	**Test 1**	**Test 2**	**Test 3**
Accuracy	0.9563	1.0000	0.5803
Recall	0.9563	1.0000	0.5803
Precision	0.9563	1.0000	0.5803

Our system using the accuracy-based approach performed better on the Test 1 dataset compared to both the system using the training strategy from Section 3.3.1 and the single model approach. On the Test 2 dataset, it did not show low performance like our other system and even outperformed the single model approach. However, its performance on the Test 3 dataset was lower compared to the single model approach.

The system using the training strategy from Section 3.3.2 had its training datasets balanced using the method described in Section 3.4 after each batch was added to the training datasets. The models were then trained with these balanced datasets, giving the system its final form. The system predicted by using method is mentioned in Section 3.6.1. The results of the system are detailed in Table66:

Table 66: Performance of the system that used accuracy based training strategy and implemented training process with balanced datasets on test sets which used in reoccurring drift experiment 2 where the reoccurring drift event occurs as an increase in classes A and B, and a decrease in class C.

Metric	**Test 1 (Balanced Datasets)**	**Test 2 (Balanced Datasets)**	**Test 3 (Balanced Datasets)**
Accuracy	0.9563	1.0000	0.5803
Recall	0.9563	1.0000	0.5803
Precision	0.9563	1.0000	0.5803

Applying the balancing approach from Section 3.4 to the training datasets of our system using the accuracy-based training strategy did not have any impact on its performance.

4.1.4.3 Reoccurring Drift Experiment 3

In this experimental design, similar to the second experimental design, while the features defining the concepts of some classes will increase, the features defining the concepts of other classes will

decrease. Additionally, an extra challenge has been introduced to test our system. In this experiment, the concepts formed after the concept drift for some classes will be identical to the concepts of other classes. Figure 55 provides an overview of the data's overall distribution. Also, Figure 56, Figure 57, and Figure 58 show the distributions for Class A, Class B, and Class C, respectively.

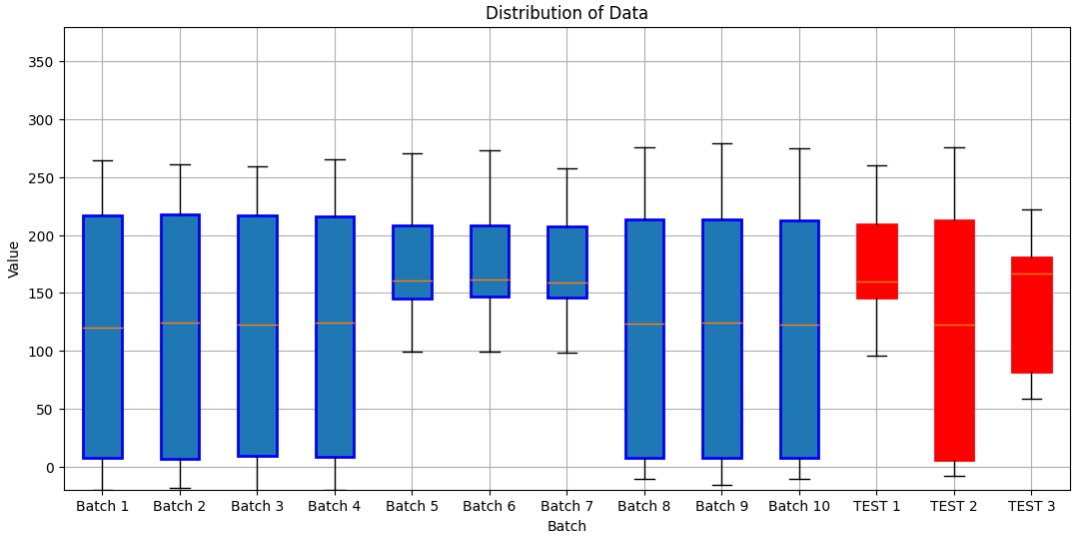


Figure 55: The Distribution of Data in Reoccurring Drift Experiment 3 Where the Reoccurring Drift Event Occurs as an Increase in Classes A and B, and a Decrease in Class C, and After the Concept Drift Event, Classes A and C Converge to the Same Concept

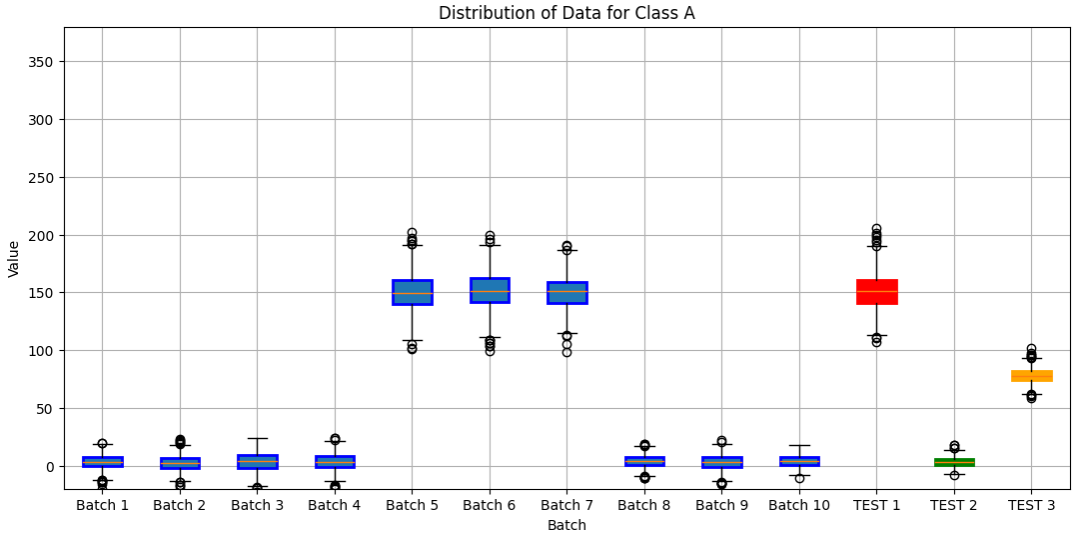


Figure 56: The Distribution of Class A Data in Reoccurring Drift Experiment 3 Where the Reoccurring Drift Event Occurs as an Increase in Classes A and B, and a Decrease in Class C, and After the Concept Drift Event, Classes A and C Converge to the Same Concept

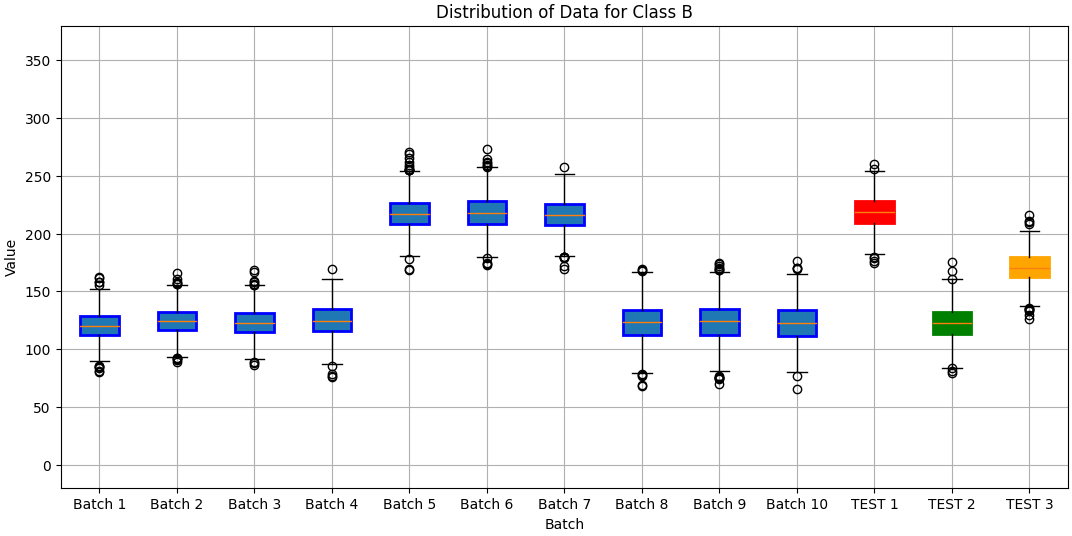


Figure 57: The Distribution of Class B Data in Reoccurring Drift Experiment 3 Where the Reoccurring Drift Event Occurs as an Increase in Classes A and B, and a Decrease in Class C, and After the Concept Drift Event, Classes A and C Converge to the Same Concept

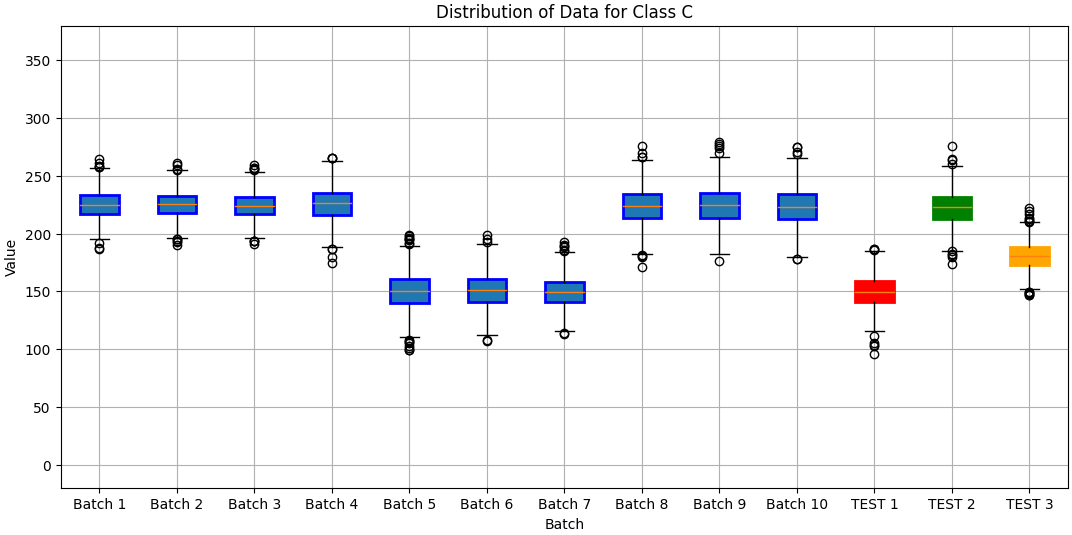


Figure 58: The Distribution of Class C Data in Reoccurring Drift Experiment 3 Where the Reoccurring Drift Event Occurs as an Increase in Classes A and B, and a Decrease in Class C, and After the Concept Drift Event, Classes A and C Converge to the Same Concept

In Table 67, the numerical characteristics of the dataset batches, such as class mean and standard deviation, are presented.

The training set consists ten batches, each containing three features and various class distributions. In Batch 1, which includes 1250 samples, class A has a mean of 3 and a standard deviation of 6, class

Table 67: Summary of dataset of reoccurring drift experiment 3 where the reoccurring drift event occurs as an increase in classes A and B, and a decrease in class C, and after the concept drift event, classes A and C converge to the same concept.

Batch/Test Set	**Samples**	**Class A**		**Class B**		**Class C**		**Type**
		Mean (μ)	SD (σ)	Mean (μ)	SD (σ)	Mean (μ)	SD (σ)	
Batch 1	1250	3	6	121	13	225	12	Training
Batch 2	1249	2	6	124	12	225	11	Training
Batch 3	1490	3	8	123	12	224	11	Training
Batch 4	1264	4	7	124	14	225	14	Training
Batch 5	1258	150	15	218	14	150	16	Training
Batch 6	1249	152	15	218	15	151	14	Training
Batch 7	1050	150	14	216	13	149	13	Training
Batch 8	1303	4	5	123	16	224	16	Training
Batch 9	1450	3	6	123	16	224	16	Training
Batch 10	548	4	5	124	16	224	16	Training
Test Set 1	1250	151	15	218	14	150	13	Test
Test Set 2	1250	3	4	123	14	222	14	Test
Test Set 3	1250	78	6	170	13	181	12	Test

B has a mean of 121 and a standard deviation of 13, and class C has a mean of 225 and a standard deviation of 12. Batch 2 contains 1249 samples, where class A has a mean of 2 and a standard deviation of 6, class B has a mean of 124 and a standard deviation of 12, and class C has a mean of 225 and a standard deviation of 11. Batch 3 comprises 1490 samples, presenting class A with a mean of 3 and a standard deviation of 8, class B with a mean of 123 and a standard deviation of 12, and class C with a mean of 224 and a standard deviation of 11. Batch 4 includes 1264 samples, featuring class A with a mean of 4 and a standard deviation of 7, class B with a mean of 124 and a standard deviation of 14, and class C with a mean of 225 and a standard deviation of 14. Batch 5 consists of 1258 samples, where class A has a mean of 150 and a standard deviation of 15, class B with a mean of 218 and a standard deviation of 14, and class C with a mean of 150 and a standard deviation of 16. Batch 6 contains 1249 samples, featuring class A with a mean of 152 and a standard deviation of 15, class B with a mean of 218 and a standard deviation of 15, and class C with a mean of 151 and a standard deviation of 14. Batch 7, with 1050 samples, shows class A having a mean of 150 and a standard deviation of 14, class B with a mean of 216 and a standard deviation of 13, and class C with a mean of 149 and a standard deviation of 13. Batch 8, comprising 1303 samples, presents class A with a mean of 4 and a standard deviation of 5, class B with a mean of 123 and a standard deviation of 16, and class C with a mean of 224 and a standard deviation of 16. Batch 9 contains 1450 samples, where class A has a mean of 3 and a standard deviation of 6, class B with a mean of 123 and a standard deviation of 16, and class C with a mean of 224 and a standard deviation of 16. Batch 10 includes 548 samples, featuring class A with a mean of 4 and a standard deviation of 5, class B with a mean of 124 and a standard deviation of 16, and class C with a mean of 224 and a standard deviation of 16.

As seen in Figure55, new concept is emerged after Batch 5, but the data come back to its initial concept after Batch 8. As illustrated in Figure56, and Figure58, the new concept of Class A which is started to

appear on Batch 5 and the new concept of Class C which is started to appear on Batch 5 are identical. It is unique challenge of the Experiment 3.

Test set 1 includes 1250 samples, with class A having a mean of 151 and a standard deviation of 15, class B with a mean of 218 and a standard deviation of 14, and class C with a mean of 150 and a standard deviation of 13. Test set 2, also with 1250 samples, shows class A having a mean of 3 and a standard deviation of 4, class B with a mean of 123 and a standard deviation of 14, and class C with a mean of 222 and a standard deviation of 14. Test set 3, containing 1250 samples, features class A with a mean of 78 and a standard deviation of 6, class B with a mean of 170 and a standard deviation of 13, and class C with a mean of 181 and a standard deviation of 12. The concept of Test 1 reflects concept which is appeared in concept drift. Concept of Test 2 is identical to initial concept of the training dataset. Test 3 has not concept that is appeared in training dataset. However, concept of the Test 3 can be evaluated as behind two concepts that is appeared in training set.

To evaluate the performance of our system, a single model was trained using an 10-batch training dataset. The single model uses the same type of model as described in our system in the Section 3.5. The performance of the single model is seen in Table68:

Table 68: Performance metrics of single model approach in reoccurring drift experiment 3 where the reoccurring drift event occurs as an increase in classes A and B, and a decrease in class C, and after the concept drift event, classes A and C converge to the same concept.

	Test 1	Test 2	Test 3
Accuracy	0.0000	0.9944	0.8275
Recall	0.0000	0.9944	0.8275
Precision	0.0000	0.9944	0.8275

The training strategy defined in Section 3.3.1 was used for training and strategy explained in Section 3.6.1 for predicting test sets. After the training process was completed, in addition to making predictions, all the collected training data were balanced using the method specified in the Section 3.4, and the models were trained and predictions made using these balanced datasets. The result of these implementation is given in Table69 and Table70:

Table 69: Performance of the system for test sets that are used in reoccurring drift experiment 3 where the reoccurring drift event occurs as an increase in classes A and B, and a decrease in class C, and after the concept drift event, classes A and C converge to the same concept.

Metric	**Test 1**	**Test 2**	**Test 3**
Accuracy	0.6637	0.3333	0.7493
Recall	0.6637	0.3333	0.7493
Precision	0.6637	0.3333	0.7493

Table 70: Performance of the system that implemented training process with balanced datasets in reoccurring drift experiment 3 where the reoccurring drift event occurs as an increase in classes A and B, and a decrease in class C, and after the concept drift event, classes A and C converge to the same concept.

Metric	**Test 1 (Balanced Datasets)**	**Test 2 (Balanced Datasets)**	**Test 3 (Balanced Datasets)**
Accuracy	0.6637	1.0000	0.5685
Recall	0.6637	1.0000	0.5685
Precision	0.6637	1.0000	0.5685

By comparing Table68, and Table69, we can say that except for the Test 1 dataset, our system performed worse than the single model approach on the other test sets. While the single model approach completely mispredicted the Test 1 dataset, our system showed acceptable performance on the Test 1 dataset. However, our system performed poorly on the Test 2 dataset and its performance on the Test 3 dataset was lower than that of the single model approach. Balancing the training datasets did not affect the performance on the Test 1 dataset, but it improved performance on Test 2 while reducing performance on the Test 3 dataset according to Table70.

The training strategy defined in Section 3.3.2 was used for training and strategy explained in Section 3.6.1 to predict. After the training process and prediction step were completed all the collected training data were balanced using the method specified in the Section 3.4, and the models were trained and predictions made using these balanced datasets. The result of these implementation is given below:

Table 71: Performance metrics of the system that used accuracy based training strategy on test sets which used in reoccurring drift experiment 3 where the reoccurring drift event occurs as an increase in classes A and B, and a decrease in class C, and after the concept drift event, classes A and C converge to the same concept.

Metric	**Test 1**	**Test 2**	**Test 3**
Accuracy	0.6795	1.0000	0.0181
Recall	0.6795	1.0000	0.0181
Precision	0.6795	1.0000	0.0181

Table 72: Performance of the system that used accuracy based training strategy and implemented training process with balanced datasets on test sets which used in reoccurring drift experiment 3 where the reoccurring drift event occurs as an increase in classes A and B, and a decrease in class C, and after the concept drift event, classes A and C converge to the same concept.

Metric	**Test 1 (Balanced Datasets)**	**Test 2 (Balanced Datasets)**	**Test 3 (Balanced Datasets)**
Accuracy	0.6795	1.0000	0.0181
Recall	0.6795	1.0000	0.0181
Precision	0.6795	1.0000	0.0181

As seen in the Table71, our system using the accuracy-based training strategy performed the same as our system using the training strategy from Section 3.3.1 on the Test 1 dataset. On the Test 2 dataset, unlike the system using the training strategy from Section 2, it did not experience any issues and successfully predicted all instances even without balancing the training datasets. However, on the Test 3 dataset, the system using the approach from Section 3.3.2 performed significantly poor. Implementing function which described in Sec3.4 for balancing training datasets of the system that uses training strategy that mentioned in Section 3.3.2 does not affected performance of the system as seen in Table72.

4.2 Experiments with Real-Word Data

In this section, the experiments are conducted with real-word data and their results are introduced. The datasets that consists of observations from real-word generally do not follow a specific concept drift scenarios completely. Real-word datasets have a complex and intricate nature instead of fully reflect an event or occurrence. On the other hand, concept drift events can be observed in real-word. Therefore, usage of datasets that consist real-word instances is important for testing our dynamic system. Although our dynamic system are successful in handling specific concept drift scenarios, it is also important to assess their performance on complex and intricate datasets that do not follow specific scenarios and where concept drift is not clearly observable.

In this study, the "ELEC 2" dataset, which contains real-world data, is used to test our system [41]. The "ELEC 2" dataset contains 45,312 instances dated from 7 May 1996 to 5 December 1998. Each row on the dataset has 9 variables which are the time stamp, the day of week, period, the New South Wales electricity demand, the Victoria electricity demand, the New South Wales electricity price, the Victoria electricity price, the scheduled electricity transfer between states, and the class label. The class variable identifies the change of the price as "UP" or "DOWN" in New South Wales. The class label is used as target variable in this research.

Ali et al. claimed that the "ELEC 2" dataset includes sudden drift and gradual drift events [42]. Also, they measured that concept drift event occurs 4531 times in "ELEC 2" dataset. Therefore, the "ELEC 2" dataset can be accepted as useful dataset to test our system against concept drift events.

In real-world scenarios, the data that reach the system does not always arrive in such a way that different concepts are separated into different batches. For instance, multiple concepts may exist within a single batch. If the data reaches the system as real-time batch fragments, the system must process these fragments as they are. On the other hand, if the entire dataset is available or the user has the option to merge or split the real-time batches before feeding them into the system, how these batches are adjusted will have an impact on the system's performance. Therefore, our study also investigates the effect of batch configuration on the performance of our system. In the study, we examined the performance of the dataset by dividing it into different batch numbers, ranging from two to 150. The results obtained from dividing the dataset into various batch numbers are given in the Section 4.2.1.

In the experiments, as the number of batches used by the system changed, the amount of data in each batch also varied because the total amount of data used remained constant. In all experiments conducted with real-world datasets, the entire dataset was utilized. On the other hand, to avoid creating bias and to fairly evaluate the system's performance, the validation and test sets were separated beforehand, with their size and the instances they contain kept constant.

To test the performance of our system, a validation set consisting of 450 instances and one test set consisting of 450 instances have been created. Two different methods have been applied to form the validation and test sets. Firstly, the last 450 instances of the dataset were assigned as the test set, and the 250 instances preceding the test set were designated as the validation set. As a second method, 450 instances were randomly selected from the entire dataset for both the test and validation sets. By testing our system with datasets that were formed by two different methods, the system's performance has been examined both in scenarios where it is required to predict data in chronological order and in situations where the data is not in chronological order, or when the system has to predict data from earlier time periods.

The performance of our systems using different training strategies, as described in the methodology chapter, has been tested on the dataset. The performance of each system using a different training strategy is presented in the corresponding subsection. In addition, the performance of our systems using different prediction strategies, as described in the methodology chapter, has been tested on the dataset. The performance of each system using a different training strategy is shown in Section 4.2.1.

4.2.1 Performance of the System According to Training Strategy

In this subsection, results of the system that uses training strategy that described in Section 3.3.1 and the system that uses training strategy that described in Section 3.3.2 are provided. Both the systems use prediction strategy which described in Section 3.6.1. By this way, it is aimed to provide comparable results according to training strategy.

The system using the strategy described in the Section 3.3.1 uses a threshold value based on the MAD (Median Absolute Deviation) value to detect concept drift while applying this strategy. As described in the Section 3.3.1, the coefficient of the threshold value based on the MAD (Median Absolute Deviation) value can be manually assigned. During the experimental phase with real-world data, two different coefficients for the MAD-based threshold value have been tested:

$$\text{Threshold} = \text{MAD} \tag{1}$$

$$\text{Threshold} = 2 \times \text{MAD} \tag{2}$$

Note: MAD that is stated in the formula is the Median Absolute Value of an already registered dataset on the system.

The comparable results of the single model approach, which trains the model using the entire dataset, the system using the training strategy described in the Section 3.3.1, and the system using the training strategy described in Section 3.3.2, on last batch dataset are presented below:

Table 73: Comparison of Performance of Different Training Approaches and Single Model Approach on Last Batch Data Based on Accuracy Metric

Approach	BEST	WORST	AVERAGE
Single Model Approach	0.789	**0.789**	**0.789**
Compare Datasets Approach (Threshold = MAD)	0.887	0.440	0.618
Compare Datasets Approach (Threshold = 2 * MAD)	**0.888**	0.438	0.759
Approach Based on Accuracy	**0.888**	0.436	0.658

Table 74: Comparison of Performance of Different Training Approaches and Single Model Approach on Last Batch Data Based on Recall Metric

Approach	BEST	WORST	AVERAGE
Single Model Approach	0.778	**0.778**	**0.778**
Compare Datasets Approach (Threshold = MAD)	0.887	0.440	0.617
Compare Datasets Approach (Threshold = 2 * MAD)	**0.909**	0.438	0.759
Approach Based on Accuracy	0.888	0.436	0.619

Table 75: Comparison of Performance of Different Training Approaches and Single Model Approach on Last Batch Data Based on Precision Metric

Approach	BEST	WORST	AVERAGE
Single Model Approach	0.751	**0.751**	0.751
Compare Datasets Approach (Threshold = MAD)	0.887	0.194	0.564
Compare Datasets Approach (Threshold = 2 * MAD)	**0.910**	0.314	**0.798**
Approach Based on Accuracy	0.894	0.190	0.658

When we examine Table73, we see that both versions of the model using the training strategy explained in Section 3.3.1, with two different thresholds, as well as the system using the training strategy described in Section 3.3.2, have achieved better results than the single-model approach in cases where they showed the best performance according to the accuracy metric. On the other hand, when looking at the worst results, it is observed that the accuracy values of all our systems are quite low. This indicates that the number of batches into which the dataset is divided and the separation points of the batches can dramatically affect the system's performance. Although the average performance of our systems with different batch selections is close to the performance of the single-model approach, it is below it. When we compare the best performance of our systems using different training strategies based on the accuracy metric, it has been observed that all training strategies yield almost identical results. In summary, according to the accuracy metric, while our systems experience dramatic performance drops depending on the batch number, they have the potential to achieve better results than the single-model approach when an appropriate batch number is selected.

When we look at Table74, it can be said that the results for the recall metric are similar to results of the accuracy metric. The best performances of our systems are close to each other. Also, their best performances exceed performance of single model approach. However, the worst performances are quite low. Although the average values for the recall metric are close to those of the single-model approach, but they lower than performance of single model approach.

In Table75, the performance of our systems and the single-model approach according to the precision metric is shown. The best performance of all our systems is better than that of the single-model approach. On the other hand, the worst performances are dramatically poor. The system using the training strategy described in Section 3.3.1, with a threshold value of 2*MAD for detecting concept drift, shows a better average performance across different batch numbers compared to the single-model approach. However, the average performance of our other systems is worse than that of the single-model approach.

The comparable results of the single model approach, which trains a model using the all dataset, the system using the training strategy described in the Section 3.3.1, and the system using the training strategy explained in Section 3.3.2, on random instances dataset are shown below:

Table 76: Comparison of Performance of Different Training Approaches and Single Model Approach on Random Instances Dataset Based on Accuracy Metric

Approach	**BEST**	**WORST**	**AVERAGE**
Single Model Approach	0.796	**0.796**	**0.796**
Compare Datasets Approach (Threshold = MAD)	**0.818**	0.464	0.662
Compare Datasets Approach (Threshold = 2 * MAD)	0.804	0.504	0.718
Approach Based on Accuracy	0.816	0.550	0.650

Table 77: Comparison of Performance of Different Training Approaches and Single Model Approach on Random Instances Dataset Based on Recall Metric

Approach	**BEST**	**WORST**	**AVERAGE**
Single Model Approach	0.663	**0.663**	0.663
Compare Datasets Approach (Threshold = MAD)	**0.818**	0.464	0.662
Compare Datasets Approach (Threshold = 2 * MAD)	0.804	0.504	**0.718**
Approach Based on Accuracy	0.816	0.480	0.650

Table 78: Comparison of Performance of Different Training Approaches and Single Model Approach on Random Instances Dataset Based on Precision Metric

Approach	**BEST**	**WORST**	**AVERAGE**
Single Model Approach	0.810	**0.810**	**0.810**
Compare Datasets Approach (Threshold = MAD)	**0.817**	0.511	0.692
Compare Datasets Approach (Threshold = 2 * MAD)	0.806	0.551	0.740
Approach Based on Accuracy	0.815	0.550	0.693

When we look at Table76, it is evident that the versions of the model using the training strategy described in Section 3.3.1, with two different thresholds, as well as the system employing the training strategy explained in Section 3.3.2, achieved better results than the single-model approach when they are demonstrating their best performances according to the accuracy metric. On the other hand, looking at their worst performances, all systems show significantly low accuracy values. The average performances of our systems with different batch sizes can be accepted as close to the average value of the single-model approach, but they are lower. In summary, according to the accuracy metric, while our systems experience dramatic performance drops depending on the batch size, they still have the potential to outperform the single-model approach when an appropriate batch size is selected.

When we look at Table77, the performance of our systems' best results according to the recall metric is quite similar. The best performance of all our systems, according to the recall metric, is better than that of the single-model approach. However, their worst performances are significantly lower. The system that uses the training strategy described in Section 3.3.1 and applies a threshold value of 2 * MAD, and the system that uses the training strategy described in Section 3.3.2 demonstrate a better average performance across different batch sizes compared to the single-model approach. On the other hand, while the average performance of the system that uses the training strategy described in Section 3.3.1 and applies a threshold value of MAD is close to that of the single-model approach, it still falls below it.

In Table78, the performance of our systems and the single-model approach according to the precision metric is presented. Except for the system that uses the training strategy described in Section 3.3.1 and applies 2*MAD as the threshold for detecting concept drift, the best performance of all our systems surpasses that of the single-model approach. On the other hand, the worst performances of all systems are remarkably poor. Moreover, it is observed that the average performance of our systems, generated by using different batch sizes, falls below the average performance of the single-model approach.

Both systems which use different training strategies achieved better results compared to the single-model approach when using appropriate batch numbers for the training phase, based on their performance on the validation sets generated by either selecting random instances or using the last batch. On the other hand, significant performance drops occurred in the systems using the two training strategies depending on the batch number. This highlights the importance of accurately determining the batch size parameter.

A hyper-parameter tuning method can be employed to determine the appropriate batch size by evaluating performance on the validation set. The comparable results of the single model approach, which trains the model using the entire dataset, the batch number tuned version of the system using the training strategy described in the Section 3.3.1, and the batch number tuned version of the system using the training strategy described in Section 3.3.2, on a test set that is last batch are given below:

Table 79: Comparison of Tuned Version of the Systems and Single Model Approach Based on Test Set Accuracy Where Test Set Is Last Batch

Approach	**Number of Batch**	**Validation Set Accuracy**	**Test Set Accuracy**
Single Model Approach	1	0.789	0.824
Compare Datasets Approach (Threshold = 2 * MAD)	146	0.888	**0.92**
Approach Based on Accuracy	25	0.888	0.888

As seen in the Table79, when the batch number that provides the best performance on the validation set is selected, both the batch number tuned version of the system using the training strategy described in the Section 3.3.1, and the batch number tuned version of the system using the training strategy described in Section 3.3.2 outperform the single-model approach on the test set.

The comparable results of the single model approach, which trains the model using the entire dataset, the batch number tuned version of the system using the training strategy described in the Section 3.3.1, and the batch number tuned version of the system using the training strategy described in Section 3.3.2, on a test set that consists random instances are given below:

Table 80: Comparison of Tuned Version of the Systems and Single Model Approach Based on Test Set Accuracy Where Test Set Consists Random Instances

Approach	Number of Batch	Validation Set Accuracy	Test Set Accuracy
Single Model Approach	1	0.796	0.736
Compare Datasets Approach (Threshold = 2 * MAD)	149	0.804	0.784
Approach Based on Accuracy	21	0.816	**0.808**

As seen in the the Table80, thanks to tuning strategy based on choosing batch number according to performance on validation set, both the batch number tuned version of the system using the training strategy described in the Section 3.3.1, and the batch number tuned version of the system using the training strategy described in Section 3.3.2 outperform the single-model approach on the test set.

Although the batch size parameter used by the system can be tuned based on validation set performance to prevent performance drops related to batch size, a detailed examination of batch size's impact on performance remains crucial for understanding the system's behavior on real-world datasets. This topic is discussed in detail in Section 4.2.1.1 and Section 4.2.1.2.

4.2.1.1 Analysis of Performance of the System That Uses Training Strategy Based on Comparison of Datasets According to Batch Number.

The Figure59 illustrates the change in the performance of our system, which uses the training strategy described in Section 3.3.1 and applies the MAD value as the threshold, on the validation set that is last batch according to the accuracy metric across different batch numbers.

As seen in the Figure59, although the system outperforms the single-model approach for many batch numbers, its performance drops significantly in certain batch numbers where a performance decline occurs. Additionally, when looking at the graph, a zigzag pattern can be observed. This suggests that even batch numbers close to the optimal one might still experience notable performance losses. No significant difference in overall performance was observed between using balanced datasets and not using them. In both cases, the system's performance remains unstable.

The Figure60 illustrates the change in the performance of our system, which uses the training strategy described in Section 3.3.1 and applies the MAD value as the threshold, on the validation set that is last batch according to the recall metric across different batch numbers.

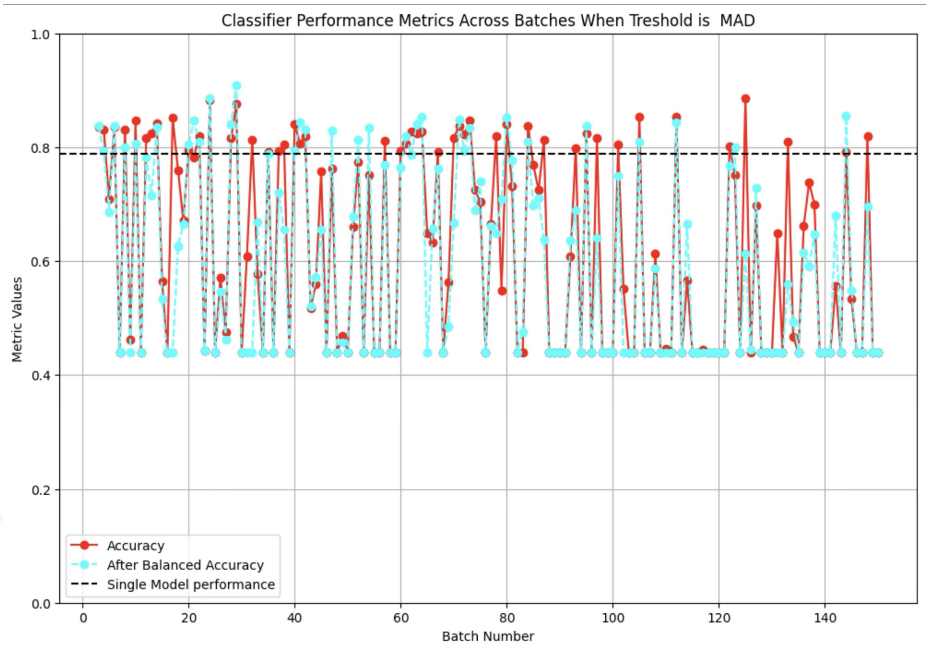


Figure 59: The Performance of The System That Uses Strategy Based On Comparison of Datasets With Threshold As MAD on The Validation Set Which is Formed By Last Batch Method According to The Accuracy Metric Across Different Batch Numbers

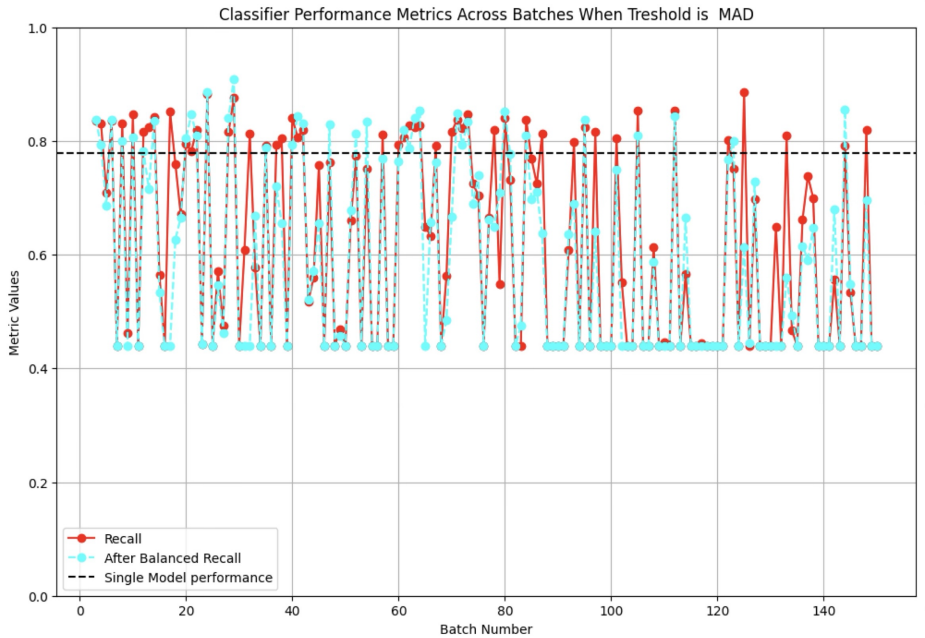


Figure 60: The Performance of The System That Uses Strategy Based On Comparison of Datasets With Threshold As MAD on The Validation Set Which is Formed By Last Batch Method According to The Recall Metric Across Different Batch Numbers

In the Figure60, a zigzag pattern can be observed for the recall metric. No significant difference in overall performance has been observed between using balanced datasets and not using them. In both cases, the system's performance remains unstable.

The Figure61 illustrates the change in the performance of our system, which uses the training strategy described in Section 3.3.1 and applies the MAD value as the threshold, on the validation set that is last batch according to the precision metric across different batch numbers.

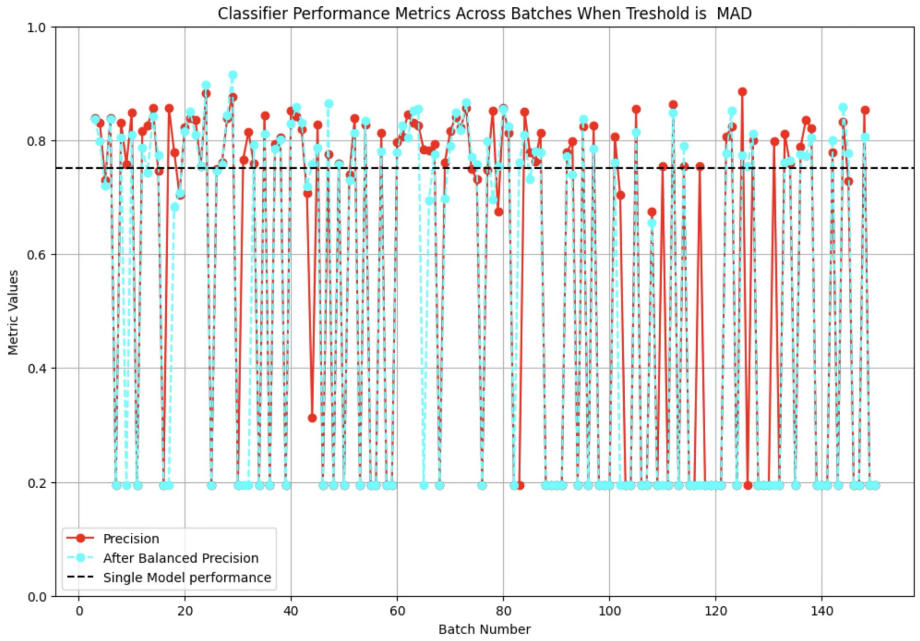


Figure 61: The Performance of The System That Uses Strategy Based On Comparison of Datasets With Threshold As MAD on The Validation Set Which is Formed By Last Batch Method According to The Precision Metric Across Different Batch Numbers

In Figure61, a zigzag pattern is evident for the precision metric. No notable difference in overall performance is detected between utilizing balanced datasets and unbalanced datasets. The system's performance continues to exhibit instability in both scenarios.

The Figure62 illustrates the change in the performance of our system, which uses the training strategy described in Section 3.3.1 and applies the MAD value as the threshold, on the validation set that consists random instances according to the accuracy metric across different batch numbers.

As shown in the Figure62, the system's performance on the validation set, composed of random instances, has surpassed the performance of the single-model approach in only a small number of batch sizes. The pattern in the graph is characterized by fluctuations, exhibiting an up-and-down trend.

The Figure63 illustrates the change in the performance of our system, which uses the training strategy described in Section 3.3.1 and applies the MAD value as the threshold, on the validation set that consists random instances according to the recall metric across different batch numbers.

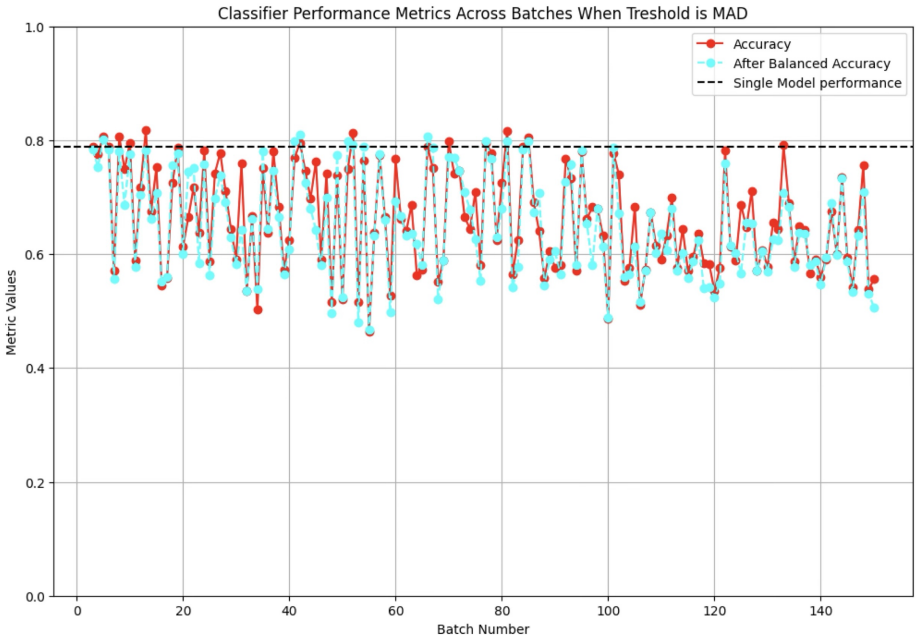


Figure 62: The Performance of The System That Uses Strategy Based On Comparison of Datasets With Threshold As MAD on The Validation Set Which Contains Random Instances According to The Accuracy Metric Across Different Batch Numbers

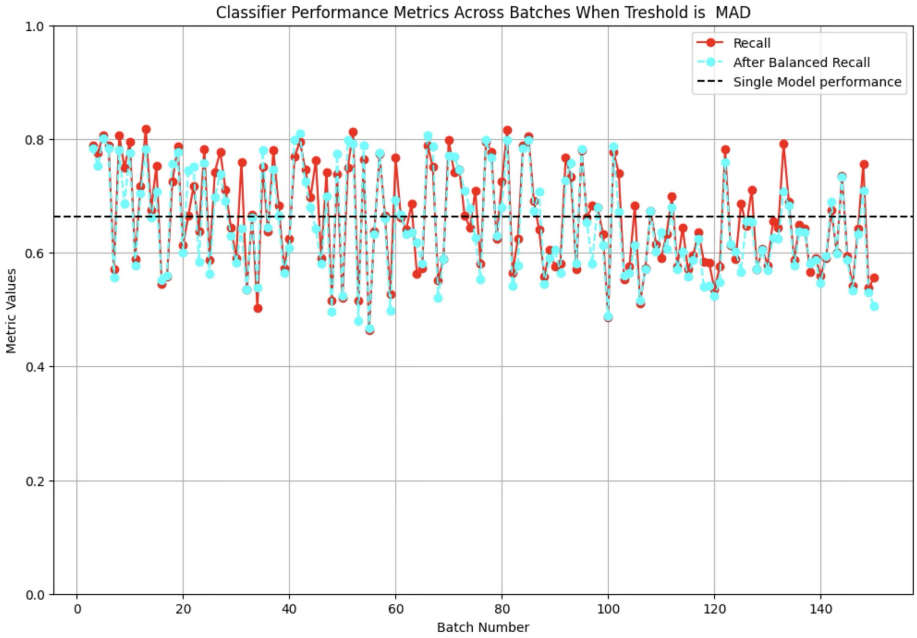


Figure 63: The Performance of The System That Uses Strategy Based On Comparison of Datasets With Threshold As MAD on The Validation Set Which Contains Random Instances According to The Recall Metric Across Different Batch Numbers

As seen in the Figure63, the system's performance according to the recall metric has surpassed that of the single-model approach in more than half of the batch numbers. The model's performance, based on the recall metric, shows a fluctuating pattern with rises and falls.

The Figure64 illustrates the change in the performance of our system, which uses the training strategy described in Section 3.3.1 and applies the MAD value as the threshold, on the validation set that consists random instances according to the precision metric across different batch numbers.

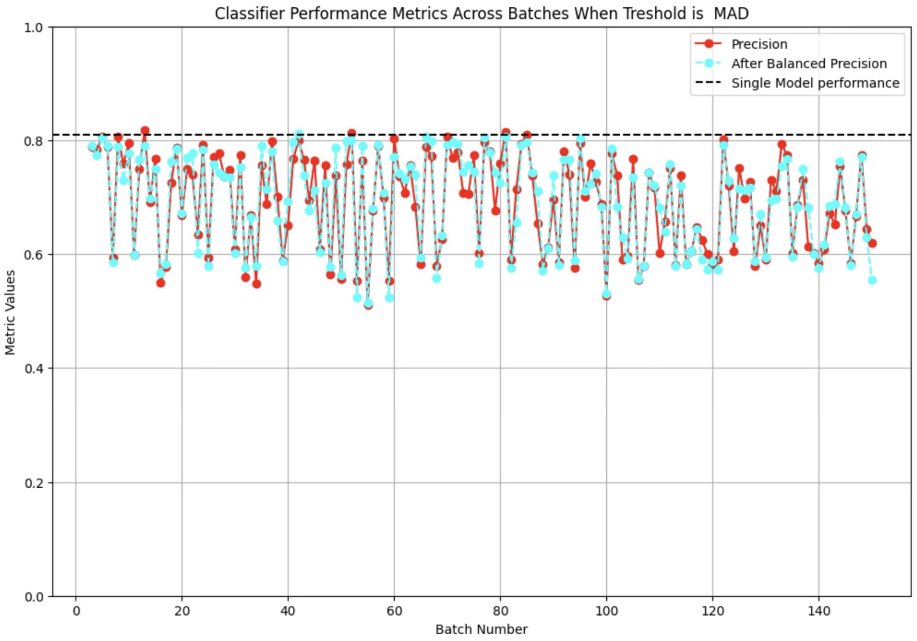


Figure 64: The Performance of The System That Uses Strategy Based On Comparison of Datasets With Threshold As MAD on The Validation Set Which Contains Random Instances According to The Precision Metric Across Different Batch Numbers

As seen in the Figure64, the system's performance generally remains below that of the single-model approach. However, in a few specific batch number configurations, the system has managed to surpass the single-model approach according to the precision metric.

The Figure65 illustrates the change in the performance of our system, which uses the training strategy described in Section 3.3.1 and applies the $2 \times$ MAD value as the threshold, on the validation set that is last batch according to the accuracy metric across different batch numbers.

As seen in the Figure65, although the system outperforms the single-model approach for many batch numbers, its performance drops significantly in specific batch numbers. There are not significant difference in overall performance was observed between using balanced datasets and not using them. In both cases, the system's performance is unstable.

The Figure66 illustrates the change in the performance of our system, which uses the training strategy described in Section 3.3.1 and applies the $2 \times$ MAD value as the threshold, on the validation set that is last batch according to the recall metric across different batch numbers.

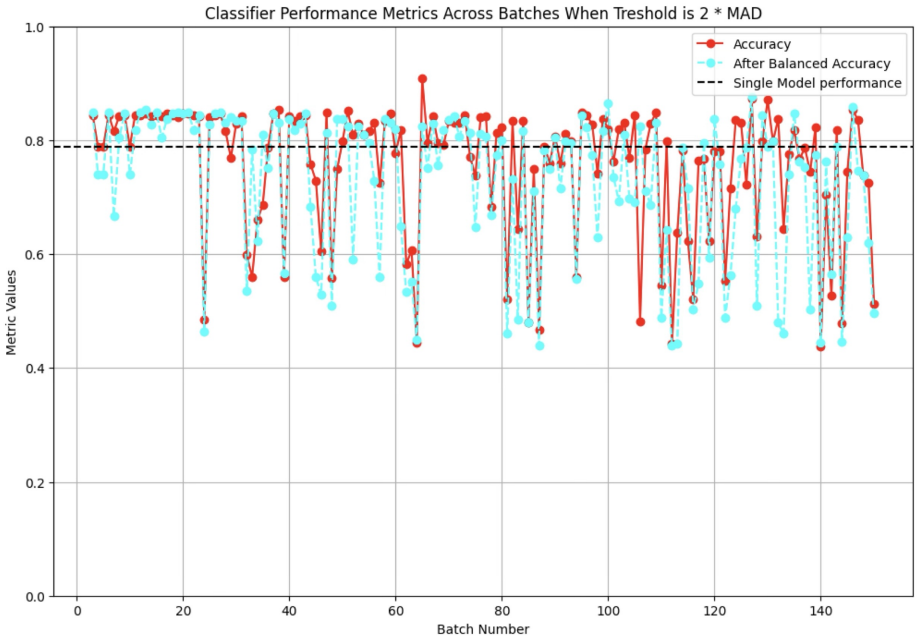


Figure 65: The Performance of The System That Uses Strategy Based On Comparison of Datasets With Threshold As 2 MAD on The Validation Set According to The Accuracy Metric Across Different Batch Numbers

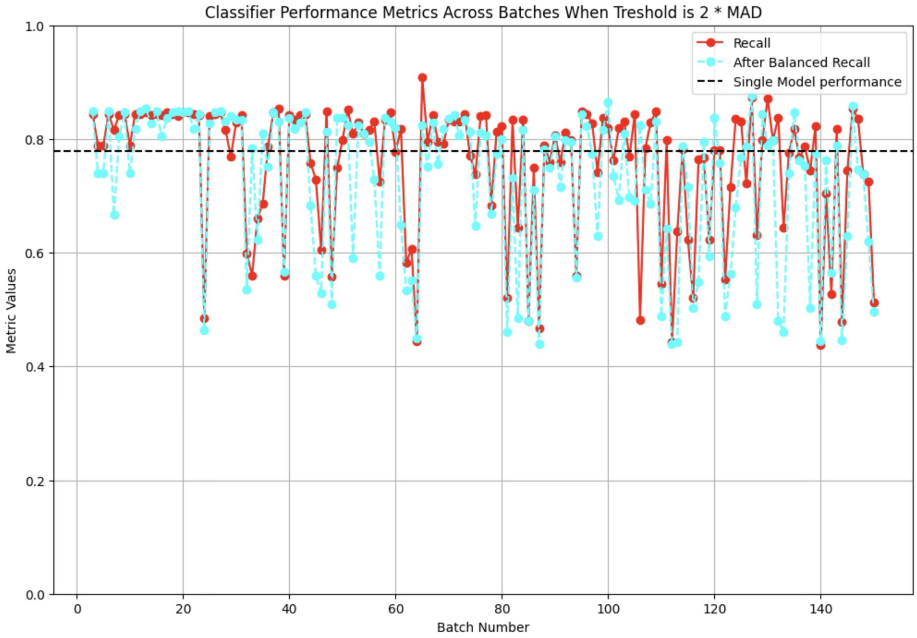


Figure 66: The Performance of The System That Uses Strategy Based On Comparison of Datasets With Threshold As 2 MAD on The Validation Set According to The Recall Metric Across Different Batch Numbers

In the Figure66, a fluctuating pattern is observed, with the system's performance rising and falling across different batch numbers. No significant difference in overall performance has been observed between using balanced datasets and not using them.

The graph below illustrates the change in the performance of our system, which uses the training strategy described in Section 3.3.1 and applies the $2 \times$ MAD value as the threshold, on the validation set that is last batch according to the precision metric across different batch numbers.

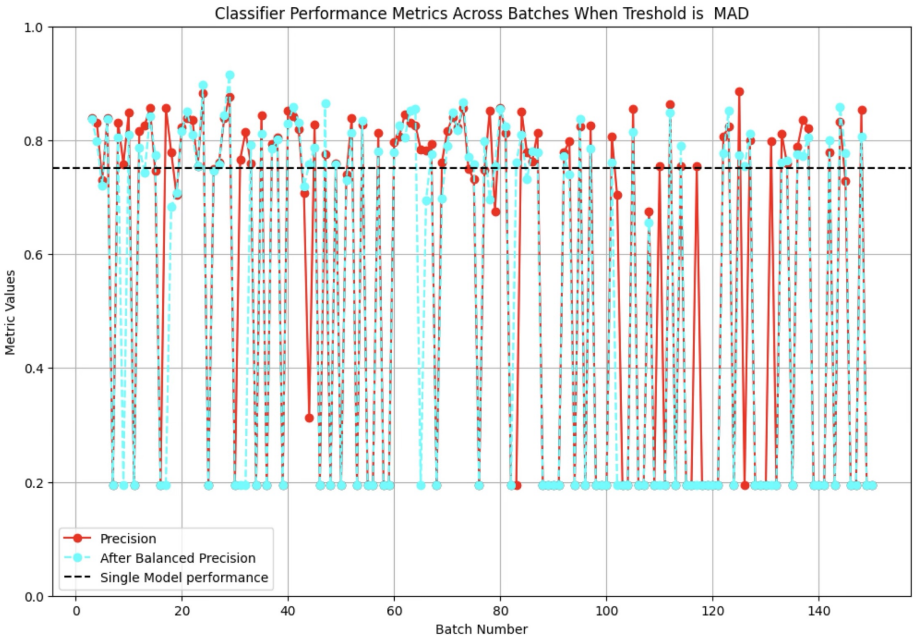


Figure 67: The Performance of The System That Uses Strategy Based On Comparison of Datasets With Threshold As 2 MAD on The Validation Set According to The Precision Metric Across Different Batch Numbers

In Figure67, a fluctuating pattern is seen for the precision metric. No notable difference in overall performance is detected between using balanced datasets and unbalanced datasets. It has been observed that models trained on both unbalanced and balanced datasets achieve better results in terms of precision compared to a single-model approach in specific batch numbers. However, on the other hand, they experience dramatic drops in performance at specific batch numbers.

4.2.1.2 Analysis of Performance of the System That Uses Training Strategy Based on Accuracy According to Batch Number.

The following graph illustrates the performance of our model, which uses a training strategy based on the decision to create to a new model according to the accuracy performance of current learners as described in Section 3.3.2. The performance of model according accuracy metric is illustrated across different batch numbers ranging from 2 to 150. The last batch was used as the validation set to conduct the measurements.

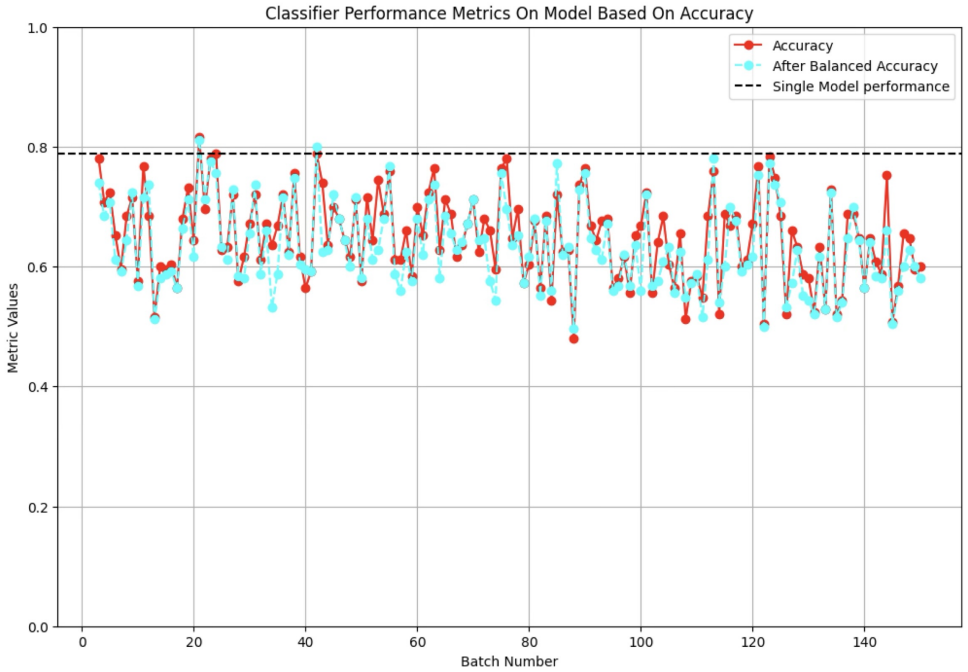


Figure 68: The Performance of The System That Uses Training Strategy Based On Accuracy on The Validation Set Which is Formed By Last Batch Method According to The Accuracy Metric Across Different Batch Numbers

As seen in the Figure68, although better results are obtained in scenarios with small percent of batch numbers compared to the single-model approach, the performance of the model remains lower for most batch numbers. Also, the model's performance fluctuates according to the number of batches.

The performance of the model, which utilizes the training strategy explained in Section 3.3.2, across different batch number configurations is presented in the Figure69. The performance according to the recall metric is presented in the graph. The last batch was used as the validation set to measure the performance.

As seen in the Figure69, the model has achieved better results compared to the single-model approach across many different batch numbers. However, significant drops in performance have occurred for specific batch numbers. The performance patterns of the models using balanced and unbalanced datasets are similar.

The performance of the model using the training strategy from Section 3.3.2, is presented in Figure70 according to the precision metric. The figure shows the model's performance across different batch number configurations. The last batch was used as the validation set to measure the model's performance.

Upon examining the Figure70, it is observed that the model performs better than the single-model approach for most batch number configurations. Furthermore, the model trained on the unbalanced dataset does not experience any dramatic drop in performance except for one batch number. On the

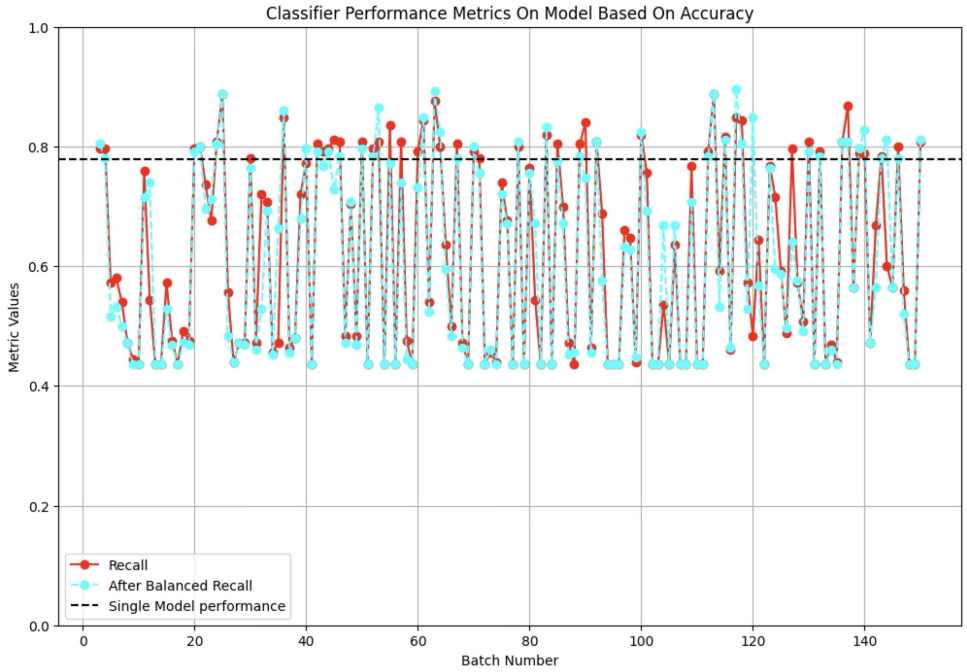


Figure 69: The Performance of The System That Uses Training Strategy Based On Accuracy on The Validation Set Which is Formed By Last Batch Method According to The Recall Metric Across Different Batch Numbers

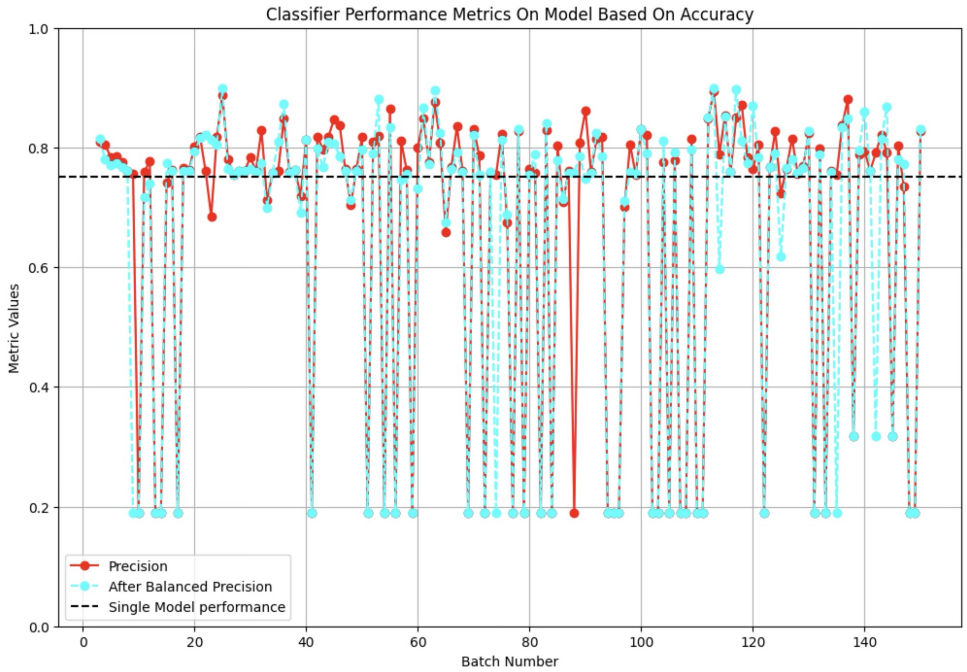


Figure 70: The Performance of The System That Uses Training Strategy Based On Accuracy on The Validation Set Which is Formed By Last Batch According to The Precision Metric Across Different Batch Numbers

other hand, the model trained on balanced datasets has undergone several significant performance drops.

The Figure71 shows the performance of the model trained using the strategy described in Section 3.3.2 according to the accuracy metric. The validation set that is used to measure the model's performance consists of randomly selected instances from the dataset.

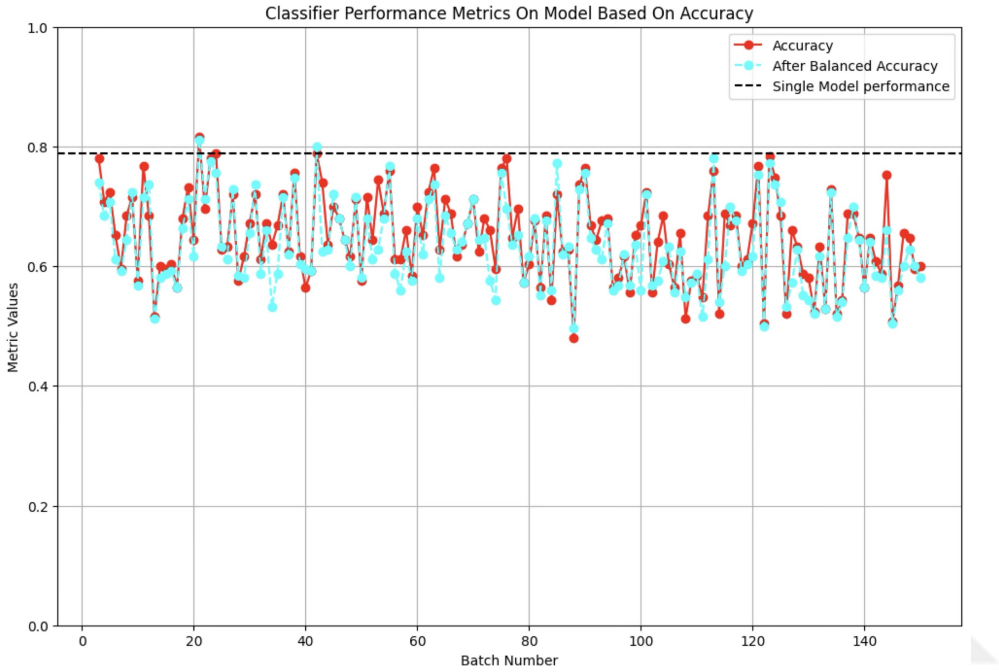


Figure 71: The Performance of The System That Uses Training Strategy Based On Accuracy on The Validation Set Which Contains Random Instances According to The Accuracy Metric Across Different Batch Numbers

As seen in the Figure71, the model's performance surpasses that of the single-model approach in scenarios with a small number of batch configurations. However, the model's performance fluctuates when trained on both balanced and unbalanced datasets.

In the Figure 72, the performance of the model trained using the strategy described in Section 3.3.2 is presented in Figure72 according to the recall metric. The validation set which is used to measure the model's performance across different batch numbers consists of randomly selected instances from the dataset.

As seen in the Figure72, the model's performance surpasses that of the single-model approach in more than half of the selected batch numbers. The model's performance fluctuates according to the recall metric,. It shows a similar pattern when trained on both balanced and unbalanced datasets.

The performance of the model using the training strategy from Section 3.3.2 is presented in Figure73 according to the precision metric. The figure shows the model's performance across different batch number configurations. The validation set that is used for measuring model's performance according to precision metric was formed by randomly selecting instances.

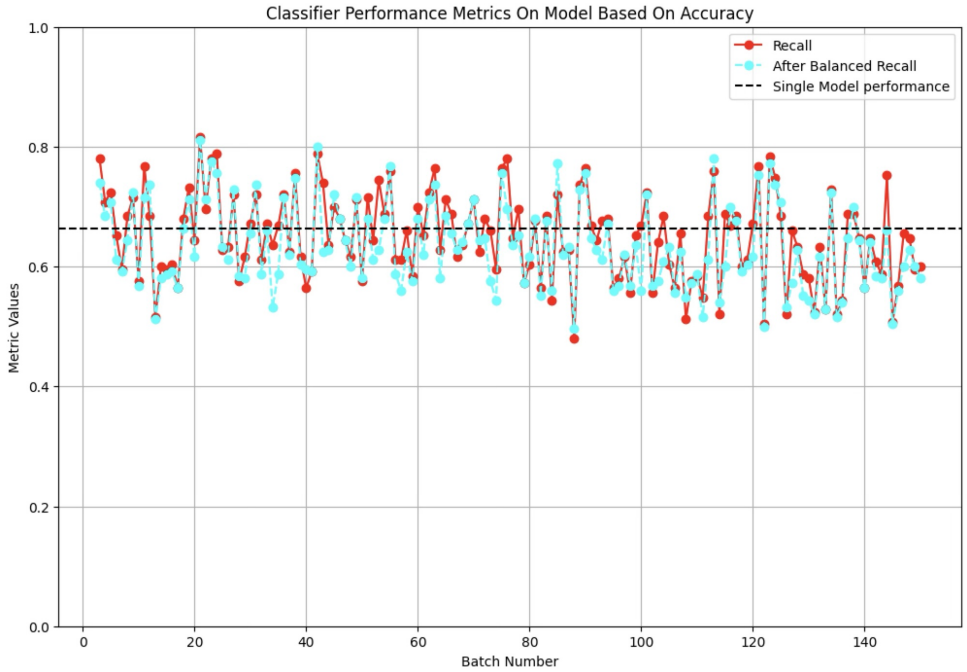


Figure 72: The Performance of The System That Uses Training Strategy Based On Accuracy on The Validation Set Which Contains Random Instances According to The Recall Metric Across Different Batch Numbers

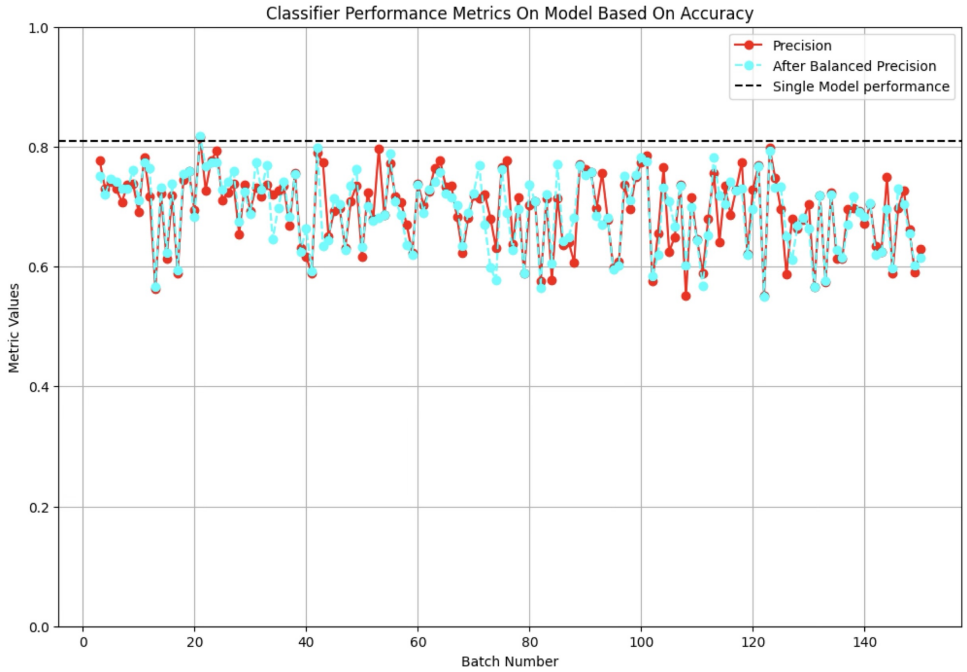


Figure 73: The Performance of The System That Uses Training Strategy Based On Accuracy on The Validation Set Which Contains Random Instances According to The Precision Metric Across Different Batch Numbers

In the Figure73, it can be seen that, based on the precision metric, better results were achieved compared to the single-model approach for only one batch number. This result was obtained by the model trained on the balanced dataset. The performance of the models fluctuates according to the selected batch number.

All models trained using different training strategies have shown fluctuating performance across batch numbers. These results were obtained using the prediction strategy described in 3.6.1. In this case, it would be beneficial to try different prediction strategies and make conclusions based on their results.

4.2.2 Performance of the System According to Prediction Strategy

In this subsection, the performances of different prediction strategies are examined. The results of the prediction strategies are described in Section 3.6.

In order to compare the prediction strategies and make inference about which strategy yields better results, the results were obtained using a specific training strategy. The training strategy described in Section 3.3.1 was employed, and the threshold value was set as $2 \times$ MAD.

The prediction strategy described in Section 3.6.2 conducts the prediction process based on a similarity score approach. The performance of the system using this strategy is presented in Figure74 and Figure75, according to the accuracy and precision metrics. The validation set was created by reserving the last batch to measure the system's performance.

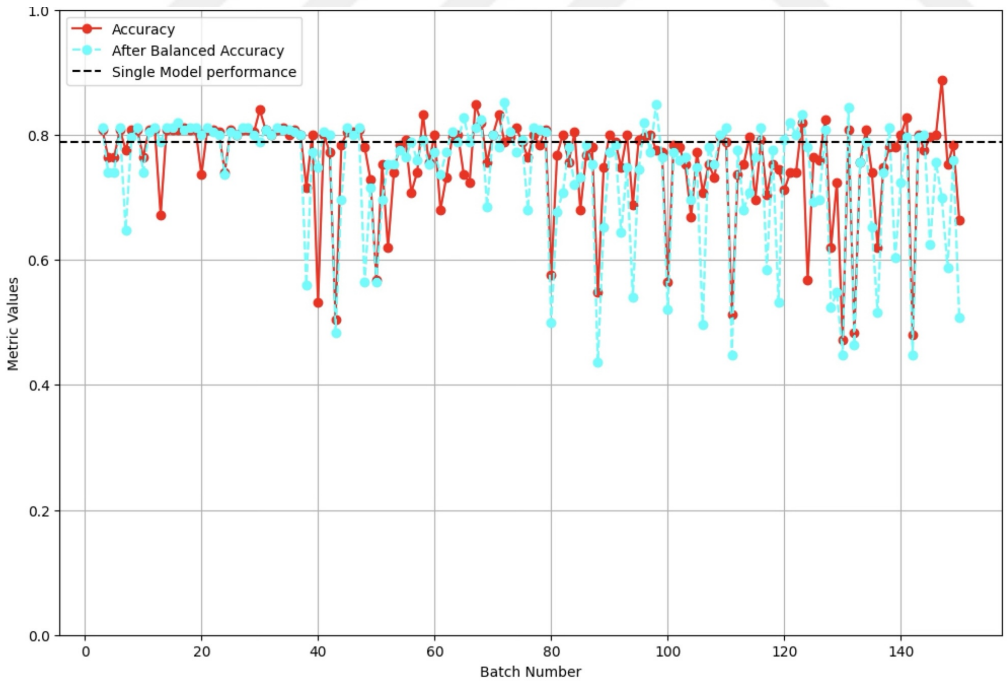


Figure 74: The Performance of The System That Uses Prediction Strategy Based on Similarity Score on The Validation Set Which is Last Batch According to The Accuracy Metric Across Different Batch Numbers

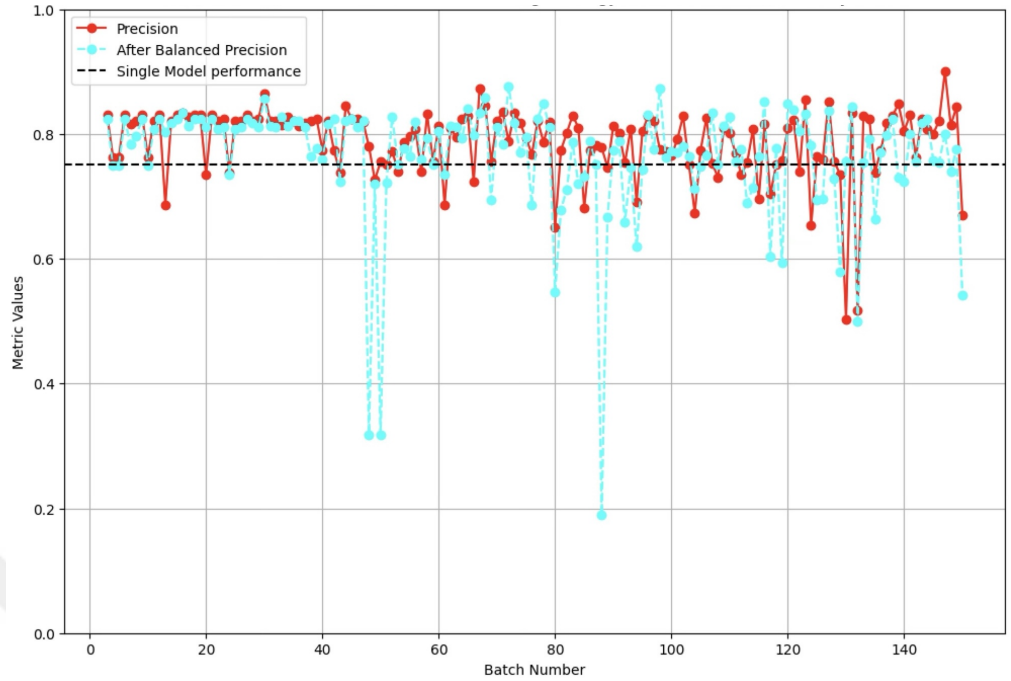


Figure 75: The Performance of The System That Uses Prediction Strategy Based on Similarity Score on The Validation Set Which is Last Batch According to The Precision Metric Across Different Batch Numbers

According to both the accuracy and precision metrics, the system has achieved better results than the single-model approach across many batch number configurations as seen in Figure74 and Figure75. However, in some batch configurations, the system experienced performance drops. In summary, even when using the prediction strategy described in Section 3.6.2, the system has suffered performance losses depending on the batch number.

The performance of the model using the prediction strategy from Section 3.6.2 and the training strategy from Section 3.3.1 across different batch number configurations is presented in the Figure76 and Figure77. In the experiments where these scores were obtained, the validation set consists of random instances.

As seen in the Figure76, performance of model is lower than single model approach except one batch number specification based on accuracy metric. Also, as seen in the Figure77, according to the precision metric, the system did not achieve better results than the single-model approach for any batch number configuration. When comparing the cases where the system uses the prediction strategy from Section 3.6.2, it was observed that the system performed worse when the validation set consisted of random instances compared to when the last batch was used as the validation set.

When the system using the prediction strategy from Section 3.6.2 is tuned based on the batch number that yielded the best accuracy on the validation set, it achieved the results shown in the Table81 on the test set.

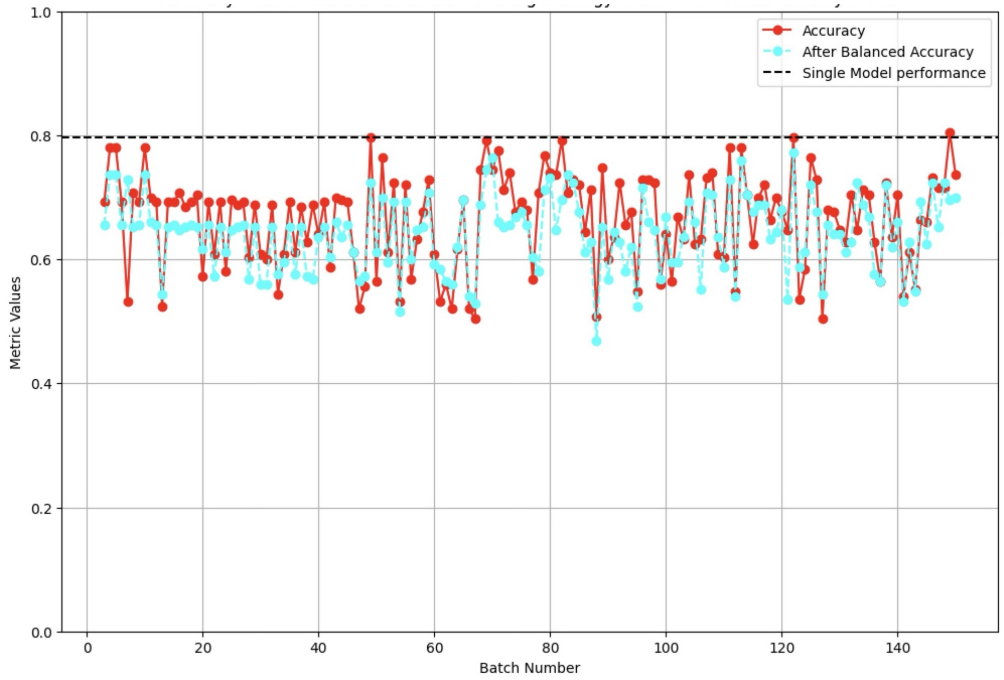


Figure 76: The Performance of The System That Uses Prediction Strategy Based on Similarity Score on The Validation Set Which Contains Random Instances According to The Accuracy Metric Across Different Batch Numbers

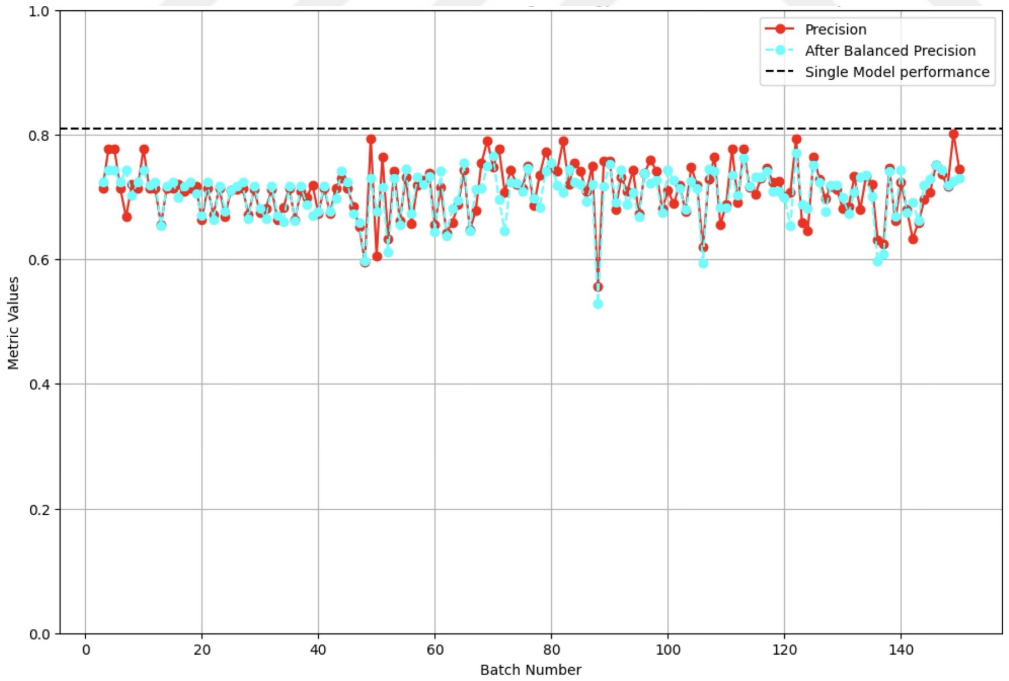


Figure 77: The Performance of The System That Uses Prediction Strategy Based on Similarity Score on The Validation Set Which Contains Random Instances According to The Precision Metric Across Different Batch Numbers

Table 81: Comparison of Tuned Version of the Systems That Use Prediction Strategy Based on Similarity Score and Single Model Approach Based on Test Set Accuracy

Approach	Number of Batch	Number of Classifier	Validation Set Accuracy	Test Set Accuracy
Last Batch - Single Model Approach	1	1	0.789	**0.824**
Last Batch - Our System	149	10	0.804	0.784
Random Set - Single Model Approach	1	1	0.796	0.736
Random Set - Our System	147	7	0.808	**0.760**

In Table81, it is observed that when the batch number used by our system is tuned based on the validation set, its performance on the test set, composed of the last batch, is lower compared to the single-model approach. On the other hand, when the test set consists of random instances, the system's performance is better than the single-model approach.

The performance of the model using the prediction strategy from Section 3.6.3 and the training strategy from Section 3.3.1 across different batch number configurations is presented in the Figure78 and Figure79. The validation set was created by reserving the last batch to measure the system's performance.

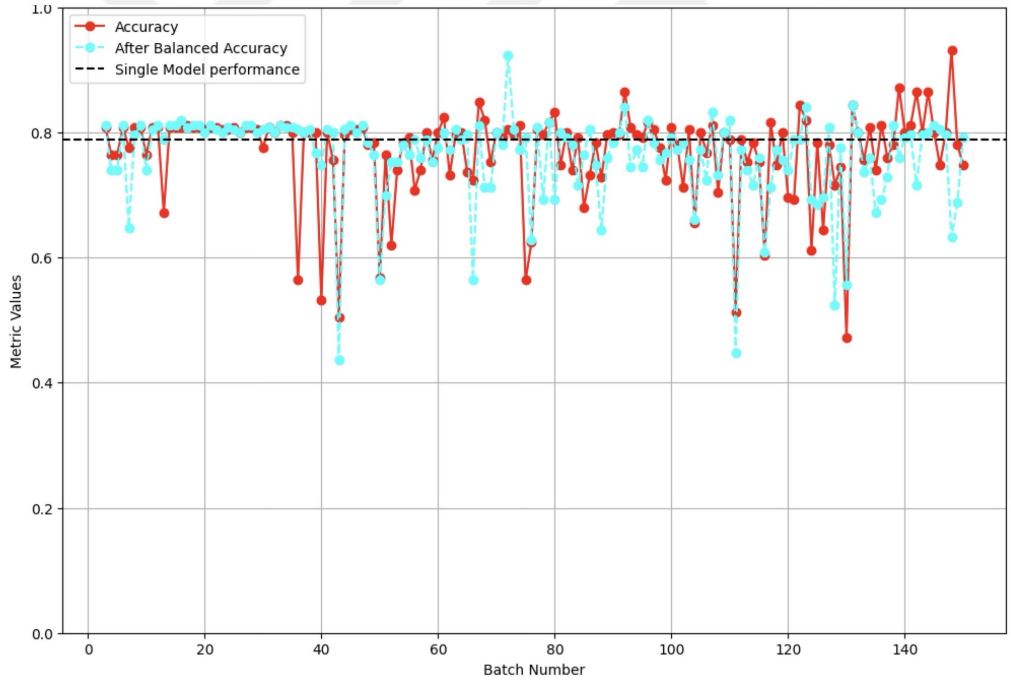


Figure 78: The Performance of The System That Uses the Diversity-Rewarding Prediction Strategy on The Validation Set Which is Last Batch According to The Accuracy Metric Across Different Batch Numbers

Our system that uses the diversity-rewarding prediction strategy described in Section 3.6.3, the fluctuating performance across different batch numbers has continued when evaluated against the validation set created from the last batch according to Figure78 and Figure79.

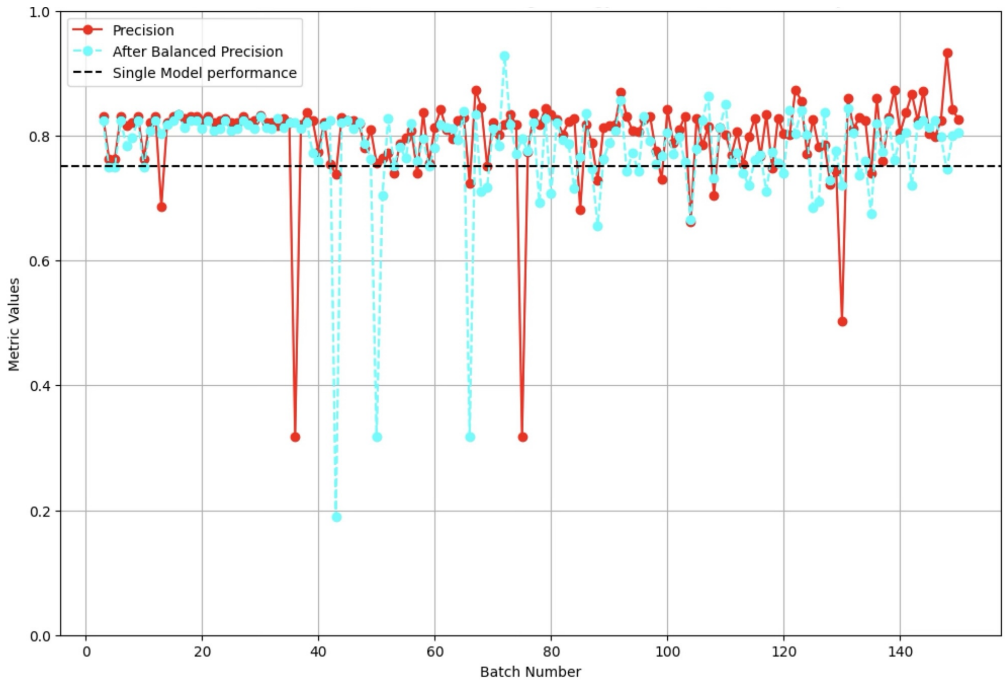


Figure 79: The Performance of The System That Uses the Diversity-Rewarding Prediction Strategy on The Validation Set Which is Last Batch According to The Precision Metric Across Different Batch Numbers

The performance of the model using the prediction strategy from Section 3.6.3 and the training strategy from Section 3.3.1 across different batch number configurations is presented in Figure80 and Figure81. The validation set was created by randomly choosing instances from dataset to evaluate our system's performance.

As shown in the Figure80, performance of model is lower than single model approach except two batch number specification based on accuracy metric. Also, as seen in the Figure81, according to the precision metric, the system did not achieve better results than the single-model approach for any batch number configuration. When comparing the cases where the system uses the prediction strategy from Section 3.6.3, it was observed that the system performed worse when the validation set consisted of random instances compared to when the last batch was used as the validation set.

The system using the prediction strategy from Section 3.6.3 is tuned based on the batch number that yielded the best accuracy on the validation set. The results are shown in the Table82.

Table 82: Comparison of Tuned Version of the Systems That Use Prediction Strategy Based on Centroid Difference and Diversity and Single Model Approach Based on Test Set Accuracy

Approach	Number of Batch	Number of Classifier	Validation Set Accuracy	Test Set Accuracy
Last Batch - Single Model Approach	1	1	0.789	**0.824**
Last Batch - Our System	148	11	0.932	0.596
Random Set - Single Model Approach	1	1	0.796	0.736
Random Set - Our System	149	10	0.804	**0.784**

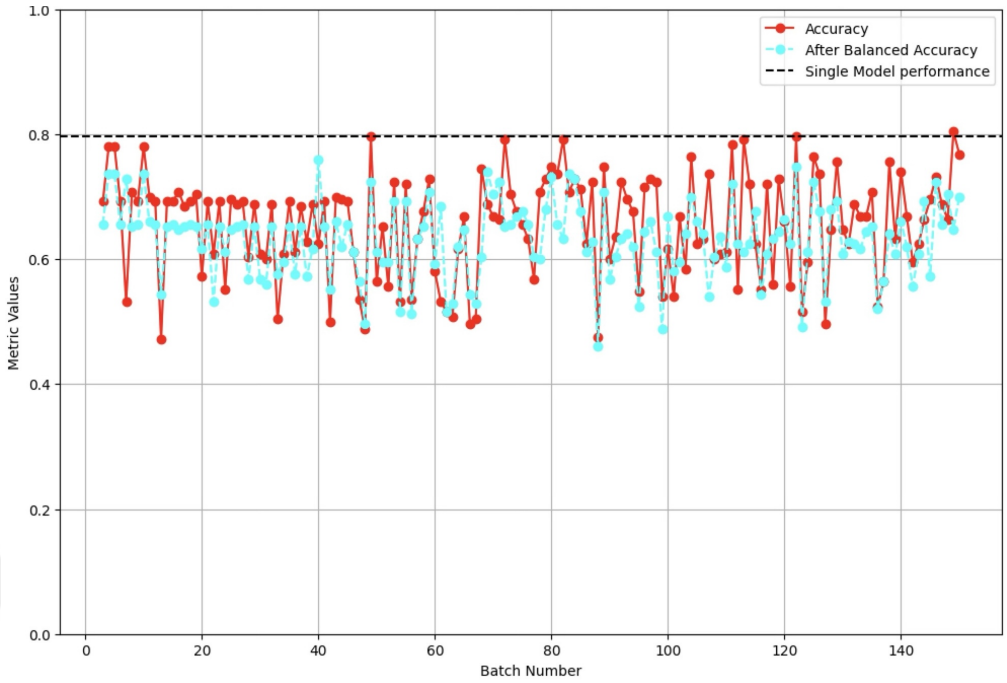


Figure 80: The Performance of The System That Use Prediction Strategy Based on Centroid Difference and Diversity on The Validation Set Which Consist Random Instances According to The Accuracy Metric Across Different Batch Numbers

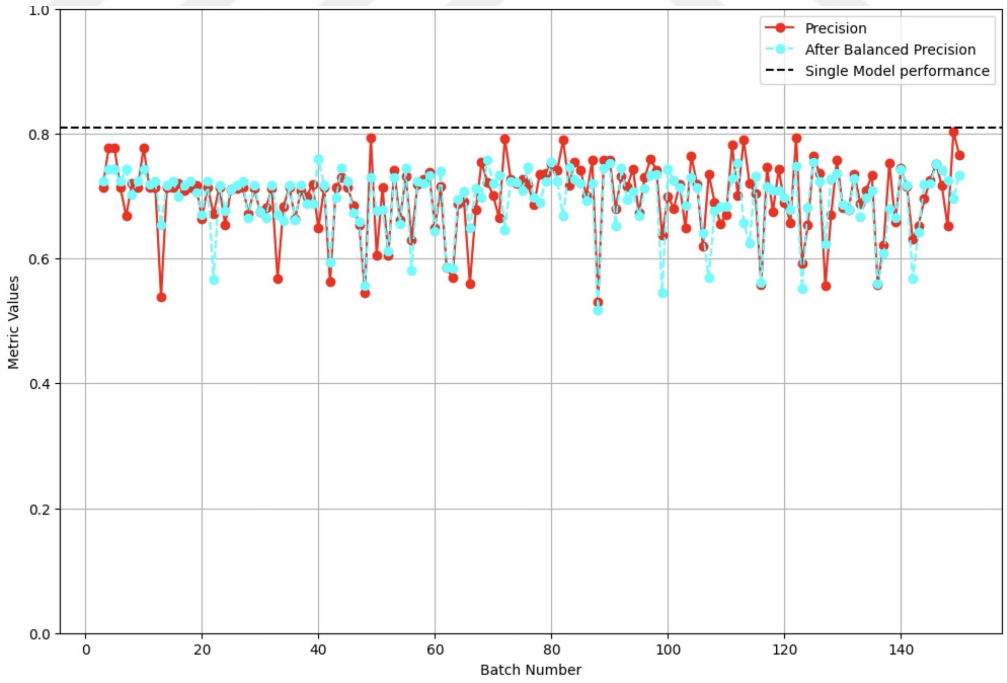


Figure 81: The Performance of The System That Uses Prediction Strategy Based on Centroid Difference and Diversity on The Validation Set Which Consist Random Instances According to The Precision Metric Across Different Batch Numbers

In the Table82, it is seen that our system performed poorly against the test set composed of the last batch, with performance falling below that of the single-model approach. On the other hand, our system received better results compared to the single-model approach when evaluated against a test set composed of randomly selected instances.

The performance of the system, which uses the prediction strategy described in Section 3.6.5 and is trained with the training strategy explained in Section 3.3.1, on the validation set created using the last batch method is shown in the figures which are Figure82 and Figure83.

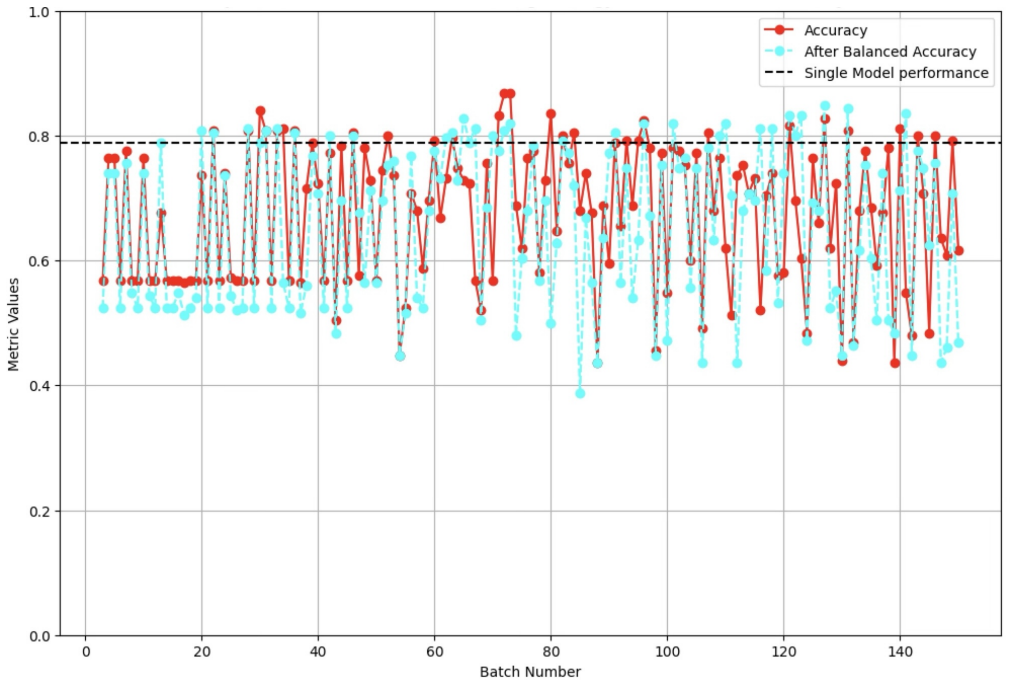


Figure 82: The Performance of The System That Uses Prediction Strategy Based on Statistical Test on The Validation Set Which is Last Batch According to The Accuracy Metric Across Different Batch Numbers

As seen in the figures which are Figure82 and Figure83, the system has performed better than the single-model approach in terms of accuracy and precision metrics across many batch numbers on the validation set created using the last batch method. On the other hand, it has experienced significant performance drops in several batch number configurations.

The performance of our system which predicts by using the prediction strategy described in Section 3.6.5 and is trained with the training strategy explained in Section 3.3.1, has been measured on a validation set consists of random instances from the dataset based on accuracy and precision metrics. The figures which are Figure84 and Figure85 show the performance across different batch number configurations.

In Figure84, it is seen that our system's performance generally are lower than single model approach's performance according to accuracy metric. On the other hand, our system's performance are better than single model approach performance in a few points. According to Figure85, our model's precision performance is lower than single model approach's performance in any batch number configuration.

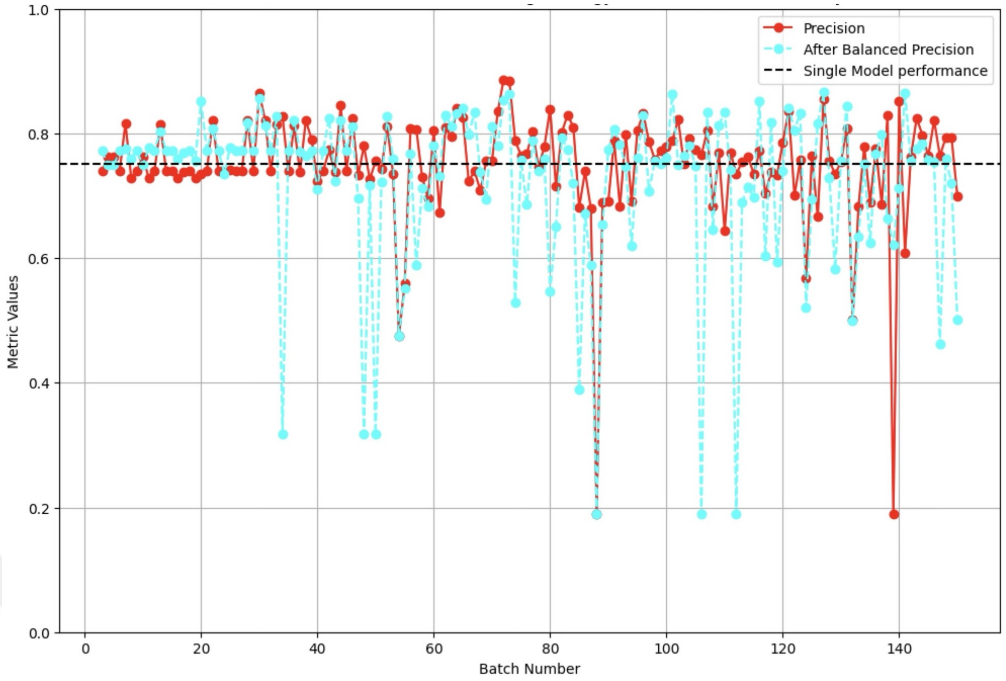


Figure 83: The Performance of The System That Uses Prediction Strategy Based on Statistical Test on The Validation Set Which is Last Batch According to The Precision Metric Across Different Batch Numbers

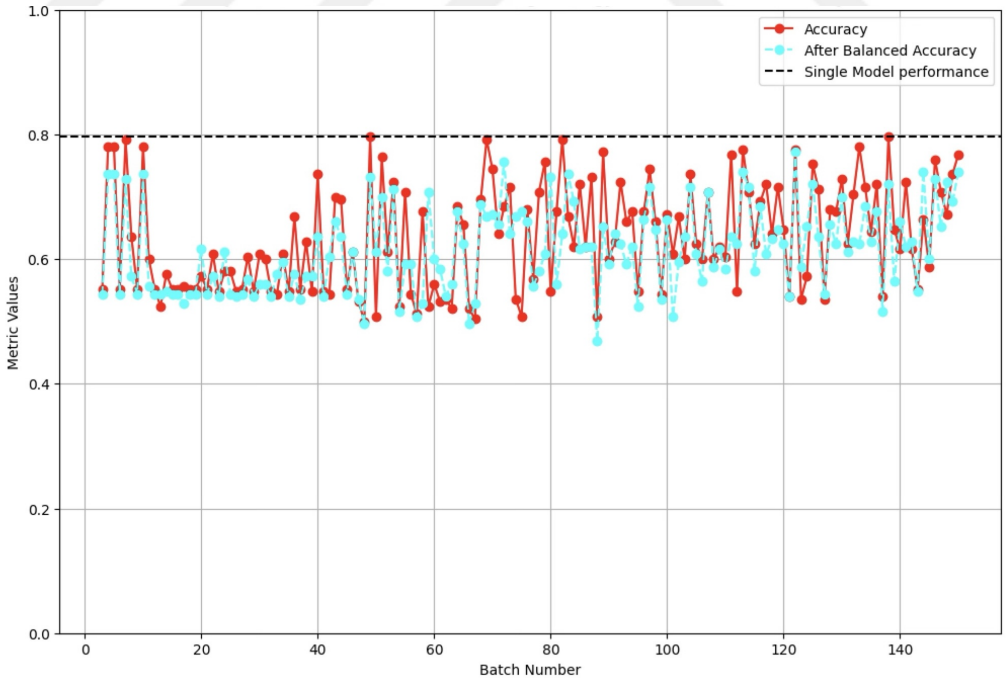


Figure 84: The Performance of The System That Uses Prediction Strategy Based on Statistical Test on The Validation Set Which Consist Random Instances According to The Accuracy Metric Across Different Batch Numbers

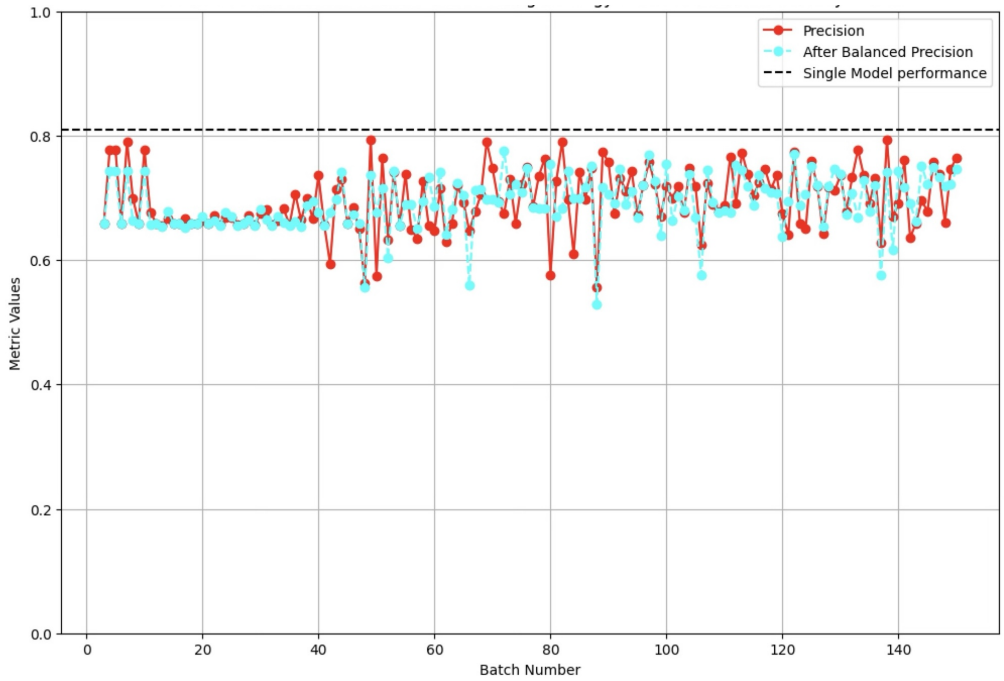


Figure 85: The Performance of The System That Uses Prediction Strategy Based on Statistical Test on The Validation Set Which Consist Random Instances According to The Precision Metric Across Different Batch Numbers

Table 83: Comparison of Tuned Version of the Systems That Use Prediction Strategy Based on a Statistical Test and Single Model Approach Based on Test Set Accuracy

Approach	Number of Batch	Number of Classifier	Validation Set Accuracy	Test Set Accuracy
Last Batch - Single Model Approach	1	1	0.789	**0.824**
Last Batch - Our System	148	11	0.868	0.804
Random Set - Single Model Approach	1	1	0.796	0.736
Random Set - Our System	49	5	0.796	**0.764**

As shown in the Table83, although our model, when the batch number is tuned, demonstrates performance close to the single-model approach on the test dataset created using the last batch method, its performance remains lower than that of the single-model approach. On the other hand, the performance of our model on the test dataset created using the last batch method is better than that of the single-model approach.

The performance of the model using the prediction strategy that described in Section 3.6.6 and the training strategy that is explained in Section 3.3.1 among different batch number configurations is shown in Figure86 and Figure87. The validation set was created using the last batch method.

As seen in Figure86, our system has exhibited fluctuating performance according to the accuracy metric. Although it achieved better results for certain batch numbers, it performed worse in most batch number configurations. As shown in Figure87, our system has exhibited fluctuating performance according to the precision metric. Before the 52^{nd} batch, the fluctuations occurred within a narrower range, but as the batch number increased, the range of these fluctuations widened.

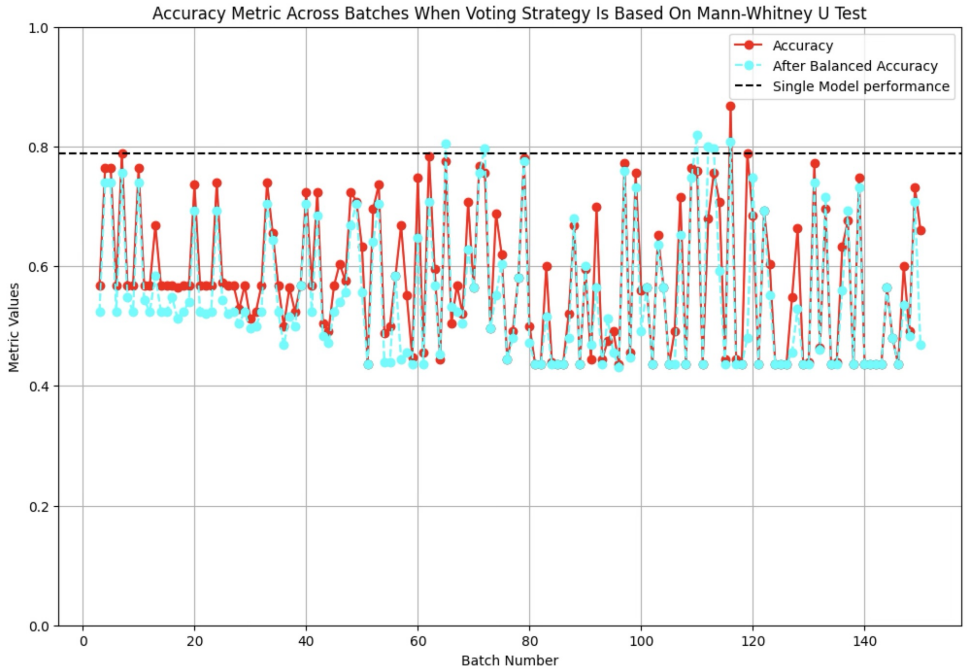


Figure 86: The Performance of The System That Uses Prediction Strategy Based on Voting Right Constraints on The Validation Set Which is Last Batch According to The Accuracy Metric Across Different Batch Numbers

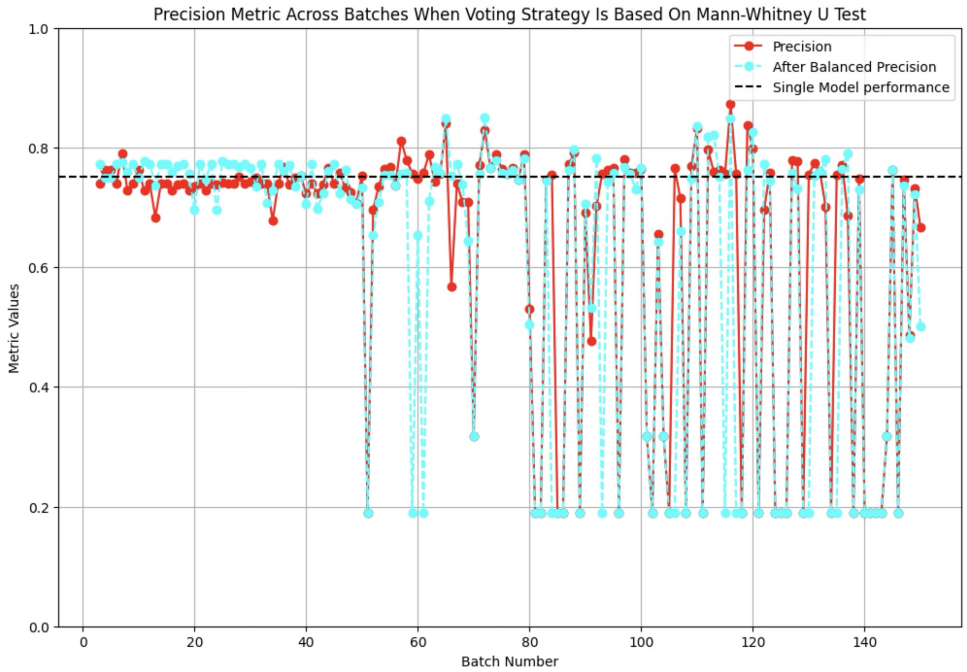


Figure 87: The Performance of The System That Uses Prediction Strategy Based on Voting Right Constraints on The Validation Set Which is Last Batch According to The Precision Metric Across Different Batch Numbers

The performance of the model using the prediction strategy that explained in Section 3.6.6 and the training strategy that is mentioned in Section 3.3.1 among different batch number settings is illustrated in the figures which are Figure88 and Figure89. The validation set consists of random instances from dataset.

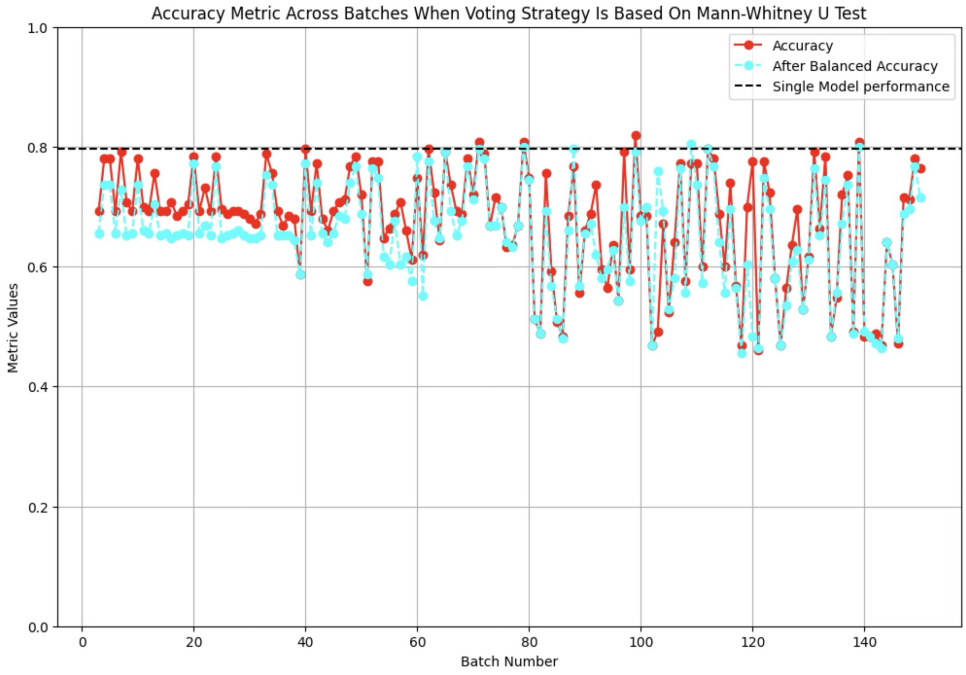


Figure 88: The Performance of The System That Uses Prediction Strategy Based on Voting Right Constraints on The Validation Set Which Consist Random Instances According to The Accuracy Metric Across Different Batch Numbers

Upon looking the figures which are Figure88 and Figure89, it is observed that the system has shown fluctuating performance according to both accuracy and precision metrics. While it achieved better results than the single-model approach for certain batch numbers, it produced worse results for other specific batch numbers.

The system using the prediction strategy from Section 3.6.6 is tuned based on the batch number that yielded the best accuracy on the validation set. The results are shown in the Table84.

Table 84: Comparison of Tuned Version of the Systems That Use Prediction Strategy Based on Give One Vote Right Each to Models Trained on Datasets That Are Not In Different Concept and Single Model Approach Based on Test Set Accuracy

Approach	Number of Batch	Number of Classifier	Validation Set Accuracy	Test Set Accuracy
Last Batch - Single Model Approach	1	1	0.789	0.824
Last Batch - Our System	116	8	0.868	**0.852**
Random Set - Single Model Approach	1	1	0.796	0.736
Random Set - Our System	99	9	0.820	**0.776**

As seen in Table84, when the batch number parameter of our model, which uses the prediction strategy described in Section 3.6.6, is tuned based on its performance on the validation set, it demonstrated

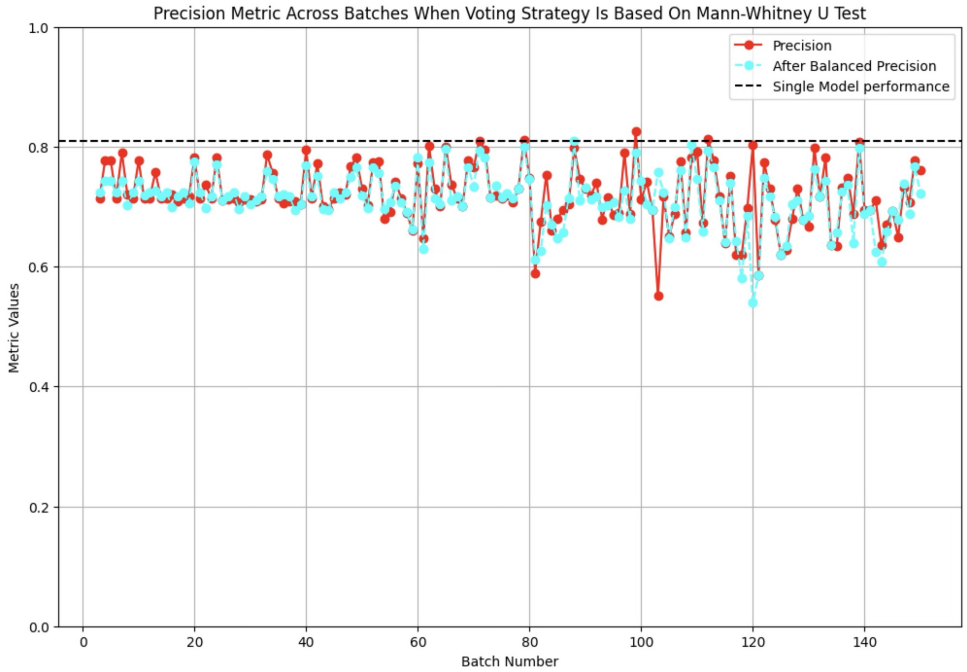


Figure 89: The Performance of The System That Uses Prediction Strategy Based on Voting Right Constraints on The Validation Set Which Consist Random Instances According to The Precision Metric Across Different Batch Numbers

better performance compared to the single-model approach on both the test dataset consisting of the last batch and the test dataset composed of random instances.

The system made predictions by using prediction strategy which is described in Section 3.6.4 for validation set which is formed by splitting last batch from dataset. Also, the system's performance was recorded under different batch number configurations. The performance of the system is illustrated according to accuracy and precision metrics in Figure90 and Figure91.

As seen in Figure90, the performance of the system is fluctuating while batch number specification is changing. There are lots of batch number configurations that provide better results than single model approach. However, the system's performance with some of the batch number specifications are lower than single model. According to Figure91, the system's performance is not stabilized according to precision metric. Also, fluctuations can be observed in larger ranges after specific batch number configuration which is 45^{th} batch.

The performance of the system which make predictions by using strategy which explained in Section 3.6.4 was measured according to accuracy and precision metric. To measure their performance an validation set which is consist of random instances from dataset was used. The performance of the system under different batch number configurations is illustrated in Figure92 and Figure93.

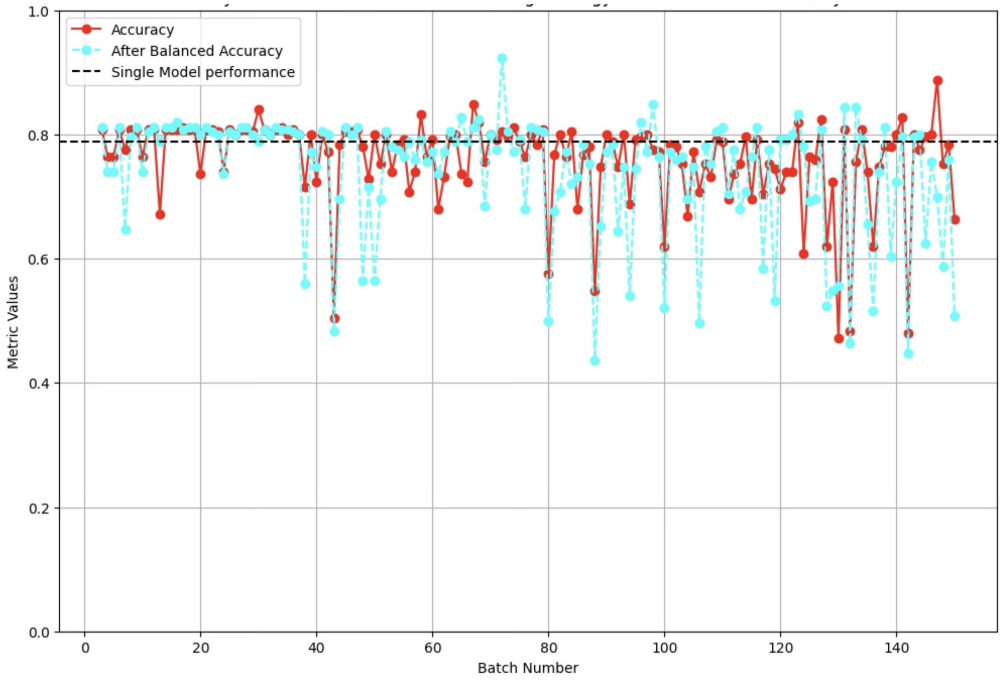


Figure 90: The Performance of The System That Uses Prediction Strategy Based on The Jensen-Shannon Divergence Score on The Validation Set Which is Last Batch According to The Accuracy Metric Across Different Batch Numbers

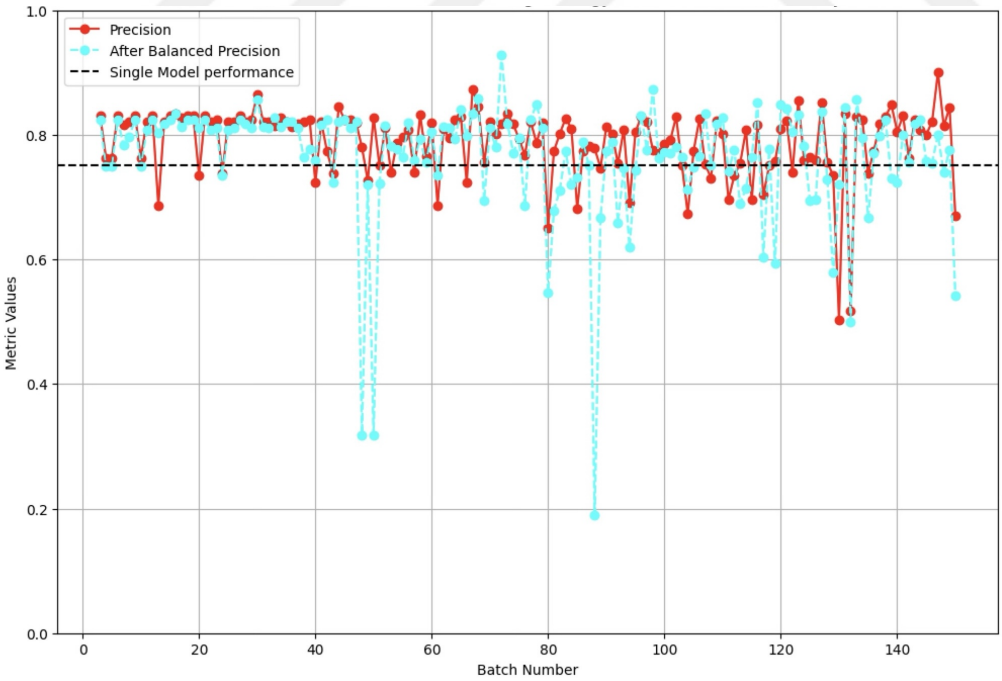


Figure 91: The Performance of The System That Uses Prediction Strategy Based on The Jensen-Shannon Divergence Score on The Validation Set Which is Last Batch According to The Precision Metric Across Different Batch Numbers

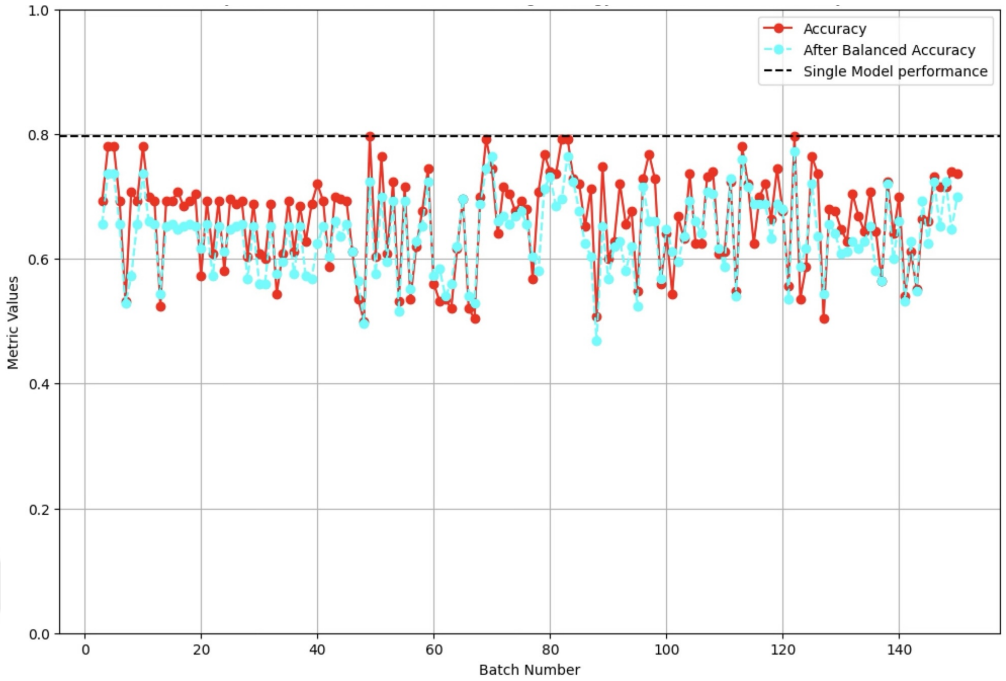


Figure 92: The Performance of The System That Uses Prediction Strategy Based on The Jensen-Shannon Divergence Score on The Validation Set Which Consist Random Instances According to The Accuracy Metric Across Different Batch Numbers

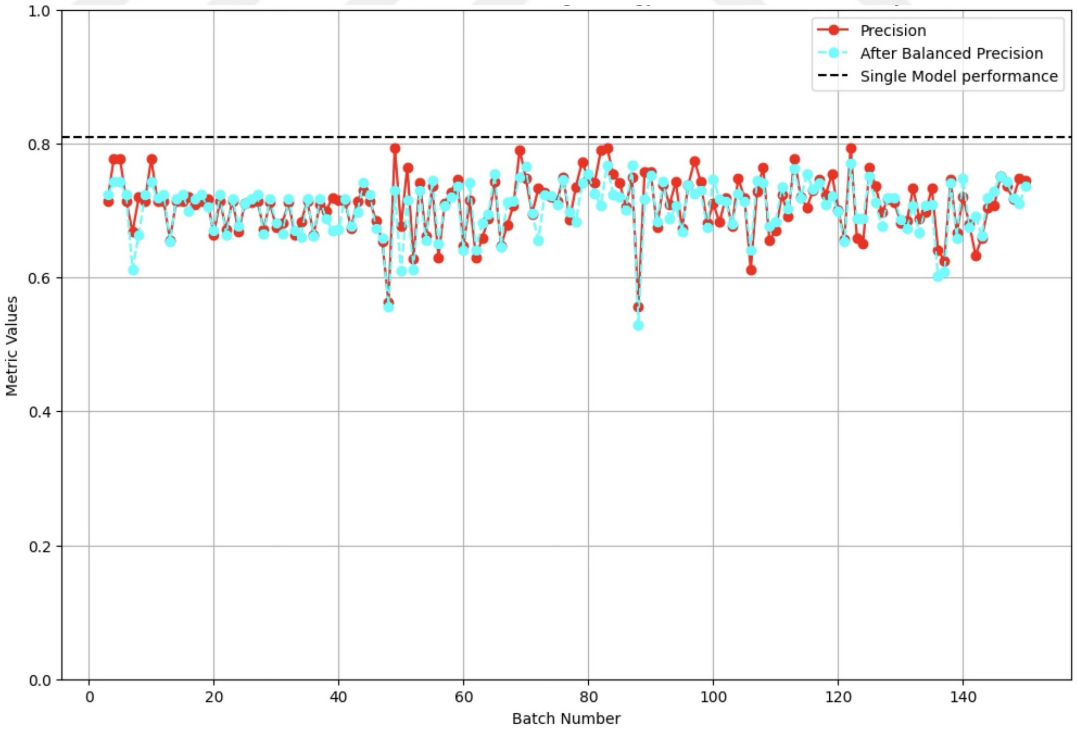


Figure 93: The Performance of The System That Uses Prediction Strategy Based on The Jensen-Shannon Divergence Score on The Validation Set Which Consist Random Instances According to The Precision Metric Across Different Batch Numbers

In Figure92, it is seen that our system's performance with most of the batch number configurations are lower than single model approach. On the other hand, our system's performance can exceed performance of single model approach by only one batch number specification among 2 to 150. In Figure93, it is observed that our system's precision performance is always lower than single model approach.

Our system's performance on test sets when the system's batch number is tuned based on performance on validation set is given in Table85.

Table 85: Comparison of Tuned Version of the Systems That Use Prediction Strategy Based on The Jensen-Shannon Divergence Score and Single Model Approach Based on Test Set Accuracy

Approach	Number of Batch	Number of Classifier	Validation Set Accuracy	Test Set Accuracy
Last Batch - Single Model Approach	1	1	0.789	**0.824**
Last Batch - Our System	147	7	0.796	0.760
Random Set - Single Model Approach	1	1	0.796	0.736
Random Set - Our System	49	5	0.796	**0.764**

As seen in Table85, the single model approach's performance is better than our system that uses prediction strategy that explained in Section 3.6.4 on test set that is formed by splitting last batch even our model's batch number is tuned. On the other hand, our system's performance is better than single model approach on test set consists random instances when our system's batch number is tuned.

CHAPTER 5

DISCUSSION

5.1 Concept Drift Scenarios

In this section, concept drift scenarios and our system's performance against them is discussed. Four different concept scenarios were simulated using synthetic data, and the performance of our system is reported in Section 4.1. Based on these experiments, the performance of different training strategies used by the system in these concept scenarios is discussed in the relevant subsections below.

5.1.1 Sudden Drift

Sudden drift is a rapid and significant change in the underlying distribution of a dataset that can abruptly degrade the performance of a predictive model. Our system aims to adapt to sudden changes and perform well in scenarios where sudden drift occurs.

The two separate strategies described in Section 3.3.1 and Section 3.3.2 have been tested in the context of sudden drift. In the experiments where sudden drift occurred, the system used the prediction strategy described in Section 3.6.1.

Sudden drift experiments are introduced in Section 4.1.1. In the sudden drift experiments, the Test 2 set shares the same concept as the concept that is less represented in the training dataset. The system has been more successful in predicting the Test 2 dataset compared to the single model approach in all experiments. The system has been more successful than the single model approach in understanding the changing concept in all sudden drift scenarios, particularly in understanding the concept that is less represented in the training dataset. The reason for this is that the single model approach tends to generalize the concept that is predominantly present in the training dataset across all datasets. In contrast, our system's approach trains separate models for each concept, enabling it to understand even those concepts that are underrepresented in the training dataset.

In the first sudden drift experiment, the system was less successful than the single model approach in predicting the test set with the concept predominantly present in the training dataset. In the second experiment, it showed the same performance than the single model approach. In the final experiment, the system achieved better results than the single model approach. Although the system experienced performance drops in some cases when predicting test sets with concepts that are prevalent in the training dataset compared to the single model approach, it generally achieved results that are better

than results of the single model approach. To understand the reason for this, the specific concept drift characteristics in each experiment should be examined.

In Experiment 1, because the classes evolved towards same direction, the single model approach was able to generalize the predominantly present concept in the training dataset effectively. On the other hand, although the system which used training strategy described in Section 3.3.1 is capable of understanding each concept separately, it does not have the advantage of the single model approach in better understanding the dominant concept in the training dataset. The single model approach generalized all data closer to the dominant concept, resulting in a performance drop when predicting the less prevalent concept in the training dataset, and therefore, it performed worse than our system. The system using the accuracy-based training strategy described in Section 3.3.2 has been more successful than the single model approach in predicting the test set with the concept predominantly present in the training dataset, even in sudden drift scenarios.

In Experiment 2, because the classes evolved towards distant points from each other, the concept drift has further separated the class boundaries. Rather than making the task more difficult, the sudden drift has actually made it easier for the models. Consequently, both the single model approach and our system with its two different training strategies demonstrated high performance. This experiment shows that when such a concept drift occurs, our system does not decrease the performance of the single model approach. It also indicates that using the system is not a disadvantage in scenarios where sudden drift does not pose a problem.

In Experiment 3, the concepts of Classes A and B underwent a sudden change, causing them to converge towards each other. In this scenario, the single model approach became less effective at understanding the predominantly present concept in the training dataset. In contrast, the system using the training strategy described in Section 3.3.1 produced better results than the single model approach in predicting the test set with the concept predominantly present in the training dataset. In contrast, our system was less successful than the single model approach in predicting a concept not present in the training dataset. This could be because the test set was situated in a concept close to the area where the two classes converged in the analytical plane. Since the single model approach tends to generalize all data to the concept in the area where the two classes converged, it was more successful in predicting a test set close to that concept. However, the system using the training strategy described in Section 3.3.2 has overcome this situation and achieved better results compared to the single model approach. This can be attributed to its focus on the performance of individual models. If a model performs worse than the single model approach on a new dataset, the model that uses accuracy-based training strategy assumes that the model does not understand that concept. Consequently, a separate model is developed for that specific concept. Therefore, the system using the training strategy described in Section 3.3.2 can predict better.

When the balancing function which described in Section 3.4was used, it did not yield better results in any of the sudden drift experiments compared to the systems where this function was not applied. In some experiments, it even produced worse results on certain datasets. Therefore, it can be assumed that balancing the training datasets is not an effective method in sudden drift scenarios.

In conclusion, our system using the accuracy based training strategy explained in Section 3.3.2 performed better towards sudden drift event than both the system using the training strategy described in Section 3.3.1 and single model approach. Balancing training sets did not improve the performance of the system that using any training strategy.

5.1.2 Gradual Drift

Gradual drift occurs when changes in the data's underlying distribution emerge progressively over a period rather than through a swift, definitive shift in concept. In this type of drift, the concept of data might temporarily revert to previous states before fully transitioning to a new concept. Throughout this transition, the data may alternate between the previous and current concepts, reflecting a slow and fluctuating shift towards a new state.

Different strategies explained in Section 3.3.1 and Section 3.3.2 have been examined in the context of gradual drift. In the experiments where gradual drift event are simulated, the system used the prediction strategy described in Section 3.6.1.

Gradual drift experiments are introduced in Section 4.1.2. In these experiments, the pre-concept drift concept is presented to the system in 5 batches, and the post-concept drift concept is presented in 6 batches. The total number of instances in the training dataset with the pre-concept drift concept is 1280. On the other hand, the number of instances with the post-concept drift concept is 1637. The prevalence of a particular concept in the training dataset will affect the performance of the single model approach. However, since the prevalence of the two concepts is not equal but relatively close to each other, this situation is not expected to create a disadvantage for the single model approach.

The Test 1 dataset represents the concept formed after the concept drift, while the Test 2 dataset represents the concept before the concept drift. The concept of the Test 3 dataset does not match any concept within the training data, but it is closer to the concept formed after the concept drift.

Our system using the training approach from Section 3.3.1 has never performed worse than the single model approach on Test 1 in any experiment design. Except for Experiment 3, its performance on Test 2 was better than that of the single model approach. The superiority between our system using the training approach from Section 3.3.1 and the single model approach varies for the Test 1 dataset. In Experiment 1, where all classes gradually increased at different rates, the single model approach yielded better results. In Experiment 2, which simulated a gradual drift where some classes increased and others decreased, our system performed better. In Experiment 3, some classes increased while others decreased, and additionally, the new concepts of the increasing and decreasing classes became identical. In this experiment design, our system using the training strategy from Section 3.3.1 did not show outstanding performance but was still better than the results of the single model approach. In the Test 3 dataset, which represents a concept that is not present in the training dataset but is an average of the two concepts present in the training dataset, the single model approach was successful in Experiment 1, while our system using the training method from Section 3.3.1 was successful in Experiment 2. In Experiment 3, both the single model approach and our system correctly predicted the entire Test 3 dataset. Based on these results, it can be concluded that while our system using the strategy from Section 3.3.1 generally shows improved performance over the single model approach in gradual drift scenarios, it cannot be said that its performance will always be better in every situation.

The impact of balancing the training datasets on the performance of the system using the training strategy from Section 3.3.1 is not consistent across all experiment designs. In Experiment 1, the performance on the Test 1 and Test 2 datasets remained unchanged, while the performance on the Test 3 dataset improved. In Experiment 2, the performance on the Test 1 dataset remained unchanged, but there was a performance drop on the Test 2 and Test 3 datasets. In Experiment 3, the performance remained the same across all three test datasets representing different concepts. Based on these results,

applying the balancing method from Section 3.4 to the training datasets of the system using the training strategy from Section 3.3.1 in gradual drift scenarios can lead to performance improvements under certain conditions, but it is also possible that it may reduce performance in some cases.

In Experiment 1, our system using the training strategy from Section 3.3.2 performed better than the single model approach on the Test 1 and Test 2 datasets, but its performance was lower on the Test 3 dataset. Although it performed worse on the Test 3 dataset compared to the single model approach, its performance showed a slight improvement compared to the system using the training strategy from Section 3.3.1. In Experiment 2, which had a design where the concepts of some classes increased while others decreased, our accuracy-based system correctly predicted all instances in the test sets representing all concepts. It outperformed both the single model approach and the system using the training strategy from Section 3.3.1. In the gradual drift scenario where different classes converged to the same concept after the concept drift, the performance was poor. Although it performed better on the Test 1 dataset compared to the single model approach and the system using the training strategy from Section 3.3.1, its performance on the Test 2 and Test 3 datasets was significantly inadequate. Based on these results, while the accuracy-based approach is successful in most gradual drift scenarios, it experiences significant performance loss in situations where the concepts of different classes become intermixed.

Balancing the training datasets of the system using the training strategy described in Section 3.3.2 with the approach from Section 3.4 had no positive or negative impact on performance in any of the gradual drift scenarios.

5.1.3 Incremental Drift

Incremental drift occurs when the underlying distribution of a dataset shifts slightly but consistently over time. These subtle shifts lead to a continuous evolution in data patterns, often complicating the detection and necessary adaptations.

To solve problems that are caused by incremental drift, strategies explained in Section 3.3.1 and Section 3.3.2 have been used in the experiments that have context of gradual drift. In the experiments where gradual drift event are simulated, the system used the prediction strategy described in Section 3.6.1.

Incremental drift experiments are introduced in Section 4.1.3. In these experiments, the training dataset includes 1 batch from the old concept and 1 batch that has completely transitioned to the new concept. Along with batches that change as they approach the new concept, a total of 9 batches were provided to the system as the training dataset. n these experiments, the Test 1 dataset was used to test the system's performance on the new concept, the Test 2 dataset was used to test its performance on the old concept, and the Test 3 dataset was used to test its performance on a concept that is not present in either but lies between the two concepts.

Our system, which uses the training strategy that described in Section 3.3.1 and prediction approach described in Section 3.6.1, has consistently outperformed the single model approach across all test sets in all experiments. While this performance advantage is small in scenarios where the single model approach works well, it is significant in experiments where the single model approach fails.

This demonstrates the benefit of our system in incremental learning scenarios where the single model approach is insufficient.

Our system, which uses the training strategy that described in Section 3.3.2 and prediction strategy described in Section 3.6.1, has consistently outperformed the single model approach across all test sets in all experiments. Furthermore, the system using the accuracy-based approach has never performed worse than the system using the training strategy from Section 3.3.1 and has sometimes demonstrated superior performance. In Experiment 1, both systems produced the same result on the Test 2 dataset, while the accuracy-based strategy outperformed on Test 1 and Test 3. In Experiment 2, the accuracy-based strategy performed better on all test sets. In Experiment 3, the accuracy-based strategy performed better on Test 1 and Test 3, while achieving the same result as the other system on Test 2.

Based on these results, it can be argued that the training strategy described in Section 3.3.2 is more successful in incremental drift scenarios compared to the training strategy described in Section 3.3.1.

In the experiments simulating incremental learning scenarios, applying the approach described in Section 3.4 to balance the training datasets had no impact on the system's performance.

5.1.4 Reoccurring Drift

Reoccurring drift refers to a concept drift phenomenon where the concept changes and then returns to its initial concept after some time. This specific type of concept drift occurrence is generally seen in dataset that contains seasonal, and iterative events.

We tried two different training strategy which are described in Section 3.3.1 and Section 3.3.2 for addressing reoccurring drift. In reoccurring drift experiments, these training strategies are used with prediction method which is explained in Section 3.6.1. By this way, we can compare our different training strategies against reoccurring drift.

Reoccurring drift experiments are presented in Section 4.1.4. In these experiments, the initial concept is represented by first four batches in the training dataset. drifted concept is represented by following three batches. Last tree batches have concept same as initial concept. The training dataset following scenario that the pre-concept drift concept returns in accordance with reoccurring drift scenario.

The Test 1 dataset represents the concept formed after the concept drift, while the Test 2 dataset represents the concept before the concept drift was occurred. The Test 3 dataset was used to test system's performance on a concept that is not present in either but lies between the two concepts.

In experiment 1 which described in Section 4.1.4.1, our system that used training strategy which described in Section 3.6.1 has better performance than single model approach on Test 1 dataset, but its performance lower than single model approach on Test 2 and Test 3 datasets. When system's datasets are balanced by method that described in Section 3.4, its performance exceeds single model approach on Test 2 dataset. Also, its performance increased on Test 3 dataset. The single model is problematic for making prediction about dataset that has concept that is less represented in dataset. Our system fix this issue. However, as our method split training dataset and assign them to separate models, it may cause to implementing training with unbalanced dataset. When we solved this problem by balancing training datasets, our system's performance exceeded single model's performance. The system that used accuracy based training strategy that is described in Section 3.3.2 did not faced with same prob-

lem. Its performance better than single model approach on both Test 1 and Test 2. Balancing datasets did not change its results. However, our both training strategies are not successful as single model approach for predicting concept that is not represented in training set, but lies between the two concepts that are represented in training set. We can claim that our system is successful against reoccurring drift which is in an increasing manner when there are relative concept sample with concept of dataset that will be predicted in training set.

In experiment 2 which is described in Section 4.1.4.2, while the features defining the concepts of some classes will increase, this change will manifest as a decrease for other classes. The training method that explained in Section 3.6.1 showed better performance than single model approach on Test 1 dataset. However, its performance lower than single model approach on Test 2 and Test 3 datasets. Balancing datasets of this system cause that its performance exceeded performance of single model approach on Test 2 dataset. The system that used accuracy based training strategy that is described in Section 3.3.2 has better performance on Test 1 and Test 2 dataset. However, both training strategies have lower performance than single model approach on Test 3.

In experiment 3 which is described in Section 4.1.4.3, an extra challenge has been introduced to test our system. The concepts for some classes will become identical to the concepts of other classes after the concept drift. The training method that explained in Section 3.6.1 have better performance on Test 1 dataset, but its performance lower than single model approach on Test 2 and Test 3 datasets. The problem about Test 2 is solved after balancing datasets. However, balancing datasets decreased its performance on Test 3. The performance of the system that used accuracy based training strategy that is explained in Section 3.3.2 is better on Test 1 and Test 2 dataset. On the other hand, both training strategies have lower performance than single model approach on Test 3.

In conclusion, our both training strategies can gather results better than single model approach for predicting datasets whose concept is represented in training set. However, their performance is insufficient on test sets that does not have concept that is represented in training dataset.

5.2 Training Strategies

In the scope of this thesis, two different training approaches which are described in 3.3 are developed. To compare their performances, experiments were conducted where the prediction strategies were kept constant and the training approach was changed. In these experiments, the prediction strategy that assigns weights based on the centroid difference between the datasets was used as the fixed prediction strategy. In the experiments, while using the training strategy described in Section 3.3.1, two different threshold values—MAD and $2 \times$ MAD—were used. Additionally, the training strategy described in Section 3.3.2 was tested on the same dataset.

Both training approaches faced the same fundamental issue. Both training strategies experienced performance drops depending on the batch number configuration. In the experiments described in Section 4.2.1, we observed that the performance of both approaches, with a batch number hyper-parameter ranging from 2 to 150, fluctuated when evaluated on both the validation set created using the last batch method and the validation set created by selecting random instances.

In all of these experiments, we observe that the system using the training strategy from Section 3.3.1, in both cases where it employed either threshold value, was able to outperform the single-model ap-

proach in at least one batch number configuration according to accuracy, recall and precision metrics. Additionally, we see that the accuracy-based approach described in Section 3.3.2 outperforms performance of the single model approach in terms of accuracy, recall, and precision metrics in at least one batch number configuration in all experiments. Analyzing the results of these experiments, it can be said that both training strategies can achieve better results than the single-model approach when an appropriate batch number is selected. However, they are also faced with significant performance drops in some batch number configurations. This emphasizes the importance of selecting the appropriate batch number configuration.

When batch number tuning was applied by selecting the batch number that gave the best performance on the validation set to be used on the test set, both training strategies successfully outperformed the single model approach on both the test set created from the last batch and the test set composed of random instances. This results demonstrate how the performance drops experienced due to batch number configurations can be mitigated.

When comparing the performance of these two different training strategies, we observe that the system using the training strategy from Section 3.3.1 performs better on the test set created from the last batch, while the system using the training strategy from Section 3.3.2 delivers better results on the test set consists of random instances when they are tuned. Upon examining the best performances on the validation set, it can be said that the system using the training strategy described in Section 3.3.1 and employing MAD as the threshold has the potential to achieve higher performance on the validation set composed of random instances. When looking at the best performances on the validation set created by separating the last batches, the system using the training strategy from Section 3.3.1 and employing $2 \times$ MAD as the threshold shows greater potential. On the other hand, their best results are quite close to each other. Based on the results of the experiments, there is not enough evidence to claim that one of the two training strategies is superior to the other.

In conclusion, both training strategies outperform the single-model approach on datasets with concept drift when the batch number is tuned. Our system mitigates the effects of the concept drift problem by training a separate model for each concept. Although the decision on whether a new concept has emerged during the training phase leads to small performance differences, it can be said that the main approach which is creating a separate learner for each concept provides a solution to the concept drift problem.

5.3 Prediction Strategies

In this study, we tried 6 different prediction strategy which are described in Section 3.6. Each of the training strategy was used with training strategy which described in Section 3.3.1, and by using threshold value which is specified as $2 \times$ MAD to determine whether concept drift occur. As they have tried with same training strategy, it is possible to compare them.

Each of the prediction strategies have its own minor differences in performance, but they all share two common and fundamental characteristics. Firstly, different prediction approaches have not been able to prevent performance fluctuations that occur depending on the batch number configurations. Secondly, all prediction strategies have succeeded in outperforming the single-model approach when optimal hyper-parameters are determined. We can conclude this from the fact that in all experiments, when testing different batch number configurations from 2 to 150, each strategy surpassed the performance

of the single-model approach in at least one batch number configuration. Based on these two fundamental and common observations for each prediction strategies, it can be concluded that the system can outperform the single-model approach when suitable hyper-parameters are determined. In other words, The issue of our system's fluctuating performance depending on the batch number configuration is more related to whether the batch number has been correctly determined rather than the specific characteristics of the prediction strategies. This issue can be solved by tuning the batch number.

To tune the batch number configuration, the system's performance is observed on validation set, and the batch number configuration that demonstrated the best performance on the validation set according to the accuracy metric has been selected. This process was implemented with two different validation set which are formed by selecting random instances and splitting last batches, and tested with their respective test sets that formed by same method as validation set.

As a result of the experiments conducted with the tuning process mentioned above, it has been observed that when the batch number is tuned, all prediction strategies performed better on the test set created by selecting random instances compared to the single-model approach. On the other hand, all systems, except for the two which are using the prediction strategy described in Section 3.6.6 that applies the Mann-Whitney U test to decide whether or not to give voting rights to the models created during training phase, and prediction strategy described in Section 3.6.1 that assigns weight based on centroid difference worse than the single-model approach on the test set composed of the last batch.

The performance of the prediction strategy based on the similarity score described in Section 3.6.2, and the prediction strategy that assigns weights to the models based on the p-score given by the Whitney U-test described in Section 3.6.5, is better on the test set created from the last batch compared to the performance on the test set composed of random instances, when considering the accuracy metric. Also, The performance of the prediction strategy that assigns weights to the models based on the p-score given by the Whitney U-test on the last batch test set is quite close to the performance of the single-model approach. In addition, the performance of the prediction strategy described in Section 3.6.4 on the test set composed of random instances and the test set created by last batching is quite similar. However, the performance of the prediction strategy based on rewarding diversity on model's training sets, as described in Section 3.6.3, has been lower on test set that is last batch. Based on these results, we can assert that, except for the strategy based on rewarding diversity, the performance of the prediction strategies does not decline on the test set composed of the last batch, and that the single-model approach works better on the last batch test set. When the batch number is tuned based on the validation set performance on the last batch test set, it has been observed that the four prediction strategies, which performed worse than the single-model approach, outperformed the single-model approach in certain batch number configurations on the validation set created by separating the last batches. Based on this observation, it can be argued that these prediction strategies also have the potential to surpass the performance of the single-model approach when the most optimal batch number is determined. This situation necessitates a discussion on the batch number tuning method.

Since we determine the batch number based on the best performance on the validation set, if the validation set and test set exhibit very different concept characteristics, the appropriate batch number for the test set may not be selected for the test set by using validation set, and it leads to lower system performance. On the other hand, since two prediction strategies yielded better results than the single-model approach when tuned using the same validation set on the same test set, it can be considered that the issue does not stem from this. In this case, the four prediction strategies that performed worse than the single-model approach when tuned should be questioned.

In conclusion, the two prediction strategies—one based on centroid difference and the other on granting voting rights to models according to the Whitney U-test results—have successfully outperformed the single-model approach on two different test sets when tuned. While more complex tuning mechanisms could be developed, it can be said that these two prediction strategies work sufficiently well with the current tuning approach.

5.4 Effect of Size of Data

It is a matter to consider how much data the system needs to operate with good performance in the case of concept drift because if the system cannot function by using data below a certain threshold, this situation must be identified and the system must be operated accordingly.

In experiments conducted with synthetic data, the number of data points fed into the system was randomly determined. The reason for conducting the experiments this way is to observe the system's performance with different amounts of data. In the three experiments simulating the sudden drift scenario, the total amount of data used was randomly determined and evenly divided into batches. On the other hand, in the experiments simulating the other three scenarios, the number of data points per batch was randomly determined. When examining the results of these experiments, no evidence was found suggesting that performance was affected by the amount of data. However, even though the amount of data used in the experiments was varied, it would not be accurate to draw a definitive conclusion regarding the relationship between data quantity and performance, as the primary purpose of these experiments was to observe the system's performance under different concept drift scenarios. For example, the system's performance was not observed with extremely low amounts of data. Additionally, since the amount of data used was randomly determined, the exact strength of the impact of increasing or decreasing data quantity on performance remains unclear. In this context, while there is no evidence showing that the amount of data affects the system in concept drift scenarios, the experiments were not designed to conclusively demonstrate that the system's performance is independent of data quantity.

In all experiments conducted with the real-word dataset, all 45,312 instances in the dataset were utilized. To ensure fair evaluation of the system's performance, the validation and test sets were separated according to the formation method that is used before training the system began. For example, if the validation and test sets were created using the random instance method, 450 random instances were first selected from the dataset to form the test set, followed by another 450 random instances to form the validation set. On the other hand, if the validation and test sets were created using the last batch method, the last 450 instances of the dataset were used to form the test set, and the next-to-last 450 instances were used to form the validation set. After the validation and test sets were created, the remaining data was used as the training dataset. In other words, in experiments conducted with the real dataset, the total amount of data used by the system remained constant. However, the number of batches sent to the system during training was varied, ranging from 3 to 150, and the system's performance was observed. In other words, the number of data points contained in the batches provided to the system during the experiments was tested in different configurations. When the system's performance was analyzed based on the number of batches used, no clear relationship was observed. In some cases, performance improved with a lower number of batches, while in others, it improved with a higher number of batches. Regardless of the training and prediction strategy employed, the system's performance fluctuated with varying batch numbers in every experiment. Since the amount of data in each batch changes depending on the determined number of batches, it can be said that no correlation

between the data quantity in batches and performance could be observed. However, since the total amount of data used by the system remained constant at all times, it is also not possible to conclude that the system's performance is entirely unaffected by data quantity.

5.5 Effect of Balancing Datasets

In this section, it is discussed the effects of balancing the datasets used to train each model during the training phase.

Our system creates a different model for each concept during the training phase. To decide when concept drift has occurred, it uses two different strategies described in Section 3.3.1 and Section 3.3.2. The system has the capability to balance the training datasets of the models created up to that point the user's command by using the method explained in Section 3.4. The system automatically updates the models with the new datasets after balancing the training datasets.

In our study, for each experimental scenario, both the performance before balancing the training datasets and the performance after balancing them were measured. During the experiments, the prediction strategy was kept constant, and the training strategy was changed to observe the effects of balancing. Additionally, the training strategy was kept constant, and the prediction strategy was changed to observe the effects of balancing as well.

When examining the system's performance with and without the balancing process applied, although there are minor differences, no significant performance differences were observed. In fact, when performance was analyzed across batch number configurations from 2 to 150, it was seen that, despite some differences at certain points, they exhibited similar trends according to the batch number.

On both real datasets and synthetic datasets that mimic concept scenarios, their best performances according to batch number configurations generally aligned. In certain experiments, balancing the training datasets decreased performance, while in others, it improved it. Based on the current experimental results, it is not possible to conclude that balancing the training datasets enhances system performance for datasets containing concept drift.

CHAPTER 6

CONCLUSION AND FUTURE WORK

Concept drift is a important problem that reduces model performance or prevents the effective utilization of available data. In the scope of this thesis, an adaptive system has been developed to address the concept drift problem and dynamically mitigate its negative effects on the performance.

Machine learning models, during the training process, capture the data features, relationships, and patterns inherent in a dataset. However, if the dataset contains multiple concepts, each with distinct relationships, features, and patterns, a single model will struggle to generalize effectively across these varied concepts. Therefore, the structure of our adaptive system is based on creating a distinct machine learning model for each concept within the training data. Our system measures the similarity of the target dataset to the concepts within the training dataset of the models when predicting a dataset. Based on this measurement, it employs a voting mechanism to produce the final prediction outputs.

In our research, 2 different training strategies were tested to train a model for each concept during the training phase. Also, 6 different prediction strategies were developed to evaluate the models' capacity to understand the concept of the dataset to be predicted and accordingly assign them voting rights. The results of these approaches are provided in Chapter 4. Their capabilities, along with their respective advantages and disadvantages, are discussed in Chapter 5.

First of all, our system demonstrates the potential to surpass the performance of a single-model approach across all combinations of training and prediction strategies tested in the experiments. However, it experiences significant performance declines when a hyper-parameters which are not appropriate are selected, particularly regarding the batch number parameter. This highlights the importance of selecting appropriate hyper-parameters.

Secondly, a method was tested that assigns a batch number based on the system's performance on a validation set to select an appropriate batch size to find optimal hyper-parameters. The results of these experiments indicated that, when hyper-parameters were tuned, both proposed training approaches, used in conjunction with the prediction strategy outlined in Section3.6.1, outperformed the single-model approach. On the other hand, while the other prediction strategies generally yielded better results than the single-model approach when hyper-parameters were tuned, cases of lower performance were also observed. Ultimately, our research introduces combinations of training and prediction approaches that surpass the performance of the single-model approach under different conditions. These findings demonstrate that our system can address the concept drift problem more effectively than a single-model approach.

Thirdly, although we have system versions that consistently outperform the single-model approach when hyper-parameters are tuned based on validation set performance, there remains potential to achieve better performance on the test set with alternative hyper-parameters. This is because the batch size that yields optimal performance is not always consistent between the validation and test sets. Moreover, significant issues can arise when the validation and test sets represent distinctly different concepts.

Finally, the system does not faced with batch number configurations problem when data follows a specific concept drift scenario. This can be observed by examining the experimental results in Section 4.1. When the dataset follows a concept drift scenario, the points at which the data is divided into batches or the number of batches becomes less significant, as each batch generally represents a single concept regardless of the batch divisions. Thus, batch number hyper-parameter mostly does not affect performance of the system on creating a separate model for each concept in the dataset using the applied training methods or on measuring the similarity between the concept in the training data and the concept in the dataset to be predicted. On the other hand, if the concept drift scenario is not clear, as it is often the case with real-world datasets, the selection of batches through which the data enters the system becomes important.

Our study addresses a gap in the literature, which predominantly focuses on detecting concept drift while deferring its solution to a later stage. The system proposed in our research not only detects concept drift but also dynamically resolves the issues caused by it through an adaptive approach that develops models tailored to each concept. Our research comprehensively addresses all potential issues that may arise from concept drift to provide a dynamic solution. For instance, our study discusses various potential situations, such as how the training phase is proceed, how the resulting models should be utilized for prediction, and how to handle potential class imbalance that may emerge during these processes. In conclusion, our study aims to fill the gap created by the fragmented understanding in the literature, where the detection, understanding, and solution of concept drift are often addressed separately. Instead, our research offers a holistic approach to tackle these interconnected aspects of the problem.

The third outcome of the research paves the way for future studies to enhance the system. One major area for future researches involves developing complex, multi-layered approaches for determining batch size, rather than simply optimizing hyper-parameters that perform best on the validation set. The process of determining batch size based on validation set performance could be refined by examining the concept differences between the validation and test sets. In other words, batch number parameter can be defined adaptively too. Secondly, the system's structure can be revised to reduce its dependency on the batch size parameter. This adaptation could mitigate performance declines caused by dependency on the batch size parameter by changing batch-based structure of the system. Additionally, in future research, instead of defining a fixed batch number and splitting the dataset equally, the number of data points in each batch could be variably determined based on performance or directly concept based criteria. Finally, while no significant variation in performance based on data quantity was observed in our study, there is insufficient evidence to conclusively demonstrate that the system operates independently of data quantity. Future research should employ designed experiments to systematically examine whether the system's performance is influenced by the amount of data used and, if so, to determine the extent of this dependency.

REFERENCES

[1] J. Lu, A. Liu, F. Dong, F. Gu, J. Gama, and G. Zhang, "Learning under concept drift: A review," *IEEE Transactions on Knowledge and Data Engineering*, vol. 31, no. 12, pp. 2346–2363, 2019.

[2] L. Yang, W. Guo, Q. Hao, A. Ciptadi, A. Ahmadzadeh, X. Xing, and G. Wang, "CADE: Detecting and explaining concept drift samples for security applications," pp. 2327–2344, Aug. 2021.

[3] N. Lu, G. Zhang, and J. Lu, "Concept drift detection via competence models," *Artificial Intelligence*, vol. 209, pp. 11–28, 2014.

[4] L. L. Minku and X. Yao, "Ddd: A new ensemble approach for dealing with concept drift," *IEEE Transactions on Knowledge and Data Engineering*, vol. 24, no. 4, pp. 619–633, 2012.

[5] A. Mallick, K. Hsieh, B. Arzani, and G. Joshi, "Matchmaker: Data drift mitigation in machine learning for large-scale systems," in *Proceedings of Machine Learning and Systems* (D. Marculescu, Y. Chi, and C. Wu, eds.), vol. 4, pp. 77–94, 2022.

[6] A. Kuppa and N.-A. Le-Khac, "Learn to adapt: Robust drift detection in security domain," *Computers and Electrical Engineering*, vol. 102, p. 108239, 2022.

[7] L. L. Minku and X. Yao, "Ddd: A new ensemble approach for dealing with concept drift," *IEEE Transactions on Knowledge and Data Engineering*, vol. 24, no. 4, pp. 619–633, 2012.

[8] B. I. F. Maciel, S. G. T. C. Santos, and R. S. M. Barros, "A lightweight concept drift detection ensemble," in *2015 IEEE 27th International Conference on Tools with Artificial Intelligence (ICTAI)*, pp. 1061–1068, 2015.

[9] Q. Hu, Y. Guo, M. Cordy, X. Xie, L. Ma, M. Papadakis, and Y. Le Traon, "An empirical study on data distribution-aware test selection for deep learning enhancement," *ACM Trans. Softw. Eng. Methodol.*, vol. 31, jul 2022.

[10] F. Hinder, V. Vaquet, and B. Hammer, "One or two things we know about concept drift – a survey on monitoring evolving environments," 2023.

[11] F. Bayram, B. S. Ahmed, and A. Kassler, "From concept drift to model degradation: An overview on performance-aware drift detectors," *Knowledge-Based Systems*, vol. 245, p. 108632, 2022.

[12] G. Fenza, M. Gallo, and V. Loia, "Drift-aware methodology for anomaly detection in smart grid," *IEEE Access*, vol. 7, pp. 9645–9657, 2019.

[13] M. A. Hashmani, S. M. Jameel, M. Rehman, and A. Inoue, "Concept drift evolution in machine learning approaches: A systematic literature review," *International Journal on Smart Sensing and Intelligent Systems*, vol. 13, pp. 1 – 16, 2020.

[14] Q. Yang, Y. Gu, and D. Wu, "Survey of incremental learning," in *2019 Chinese Control And Decision Conference (CCDC)*, pp. 399–404, 2019.

[15] Z. Yang, S. Al-Dahidi, P. Baraldi, E. Zio, and L. Montelatici, "A novel concept drift detection method for incremental learning in nonstationary environments," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 31, no. 1, pp. 309–320, 2020.

[16] F. Jemili, K. Jouini, and O. Korbaa, "Intrusion detection based on concept drift detection online incremental learning," 08 2023.

[17] R. S. M. de Barros and S. G. T. de Carvalho Santos, "An overview and comprehensive comparison of ensembles for concept drift," *Information Fusion*, vol. 52, pp. 213–244, 2019.

[18] G. Andresini, F. Pendlebury, F. Pierazzi, C. Loglisci, A. Appice, and L. Cavallaro, "Insomnia: Towards concept-drift robustness in network intrusion detection," in *Proceedings of the 14th ACM Workshop on Artificial Intelligence and Security*, AISec '21, (New York, NY, USA), p. 111–122, Association for Computing Machinery, 2021.

[19] J. He, R. Mao, Z. Shao, and F. Zhu, "Incremental learning in online scenario," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2020.

[20] M. Salehi, H. Mirzaei, D. Hendrycks, Y. Li, M. H. Rohban, and M. Sabokrou, "A unified survey on anomaly, novelty, open-set, and out-of-distribution detection: Solutions and future challenges," 2022.

[21] J. Yang, K. Zhou, Y. Li, and Z. Liu, "Generalized out-of-distribution detection: A survey," 2024.

[22] S. Bulusu, B. Kailkhura, B. Li, P. K. Varshney, and D. Song, "Anomalous example detection in deep learning: A survey," *IEEE Access*, vol. 8, pp. 132330–132347, 2020.

[23] D. Berend, X. Xie, L. Ma, L. Zhou, Y. Liu, C. Xu, and J. Zhao, "Cats are not fish: deep learning testing calls for out-of-distribution awareness," in *Proceedings of the 35th IEEE/ACM International Conference on Automated Software Engineering*, ASE '20, (New York, NY, USA), p. 1041–1052, Association for Computing Machinery, 2021.

[24] J. Lu, A. Liu, F. Dong, F. Gu, J. Gama, and G. Zhang, "Learning under concept drift: A review," *IEEE Transactions on Knowledge and Data Engineering*, vol. 31, no. 12, pp. 2346–2363, 2019.

[25] F. Hinder, V. Vaquet, and B. Hammer, "One or two things we know about concept drift—a survey on monitoring in evolving environments. part a: detecting concept drift," *Frontiers in Artificial Intelligence*, vol. 7, 2024.

[26] F. Hinder, V. Vaquet, and B. Hammer, "One or two things we know about concept drift – a survey on monitoring evolving environments," 2023.

[27] A. Suárez-Cetrulo, D. Quintana, and A. Cervantes, "A survey on machine learning for recurring concept drifting data streams," *Expert Systems with Applications*, vol. 213, p. 118934, 10 2022.

[28] I. Žliobaitė, "Learning under concept drift: an overview," 2010.

[29] B. Krawczyk and M. Woźniak, "Reacting to different types of concept drift with adaptive and incremental one-class classifiers," in *2015 IEEE 2nd International Conference on Cybernetics (CYBCONF)*, pp. 30–35, 2015.

[30] N. Liu, J. Huang, and L. Cui, "A framework for online process concept drift detection from event streams," in *2018 IEEE International Conference on Services Computing (SCC)*, pp. 105–112, 2018.

[31] J. Tian, X. Liu, and M. Li, "An incremental learning ensemble method for imbalanced credit scoring," in *2019 IEEE Symposium Series on Computational Intelligence (SSCI)*, pp. 754–759, 2019.

[32] D. Elreedy and A. F. Atiya, "A comprehensive analysis of synthetic minority oversampling technique (smote) for handling class imbalance," *Information Sciences*, vol. 505, pp. 32–64, 2019.

[33] J. Liu, "Importance-smote: a synthetic minority oversampling method for noisy imbalanced data," *Soft Comput.*, vol. 26, p. 1141–1163, feb 2022.

[34] Z. Wan, Y. Zhang, and H. He, "Variational autoencoder based synthetic data generation for imbalanced learning," in *2017 IEEE Symposium Series on Computational Intelligence (SSCI)*, pp. 1–7, 2017.

[35] S. Wang, Y. Dai, J. Shen, and J. Xuan, "Research on expansion and classification of imbalanced data based on smote algorithm," 2021.

[36] N. V. Chawla, K. W. Bowyer, L. O. Hall, and W. P. Kegelmeyer, "Smote: Synthetic minority over-sampling technique," *Journal of Artificial Intelligence Research*, vol. 16, p. 321–357, June 2002.

[37] H. Hu, M. M. Kantardzic, and T. S. Sethi, "No free lunch theorem for concept drift detection in streaming data classification: A review," *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, vol. 10, 2019.

[38] F. Nielsen, "On the jensen–shannon symmetrization of distances relying on abstract means," *Entropy*, vol. 21, no. 5, 2019.

[39] M. P. Fay and M. A. Proschan, "Wilcoxon-Mann-Whitney or t-test? On assumptions for hypothesis tests and multiple interpretations of decision rules," *Statistics Surveys*, vol. 4, no. none, pp. 1 – 39, 2010.

[40] S. Farhadpour, T. A. Warner, and A. E. Maxwell, "Selecting and interpreting multiclass loss and accuracy assessment metrics for classifications with class imbalance: Guidance and best practices," *Remote Sensing*, vol. 16, no. 3, 2024.

[41] M. Harries, "Splice-2 comparative evaluation: Electricity pricing," technical report, University of New South Wales, School of Computer Science and Engineering, 1999.

[42] U. Ali and T. Mahmood, "A novel framework for concept drift detection for classification problems in data streams," 08 2023.