UNIVERSITY OF THESSALY

SCHOOL OF ENGINEERING

DEPARTMENT OF ELECTRICAL AND COMPUTER ENGINEERING

METHODS FOR ADDRESSING IMBALANCED DATA

Diploma Thesis

VASILEIOS SAVVAS TSIMPERIS

IOANNIS KARVELIS

Supervisor: MICHAEL VASSILAKOPOULOS

September 2024

UNIVERSITY OF THESSALY

SCHOOL OF ENGINEERING

DEPARTMENT OF ELECTRICAL AND COMPUTER ENGINEERING

METHODS FOR ADDRESSING IMBALANCED DATA

Diploma Thesis

VASILEIOS SAVVAS TSIMPERIS

IOANNIS KARVELIS

Supervisor: MICHAEL VASSILAKOPOULOS

September 2024

ΠΑΝΕΠΙΣΤΗΜΙΟ ΘΕΣΣΑΛΙΑΣ

ΠΟΛΥΤΕΧΝΙΚΗ ΣΧΟΛΗ

ΤΜΗΜΑ ΗΛΕΚΤΡΟΛΟΓΩΝ ΜΗΧΑΝΙΚΩΝ ΚΑΙ ΜΗΧΑΝΙΚΩΝ ΥΠΟΛΟΓΙΣΤΩΝ

ΜΕΘΟΔΟΙ ΓΙΑ ΤΗΝ ΑΝΤΙΜΕΤΩΠΙΣΗ ΑΝΙΣΟΡΡΟΠΩΝ ΔΕΔΟΜΕΝΩΝ

Διπλωματική Εργασία

Βασίλειος Σάββας Τσιμπέρης

Ιωάννης Καρβέλης

Επιβλέπων/πουσα: ΜΙΧΑΗΛ ΒΑΣΙΛΑΚΟΠΟΥΛΟΣ

Σεπτέμβριος 2024

Approved by the Examination Committee:

Supervisor	**MICHAEL VASSILAKOPOULOS** Chairman and Professor, Department of Electrical and Computer Engineering, University of Thessaly
Member	**ATHANASIOS FEVGAS** Laboratory Teaching Staff, Department of Electrical and Computer Engineering, University of Thessaly
Member	**ELENI TOUSIDOU** Laboratory Teaching Staff, Department of Electrical and Computer Engineering, University of Thessaly

Acknowledgements

With this thesis we close our undergraduate cycle at the University of Thessaly. 5 excellent years filled with unforgettable experiences and moments.We came into contact with new people, we made new friends, we learned many things that we will remember and be grateful for.We would like to thank you all individually.

First and most importantly, we would like to sincerely thank Michael Vasilakopoulos, our supervisor , professor and president of the Department of Electrical and Computer Engineering, for his constant support, direction, and insightful analysis over the duration of this project. Their knowledge and support have been much needed to help this thesis be successfully completed.

Next, we are deeply grateful to Professor Athanasios Fevgas and Eleni Tousidou for graciously agreeing to participate in examination committee, evaluating this thesis .

We are also profoundly grateful to the University of Thessaly and the Department of Electrical and Computer Engineering for providing us with the opportunity , the knowledge and resources to undertake this research. The facilities and academic environment have been conducive to our learning and development.

Especially thanks to all the professors, whose direction and lectures have substantially advanced us academically . This research has a solid foundation due to your dedication in education, incisive lectures, and advice.

Especially thanks to our colleagues for their helpful criticism, friendship, and for traveling this road with us. Their help and mental encouragement have been very invaluable.

We also like to thank the technical staff and library personnel who enabled us to obtain the equipment and materials needed for our study.Their helpfulness and efficiency definitely enhanced the learning process.

As of our family and friends, we truly value your consistent support, patience, and encouragement. Your trust in our capacity has kept us constantly driven.

Finally, we would wish to thank all those who supported us and this project in one way or another. Your help and encouragement have been highly appreciated.Without you, this success would not have been possible. We deeply thank you all!

DISCLAIMER ON ACADEMIC ETHICS AND INTELLECTUAL PROPERTY RIGHTS

Diploma Thesis

METHODS FOR ADDRESSING IMBALANCED DATA

VASILEIOS SAVVAS TSIMPERIS

IOANNIS KARVELIS

Abstract

Methods for addressing unbalanced data in machine learning are investigated in this thesis. Prediction models' efficacy can be seriously hampered by imbalanced data—that is, by disproportionate representation of particular classes. Various approaches for obtaining dataset balance are investigated in this work together with their efficiency in improving model accuracy.

To simplify practical use, a web application built on Django was created. This program lets users upload their data in CSV format, generates thorough statistics on data imbalance, and presents several balancing techniques to apply. The website provides a user friendly tool , easy to use, either as a first introduction to imbalanced data or as a balancing tool for someone working with it.

Among other approaches, the chosen balancing ones are Random Under-Sampling, Over-Sampling, Adaptive Synthetic Sampling (ADASYN), and Synthetic Minority Over-sampling Technique (SMote). Bagging,Boosting and stacking methods are also implemented.

Using a number of criteria, including accuracy, precision, recall, F1-score, Gmean,MCC and area under the ROC curve, the effectiveness of these strategies was evaluated. Using balancing techniques shows that models' performance on datasets with imbalanced classes can be much enhanced.

This work improves the field by means of a comprehensive analysis of data balancing methods and a user-friendly tool for practitioners handling imbalanced data. The outcomes underline the need of selecting appropriate techniques for data balancing depending on the particular features of the dataset and the intended usage.

Keywords:

imbalanced data, data balancing, machine learning, SMOTE, ADASYN, Django, web application, data analysis, Bagging, Boosting, Stacking, Imbalance Ratio, file, SVM, Classification, Analysis, User Profile, Python, metrics, accuracy.

Διπλωματική Εργασία

ΜΕΘΟΔΟΙ ΓΙΑ ΤΗΝ ΑΝΤΙΜΕΤΩΠΙΣΗ ΑΝΙΣΟΡΡΟΠΩΝ ΔΕΔΟΜΕΝΩΝ

Βασίλειος Σάββας Τσιμπέρης

Ιωάννης Καρβέλης

Περίληψη

Στην παρούσα διατριβή διερευνώνται μέθοδοι για την αντιμετώπιση μη ισορροπημένων δεδομένων στη μηχανική μάθηση. Η αποτελεσματικότητα των μοντέλων πρόβλεψης μπορεί να παρεμποδιστεί σοβαρά από τα ανισόρροπα δεδομένα - δηλαδή από τη δυσανάλογη εκπροσώπηση συγκεκριμένων κλάσεων. Στην παρούσα εργασία διερευνώνται διάφορες προσεγγίσεις για την επίτευξη ισορροπίας των συνόλων δεδομένων καθώς και η αποτελεσματικότητά τους στη βελτίωση της ακρίβειας των μοντέλων.

Για την απλούστευση της πρακτικής χρήσης, δημιουργήθηκε μια διαδικτυακή εφαρμογή βασισμένη στη Django. Η ιστοσελίδα αυτή επιτρέπει στους χρήστες να στέλνουν τα δεδομένα τους σε μορφή CSV, να παράγει λεπτομερή στατιστικά στοιχεία σχετικά με την ανισορροπία των δεδομένων και έπειτα παρουσιάζει διάφορες τεχνικές εξισορρόπησης προς εφαρμογή. Μεταξύ άλλων προσεγγίσεων, οι επιλεγμένες εξισορροπητικές είναι η τυχαία υποδειγματοληψία, η υπερδειγματοληψία, η προσαρμοστική συνθετική δειγματοληψία (ADASYN) και η συνθετική τεχνική υπερδειγματοληψίας μειοψηφίας (SMote). Εφαρμόζονται επίσης οι μέθοδοι Bagging,Boosting και Stacking.

Χρησιμοποιώντας μια σειρά κριτηρίων, όπως τα accuracy, precision, recall, F1-score, Gmean,MCC και η περιοχή κάτω από την καμπύλη ROC, αξιολογήθηκε η αποτελεσματικότητα αυτών των στρατηγικών. Η χρήση τεχνικών εξισορρόπησης δείχνει ότι η απόδοση των μοντέλων σε σύνολα δεδομένων με ανισόρροπες κλάσεις μπορεί να βελτιωθεί σημαντικά.

Η παρούσα εργασία αντιμετωπίζει το πρόβλημα ανισορροπίας των δεδομένων, μέσω μιας ολοκληρωμένης ανάλυσης των μεθόδων εξισορρόπησης δεδομένων και ενός φιλικού προς τον χρήστη εργαλείου για τους επαγγελματίες που χειρίζονται ανισόρροπα δεδομένα. Τα αποτελέσματα υπογραμμίζουν την ανάγκη επιλογής κατάλληλων τεχνικών για την εξισορρόπηση των δεδομένων ανάλογα με τα ιδιαίτερα χαρακτηριστικά του συνόλου δεδομένων και την προβλεπόμενη χρήση.

Λέξεις-κλειδιά:
ανισόρροπα δεδομένα, εξισορρόπηση δεδομένων, μηχανική μάθηση, SMOTE, ADASYN, Django, διαδικτυακή εφαρμογή, ανάλυση δεδομένων, bagging, boosting, stacking, Imbalance Ratio, αρχείο, SVM, ταξινόμηση, ανάλυση, προφίλ χρήστη, Python, μετρήσεις, ακρίβεια.

Table of contents

Acknowledgements **ix**

Abstract **xii**

Περίληψη **xiv**

Table of contents **xvii**

List of figures **xxiii**

List of tables **xxv**

Abbreviations **xxvii**

1 Introduction **1**

1.1 Problem Statement 2

1.2 Thesis Scope 3

1.2.1 Key Findings 5

1.3 Thesis Contribution 5

1.4 Thesis Organisation 6

2 Related Work **9**

2.1 Empirical study-Imbalance Degrees 9

2.2 Metrics in the Vector Space 10

2.2.1 Euclidean Distance 10

2.2.2 Chebyshev Distance 10

2.3 f-divergences 11

2.3.1 Kullback-Leibler (KL) Divergence 11

2.3.2 Hellinger Distance 11
2.3.3 Total Variation Distance 11
2.3.4 Chi-square Divergence 12
2.4 Towards Label Imbalance 12
2.5 Advanced Techniques in Imbalanced Data Handling 12
2.6 Cost Sensitive Learning 13
2.7 Resampling Techniques and Classifiers 14
2.8 Boosting 15
2.9 Hybrid Sampling 16
2.10 ROC Curves 16

3 Basic Concepts 17
3.1 Machine Learning Fundamentals 17
3.1.1 Supervised Learning 17
3.1.2 Classification 18
3.2 Evaluation Metrics 18
3.2.1 Accuracy 18
3.2.2 Precision and Recall 19
3.2.3 F1-Score 19
3.2.4 AUC-ROC Curve 19
3.2.5 MCC (Matthews Correlation Coefficient) 20
3.2.6 G-mean 20
3.3 Imbalanced Data Techniques 21
3.3.1 Data-Level Methods 21
3.3.2 Algorithm-Level Methods 21
3.4 Bagging and Boosting 22
3.4.1 Bagging (Bootstrap Aggregating) 22
3.4.2 Boosting 23
3.5 Text Preprocessing (NLP methods) 23
3.5.1 Tokenization 23
3.5.2 Stop Words Removal 24
3.5.3 Stemming and Lemmatization 24
3.6 Feature Extraction and Selection 24

3.6.1 Vectorization 25
3.6.2 Feature Selection 25
3.7 Summary 26

4 Definition of the problem 27
4.1 Understanding Imbalanced Data 28
4.1.1 Definition 28
4.1.2 Causes of Imbalance 28
4.1.3 The Nature of Multi-Label Imbalanced Data 29
4.2 Impact of Imbalanced Data on Machine Learning 30
4.2.1 Model Bias 30
4.2.2 Evaluation Metrics 30
4.2.3 Evaluation Complexity in Multi-Label Contexts 31
4.3 Challenges in Handling Imbalanced Data 31
4.3.1 Data-Level Challenges 31
4.3.2 Algorithm-Level Challenges 32
4.3.3 Cost-Sensitive Learning in Multi-Label Contexts 32
4.3.4 Real-World Constraints 33
4.3.5 Evolving Data Streams and Scalability Concerns 33
4.4 Need for Effective Solutions 33
4.4.1 Importance of Addressing Imbalance 33
4.4.2 Current Gaps and Limitations 34
4.4.3 Emerging Techniques and Future Directions 34
4.5 Chapter Summary 34

5 Methodology I 35
5.1 Datasets 35
5.2 Data Preprocessing 36
5.3 Text Vectorization and Data Splitting 36
5.4 Implementation Details 37
5.4.1 Visualization Tools 37
5.4.2 Tools and Libraries used for preprocessing tasks 37
5.4.3 Tools and Libraries used for the methods and algorithms 39

5.5 Classifiers . . . 39
5.5.1 Random Forest Classifier . . . 40
5.5.2 Logistic Regression Classifier . . . 40
5.5.3 KNN Classifier . . . 40
5.5.4 XGB Classifier . . . 40
5.5.5 MLP Classifier . . . 41
5.6 Support Vector Machines . . . 41

6 Methodology II 43
6.1 Over-Sampling Methods/Algorithms . . . 43
6.1.1 SMOTE . . . 43
6.1.2 ADASYN . . . 45
6.2 One Sided Selections . . . 47
6.3 Boosting . . . 48
6.3.1 XGBoost . . . 48
6.3.2 LightGBM . . . 48
6.3.3 CatBoost . . . 49
6.3.4 AdaBoost . . . 49
6.3.5 AdaUBoost . . . 49
6.3.6 AsymBoost . . . 50
6.3.7 CompatibleAdaBoost . . . 50
6.3.8 GradientBoosting . . . 50
6.3.9 HistGradientBoosting . . . 51
6.3.10 SmoteBoost . . . 51
6.3.11 RUSBoost . . . 51
6.3.12 AdaCost . . . 52
6.3.13 KmeansSMoteBoost . . . 52
6.3.14 OverBoost . . . 52
6.3.15 SelfPacedEnsemble . . . 52
6.4 Bagging . . . 53
6.4.1 Standard Bagging . . . 54
6.4.2 OverBagging . . . 54
6.4.3 UnderBagging . . . 54

6.4.4 UnderOverBagging . . . 54
6.5 SVM . . . 55
6.5.1 LinearSVM . . . 55
6.5.2 PolynomialSVM . . . 55
6.5.3 RBFSVM . . . 56
6.5.4 SigmoidSVM . . . 56

7 Web Application Development 57
7.1 Django Framework Overview . . . 57
7.2 Application Architecture . . . 58
7.3 HTML . . . 58
7.3.1 Models . . . 58
7.3.2 Views . . . 60
7.3.3 Templates . . . 62
7.3.4 URL Routing . . . 62
7.4 Key Features and Functionality . . . 62
7.4.1 User Authentication and Account Management . . . 62
7.4.2 Data Upload . . . 63
7.4.3 Statistical Analysis Results . . . 63
7.4.4 Data Balancing . . . 63

8 Web Application Tutorial 65
8.1 Getting Started . . . 65
8.1.1 Installation . . . 65
8.1.2 Accessing the Website . . . 66
8.2 Menu . . . 68
8.2.1 Analyze . . . 68
8.2.2 Classify . . . 69
8.2.3 Contact us . . . 75
8.2.4 Account Management . . . 76

9 Summary 79
9.1 Future extensions . . . 80
9.1.1 Advanced Deep Learning Architectures . . . 81

9.1.2 Expanded Dataset Diversity 81
9.1.3 Website Enhancements 81
9.1.4 Advanced Ensemble Methods 82
9.1.5 Theoretical Analysis 82
9.1.6 Real-time Learning and Adaptation 82
9.1.7 Fairness and Bias Mitigation 82

Bibliography **85**

List of figures

8.1 URL Generation. 66

8.2 Login/Register. 67

8.3 Register Form. 67

8.4 Menu/Tools. 68

8.5 Analyze Section 68

8.6 Label Distribution Graph 69

8.7 Classify-Step 1 70

8.8 Classify-Step 2 70

8.9 Classify-Step 3 71

8.10 Classify-Step 4 71

8.11 Classify-Step 5 72

8.12 Classify-Step 6 - Random Forest 73

8.13 Classify- Parameter Overview 74

8.14 ”Classification Sequence” window 75

8.15 Contact Us 76

8.16 Account Management Menu 77

8.17 My Datasets page 77

List of tables

6.1 SMOTE with Classifiers 44
6.2 ADASYN with Classifiers 46
6.3 OSS with Classifiers 47
6.4 Boosting Methods 53
6.5 Bagging Methods 55
6.6 SVM-kernels 56

Abbreviations

etc	et cetera
SVM	Support Vector Machine
RF	Random Forest
KNN	K-Nearest Neighbors
LR	Logistic Regression
OSS	One-Side Selections
XGB	XGBoost - Extreme Gradient Boost
TF-IDF	
AWH-SMOTE	Attribute Weighted Smote
SHSampler	Similarity Hybrid Sampler
LCT	Loss Conditional Training
TPR	True Positive Rate
FPR	False Positive Rate
ROS	Random Over Sampling
ENN	Editing Nearest Neighbor
CNN	Condensed Nearest Neighbor
TL	Tensor-Flow
AMCS	adaptive multiple classifier system
MOF	Mass Ratio Variance
ROC	Receiver Operating Characteristics
AUC	Area Under Curve
MCC	Matthews Correlation Coefficient
TN	True-Negative
FN	False-Negative
TP	True-Positive

FP	False-Positive
NLP	Natural Language Processing
ORM	(Object-Relational Mapping)
XSS	Cross Site Scripting
CSRF	Cross site request forgery
MVT	Model-View-Template
MVC	Model-View-Controller
RBF	Radial Basis Function
RBFSVM	Radial Basis Function SVM
IDE	Integrated Development Environment

Chapter 1

Introduction

In the rapidly evolving fields of machine learning and data science, the quality and balance of datasets used to train predictive models significantly affect their performance, reliability, and real-world applicability [1]. One of the primary challenges researchers and practitioners encounter is the prevalence of imbalanced datasets, where some classes are substantially underrepresented relative to others. This imbalance may lead to biased models that disproportionately favor the majority class, thereby compromising the predictive performance on the minority class [2].

The problem of class imbalance is not merely a theoretical concern but has profound implications across various domains. In practical applications, datasets often exhibit notable class imbalances:

- In healthcare, disease diagnosis datasets typically contain far more instances of healthy cases than occurrences of rare diseases [3]. This imbalance can lead to models that excel at identifying normal cases but struggle to detect critical, albeit infrequent, health conditions.

- In finance, datasets for fraud detection usually include a significantly higher number of legitimate transactions compared to fraudulent ones [4]. The rarity of fraud cases can result in models that are overly optimistic in predicting legitimate transactions but fail to identify subtle patterns of fraudulent activity.

- In environmental monitoring, datasets may contain an abundance of normal weather patterns but relatively few instances of extreme events or natural disasters [5].

This imbalance can lead models to be unduly confident in their ability to predict the majority class, thereby impairing their capacity to accurately identify and classify the minority class. The consequences of this bias are significant: in finance, it may result in undetectable fraud; in healthcare, it could mean missed diagnoses of rare but severe conditions; in environmental science, it might lead to failure in predicting critical events.

The challenge becomes even more complex in the context of multi-label datasets—where each instance can be associated with multiple labels concurrently [6]. Such datasets add layers of complexity to the problem by incorporating specific label combinations in addition to the frequency of individual labels, thus creating multi-dimensional imbalance. In this multi-label environment, conventional techniques that perform effectively on single-label imbalanced datasets often falter, resulting in suboptimal performance and potentially misleading results.

Addressing these challenges requires robust methods to balance the data, ensuring that models perform well across all classes. Effective data balancing techniques are crucial to mitigate the risks of biased predictions and to ensure that the insights derived from machine learning models are reliable and relevant in practical settings. This work aims to investigate and evaluate various approaches to mitigate class imbalance, particularly in multi-label datasets, thereby enhancing the overall accuracy, fairness, and applicability of predictive models across diverse domains.

1.1 Problem Statement

This thesis addresses the critical issue of class imbalance in multi-label datasets and its profound impact on the performance of machine learning algorithms. In real-world applications, datasets often exhibit skewed distributions where certain classes or labels are significantly underrepresented, compromising the ability of models to learn effectively [1]. The challenge is further compounded in multi-label classification scenarios, where each data point can be associated with multiple labels, making it even more difficult for models to discern and learn from these complex patterns [6].

Traditional machine learning methods often struggle to perform well on minority classes in such environments, resulting in biased models with either overly optimistic or misleading performance metrics. This imbalance has significant consequences:

- Models trained on imbalanced data are prone to high error rates when predicting minority classes, leading to inaccurate or incomplete predictions.

- In critical applications such as medical diagnosis, fraud detection, or recommendation systems, these errors can have severe repercussions.
- Misrepresentation of performance caused by imbalanced data can lead to flawed decision-making and the deployment of suboptimal models in practical settings.

The implications of these issues extend beyond mere statistical concerns. In healthcare, for instance, a model trained on imbalanced data might excel at identifying common conditions but fail to detect rare, life-threatening diseases [3]. In financial fraud detection, models might accurately classify the majority of legitimate transactions but miss subtle patterns of fraudulent activity, potentially resulting in significant financial losses [4].

This thesis seeks to address these challenges by exploring and implementing various techniques for balancing multi-label datasets. The primary goal is to enhance machine learning models by ensuring they are trained on data that more accurately reflects the full spectrum of classes and label combinations present in the dataset. This approach aims to improve not only the overall accuracy of the models but also their fairness and reliability across all classes, particularly the underrepresented ones.

Furthermore, the development of a user-friendly website as part of this thesis creates a practical tool to fulfill the purpose of the work. This web application aims to bridge the gap between advanced data balancing techniques and their practical implementation, providing researchers and practitioners with an accessible platform to analyze and balance their datasets effectively.

1.2 Thesis Scope

The scope of this thesis encompasses several key objectives, each designed to address different aspects of the class imbalance problem in multi-label datasets:

1. To develop a deep understanding of the theoretical foundations of data imbalance and its implications in machine learning and data science [1]. This includes exploring the mathematical and statistical principles underlying class imbalance and its effects on model performance.

2. To investigate and evaluate various methods for balancing imbalanced datasets [2, 1]. This objective includes:

    - Analyzing resampling techniques such as oversampling, undersampling, and hybrid methods.
    - Exploring algorithm-level methods that modify existing learning algorithms to handle imbalanced data more effectively.
    - Investigating ensemble techniques like bagging and boosting, and their effectiveness in addressing class imbalance.

    The goal is to prevent incorrect conclusions and decisions from being drawn from the interpretation of imbalanced data.

3. To develop a Django-based, user-friendly web application that provides users with tools to analyze and balance their datasets. This application aims to:

    - Offer an intuitive interface for dataset upload and visualization.
    - Implement various balancing techniques that users can apply to their datasets.
    - Provide performance metrics and visualizations to help users understand the impact of balancing methods.

4. To evaluate and compare the performance of different balancing techniques using relevant metrics [1, 7]. This includes:

    - Implementing metrics such as precision, recall, F1-score, and AUC-ROC.
    - Conducting comprehensive experiments to illustrate the effectiveness of the implemented methods across various datasets and scenarios.

5. To clarify the appropriate use of different balancing methods, including:

    - Resampling Techniques: Analyzing methods like Random Over-Sampling, Random Under-Sampling, and SMOTE [2].
    - Algorithm-Level Methods: Exploring cost-sensitive learning and adjusting class weights [1].
    - Hybrid Methods: Investigating combinations of resampling and algorithm-level approaches.

- Bagging-Boosting algorithms: Evaluating ensemble methods like Random Forest [8] and Gradient Boosting [9] in the context of imbalanced data.

1.2.1 Key Findings

To address the problem of class imbalance in multi-label datasets, a comprehensive evaluation of numerous models and algorithm combinations was conducted throughout this study. Despite testing a wide range of strategies, including those frequently mentioned in literature and applied in various projects and papers, not all approaches performed equally in our specific context.

One of the most significant findings was that the combination of the ADASYN (Adaptive Synthetic) sampling method [10] with the Random Forest classifier [8] and with Logistic Regression consistently produced superior performance across several metrics for multiple datasets. This finding highlights the importance of not only choosing appropriate balancing techniques but also considering their interaction with specific machine learning algorithms.

1.3 Thesis Contribution

This thesis makes several important contributions to the field of machine learning, particularly in addressing class imbalance in multi-label datasets:

1. Comprehensive Evaluation of Balancing Strategies: The thesis provides an in-depth study and comparison of various data balancing strategies, highlighting their advantages and disadvantages. This evaluation goes beyond simple accuracy metrics, considering the nuanced performance across different classes and label combinations. The findings contribute to the broader understanding of when and how to apply specific balancing techniques in multi-label contexts.

2. Development of a User-Friendly Tool: A significant practical contribution is the development of a Django-based web application designed to assist users in tackling the challenges of class imbalance in multi-label datasets. Key features of this tool include:

   - An intuitive interface for dataset upload and visualization.
   - Implementation of various balancing techniques that users can apply with ease.

- Real-time performance metrics and visualizations to help users understand the impact of different balancing methods.
- Integration of state-of-the-art machine learning techniques, enabling users to achieve improved model performance.

This platform not only streamlines the complex data preprocessing process but also makes advanced balancing techniques accessible to a wider audience of researchers and practitioners.

3. Practical Relevance Through Case Studies: The thesis demonstrates the practical relevance of balancing methods through comprehensive case studies. These studies provide practitioners with insightful analysis and guidelines for applying balancing techniques in real-world scenarios across various domains.

4. Scalability and Future-Proofing: While the website is in its initial stage, scalability has been a key consideration in its design. The application's modular architecture allows for easy incorporation of new algorithms and balancing strategies as they become available. Future enhancements could include:

   - Support for larger datasets and more complex multi-label structures.
   - Cloud-based processing for computationally intensive tasks.
   - Integration of more sophisticated machine learning models and emerging balancing techniques.

   The platform's adaptability ensures that it can evolve to meet the growing needs of users in both academia and industry, making it a valuable tool for a wide range of applications.

By addressing these aspects, this thesis not only contributes to the theoretical understanding of class imbalance in multi-label datasets but also provides practical tools and insights for researchers and practitioners working with imbalanced data in real-world applications.

1.4 Thesis Organisation

Nine chapters make up this thesis, each building on the one before to offer a thorough analysis of the difficulties and fixes for class imbalance in multi-label datasets as well as the evolution of a supporting web application. Chapter 1 presents the problem, specifies the extent of the study, and lists the main points of contribution of the thesis. Reviewing related work, Chapter 2 summarizes current research on data imbalance, balancing methods, and their application in machine learning. Including fundamental ideas in machine learning and class imbalance, Chapter 3 addresses the fundamental ideas required for grasp of the problem. In-depth discussion of the particular difficulties presented by imbalanced multi-label datasets and their effects on predictive modeling follows Chapter 4 definition of the problem. Emphasizing on the preprocessing tools, libraries, and classifiers applied in this work, Chapter 5 offers the first section of the methodology. Chapter 6 carries on the approach by exploring the several balancing techniques and algorithms used to solve data imbalance, and their application. Including its design, features, and technical elements, Chapter 7 details the evolution of the Django-based web application Acting as a guide on how to use the tool for data analysis and balancing, Chapter 8 functions as a web application tutorial. Chapter 9 finishes the thesis by aggregating the results, debating the consequences, and suggesting future directions of research.

Chapter 2

Related Work

An overview of the body of current research pertinent to the formulation of techniques for managing unbalanced data is given in this part. It addresses several strategies and methods that have been suggested and applied in past research, therefore stressing their advantages and drawbacks.

2.1 Empirical study-Imbalance Degrees

In machine learning, a dataset's degree of imbalance is a major determinant of both evaluation and model performance. Building strong and accurate predictive models depends on an awareness of and resolution of this imbalance using suitable methods, particularly in fields where minority class forecasts are critical.

In his research, Jonathan [11] insists the imbalance ratio cannot be considered a measure that has a satisfactory resolution for multi-class problems.Experimental data indicates that imbalance-degree is a more suitable summary compared to imbalance-ratio. In the context of multi-class classification, the former not only separates class distributions from groups with the same value but also establishes a stronger link with the challenges that arise from distorted class distributions in the learning process.

Therefore, we utilized the following mathematical formulas he proposed for the distance and similarity functions used to instantiate the imbalance degree in the empirical studies:

2.2 Metrics in the Vector Space

2.2.1 Euclidean Distance

The Euclidean distance is the “ordinary” straight-line distance between two points in Euclidean space. For two points p and q in n-dimensional space, it’s calculated as:

$$d(p,q) = \sqrt{\sum_{i=1}^{n}(p_i - q_i)^2}$$

The Euclidean distance is the most intuitive and commonly used distance metric. It’s sensitive to the scale of the features and works well when the data is dense or continuous. This metric is widely used in many machine learning algorithms like k-nearest neighbors (k-NN) and k-means clustering. It’s also common in recommendation systems and image processing tasks.

2.2.2 Chebyshev Distance

The Chebyshev distance, also known as L_∞ norm, between two vectors is the greatest of their differences along any coordinate dimension. For two points p and q, it’s defined as:

$$d(p,q) = \max_i |p_i - q_i|$$

This distance represents the maximum difference between any two corresponding elements of the vectors. It’s less sensitive to outliers compared to Euclidean distance and is invariant under rotations but not under arbitrary linear transformations. The Chebyshev distance is often used in warehouse logistics for calculating the time needed to move items. It’s also applied in chess for calculating the minimum number of moves for a king to reach a specific square. In data science, it can be useful in some clustering algorithms and anomaly detection tasks.

2.3 f-divergences

2.3.1 Kullback-Leibler (KL) Divergence

KL divergence measures the difference between two probability distributions P and Q. It's defined as:

$$KL(P||Q) = \sum_x P(x) \log \left(\frac{P(x)}{Q(x)} \right)$$

This divergence is not symmetric, meaning $KL(P||Q) \neq KL(Q||P)$. It's always non-negative and equals zero if and only if P and Q are identical. KL divergence is widely used in information theory and machine learning. It's applied in variational inference and model selection tasks. In natural language processing, it's often used for comparing language models.

2.3.2 Hellinger Distance

The Hellinger distance is used to quantify the similarity between two probability distributions. For discrete probability distributions P and Q, it's defined as:

$$H(P, Q) = \frac{1}{\sqrt{2}} \sqrt{\sum_x (\sqrt{P(x)} - \sqrt{Q(x)})^2}$$

This distance is symmetric, meaning $H(P, Q) = H(Q, P)$. It's bounded between 0 and 1 and satisfies the triangle inequality, making it a true metric. The Hellinger distance is used in statistics for hypothesis testing. In machine learning, it's applied for measuring the distance between distributions. It's also useful in image processing and computer vision tasks.

2.3.3 Total Variation Distance

The total variation distance is a measure of the largest possible difference between the probabilities that two probability distributions can assign to the same event. For discrete probability distributions P and Q, it's defined as:

$$TV(P, Q) = \frac{1}{2} \sum_x |P(x) - Q(x)|$$

This distance is symmetric and bounded between 0 and 1. It satisfies the triangle inequality and is related to the L_1 distance between probability distributions. The total variation distance is used in probability theory and statistics. In machine learning, it's applied for model comparison and selection. It's also useful in analyzing Markov chain convergence.

2.3.4 Chi-square Divergence

The Chi-square divergence measures how unlikely it is that one distribution was drawn from the population represented by the other. It's defined as:

$$\chi^2(P, Q) = \sum_x \frac{(P(x) - Q(x))^2}{Q(x)}$$

This divergence is not symmetric and is always non-negative. Compared to KL divergence, it's more sensitive to small differences in low-probability regions. The Chi-square divergence is widely used in statistical hypothesis testing. In machine learning, it's applied in feature selection tasks. It's also used in natural language processing for comparing word distributions.

2.4 Towards Label Imbalance

Label imbalance deflects on the performance of multi-label classifiers, forcing them to produce slower and mistaken classification results.In Xin's,Yuqi's,Fei's and Xiaofeng's research [12], we found some similar conclusions.Label imbalance may restrict the performance of multi-label classifiers, particularly in cases where some labels are missing.They applied pseudo labeling approach, which enables often used networks to be implemented in partially labeled environments without the necessity of extra intricate structures and also created a dynamic training program to lower the dimension of the labeling space and so help to minimize the imbalance.

2.5 Advanced Techniques in Imbalanced Data Handling

Recent developments in imbalanced dataset handling have produced more advanced methods addressing the constraints of conventional approaches. One such technique is resampling grounded in Deep Learning. [13] Unlike conventional approaches depending on simple resampling techniques, this one generates synthetic data points using neural networks. Recent research, for instance, show how Generative Adversarial Networks (GANs) as Junlan Chen et al. [14] proposed, can produce quite realistic synthetic samples that enhance the performance of classifiers on imbalanced datasets. This method considers the underlying distribution of the minority class and generates data points not only similar but also varied, so preventing overfitting, surpassing basic oversampling.

2.6 Cost Sensitive Learning

Managing multi-label imbalanced datasets—where different labels may have different degrees of importance or cost connected with misclassification—requires the critical method of cost-sensitive learning. The difficulty of imbalance is compounded in multi-label classification since every instance may be connected to several labels, each of which may be underrepresented. In such situations, traditional classifiers usually fall short since they usually reduce total error rates without considering the various costs of misclassifying minority versus majority labels. Cost-sensitive learning solves this by including these expenses straight into the learning process, so guaranteeing that the model not only forecasts accurately but also reduces the penalties for most important mistakes. [15]

Because they offer a strong statistical framework for assessing the performance of cost-sensitive classifiers, Dragos D. Margineantu and Thomas G. Dietterich [16] suggest that bootstrap techniques are especially crucial in this environment. Bootstrapping generates several resampled datasets that let one estimate confidence intervals for the expected cost connected with each label and the expected variation in costs among several classifiers. In multi-label situations when the interaction of several labels complicates the evaluation process, this is especially important. It would be difficult to ascertain whether improvements in classifier performance are statistically significant or if they merely reflect random data variation without such exact statistical techniques. Thus, the mix of cost-sensitive learning and bootstrap methods guarantees that the classifiers are both effective and reliable in managing the complexity of multi-label imbalanced datasets, so producing more accurate and fair predictions across all labels.

2.7 Resampling Techniques and Classifiers

In machine learning, data imbalance is the unequal dispersion of classes inside a dataset. Oversampling and undersampling as the resampling technique, helps to tackle this problem. Mohammed et al. [17], employed machine learning methods with varying hyperparameters using a public imbalanced dataset from Kaggle's Santander Customer Transaction Prediction. The main conclusions revealed that for several classifiers, oversampling performed better than undersampling for binary classification and yielded higher ratings in several evaluation criteria.

Piboon's and Krung's [18] work though ,suggests a combination approach for class imbalance problems in machine learning employing the MOF, undersampling and oversampling. The method combines both approaches, synthesizes cases from a minority class and eliminates sounds from a majority class. Particularly for datasets with minor imbalance ratios, the results revealed better recall and F1-scores, implying that undersampling can be useful when used in concert with oversampling .

Oversampling can have it's drawbacks.Ahsan et al. [19] Priya and Annie [20]agree that oversampling techniques can greatly enlarge the training dataset. Longer training timeframes and more computational resource needs resulting from this additional complexity could not be possible in every situation and moreover result in overfitting.

Hartono's and Eriato's [21] research proposes a hybrid approach with a distance feature to determine safe samples in SMOTE oversampling. The method handles multi-class imbalances and overlapping, ensuring the best sample with the lowest imbalance ratio is selected. This approach is effective in handling overlapping and determining safe samples.

In this work, two oversampling techniques in SMOTE: MultiRand Bal and Hybrid Approach with Distance Feature are compared in this work. With random determination, the Hybrid Approach seeks to reduce Safe Region overlapping samples. The study revealed a clear link between overlapping and multi-class imbalance on classification accuracy; increased overlap produces less accuracy. Class Balance Accuracy, however, exhibited no appreciable variation in the hybrid approach with distance feature, implying that the sample selection for the Safe Region fits multi-class imbalance in minority classes.

In Torah's et al. [22]Development of the Smote algorithm, the AWH-Smote algorithm presents a more representative kNN utilizing attribute weighting and a novel notion for choosing sample methods using occurrence data in the kNN hub. Results revealed that weighting at-

tribute in kNN generates more representative neighbors and noise, therefore improving the accuracy of current methods by up to 9%. Comparatively to other oversampling techniques, AWH-Smote also demonstrated improved performance on minority precision and minority f-measure. Applying various attribute weighting strategies, noise reduction approaches, and AWH-SMote in multiclass imbalanced data sets is the main emphasis of next work.
In Tariq's et al. project [23], various oversampling techniques are tested among with different classifiers . KNN classifier with SmoteTomek achieved the highest accuracy score of 83.7
Eréndira's et al.study [24] examines the effectiveness of heuristic sampling methods on deep learning neural networks in big data scenarios, focusing on cleaning strategies. Their research uses big data, multi-class imbalanced datasets from hyper-spectral remote sensing images. Results show that ROS and SMOTE are competitive strategies, but ROS outperforms SMOTE. Hybrid methods like SMOTE+TL, SMOTE+ENN, SMOTE+OSS, and SMOTE+CNN do not improve SMOTE's performance. They suggest that cleaning strategies should be applied to ANN output for better classification results.
Yijinga et al. [25] worked on AMCS to handle imbalanced data classification problems. Using logging data, AMCS efficiently identifies oil layers, therefore saving misclassification costs by allocating data to lower levels.

2.8 Boosting

Since boosting emphasizes misclassified events and adaptive weighting, it is frequently more successful at managing multi-class imbalanced datasets. Although bagging is helpful, in the event of class imbalance it could call for other tactics to get comparable degrees of performance [26].
With an emphasis on the need of strong techniques to maximize information without depending on user-introduced bias, Xiaohui's and Mohamed' essay [27] addresses the topic of face recognition with imbalanced training data. The authors suggest a multi-class boosting technique using subsets of samples to train an ensemble so suppressing face recognition failures. In high unbalanced situations, the approach exhibits better performance than other techniques.

Jafar's et al. [28], reviews the performance of 14 boosting ensemble algorithms for dealing with multi-class imbalanced data. The study discovered that CatBoost performs better than the majority of other boosting techniques in MAUC, G-mean, and MMCC on 15 standard datasets. XGBoost and LogitBoost are more effective in large datasets, with SMOTEBoost being ranked the second best algorithm. Nevertheless, they discovered that boosting techniques struggle to excel on extremely imbalanced large multi-class datasets, a finding that aligns with our own observations.

2.9 Hybrid Sampling

Zhong et al. [29] proposed a similarity-based hybrid sampling techinque to counter the mutuality of the data. SHSampler uses sample similarity and dissimilarity estimates to identify data complexity factors. SHSampler performs a relative majority reduction in sampling and a relative minority increase in sampling to mitigate the negative effects of data issues and highlight the importance of minority groups.

2.10 ROC Curves

In multi-class unbalanced data, ROC curves offer a strong and all-encompassing structure for assessing and enhancing classifier performance. They enable one to make wise choices on model selection and threshold tuning and to grasp the trade-offs between several kinds of mistakes.
Kelsey,Shuai,Swarna and Carlo [30] suggest training over a family of loss functions instead of a single loss function in order to lower the sensitivity to hyperparameter selections and build more broad models.They suggested using LCT to train binary classification models with strong class imbalance. This made the ROC curves better and made the model less sensitive to decisions about hyperparameters by using a substitute for maximizing along several TPR and FPR tradeoffs.

Chapter 3

Basic Concepts

This chapter will address the fundamental concepts essential for comprehending the approaches and methods covered in this thesis. These basic ideas provide the foundation for the development and implementation of models designed to handle imbalanced data [1]. By establishing a solid understanding of these concepts, readers will be better equipped to grasp the more complex techniques and methodologies presented in subsequent chapters.

3.1 Machine Learning Fundamentals

3.1.1 Supervised Learning

Supervised learning is a cornerstone of machine learning, where models are trained on labeled datasets [31]. In this paradigm, the algorithm learns to map input data to known output labels, effectively learning from 'experience' provided by the training data. This approach is particularly useful in scenarios where we have historical data with known outcomes.

- Definition: In supervised learning, a model is trained on a labeled dataset that includes the correct output for each input sample. The model gains the ability to make predictions on unfamiliar data by drawing conclusions from these examples [32].

- Examples: Common applications include classification tasks, where the model predicts categorical labels, and regression tasks, where the model predicts continuous values. These form the basis for many real-world applications in various domains.

3.1.2 Classification

Classification, an essential aspect of supervised learning, is vital in various real-world scenarios, such as detecting spam and diagnosing medical conditions [31].

- Definition: Classification is a subset of supervised learning aimed at predicting a categorical label for a given input. The model learns to assign input data to predefined categories or classes based on patterns observed in the training data.

- Examples: Real-world applications of classification are diverse and include spam detection in email systems, medical diagnosis based on patient data, and sentiment analysis in natural language processing. Each of these examples involves assigning inputs to discrete categories, illustrating the wide applicability of classification techniques.

3.2 Evaluation Metrics

Evaluation metrics are essential for assessing the effectiveness and performance of machine learning models, particularly when dealing with imbalanced datasets [1]. This thesis commonly utilizes the following metrics to offer a thorough assessment of model performance.

3.2.1 Accuracy

Accuracy is often the first metric considered in classification tasks, but it has limitations, particularly in imbalanced datasets [1].

- Definition: Accuracy is calculated as the ratio of correctly predicted instances to the total number of instances. It provides an overall measure of how often the model's predictions are correct.

- Limitations: For imbalanced datasets, accuracy can be misleading as it does not distinguish between the performance on minority and majority classes. A model that always predicts the majority class might have high accuracy but fail completely on the minority class, which is often the class of interest in many applications.

3.2.2 Precision and Recall

Precision and recall provide a more nuanced view of model performance, especially for the minority class in imbalanced datasets [1].

- Precision: This metric is calculated as the ratio of true positive predictions to the total number of positive predictions (true positives + false positives). It measures the model's ability to avoid labeling negative instances as positive.
- Recall: Also known as sensitivity or true positive rate, recall is the ratio of true positive predictions to the total number of actual positive instances (true positives + false negatives). It measures the model's ability to find all positive instances.
- Importance: These metrics are particularly useful for understanding the performance of the model on the minority class. High precision indicates that when the model predicts the positive class, it is often correct, while high recall indicates that the model correctly identifies a large proportion of the actual positive instances.

3.2.3 F1-Score

The F1-score provides a single metric that balances precision and recall, making it particularly useful for imbalanced datasets [1].

- Definition: The F1-score is the harmonic mean of precision and recall, providing a balance between these two metrics. It is calculated as 2 * (Precision * Recall) / (Precision + Recall).
- Importance: This metric is especially helpful for imbalanced datasets as it provides a single value that captures both the precision and recall of the model. A high F1-score indicates that the model has both good precision and good recall, suggesting balanced performance across classes.

3.2.4 AUC-ROC Curve

The Area Under the Receiver Operating Characteristic (AUC-ROC) curve is a powerful metric for evaluating binary classification models, especially on imbalanced datasets [7].

- Definition: The AUC-ROC curve measures the model's ability to distinguish between classes across various threshold settings. It plots the True Positive Rate against the False Positive Rate at different classification thresholds.
- Importance: This metric is particularly useful for evaluating model performance on imbalanced datasets as it shows how well a classifier can distinguish between classes regardless of the chosen threshold. An AUC of 1.0 represents a perfect model, while an AUC of 0.5 represents a model that performs no better than random guessing.

3.2.5 MCC (Matthews Correlation Coefficient)

The Matthews Correlation Coefficient is a metric that provides a balanced measure of model performance, even with classes of very different sizes [33].

- Definition: MCC is a metric that takes into account true and false positives and negatives. It is generally regarded as a balanced measure, even when the classes are of very different sizes, making it particularly useful for imbalanced datasets.
- Importance: MCC provides a comprehensive evaluation of model performance across all classes and suggests the correct choice of classifier. It ranges from -1 to +1, where +1 represents perfect prediction, 0 represents random prediction, and -1 represents total disagreement between prediction and observation.
- Formula:

$$\text{MCC} = \frac{TP \times TN - FP \times FN}{\sqrt{(TP + FP)(TP + FN)(TN + FP)(TN + FN)}} \tag{3.1}$$

  Where TP = True Positives, TN = True Negatives, FP = False Positives, and FN = False Negatives.

3.2.6 G-mean

The Geometric Mean (G-mean) is particularly useful for assessing model performance on imbalanced datasets [34].

- Definition: G-mean is calculated as the geometric mean of the sensitivity (recall) and specificity. Specificity is the true negative rate, measuring the proportion of actual negative instances that were correctly identified.

- Importance: This metric is useful for assessing the balance between classification performance on the majority and minority classes. A high G-mean indicates that the model performs well on both classes, not just the majority class.

- Formula:

$$\text{G-Mean} = \sqrt{\text{Sensitivity} \times \text{Specificity}} \tag{3.2}$$

  Where Sensitivity = TP / (TP + FN) and Specificity = TN / (TN + FP).

3.3 Imbalanced Data Techniques

Techniques for handling imbalanced data are crucial for improving model performance when one class significantly outweighs the other. These techniques can be broadly categorized into data-level and algorithm-level methods [1].

3.3.1 Data-Level Methods

Data-level methods focus on modifying the distribution of the training data to address class imbalance.

- Under-Sampling: This technique involves reducing the number of instances in the majority class to balance the dataset. It can be done randomly or through more sophisticated methods that aim to preserve important information. While effective, under-sampling risks losing potentially valuable information from the majority class [1].

- Over-Sampling: This approach increases the number of instances in the minority class, either through simple duplication of existing instances or through synthetic generation of new instances. Synthetic generation, such as SMOTE (Synthetic Minority Over-sampling Technique), creates new instances by interpolating between existing minority class instances [2]. While over-sampling helps balance the dataset, it can lead to overfitting if not carefully implemented.

3.3.2 Algorithm-Level Methods

Algorithm-level methods involve modifying the learning algorithm itself to better handle imbalanced data [1].

- Cost-Sensitive Learning: This method involves modifying the learning algorithm to assign different misclassification costs to different classes. By assigning a higher cost to misclassifying minority class instances, the model is encouraged to pay more attention to these instances during training. This approach can be effective but requires careful tuning of the cost matrix.

- Ensemble Methods: These techniques combine multiple models to improve performance on imbalanced datasets. For example, the Balanced Random Forest algorithm creates an ensemble of decision trees, each trained on a balanced subset of the data [8]. By aggregating predictions from multiple balanced models, ensemble methods can achieve better overall performance on imbalanced datasets.

3.4 Bagging and Boosting

Bagging and Boosting are ensemble learning techniques that combine multiple models to improve overall performance and robustness.

3.4.1 Bagging (Bootstrap Aggregating)

Bagging is an ensemble method that creates multiple subsets of the original dataset and trains a separate model on each subset [8].

- Process:Bagging is a technique that generates several smaller datasets from the original data by using bootstrapping (random sampling with replacement). Each subset is used to train a distinct model. The ultimate prediction is generated by combining the predictions from all models, usually by averaging for regression tasks or majority voting for classification tasks.

- Importance: Bagging helps reduce variance and prevents overfitting. By training models on different subsets of the data, bagging creates diverse models that can capture different aspects of the underlying patterns in the data. This diversity helps improve the overall robustness and generalization of the ensemble.

3.4.2 Boosting

Boosting is an ensemble method that sequentially trains models, with each model focusing on the errors of the previous ones [9].

- Definition: Boosting involves training a sequence of weak learners (models that perform slightly better than random guessing) on modified versions of the original dataset. Each subsequent model pays more attention to the instances that were misclassified by previous models. The final prediction is typically a weighted sum of the individual models' predictions, where the weights are based on each model's performance.

- Importance: Boosting is effective at reducing both bias and variance. By focusing on the mistakes of previous models, boosting can create a strong learner that performs well on difficult-to-classify instances. Popular boosting algorithms include AdaBoost, Gradient Boosting, and XGBoost, which have shown excellent performance across various machine learning tasks.

3.5 Text Preprocessing (NLP methods)

Text preprocessing is a crucial step in preparing textual data for machine learning models. In this thesis, we apply several fundamental Natural Language Processing (NLP) techniques to enhance the performance of our selected models [35].

3.5.1 Tokenization

Tokenization is often the first step in processing textual data for machine learning [35].

- Definition: Tokenization is the process of splitting text into individual words, phrases, symbols, or other meaningful elements called tokens. This can involve separating words by spaces, handling punctuation, and dealing with special cases like contractions or hyphenated words.

- Importance: Tokenization converts raw text into a format that machine learning models can use. It's a crucial step that allows models to work with discrete units of text, enabling further processing and analysis.

3.5.2 Stop Words Removal

Removing stop words helps focus the model on the most meaningful words in the text [35].

- Definition: Stop words are common words that typically do not contribute much to the overall meaning of a text. These often include articles, prepositions, and pronouns. Removing these words reduces the noise in the data and allows the model to focus on the more important, content-bearing words.
- Examples: Common English stop words include ”and,” ”the,” ”is,” ”in,” ”at,” and ”on.” The specific list of stop words can vary depending on the task and the domain of the text being analyzed.

3.5.3 Stemming and Lemmatization

Stemming and lemmatization are techniques used to reduce words to their base or root form, which can help in standardizing the vocabulary [35].

- Stemming: This process involves reducing words to their stem or root form by removing suffixes. For example, ”running,” ”runs,” and ”runner” might all be reduced to the stem ”run.” Stemming is generally faster but can sometimes produce non-words or fail to capture the correct meaning in context.
- Lemmatization: This technique reduces words to their base or dictionary form, known as the lemma. Unlike stemming, lemmatization considers the context and part of speech of the word to apply the correct transformation. For example, ”better” would be lemmatized to ”good” if used as an adjective.
- Importance: Both techniques help in standardizing words, reducing the vocabulary size, and improving the model’s ability to recognize similar words. This can lead to better performance, especially when dealing with limited training data.

3.6 Feature Extraction and Selection

Feature extraction and selection are crucial steps in preparing data for machine learning models, especially when dealing with text data [36].

3.6.1 Vectorization

Vectorization is the process of converting text data into numerical vectors that machine learning algorithms can process [35].

- Definition: Vectorization involves transforming text or categorical data into numerical vectors. This step is necessary because most machine learning algorithms work with numerical data.
- Techniques: Common vectorization techniques include:
  - TF-IDF (Term Frequency-Inverse Document Frequency): This technique shows how significant a word is within a specific document compared to a group of documents. The more a word is mentioned in the document, the higher its weight, but this is balanced out by how often the word appears in the corpus [35].
  - Word Embeddings: Techniques like Word2Vec [37], GloVe [38], or FastText create dense vector representations of words that capture semantic relationships. These embeddings can be pre-trained on large corpora or trained on the specific dataset.

3.6.2 Feature Selection

Feature selection involves choosing the most relevant features to improve model performance and reduce computational complexity [36].

- Definition: Feature selection is the process of selecting a subset of relevant features for use in model construction. It helps in reducing overfitting, improving model performance, and reducing training time.
- Techniques: Common feature selection methods include:
  - Chi-square test: This statistical test measures the dependence between a feature and the target variable, helping identify the most relevant features for classification tasks.
  - Mutual Information: This method assesses the interdependence between two variables, aiding in the discovery of characteristics that are most closely linked to the target variable.

3.7 Summary

This chapter has introduced the fundamental concepts necessary for understanding the approaches described in this thesis. We have discussed machine learning principles, focusing on supervised learning and classification tasks [31, 32]. We explored various evaluation metrics crucial for assessing model performance, especially in the context of imbalanced datasets.

We also examined techniques for handling imbalanced data, including both data-level methods like under-sampling and over-sampling, and algorithm-level methods such as cost-sensitive learning and ensemble methods [1]

Chapter 4

Definition of the problem

In the rapidly evolving fields of data science and machine learning, imbalanced data presents a significant and persistent challenge. This issue is pervasive across numerous practical applications, where datasets often exhibit a substantial disparity in the number of instances between different classes. Such imbalance is not merely a statistical curiosity but a fundamental problem that can lead to biased models. These models, while potentially performing well on the majority class, often display poor performance on the minority class. This discrepancy results in misleading evaluations and can ultimately lead to suboptimal decision-making in real-world scenarios.

The development of robust, fair, and accurate predictive models is contingent upon effectively addressing this imbalance. It's not just about improving model accuracy; it's about ensuring that the models are truly representative and useful across all classes, regardless of their relative frequencies in the training data.

This chapter aims to provide a comprehensive definition of the imbalanced data problem, delve into its profound impact on machine learning models, and explore the challenges and motivations for developing effective solutions. Our focus is particularly on the management and handling of multi-label datasets, which are increasingly common in real-world machine learning applications and present their own unique set of challenges in the context of data imbalance.

By thoroughly understanding the problem, we lay the crucial groundwork for the subsequent development and evaluation of strategies to mitigate data imbalance. This understanding is not just academic; it's a prerequisite for creating machine learning models that can perform reliably and fairly in the complex, often imbalanced landscapes of real-world data.

4.1 Understanding Imbalanced Data

4.1.1 Definition

Imbalanced data refers to datasets characterized by a non-uniform distribution of examples across various classes. In such datasets, one class—typically referred to as the majority class—significantly outnumbers the other classes, which are consequently termed minority classes. This imbalance can be extreme in many real-world scenarios.

For instance, in a fraud detection dataset, there might be 99,000 non-fraudulent transactions and only 1,000 fraudulent ones. This 99:1 ratio represents a significant class imbalance. It's important to note that the degree of imbalance can vary widely across different domains and applications. In some cases, the ratio might be even more extreme, such as 999:1 or worse. Understanding the specific nature and degree of imbalance in a dataset is crucial for developing appropriate strategies to address it.

4.1.2 Causes of Imbalance

The causes of data imbalance are diverse and often rooted in the nature of the phenomena being studied or the data collection process itself. Two primary causes are natural occurrence and data collection bias.

Natural occurrence refers to situations where the events or instances of interest are inherently rare. In medical studies, for example, certain diseases, particularly rare conditions, naturally occur less frequently in the population. A study on a rare genetic disorder might have data from thousands of healthy individuals but only a handful of affected patients. Similarly, in financial systems, fraudulent transactions are, by their nature, much less common than legitimate ones. The system's effectiveness in preventing fraud may further exacerbate this imbalance. In manufacturing quality control, defective products are (ideally) far less common than non-defective ones. These natural imbalances reflect the real-world distribution of events but pose significant challenges for machine learning models.

Data collection bias, on the other hand, occurs when the imbalance is a result of the data collection process itself. This can happen due to sampling bias in surveys or studies that oversample certain groups while undersampling others, perhaps due to accessibility issues, cost constraints, or unconscious biases in the study design. Technical limitations in data collection methods might make it easier to collect data for certain classes over others. Additionally, privacy or ethical concerns can limit the collection of data for certain classes, particularly in sensitive domains like healthcare or finance. Understanding the source of the bias is crucial for addressing it, either through data collection strategies or appropriate modeling techniques.

4.1.3 The Nature of Multi-Label Imbalanced Data

Multi-label imbalanced datasets present a more complex scenario where each instance can be associated with multiple labels simultaneously, coupled with an imbalanced distribution of these labels. This creates unique challenges that are distinct from those encountered in single-label or balanced datasets.

Multi-Label datasets typically exhibit several key characteristics. First, each data point can belong to several categories at once, leading to a combinatorial explosion of possible label sets. For example, in image classification, a single image might be labeled as both ”landscape” and ”sunset,” or in text classification, a document might be categorized under ”technology,” ”business,” and ”innovation” simultaneously.

Second, the frequency of different label combinations varies greatly. Some combinations may be very common, while others are extremely rare. This imbalance often follows a power-law distribution, where a small number of label combinations account for a large portion of the instances.

Third, label combinations that occur infrequently have relatively few instances in the dataset. This scarcity makes it challenging for models to learn these rare patterns effectively. The problem is compounded when these rare combinations are often the most interesting or critical ones in many applications.

Lastly, the multiplicity of labels and their combinations makes it more difficult and time-consuming to gather and annotate large datasets accurately. This can lead to further imbalances and potential biases in the collected data.

These characteristics of multi-label datasets introduce additional layers of complexity to the imbalanced data problem. They require sophisticated approaches that can handle not just the imbalance between individual labels, but also the intricate relationships and co-occurrences among multiple labels.

4.2 Impact of Imbalanced Data on Machine Learning

4.2.1 Model Bias

Machine learning models trained on imbalanced data often develop a bias towards the majority class. This bias is not just a statistical anomaly but a fundamental issue that can severely impact the model's real-world utility. The consequences of this bias are multifaceted and significant.

Firstly, models tend to perform poorly in predicting the minority class. This is particularly problematic when the minority class represents critical events (e.g., fraud, rare diseases) that are crucial to identify accurately. The model's inability to correctly identify these rare but important instances can have severe real-world consequences.

Secondly, despite poor performance on the minority class, these models often demonstrate high overall accuracy due to the predominance of the majority class. This can create a false sense of model effectiveness. For example, in a fraud detection scenario with a 99:1 ratio of non-fraudulent to fraudulent transactions, a model that predicts all transactions as non-fraudulent would achieve 99

Lastly, models may overfit to the patterns in the majority class, failing to generalize well to the underrepresented minority class. This overfitting can lead to poor performance when the model encounters new, unseen data, particularly instances from the minority class.

4.2.2 Evaluation Metrics

The challenge of imbalanced data extends to the evaluation of model performance. Traditional evaluation metrics can be misleading when applied to imbalanced datasets. Accuracy, as illustrated in the fraud detection example, can be highly misleading. A model predicting all instances as the majority class can achieve high accuracy without providing any valuable insights about the minority class.

To address these limitations, specialized metrics have been developed. These include Precision, which measures the accuracy of positive predictions; Recall, which indicates the proportion of actual positive cases that were correctly identified; and the F1-Score, which is the harmonic mean of precision and recall, providing a balanced measure. Additionally, the Matthews Correlation Coefficient (MCC) provides a balanced measure even for classes of very different sizes, while the G-mean (geometric mean of sensitivity and specificity) is useful for assessing the balance between classification performance on the majority and minority classes. The Area Under the Receiver Operating Characteristic Curve (AUC-ROC) measures the model's ability to distinguish between classes across various threshold settings.

These metrics provide a more nuanced and realistic evaluation of model performance on imbalanced datasets, helping to uncover issues that might be masked by simple accuracy measurements.

4.2.3 Evaluation Complexity in Multi-Label Contexts

The evaluation of model performance becomes even more intricate in multi-label settings. Standard metrics may not adequately capture the nuances of performance across multiple, potentially imbalanced labels. A model might perform well on some labels but poorly on others, and this variation needs to be accounted for in the evaluation process.

The relationships and co-occurrences between labels add another layer of complexity. Metrics need to consider not just individual label performance but also how well the model captures label interactions. In multi-label classification, predictions can be partially correct. For instance, a model might correctly predict some but not all labels for an instance. Evaluation metrics need to account for this partial correctness.

Besides evaluating performance on individual labels, it's often necessary to assess how well the model predicts entire sets of labels. This requires set-based evaluation metrics that can handle the combinatorial nature of multi-label predictions. These factors necessitate the use of specialized multi-label evaluation metrics and careful interpretation of results to provide a comprehensive assessment of model performance in multi-label, imbalanced data scenarios.

4.3 Challenges in Handling Imbalanced Data

4.3.1 Data-Level Challenges

Under-Sampling: This technique involves reducing the number of majority class instances to balance the dataset. While effective in addressing imbalance, it comes with significant drawbacks. There's a potential loss of important information from the majority class, risking under-representation of its variability. If not performed carefully, under-sampling can introduce bias, and determining the optimal under-sampling ratio remains a challenge.

Over-Sampling: This method increases the number of minority class instances, either through duplication or synthetic production. While it preserves all original data, it presents its own set of challenges. There's a risk of overfitting, especially when exact duplicates are used, and potential introduction of noise when creating synthetic samples. Over-sampling also increases computational cost due to a larger training dataset, and generating realistic synthetic samples for complex data types can be difficult.

4.3.2 Algorithm-Level Challenges

Cost-Sensitive Learning: This approach involves modifying algorithms to assign higher misclassification costs to minority class instances. While theoretically sound, it presents several practical challenges. There's a computational and financial burden in modifying and maintaining custom algorithms. Determining appropriate cost matrices is difficult, especially in multi-class problems, and there's potential for introducing new biases if costs are not carefully calibrated.

Model Adaptation: Adapting current methods for unbalanced data requires both extensive knowledge and substantial computational capacity. It necessitates a deep understanding of both the algorithm and the problem domain, coupled with high computational requirements for training and tuning adapted models. Maintaining model interpretability after adaptation can be challenging, as is generalizing these adaptations across different problem domains.

4.3.3 Cost-Sensitive Learning in Multi-Label Contexts

Cost-sensitive learning in multi-label environments presents unique challenges and opportunities. There's complexity in assigning different misclassification costs to every label or label combination, requiring sophisticated cost functions that can handle label dependencies. Balancing costs across multiple, potentially conflicting labels is challenging, and larger datasets are often required to effectively learn cost-sensitive decision boundaries in high-dimensional label spaces.

4.3.4 Real-World Constraints

Dynamic Imbalance: Many practical applications involve changing imbalance ratios over time, necessitating ongoing model adaptation. This creates challenges in detecting and quantifying changes in class distribution, and requires efficient online learning algorithms that can adapt in real-time. Maintaining model performance during transition periods is difficult, and robust evaluation metrics that can handle changing distributions are necessary.

Data Scarcity: Obtaining sufficient information for minority class cases can be challenging, particularly in rare event scenarios. There's limited availability of real-world examples for rare classes, often compounded by ethical and practical constraints in collecting sensitive minority class data. This creates challenges in validating models with limited minority class samples and necessitates creative data augmentation techniques that respect the underlying data distribution.

4.3.5 Evolving Data Streams and Scalability Concerns

Dynamic settings, such as those involving streaming data, require sophisticated approaches. There's a need for online learning systems capable of adapting to changing data distributions while balancing adaptability with stability. Scalability becomes a significant issue when applying traditional imbalance-handling methods to large-scale datasets, often requiring approximative algorithms or distributed computing frameworks to handle big data scenarios.

4.4 Need for Effective Solutions

4.4.1 Importance of Addressing Imbalance

Addressing class imbalance is crucial in many high-stakes applications. In healthcare, accurate classification of rare diseases can lead to early detection and improved patient outcomes. Cybersecurity relies on identifying infrequent but severe threats for robust system protection. In fraud detection, the ability to spot rare fraudulent activities can save significant financial resources, while in environmental monitoring, detecting uncommon events can be crucial for ecological preservation.

4.4.2 Current Gaps and Limitations

Existing methods for managing unbalanced data face several limitations. Many suffer from a lack of scalability to very large datasets or high-dimensional feature spaces, and have limited adaptability to changing data distributions or new domains. There's often difficulty in generalizing across diverse problem types and data modalities. Furthermore, there's a pressing need for more intuitive and accessible tools that allow data scientists and domain experts to effectively address imbalance in their specific contexts.

4.4.3 Emerging Techniques and Future Directions

Recent advancements offer promising avenues for addressing imbalance. The use of generative adversarial networks (GANs) for creating high-quality synthetic minority class samples has shown potential. Multidisciplinary approaches combining machine learning with domain-specific knowledge are increasingly important. There's also a growing emphasis on developing fair and explainable models, especially in sensitive applications where biased results can have significant ethical implications. Future research should focus on adaptive techniques that can automatically adjust to different types and degrees of imbalance, enhancing the robustness and applicability of imbalance-handling methods across various domains.

4.5 Chapter Summary

We have examined the difficulties in managing imbalanced data at both the data and algorithm levels as well as the practical restrictions that complicate these attempts. Clearly, there is a need for good answers, especially in high-stakes applications where minority class forecasts are crucial. The ideas covered here set the foundation for the next chapters, when we will explore particular techniques and approaches for reducing data imbalance and raising model performance.

Chapter 5

Methodology I

The approaches used in this thesis to solve the imbalanced data issue are described in this chapter. It addresses preparation, data collecting and the libraries used in this project. Moreover the classifiers combined with the methods are also presented. The next chapter emphasizes on each algorithm and method we used to address the imbalance. Every action is thoroughly described to give a full picture of the strategy followed.

5.1 Datasets

In this project, the following imbalanced datasets were used:

- Huffpost dataset (`https://huggingface.co/datasets/khalidalt/HuffPost`). A dataset of approximately 200K news headlines from the year 2012 to 2018 was collected from HuffPost.

- Indonesian 'smsa' dataset (`https://huggingface.co/datasets/indonlp/indonlu`). The comments and reviews in Indonesian that make up this sentence-level sentiment analysis dataset were gathered from various web sources.The SmSA dataset allows for the following three possible sentiments: neutral, negative, and positive.

- Amishshah dataset (`https://huggingface.co/datasets/amishshah/imbalanced`). This dataset contains the lyrics of random songs belonging to different types of music.

- Twitter financial news dataset (`https://www.kaggle.com/datasets/sulphatet/twitter-financial-news`). This dataset contains annotated corpus of tweets on financial News. Tweets about money are categorized using this dataset according to their subject.

- Emotion dataset (`https://huggingface.co/datasets/dair-ai/emotion`). This dataset contains English Twitter messages with six basic emotions: anger, fear, joy, love, sadness, and surprise.
- kinnews and kirnews datasets (`https://huggingface.co/datasets/andreniyongabo/kinnews_kirnews`). These datasets were collected from both Rwandan and Burundi news websites and newspapers.

5.2 Data Preprocessing

To preprocess the text data, several steps were implemented using Python, specifically the Natural Language Toolkit (NLTK) library. Eliminating undesired characters and stopwords helped to clean the text so that the data would be in a proper state for the next investigation. The clean-text function implemented, which receives a text string as input and performs several cleaning operations, is responsible for:

- Removing Accents and Non-ASCII Characters.
- Removing Punctuation.
- Removing Extra Whitespace.
- Removing Stopwords.
- Converting to Lowercase.

By efficiently cleaning and standardizing the text input, this prepocessing method removes noise and irrelevant components, enhancing the quality and performance of the models we are going to use in order to handle the imbalance of the data.

5.3 Text Vectorization and Data Splitting

After the data cleaning, the text is turned into numerical features fit for machine learning techniques. We used the TF-IDF vectorizing method for this purpose. To assess the models' performance, the dataset also comprised training and test sets.

TF-IDF features are produced from cleaned text data using the TfidfVectorizer in the sklearn module [39]. The text data is converted into TF-IDF vectors by means of the transform algorithm, which learns vocabulary and inverse document frequency from the corpus. Every text entry is shown as a numerical vector that, considering the whole dataset, catches the significance of every word.

The feature matrix and the target variable are divided into training and testing sets. Creating training and test sets helps to evaluate model performance objectively. Validation of the efficacy of the data balancing techniques applied in the project depends on an understanding of how effectively the model generalizes to new, unknown data.

5.4 Implementation Details

Several Python libraries were utilized to complete the chores required for processing and evaluating unbalanced data. In the stages of the project's preprocessing, modeling, and evaluation, every library fulfills a certain need.

5.4.1 Visualization Tools

For visualization of the data we used the Matplotlib library and 'matplotlib.pyplot' tool for creating static, animated, and interactive visualizations in Python in order to apply EDA analysis.This tool is crucial for analyzing findings, correlations, and data distributions. It lets data scientists and researchers design a range of plots—including line graphs, bar charts, histograms, scatter plots, and more.Exploratory data analysis depends critically on visualizing data. It clarifies fundamental trends, finds anomalies, and presents results in an understandable style.

5.4.2 Tools and Libraries used for preprocessing tasks

In the preprocessing phase of our research, we utilized several powerful Python libraries and tools to clean, normalize, and prepare our text data for analysis. Each of these tools played a crucial role in ensuring the quality and consistency of our data:

- **pandas:** This library was instrumental in our data handling and manipulation processes [40]. We used pandas for:

  - Loading and saving our datasets in various formats (CSV, Excel, etc.)
  - Efficient data manipulation through its DataFrame structure
  - Handling missing values and data cleaning
  - Basic statistical analysis of our dataset
  - Merging and reshaping datasets when necessary

  Pandas' ability to handle large datasets efficiently made it invaluable for processing our multi-label data.

- **nltk (Natural Language Toolkit):** NLTK was essential for our text preprocessing tasks [41]. We utilized it for:

  - Tokenization: breaking down text into individual words or tokens
  - Removing stopwords: eliminating common words that don't carry significant meaning
  - Stemming and lemmatization: reducing words to their base or dictionary form
  - Part-of-speech tagging: identifying the grammatical parts of speech in our text data
  - Named entity recognition: identifying and classifying named entities in the text

  NLTK's comprehensive set of text processing tools allowed us to clean and normalize our text data effectively, preparing it for further analysis and model training.

- **unidecode:** This library played a crucial role in normalizing our text data [42]. We used unidecode for:

  - Converting Unicode characters to their closest ASCII equivalents
  - Handling text with accents, diacritics, and non-Latin characters
  - Ensuring consistency in character encoding across our dataset
  - Simplifying text comparison and analysis by reducing character variety

  Unidecode helped us address challenges related to multilingual text and special characters, ensuring our models could process the text data uniformly.

- **string:** The Python string module provided essential string manipulation functions [43]. We utilized it for:
  - Removing punctuation from our text data
  - Performing case conversions (e.g., converting text to lowercase)
  - Stripping whitespace from the beginning and end of strings
  - Replacing or removing specific characters or substrings

  The string module's functions allowed us to standardize our text data, removing elements that could introduce noise or inconsistencies in our analysis.

These tools and libraries, working in concert, enabled us to transform raw, unstructured text data into a clean, normalized format suitable for our multi-label classification tasks. Their use was crucial in ensuring the quality and consistency of our input data, which in turn contributed significantly to the effectiveness of our machine learning models.

5.4.3 Tools and Libraries used for the methods and algorithms

- scikit-learn: A machine learning library used for model training, text vectorization, and evaluation metrics [44].
- imbalanced-learn: A library that provides tools to handle imbalanced datasets, including metrics such as geometric mean [45].
- The XGBoost library provides powerful tools for model training, particularly for handling imbalanced datasets and improving predictive performance. The XGBClassifier is used to train the model, which includes hyperparameter tuning to optimize performance [46].

5.5 Classifiers

The following classifiers are combined with the methods and algorithms used to address the imbalanced data in our project:

5.5.1 Random Forest Classifier

Building several decision trees and aggregating their predictions, the flexible ensemble learning tool Random Forest classifier increases accuracy and control overfitting. [47] Leveraging its bagging and feature randomization to capture intricate patterns, Random Forest can effectively manage many classes and label dependencies when applied to multi-label unbalanced data. Its strengths are simplicity of understanding, handling of high-dimensional data, and noise tolerance. Its shortcomings in this context, however, center possible bias toward the majority classes since it may find it difficult to fairly balance forecasts across all labels, particularly if the mismatch is significant.

5.5.2 Logistic Regression Classifier

Logistic Regression is a linear model used for binary classification that can be extended to multi-label classification by fitting one model per label (one-vs-rest approach). [48] Particularly for large datasets with a linear decision limit, its simplicity, interpretability, and efficiency are its major strengths. Logistic Regression struggles to capture complicated label dependencies and performs poorly on minority classes when used to multi-label imbalanced data, though.

5.5.3 KNN Classifier

A non-parametric, instance-based learning method, the k-Nearest Neighbors (KNN) classifier labels data points according on the majority label of their k-nearest neighbors. [49] By considering the labels of surrounding instances, KNN can naturally manage several labels when applied to multi-label unbalanced data. Its capacity to capture complicated label associations and simplicity, flexibility, and efficacy with well-distributed data define its strengths. But given big datasets and class imbalance, its flaws become clear. Since KNN must compute distances for all instances, it can be computationally costly and biassed toward majority classes.

5.5.4 XGB Classifier

Using boosting techniques to combine several weak learners—usually decision trees—the XGBoost classifier is a strong, scalable ensemble learning system. [50] Because of its boosting structure, which iteratively increases model performance, XGBoost can efficiently simulate complicated label dependencies and interactions applied to multi-label imbalanced data. High accuracy, strong resistance to overfitting, and feature interaction handling capability help to define its strengths. Missing data handling is another. Its shortcomings, then, include sensitivity to hyperparameter settings and possible overfitting on noisy data.

5.5.5 MLP Classifier

The MLP classifier is a type of feed-forward artificial neural network that consists of multiple layers of nodes, allowing it to model complex, non-linear relationships. [51] MLP may use its deep learning structure to detect complex patterns and dependencies among labels when applied for multi-label unbalanced data. Its strengths are great performance in learning non-linear relationships, high flexibility, and handling of big and high-dimensional data. Its shortcomings, however, are notable given imbalanced data since MLPs depend on a lot of training data, are computationally demanding, and can overfit readily without appropriate regularizing.

5.6 Support Vector Machines

SVMs are powerful classification algorithms that work by finding the optimal hyperplane that maximizes the margin between different classes. [48]SVMs can be tuned for multi-label imbalanced data by means of one-vs-rest or one-vs-one approaches to manage several labels. Their strengths are in clear decision boundaries, robustness to overfitting (particularly with suitable kernel functions), and effectiveness in high-dimensional environments. SVMs struggle with imbalanced data, though, since they often favor majority classes and could need huge processing resources for big datasets.

Chapter 6

Methodology II

This chapter presents in detail all the methods and algorithms used in this work, their advantages and disadvantages and further comments on their use and specificity. Each method is combined and tested on different classifiers. The metrics shown in tables for each method used , refer to the ”twitter_financial_news.csv” dataset. This example experiment contains every method and algorithm used in this work.

6.1 Over-Sampling Methods/Algorithms

6.1.1 SMOTE

To equal the class distribution, Smote creates synthetic examples for the minority class. Unlike basic oversampling, which replicates current minority class examples, Smote generates new, synthetic examples by interpolating between existing samples [52].

To initialize SMOTE we followed the steps below:

- The SMOTE class is imported from the imblearn library.
- The dataset is split into training and testing sets using ’train_test_split’ tool from sklearn library. This separation is crucial to evaluate model performance on unseen data.
- An instance of the SMOTE class is later created. The ’random_state’ parameter ensures reproducibility by controlling the random number generation used during the resampling process.

After the initialization, SMOTE is applied via the ’fit_resample’ method. The resampled feature matrix and target labels are produced. These contain synthetic examples generated by SMOTE, which increase the number of minority class instances.

SMOTE’s main benefit comes from its capacity to create synthetic rather than replicas of the minority class, therefore improving their representation. This method not only helps to balance the class distribution but also offers a more varied dataset that could lower bias and enhance model performance. [53] SMOTE has certain shortcomings, though. Particularly if the synthetic samples it creates closely match current data points, overfitting may result from them, therefore affecting the generalizing capacity of the model. Furthermore because of the larger dataset, SMOTE can raise computational complexity and training time. Synthetic samples with noise from the minority class run the danger of degrading model performance as well. Moreover, SMOTE could find it difficult with high-dimensional data and intricate class distributions, so producing less good synthetic examples [54]. Ultimately, SMOTE does not naturally balance the majority class, even when it addresses imbalance in the minority class; this may call for the employment of other approaches. Therefore, even while SMOTE is a useful instrument for reducing class imbalance, its constraints should be properly thought out and resolved to maximize model performance.

In the table below,we present the performance metrics of SMOTE applied on different classifiers.

Table 6.1: SMOTE with Classifiers

Method	**Accuracy**	**Precision**	**Recall**	**F1 score**	**Gmean**	**MCC**	**ROC AUC**
RF without SMOTE	0.78	0.79	0.78	0.77	0.87	0.75	0.97
RF-SMOTE	0.79	0.79	0.78	0.78	0.72	0.76	0.98
LogisticRegression-SMOTE	0.81	0.81	0.81	0.81	0.89	0.79	0.98
KNN-SMOTE	0.33	0.83	0.33	0.35	0.55	0.37	0.67
XGB-SMOTE	0.78	0.78	0.78	0.78	0.75	0.75	0.97
MLP-SMOTE	0.81	0.82	0.82	0.81	0.75	0.79	0.98
SVM(linear)-SMOTE	0.82	0.83	0.82	0.82	0.90	0.80	0.98

- SMOTE is generally effective in improving the performance of classifiers on imbalanced data. This is evidenced by the enhanced performance metrics (accuracy, precision, recall, F1 score, Gmean, MCC, and ROC AUC) when SMOTE is applied.

- Top performers are SVM (linear), logistic regression, and MLP with SMOTE. On all measures, they show good results; especially in recall and F1 score, which are vital for handling skewed datasets. This implies that these classifiers are quite good in balancing the class distributions and appropriately classifying minority classes when used with SMOTE.

- For this specific multi-label imbalanced data situation, SVM (linear) with SMOTE exhibits the best performance and is therefore the most appropriate method. In almost all measures it beats other classifiers.

- Though somewhat below SVM, Logistic Regression, and MLP,RF and XGB with SMOTE demonstrate good results. Although their ROC AUC ratings indicate they are still dependable decisions, they might not be as best as the top performers in this particular field.

- KNN with SMOTE suffers especially in accuracy, recall, and F1 score. Its low recall, while great accuracy, shows that it misses many cases of the minority class, hence unfit for imbalanced multi-label classification problems.

6.1.2 ADASYN

Using synthetic instances for minority classes, ADASYN is a resampling method meant to solve class imbalance. Unlike SMOTE, ADASYN emphasizes on generating more synthetic samples for minority class events that are difficult to learn, hence dynamically increasing the density in these regions [53]. A more balanced and sophisticated dataset follows from this.In our work , ADASYN provided the best results overall.

ADASYN's adaptive approach—which concentrates on challenging-to-learn events to increase the classifier's capacity to generalize from the minority class—is its main advantage. By so lessening the tendency towards majority classes, this can improve classification performance—especially in terms of memory and F1 score. Furthermore, ADASYN's adaptability lets it be integrated with several classifiers and extended to multi-label situations, thereby managing several minority classes at once [55].

However, ADASYN also has some disadvantages.ADASYN's adaptive character brings extra computing complexity and can call for more fine-tuning than more basic techniques like

SMOTE. Overfitting is another possibility, particularly in cases when the synthetic data distribution does not fairly reflect the underlying distribution. Moreover, the efficacy of ADASYN can be sensitive to its parameters, including the number of nearest neighbors, thus careful selection is necessary to get best results. Though it has difficulties linked to complexity and parameter sensitivity, ADASYN provides a comprehensive way to handle multi-label imbalanced data overall, with great advantages in enhancing minority class learning.
In the table below,we present the performance metrics of ADASYN applied on different classifiers.

Table 6.2: ADASYN with Classifiers

Method	**Accuracy**	**Precision**	**Recall**	**F1 score**	**Gmean**	**MCC**	**ROC AUC**
ADASYN + RF	0.97	0.97	0.97	0.97	0.99	0.97	0.999
ADASYN + XGB	0.92	0.92	0.92	0.92	0.95	0.91	0.998
ADASYN + LR	0.97	0.97	0.97	0.97	0.98	0.97	0.999

ADASYN produced the best results in our project , especially combined with RF classifier. We found that combining ADASYN with the RF classifier can efficiently solve the shortcomings of ADASYN while using the strengths of both approaches, hence producing better performance in managing multi-label imbalanced datasets. Random Forest's strong ensemble method helps ADASYN's main weakness—the potential of overfitting resulting from synthetic data—to be lessened by building many decision trees and averaging their predictions. Furthermore, Random Forest's capacity to capture complicated relationships and manage high-dimensional data supports ADASYN's adaptive synthesis of synthetic samples, therefore assuring that challenging-to-learn minority cases are well-represented and taught by the model. By so boosting the generalizing capacity of the model and so balancing the class distribution, the combination improves the performance metrics of the classifier including recall and F1 score. By means of this synergy between ADASYN's adaptive sampling and Random Forest's ensemble learning, the problems of multi-label imbalanced datasets are effectively addressed, thereby offering enhanced accuracy and robustness.

6.2 One Sided Selections

Aiming to minimize the size of the majority class in imbalanced datasets by deleting duplicate or noisy instances while keeping the most informative samples, one-sided selection methods are undersampling techniques. These techniques enhance the equilibrium between the majority and minority classes without sacrificing important data. One-sided selection's main benefits are its capacity to lower the computational load and storage needs by removing pointless majority class samples, so improving the classifier's efficiency and concentration. Furthermore, one-sided selection enhances the general model performance and resilience by data cleaning. Its shortcomings, however, include the possibility of losing important data should the selection criteria be not well defined, therefore compromising the generalizing capacity of the model. One-sided selection strategies might be difficult to apply in multi-label situations since they must consider several interdependent labels, therefore making it difficult to find and keep the most relevant samples [56].
In the table below, the performance metrics of OSS methods when applied on different classifiers is shown.

Table 6.3: OSS with Classifiers

Method	Accuracy	Precision	Recall	F1 score	Gmean	MCC	ROC AUC
RandomForest without OSS	0.78	0.79	0.78	0.77	0.87	0.75	0.97
RF-OSS	0.75	0.78	0.75	0.75	0.85	0.72	0.97
GradientBoosting-OSS	0.68	0.73	0.68	0.68	0.80	0.65	0.93
LogisticRegression-OSS	0.64	0.74	0.64	0.62	0.77	0.61	0.98
KNNclassifier-OSS	0.64	0.66	0.64	0.62	0.78	0.60	0.88
XGBclassifier-OSS	0.75	0.77	0.75	0.75	0.84	0.72	0.93
MLPclassifier-OSS	0.78	0.79	0.78	0.78	0.87	0.76	0.98

On this specific multi-label imbalanced dataset, one-sided selection (OSS) did not assist to improve the performance of classifiers. Generally speaking, OSS resulted in worse crucial performance measures like accuracy, recall, F1 score, and MCC. This implies that OSS's method of lowering majority class instances did not significantly improve the classifiers' capacity to manage imbalance for this dataset and might have even deleted important data.

6.3 Boosting

Boosting is a strong ensemble method forming a strong classifier by aggregating the predictions of several weak learners. By concentrating on hard-to-classify events, usually those from the minority classes, boosting techniques can dramatically enhance classification performance when used on imbalanced datasets. In multilabel situations—where every instance can belong to several classes—often with an uneven distribution—this method is especially helpful. The several boosting techniques applied in this research are briefly summarized below together with their special qualities and how they help to improve the performance of the model on imbalanced data.

6.3.1 XGBoost

Especially on big and complicated datasets, XGBoost is a very efficient and scalable boosting method with well-known speed and performance. Its perfect for managing high-dimensional unbalanced data since its performance results from the application of modern optimization strategies like parallel processing and distributed computing. XGBoost also uses L1 and L2 regularization to help avoid overfitting—a typical problem with imbalanced datasets, in which case the model could otherwise concentrate too much on the majority class. XGBoost's intricacy can be a two-edged blade, too; it calls for careful parameter optimization to reach best performance, especially in situations with significant class imbalance. Furthermore, its memory consumption can be really high, which could provide difficulties dealing with rather big datasets [57].

6.3.2 LightGBM

LightGBM is ideally suited for real-time applications since it makes use of a histogram-based technique that speeds up the training process and lowers memory use. LightGBM's capacity to naturally manage categorical features is one of its benefits; this helps to minimize the preprocessing chores needed before model training. In skewed datasets, where categorical data may be rather important, this is especially helpful. By concentrating more on minority classes and hence modifying the learning process, LightGBM additionally offers special settings like is ”_unbalance” and ”scale_pos_weight” that directly address class imbalance. Like XGBoost, though, it depends on careful adjustment and its complexity could be too much [58].

6.3.3 CatBoost

CatBoost distinguishes itself for its autonomous handling of categorical variables, therefore removing the need for significant data preparation. When working with imbalanced datasets where categorical data is rather common, this function is especially useful. CatBoost is also a strong candidate for classification problems since it shines in avoiding overfitting even in tiny or skewed datasets. With little hyperparameter adjustment, it also provides good out-of-the-box performance. Though CatBoost is quite good in many situations, it can be slower than LightGBM, especially with very big datasets, and may call for additional processing resources [59].

6.3.4 AdaBoost

One of the first boosting methods, AdaBoost remains popular for its simplicity and efficiency. It is especially helpful for enhancing the classification of minority classes in imbalanced datasets since it iteratively concentrates on the examples that are toughest to categorize. AdaBoost may thus need to be supplemented with other approaches, such sampling strategies, to increase its performance on minority classes since it is not intrinsically suited to manage class imbalance. Furthermore sensitive to noisy data and outliers, AdaBoost can cause overfitting [60].

6.3.5 AdaUBoost

Designed especially to solve class imbalance, AdaUBoost is a variant of AdaBoost. It guarantees that these cases get greater attention during training by giving the minority class more weights, hence changing the conventional boosting procedure. This change improves the classifier to perform better on minority classes, so AdaUBoost is especially useful in situations with appreciable class imbalance. Nonetheless, AdaUBoost's efficiency mostly relies on the degree of imbalance and may need careful weight adjustment mechanism tuning to prevent overfitting to the minority class. The more attention paid to the minority class, the model might underperform on some minority labels and overfit to others. Extreme label distribution in the dataset could caused this focus imbalance to reduce metrics.

6.3.6 AsymBoost

AsymBoost is meant to manage imbalanced datasets' asymmetric costs related to misclassification. AsymBoost guarantees that the model pays greater attention to correctly categorizing minority class instances by imposing a bigger penalty on mistakes in the minority class, therefore addressing issues where the cost of misclassifying minority class instances is substantial, as in medical diagnosis or fraud detection. AsymBoost does, however, efficiently correct the imbalance, but it could also cause overfitting to the minority class, especially if the model gets overly focused on these cases at the price of general model accuracy. AsymBoost penalizes minority misclassifications throughout the boosting process. This approach fails in multilabel contexts but performs well for binary imbalanced data. Strong penalties lead the model to overlook some labels and overfit some others [61].

6.3.7 CompatibleAdaBoost

Designed to remain compatible with particular data properties, such imbalance or noise, CompatibleAdaBoost is an adaptation of the conventional AdaBoost algorithm. Particularly from the minority class, this approach modifies the boosting process to better concentrate on the most difficult cases, so improving the classification performance in imbalanced datasets. Nevertheless, the type of the imbalance and the existence of noise in the data will affect the efficacy of the algorithm.Handling many labels with varying imbalances in multilabel imbalanced datasets can overwhelm the method. Compatible AdaBoost could not be able to manage multilabel relationships, hence performance decreased. The approach may not scale effectively when handling several classes, reducing precision and recall among labels.

6.3.8 GradientBoosting

Built sequentially, gradient boosting is a flexible and strong ensemble method whereby each new model fixes the mistakes made by the one before it. It is quite flexible and can be personalized to solve class imbalance by changing the loss function to give minority classes more weight. GradientBoosting is a flexible tool in managing multilabel imbalanced datasets since of this adaptability. Nevertheless, especially in the presence of considerable class imbalance, it can be computationally demanding and may require careful tuning to balance the trade-off between model complexity and performance, just as other boosting techniques [62].

6.3.9 HistGradientBoosting

A variation of GradientBoosting, HistGradientBoosting uses histogram-based methods to hasten the training process and lower computational complexity. For big datasets especially, this approach is quite efficient since it greatly lowers memory use without compromising accuracy. HistGradientBoosting can be tuned in the framework of imbalanced data to give minority class correct classification top priority, so enhancing the general model performance. Though it is more efficient than conventional GradientBoosting, especially in highly imbalanced situations, it could still need careful tuning to get the best results.

6.3.10 SmoteBoost

SMOTE and boosting are combined under SmoteBoost. Before using the boosting technique, Smote creates synthetic samples for the minority class, so reducing the class imbalance by raising minority class representation in the training data. In especially highly imbalanced datasets, this method produces more accurate and balanced forecasts. The synthetic samples produced by Smote could, however, introduce noise, particularly if the original minority class instances are not well-separated from the majority class, so lowering the general model efficiency.Because the classifier finds it difficult to separate closely related labels, this noise compromised the performance of the model and resulted in poor metrics especially in precision and recall [63].

6.3.11 RUSBoost

RUSBoost generates a more balanced dataset by lowering the majority class instance count, so enabling the boosting algorithm to concentrate more on the minority class. On the minority class, this method can help to improve classification performance. RUSBoost's drawback is the possible loss of important information from the majority class, which can lower general model performance particularly in cases of too aggressive under-sampling. Particularly for majority classes that were under-sampled, this information loss caused poor performance since the model did not have enough data to learn effectively, generating low accuracy and recall [64].

6.3.12 AdaCost

AdaCost is a cost-sensitive variation of AdaBoost whereby, depending on the class, misclassification errors are assigned varying costs. Misclassifying minority class instances in imbalanced datasets usually has more cost. AdaCost solves this by changing the boosting process to reduce these expensive mistakes, so improving the performance on minority groups. Like other cost-sensitive strategies, though, the difficulty is in correctly establishing the cost limits.Due to the difficulty of cost changes be closely matched to the particular imbalances in the data, the model may started to favor some labels over others. Poor overall measures follow from the classifier overcompensating for some labels at the expense of others, so producing an imbalanced performance across the several classes [65].

6.3.13 KmeansSMoteBoost

Advanced method KmeansSMoteBoost combines SMOTE and boosting with K-means clustering. Clusters in the data are found using K-means; SMote is then used inside these clusters to create synthetic samples for the minority class. When data shows obvious clustering tendencies, this method is especially successful. For large datasets though with complicated structures it demands major computational resources and so introduces noise and artifacts into the training data, confusing the boosting algorithm and leading to poor classification metrics.

6.3.14 OverBoost

OverBoost guarantees that the minority class instances get enough attention during training by repeatedly sampling and boosting them, so helping to reduce the consequences of imbalance. On the minority class, this method can considerably raise the performance of the model. Over-sampling however, brings noise and redundancy, especially if the same minority class instances are repeatedly sampled and causing overfitting and decreased generalization to unprocessed data.

6.3.15 SelfPacedEnsemble

Combining boosting with self-paced learning ideas, SelfPacedEnsemble is a novel ensemble learning method. It progressively adds simpler samples to the model while iteratively concentrates on the most difficult samples—especially from the minority class. This self-paced approach guarantees that the model pays enough attention to all classes, especially in unbalanced environments, so helping to balance the learning process.However, the self-paced learning process can be slower than the other boosting methods and correct tuning is needed [66].

The following table includes the performance metrics of the Boosting methods:

Table 6.4: Boosting Methods

Method	Accuracy	Precision	Recall	F1 score	Gmean	MCC	ROC AUC
XGBoost	0.77	0.78	0.77	0.77	0.86	0.74	0.97
LightGBM	0.78	0.79	0.78	0.78	0.87	0.76	0.97
CatBoost	0.66	0.70	0.66	0.66	0.79	0.63	0.94
AdaBoost	0.27	0.15	0.27	0.15	0.47	0.21	0.6
AdaUBoost	0.04	0.13	0.04	0.04	0.2	0.09	0.59
AsymBoost	0.27	0.18	0.27	0.15	0.47	0.19	0.6
CompatibleAdaBoost	0.26	0.15	0.26	0.14	0.46	0.19	0.6
GradientBoosting	0.72	0.75	0.72	0.71	0.82	0.69	0.92
HistGradientBoosting	0.69	0.70	0.69	0.68	0.81	0.65	0.93
SMOTEBoost	0.03	0.19	0.03	0.03	0.16	0.07	0.56
RUSBoost	0.3	0.36	0.3	0.23	0.50	0.21	0.66
AdaCost	0.22	0.09	0.22	0.09	0.42	0.07	0.64
KmeansSMOTEBoost	0.03	0.1	0.03	0.03	0.16	0.06	0.56
OverBoost	0.11	0.15	0.11	0.08	0.32	0.16	0.6
SelfPacedEnsemble	0.49	0.60	0.49	0.5	0.68	0.47	0.92

6.4 Bagging

Bagging is an ensemble learning technique designed to improve the stability and accuracy of machine learning algorithms. It generates several iterations of a dataset using bootstrapping—random sampling with replacement—then trains a model on every version. These models' predictions are then combined usually by voting or averaging to generate a final prediction. Bagging can be especially helpful in the framework of imbalanced datasets since it diversifies the training samples so helps to lower variance and prevent overfitting. Below, we discuss the performance of the different bagging methods used in this project.

6.4.1 Standard Bagging

Standard Bagging proved effective since it averages forecasts from several models, so lowering the variance. Its drawback, then, is that it does not directly solve the class imbalance, which might produce biassed models supporting the majority class. Still, its ensemble character enabled it to get respectable performance on several criteria [67].

6.4.2 OverBagging

OverBagging aims to balance classes by oversampling the minority class. In this case, though, it resulted in overfitting, as the model probably became too tailored to the minority class samples. Since the model lost its capacity to generalize, this overfitting produced poor classification results and a decline in performance across all measures [68].

6.4.3 UnderBagging

Underbagging balances the data by shrinking the majority class size. This sometimes results in the loss of important knowledge from the majority class, which might underperformance of the model. Underbagging's low recall and accuracy suggest that, despite its high ROC AUC, its reduced data diversity most likely caused it to miss enough cases [69].

6.4.4 UnderOverBagging

Combining under-sampling of the majority class with over-sampling of the minority class under OverBagging seeks to balance data diversity and representation. This approach most certainly produced the most strong model since it avoided the overfitting (as shown in OverBagging) and information loss (as shown in UnderBagging). UnderOverBagging handled the imbalanced data successfully by combining the strengths of both methods, so performing well across all measures.

The metrics of Bagging methods are presented in the table below:

Table 6.5: Bagging Methods

Method	Accuracy	Precision	Recall	F1 score	Gmean	MCC	ROC AUC
Bagging	0.72	0.74	0.72	0.72	0.83	0.69	0.93
OverBagging	0.27	0.16	0.27	0.14	0.47	0.21	0.59
UnderBagging	0.48	0.72	0.49	0.45	0.68	0.50	0.97
UnderOverBagging	0.74	0.74	0.74	0.73	0.84	0.71	0.94

6.5 SVM

Support Vector Machines (SVM) are powerful supervised learning models used for classification tasks. In a high-dimensional space, SVM finds the hyperplane most suited to divide the data points of several classes. SVM can be especially useful in the framework of imbalanced datasets since it emphasizes on maximizing the margin between classes, so enabling the distinction of minority class examples. But especially in multi-label and imbalanced data, the kernel function used in SVM—Linear, Polynomial, RBF or Sigmoid, much affects its performance. Every kernel handles imbalanced class distributions and challenging decision limits differently, with strengths and shortcomings. The performance of four distinct SVM techniques used in this project is examined below together with an analysis of why each performed as it did.

6.5.1 LinearSVM

Finding the hyperplane that best divides the classes helps Linear SVM to classify data points using a linear kernel. When the classes are either linearly separable or near to being so, it works well. Large datasets are fit for the linear kernel since it is computationally efficient and performs effectively with high-dimensional data. Linear approach provided a good balance between model simplicity and classification power.

6.5.2 PolynomialSVM

Polynomial SVM models provide more complicated decision boundaries by using a poisson kernel. The degree of the polyn shapes the decision surface's complexity. Higher degrees run the danger of overfitting, particularly in noisy or imbalanced datasets, even while they can catch more complex trends in the data. Polynomial SVM found it difficult to generalize in this setting. The tendency of the polyn kernel to overfit when applied to imbalanced datasets, where the decision boundaries might become too complicated, capturing noise instead of the underlying patterns, can help to explain the poor performance [70].

6.5.3 RBFSVM

RBFSVM employs the Radial Basis Function (RBF) kernel. By mapping input data into a higher-dimensional space, the RBF kernel enables the creation of more flexible decision boundaries able to capture the underlying patterns in challenging datasets. The RBF kernel performed well most of the time since its adaptability let it represent intricate relationships in the data. Its slight inclination to overfit to the majority class in an imbalanced dataset, however, may help to explain why its performance lacked that of Linear SVM in terms of accuracy and MCC [71].

6.5.4 SigmoidSVM

The sigmoid kernel, behaves similarly to a neural network with one hidden layer. Sigmoid SVM is helpful when data shows a sigmoid-like distribution in feature space as the sigmoid kernel can capture non-linear interactions, which occured in our task making it a good fit. Its performance was almost that of Linear SVM, suggesting that although without the overfitting problems observed in Polynomial SVM, the underlying patterns of the data might have some non-linear elements that the sigmoid kernel was able to effectively exploit [72].
Here is the metric table for different SVM kernels:

Table 6.6: SVM-kernels

Method	**Accuracy**	**Precision**	**Recall**	**F1 score**	**Gmean**	**MCC**	**ROC AUC**
LinearSVM	0.81	0.82	0.81	0.81	0.89	0.79	0.99
PolynomialSVM	0.38	0.77	0.38	0.36	0.57	0.37	0.97
RBFSVM	0.74	0.80	0.74	0.74	0.84	0.72	0.98
SigmoidSVM	0.80	0.82	0.80	0.80	0.88	0.79	0.99

Chapter 7

Web Application Development

This chapter explores the web application development process employing a high-level Python web framework called Django. This part offers a thorough summary of the codebase supporting our application, including the structure and capabilities of the Django tools and utilities applied to support different features.

7.1 Django Framework Overview

We used Django framework to implement the web page used in this project.Django is a high-level Python web framework that encourages rapid development and clean, pragmatic design. [73] It was developed to offer a strong collection of components and a clear design, therefore streamlining the creation process of sophisticated, database-driven websites. Using a ”batteries-included” approach, Django provides a wide spectrum of built-in capabilities like an ORM for database interactions, a simple URL routing system, a flexible template engine, and thorough authentication procedures. Emphasizing reusability and ”DRY” (Don’t Repeat Yourself) ideas, which lower redundancy and improve maintainability, is one of Django’s strongest suit. The modular design of the framework lets builders easily plug in various parts and outside tools. Development of safe web applications favors Django because of its robust security features—protection against typical web vulnerabilities such SQL injection,XSS and CSRF. Furthermore, the active community of Django and thorough documentation offer developers great tools and help. Within the framework of this thesis, Django was selected for its capacity to enable the fast building of a strong online application enabling users to upload CSV files, execute statistical analysis, identify data imbalances, and apply balancing methods. The effective creation and implementation of the web-based solution of the project benefited much from the scalability, dependability, and simplicity of use of the framework.

7.2 Application Architecture

Designed especially for Django, the web application created for this thesis follows the MVT architectural pattern, a variation of the MVC architecture. Three key components: models, views, and templates, separate the program in this design, therefore guaranteeing a clear separation of concerns and improving maintainability. [74]

7.3 HTML

HTML was crucial in this project in terms of information arrangement and layout, so guaranteeing a clear, understandable interface accessible to consumers. By use of HTML, the project was able to specify the framework of user inputs—forms for uploading CSV files, for example—as well as the display of data outputs—including tables for statistical analysis and visual summaries of data imbalance. HTML tags made it possible for material to be presented consistently across several pages, therefore enhancing the user interface. Furthermore, by including Django's templating engine, HTML templates dynamically displayed user-specific data, including immediately on the web pages the outcomes of data balancing techniques. The interactivity and usefulness of the program were much improved by this capacity to dynamically create and present materials depending on user inputs. HTML's overall contribution was vital in organizing the web application, allowing user interaction, and guaranteeing that the program successfully presented the findings and functionality to the end-users.

7.3.1 Models

Python classes defined in Django that determine the data structure of the application constitute the models there. Every model fits one single database table; Django's ORM manages the conversion from the model to the database table. We constructed the following models for our website:

- 'User': The User model functions as a tailored variation of Django's standard user login mechanism. Inheriting from 'AbstractUser', this model keeps all of Django's built-in user management capabilities—including authentication, rights, and user-related data handling. Additional fields, such ’phone’ and ’company’, improve the User model by allowing to store particular data pertinent to the context of the application. The phone field lets the program save a user’s contact number, which might be checked or used for correspondence. The ’company’ field is the name of the company or organization the user is connected to, which helps the application to accommodate functionalities particular to businesses. While storing extra data that improves user profiles inside the application, this model is absolutely important in managing and authenticating users.

- ’Datasets’: The ’Dataset’ model is meant to show the datasets users enter and oversee into the application. By means of a foreign key relationship, every Dataset instance is connected to a designated user, so guaranteeing the association of datasets with their respective owners. Fields including ’name’, which denotes the dataset, and file—which keeps the real dataset file in a specified directory (datasets) are part of the model. Providing a chronology for dataset management, the uploaded_at field automatically notes the timestamp upon dataset upload. This model’s main goal is to help users to manage and track the datasets they are working with in different machine learning applications by means of storage, retrieval, and organization of datasets uploaded by them.

- ’Classification’: Storing the results and configurations of machine learning classification tasks carried out on datasets inside the application depends mostly on the Classification model. It is connected to the Dataset model to guarantee that a particular dataset corresponds with every classification job. Including the ’target_column’ field, which marks the column used as the target variable in the dataset, the model gathers basic knowledge on the classification process. Furthermore storing ’sampler’ and ’model’ fields, respectively, it details the sampling technique and machine learning model used. Both the sampler’s and the model’s parameters are kept in JSON format, so allowing flexible configuration storage. The model also records several evaluation criteria including accuracy, precision, recall, F1 score, ROC AUC, geometric mean, and Matthews correlation coefficient. Crucially for evaluating and contrasting the efficiency of several approaches to handle imbalanced data, the ’Classification’ model aims to offer a complete record of the classification experiments carried out together with the methods, parameters, and performance measures.

7.3.2 Views

The Views are responsible for handling the business logic of the application. Every view function in Django answers web requests, handles them, interacts with the model as needed. Views were applied in this project to control operations including data preparation, imbalance detection, file uploads, and application of balancing strategies. The views line up with the models and templates to guarantee that the right data is produced in the right way. Here is a breakdown of the views we used in our code:

- ’ClassifyWizard’: For machine learning classification chores, this perspective enables a multi-step form process. It helps consumers choose a dataset, define target columns, pick a sampling technique, and set machine learning models. Every step gathers particular user inputs and, once finished, treats them to produce a preview of the classification configuration. This wizard-based approach guarantees a methodical and understandable way to set up intricate classification systems.

- ’update_steps’: This perspective addresses the asynchronous processing of several steps required in training and preparation of a machine learning model. It handles chores including loading datasets, text data cleaning and vectorizing, data split into training and testing sets, application of sampling methods, training models, and performance evaluation of them. It gives the user thorough comments on the completion and outcomes of the process as well as updates on the state of every stage.

- ’training’: Capturing parameters related to datasets, target columns, samplers, and models, the ’training’ view sets up the first conditions for a machine learning training session. It provides a framework for the training process and generates a template enabling users to run their models depending on the given setups.

- ’home’: Acting as the application’s landing page, this perspective offers a broad introduction and navigational center allowing users to access several aspects of the website. Its design is to greet users and provide simple access to other application features.

- ’login_view’: This view controls user authentication by addressing login inquiries. It authenticates user credentials, handles login forms, and either logs in the user or shows an error notice should authentication prove difficult. Good logons send users to the home page.

- ’logout_view’: This view handles user logouts by ending the current user session and redirecting users to the home page. It ensures that users can securely log out of their accounts.

- ’register’: This view lets fresh users register on the network. It tries to create a new user, validates user input, and guarantees password confirmation, so managing the registration process. Should registration go through, the user logs in and finds their way to the home page.

- ’analyze’: Dataset uploading and analysis are handled by this view. It lets users view data analytics using predefined graphs and upload CSV files. Depending on user input, the view also pulls current datasets from the database for study.

- ’results’: The view shows analytics or past computed results. It allows consumers to view the result of data analysis or model training by retrieving and displaying information kept in their session.

- ’my_datasets’: This view gives users a list of datasets they uploaded. It handles file uploads and saves newly created datasets to the database, so supporting viewing and uploading of fresh datasets. Users view and control their datasets as needed.

- ’view_dataset’: This view enables users to view the contents of a specific dataset. It paginates the dataset’s records for easier navigation and displays dataset headers and data, allowing users to explore and analyze the data.

- ’delete_dataset’: This view handles the deletion of a specific dataset from the database. It ensures that users can remove datasets they no longer need, maintaining a clean and organized dataset list.

- ’my_classifications’: This view shows a list of classification results connected to user-owned datasets. It lets users go over and control their classification history by giving a summary of past classification activities and outcomes.

- ’view_classification’: This view lets users view comprehensive data about a given classification outcome. It allows users to examine and understand the results by presenting measures and criteria connected to the classification task.

- ’delete_classification’: This view handles the deletion of a specific classification result from the database. It provides users with the ability to remove unwanted classification results and keep their records up to date.
- ’contact’: This view lets users access a contact form or contact page so they may get in touch with the website managers or support staff. It provides a means of user inquiry or feedback submission.

7.3.3 Templates

HTML for the user interface is produced using templates. With placeholders for data the view will load, Django’s template engine lets HTML be dynamically generated. Designed to offer an easy-to-use interface for uploading datasets, viewing data analytics, and dealing with the data balancing tools, the templates in this application Following the MVT design guarantees that the presenting layer is kept apart from the business logic.

7.3.4 URL Routing

The URL routing system of Django links URLs to related view methods. The URL patterns and their corresponding views were defined from a URL configuration file (urls.py). Clean and understandable URL structures made possible by this modular approach to URL routing improve user experience and increase navigability of the application.

7.4 Key Features and Functionality

This section explains every aspect of our website ,it’s key features and attributes in detail.

7.4.1 User Authentication and Account Management

Strong user authentication and account management tools included into our web application guarantee a safe and customized experience. Users may register using a distinct username and password, which are safely kept under encryption to guard private data and also keep the processed datasets history. This simple account building method guarantees that every user has a unique and safe identity inside the system and calls for little information. Users can sign in once registered to see their customized dashboard, where they may upload datasets, view past studies and interact with the other tools of the service. By limiting access to authorized users, the sign-in feature not only improves security but also helps the program to preserve unique user sessions, therefore enabling a more personalized experience for any use. Every user has a profile where their data history is kept so they may quickly review and control the previously submitted and examined datasets, allowing users to trace the development of their data analyses over time.

7.4.2 Data Upload

The main format supported for data processing is CSV files, so the application offers an easy file uploading interface. Clear instructions and error handling systems in place to direct users in case of erroneous file formats or other problems help this procedure to be user-friendly. The program starts a preparation procedure automatically after a file is submitted to ready the data for additional study. Important chores such identifying and managing missing values, changing data types as needed, and guaranteeing correct data structure for the next analysis phases comprise this preprocessing. Automating these preprocessing tasks guarantees that the data is in an ideal condition for statistical analysis and imbalance identification, therefore lowering the possible user error. This function not only streamlines user workflow but also improves the correctness and dependability of the analysis carried out inside the program.

7.4.3 Statistical Analysis Results

After uploading a dataset users can quickly access a variety of analytical tools (”Classification”) that automatically compute and show the several statistical measures mentioned on Chapter 3, following upload. This ability lets users rapidly evaluate the features of their data, including its distribution and central patterns, therefore laying a strong basis for well-informed decisions.

7.4.4 Data Balancing

The web application features a powerful ”Classification” tool that centralizes both imbalance detection and data balancing operations, streamlining the process for users. This application automatically examines the provided dataset to find any class imbalances. When users find imbalance, the tool applies the different data balancing methods discussed in Chapter 6. The ”Classification” tool streamlines what may otherwise be a complicated and time-consuming procedure by combining these features into a single, user-friendly interface. With only a few clicks, users may rapidly create balanced datasets more suited for training accurate and dependable machine learning models. This flawless integration of imbalance identification and correction not only improves the application’s usability but also guarantees that users may boldly prepare their data for further analysis or model training.

The next chapter of this report, contains a detailed tutorial of this website.

Chapter 8

Web Application Tutorial

This chapter provides a comprehensive tutorial designed to guide users through the essential features and functionalities of our web application.Starting with the fundamentals ,creating an account and logging in. Then the course moves to more complex processes including uploading datasets, running statistical studies, identifying data imbalances, and using the Classification tool. The chapter also addresses how to handle and make use of the user's past dataset within the program in order to monitor and go back over time with analytical development.

8.1 Getting Started

8.1.1 Installation

Proper setup of your development environment is crucial for starting to use the web application created in this project, using VS CODE for IDE. Installation of several libraries and packages is vital to guarantee that the application operates as expected in all respects. Correct installation of these libraries should provide a seamless running and testing experience locally on your machine.
NOTE: The whole project's folder should be opened on the IDE.
The following libraries, whose attributes and operation are mentioned in Chapter 5 should be installed:

1. 'unidecode' : pip install unidecode

2. ’sklearn’ : pip install scikit-learn

3. ’imblearn’ : pip install imbalanced-learn

4. ’nltk’ : pip install nltk

Also the following Django-tool-libraries should be installed:

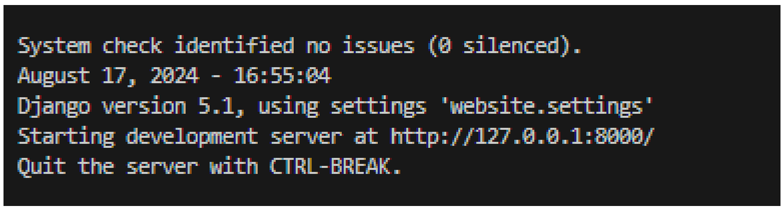
1. ’django’ : pip install django

2. ’crispyforms’ : pip install django-crispy-forms

3. ’formtools’ : pip install django-formtools

8.1.2 Accessing the Website

After the successful installation of the libraries, the user should run the following command on terminal in order to start the session with the server: ’python manage.py runserver’. Then, the url is generated in the terminal:



```
System check identified no issues (0 silenced).
August 17, 2024 - 16:55:04
Django version 5.1, using settings 'website.settings'
Starting development server at http://127.0.0.1:8000/
Quit the server with CTRL-BREAK.
```

Figure 8.1: URL Generation.

The user should then ctrl+click on the url generated as shown in picture 8.1.The website should now open on default browser.
After the web page opens, the user can sign in to an already registered account. If the user do not have an existing account, he must register as shown in 8.2.

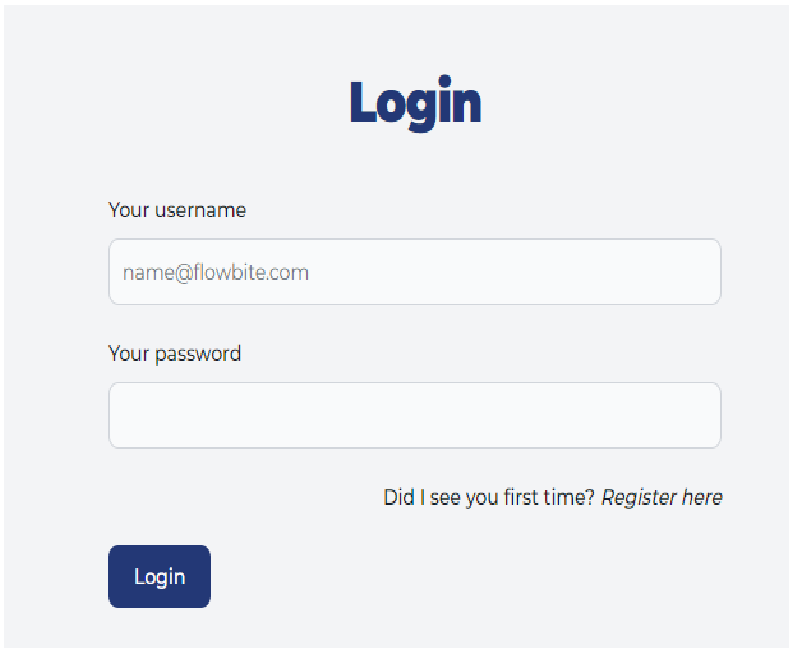


Figure 8.2: Login/Register.

If the user wishes to register he has to complete the form shown in picture 8.3

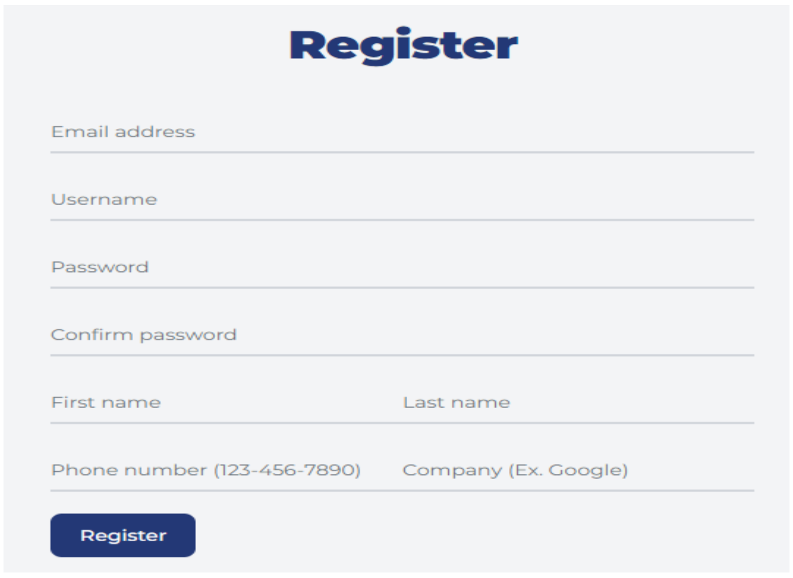


Figure 8.3: Register Form.

8.2 Menu

Once logged in the user has access to the website's menu and tools shown in picture 8.4.

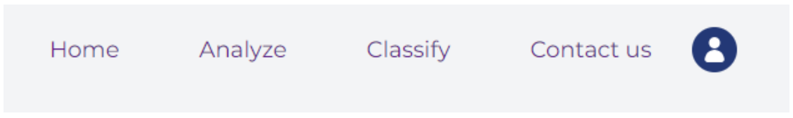


Figure 8.4: Menu/Tools.

8.2.1 Analyze

First operation available from the menu of the website is the Analyze part, which offers users a simple interface to upload their CSV files for investigation. Once a dataset is entered, the system automatically examines the data to find and evaluate any imbalances.

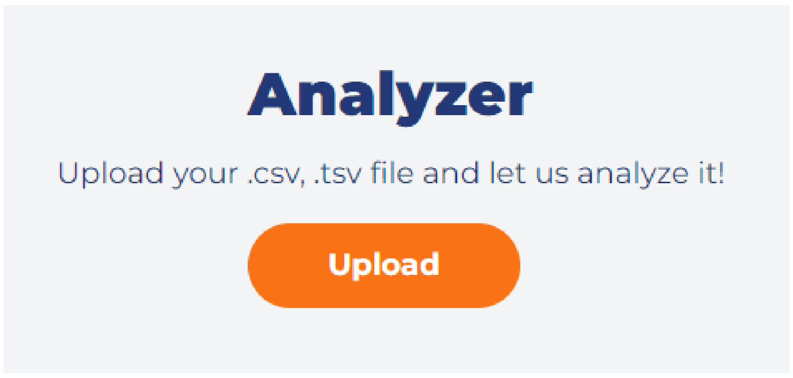


Figure 8.5: Analyze Section

Analyze also includes a visualization graph,shown in picture 8.6, depicting the label distribution within the dataset. Users would especially benefit from this graph, which visually highlights any variations between the classes and helps them rapidly grasp the balance or imbalance of their data. Users who must grasp the distribution of their data before moving on with more research or machine learning assignments depend on this instantaneous feedback—which combines numerical and graphical insights. The user-friendly design guarantees that the structure of their dataset can be quickly evaluated and interpreted even by people with limited technological background.

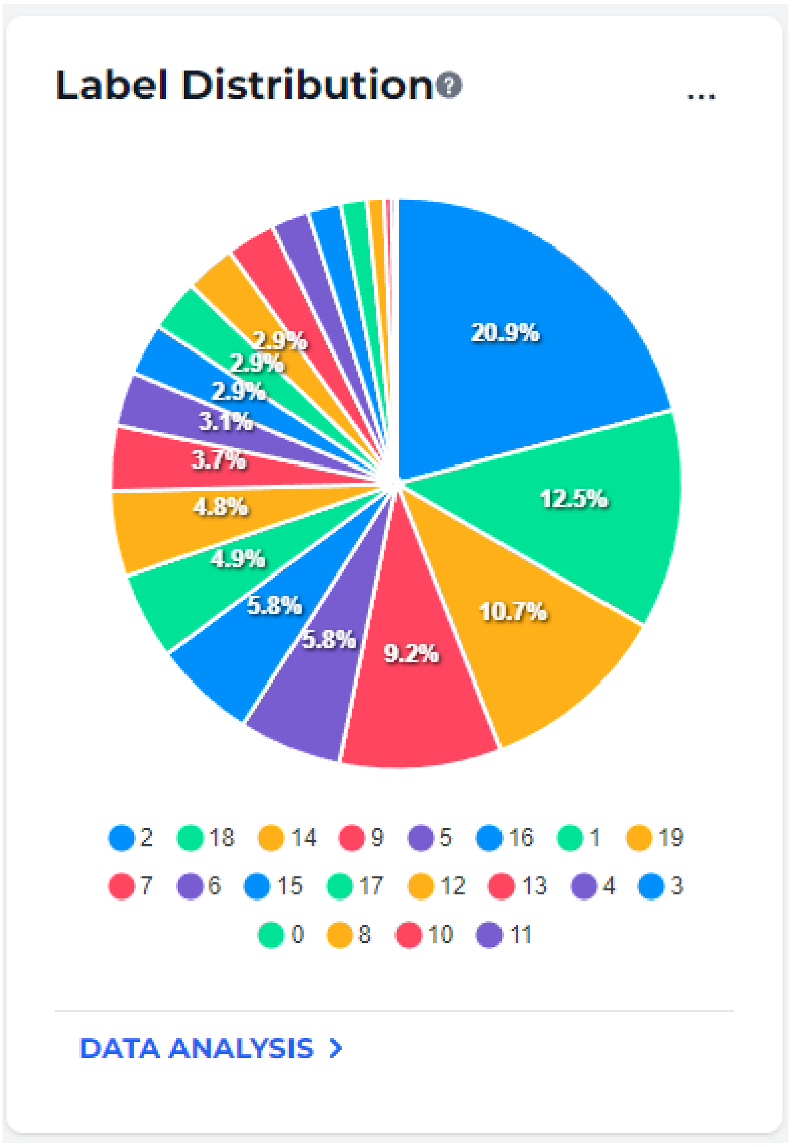
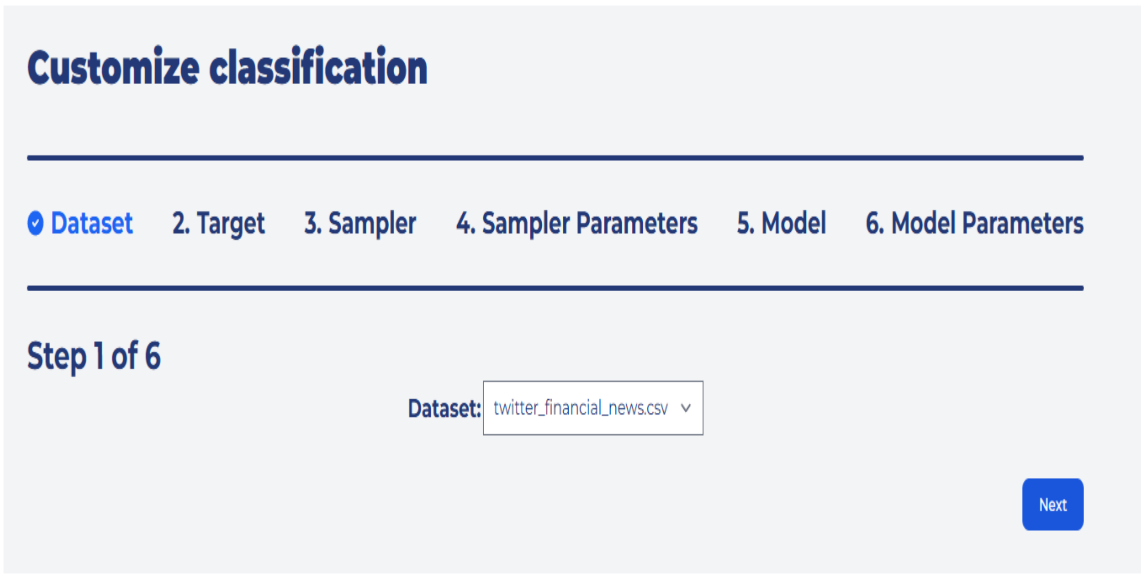


Figure 8.6: Label Distribution Graph

8.2.2 Classify

Core to the website, the Classification Tool enables users to directly apply machine learning models to their datasets over the web interface. Users may choose this tool to apply several classification algorithms meant to solve data imbalance following the data upload and analysis. The tool provides a variety of choices so that users may evaluate several models and methods to identify the best fit for their particular dataset. The outcomes include performance indicators including accuracy, precision, recall, and more, so guiding consumers toward the model most appropriate for their circumstances. Data scientists and analysts wishing to rapidly test and implement machine learning models without requiring extensive coding will find this part particularly helpful.

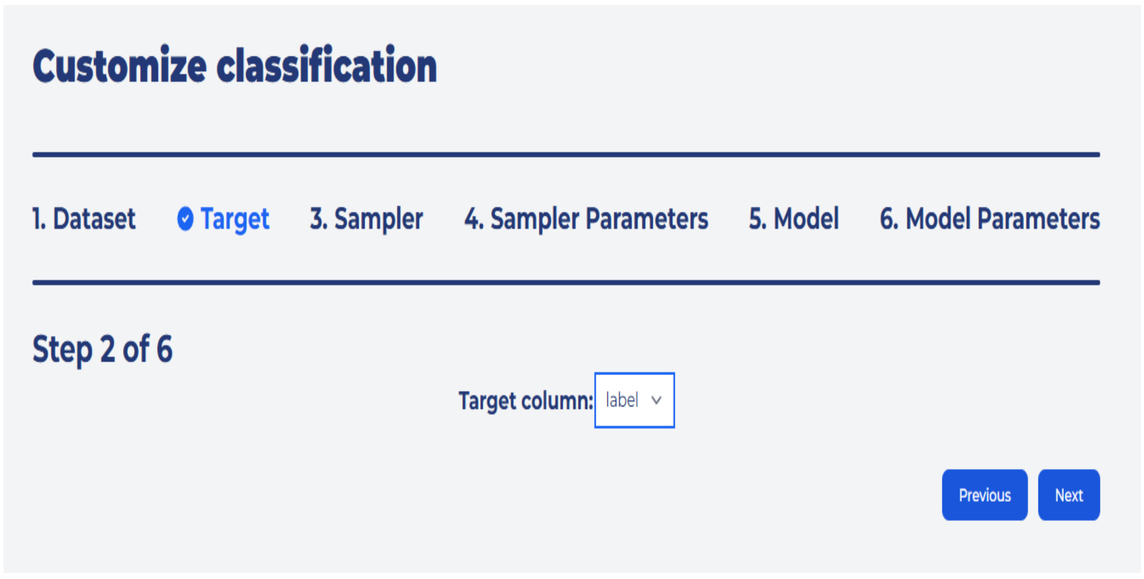
The classification tool consists of 6 steps.The first one is choosing the dataset for classification:



Dataset: twitter_financial_news.csv

Figure 8.7: Classify-Step 1

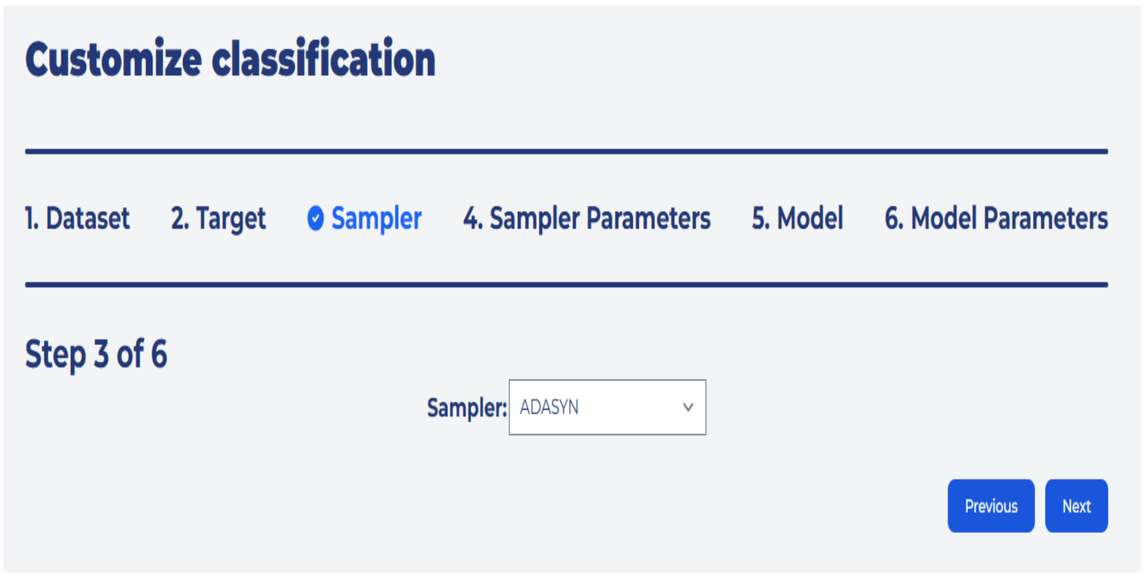
In the next step, the user should choose the target column:



Target column: label

Figure 8.8: Classify-Step 2

In Step 3, the user picks the sampler he wants to test (e.g. ADASYN as shown in picture 8.9):


Customize classification

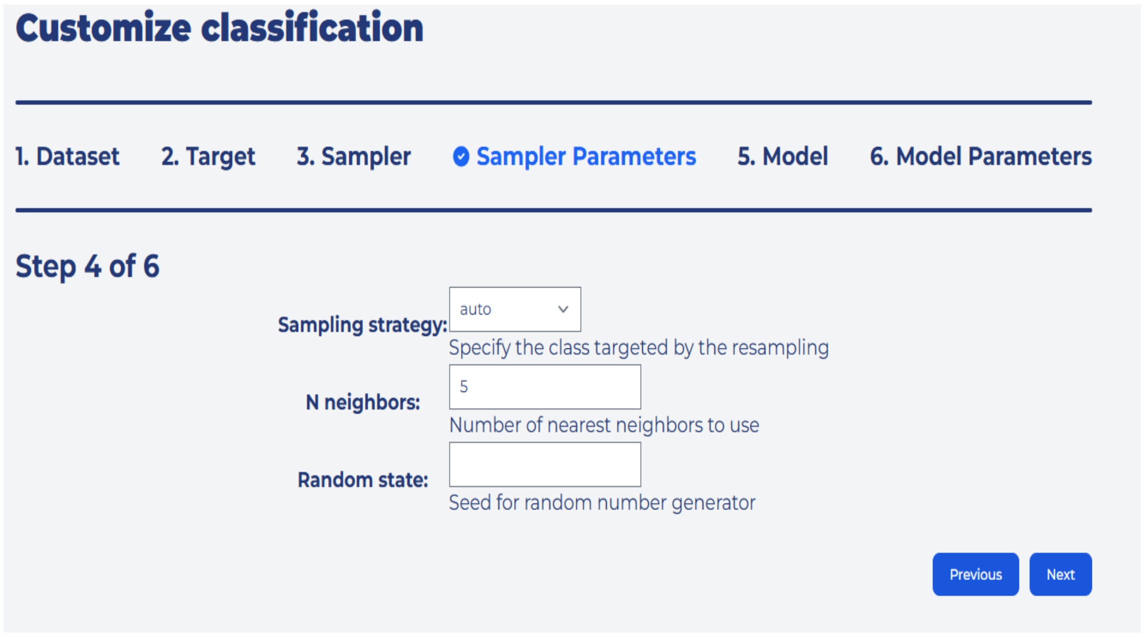
1. Dataset 2. Target ✓ Sampler 4. Sampler Parameters 5. Model 6. Model Parameters

Step 3 of 6

Sampler: ADASYN

Figure 8.9: Classify-Step 3

Step 4 is about fine-tuning, how the dataset will be adjusted to handle imbalances before the classification model is trained. By setting these parameters, the user can influence the effectiveness of the resampling process and, ultimately, the performance of the classification model. As show in picture 8.10 the user can set the sampling strategy, the random state seed and the number of nearest neighbors.


Customize classification

1. Dataset 2. Target 3. Sampler ✓ Sampler Parameters 5. Model 6. Model Parameters

Step 4 of 6

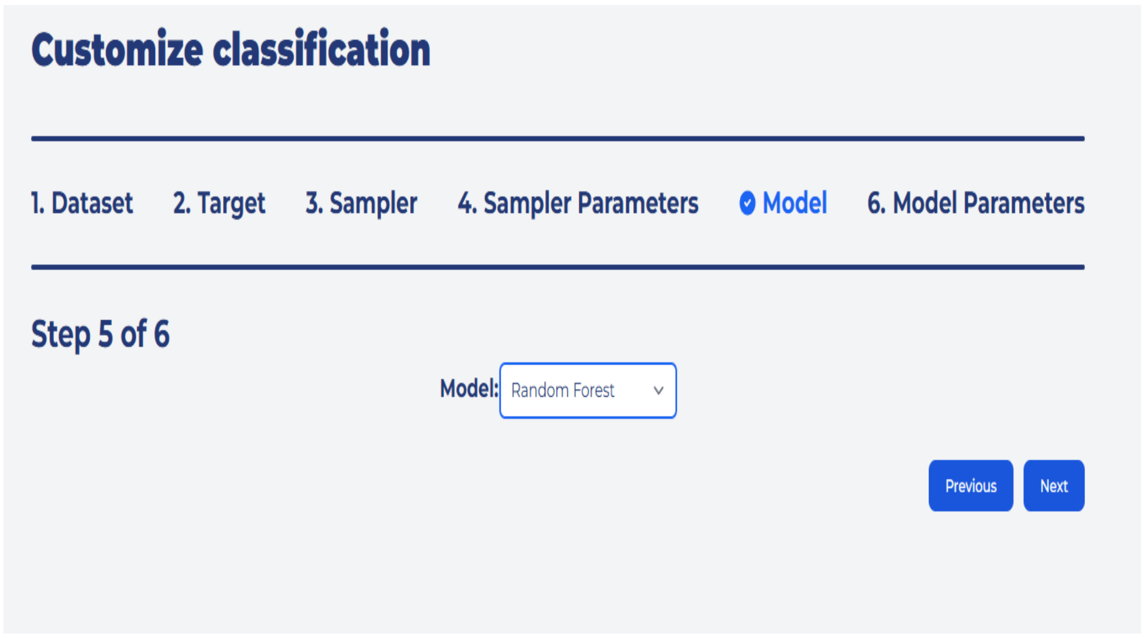
Sampling strategy: auto
Specify the class targeted by the resampling

N neighbors: 5
Number of nearest neighbors to use

Random state:
Seed for random number generator

Figure 8.10: Classify-Step 4

In Step 5 the user can select which model/classifier he wants to use. As shown in picture 8.11 the chosen model is Random Forest:



Model: Random Forest

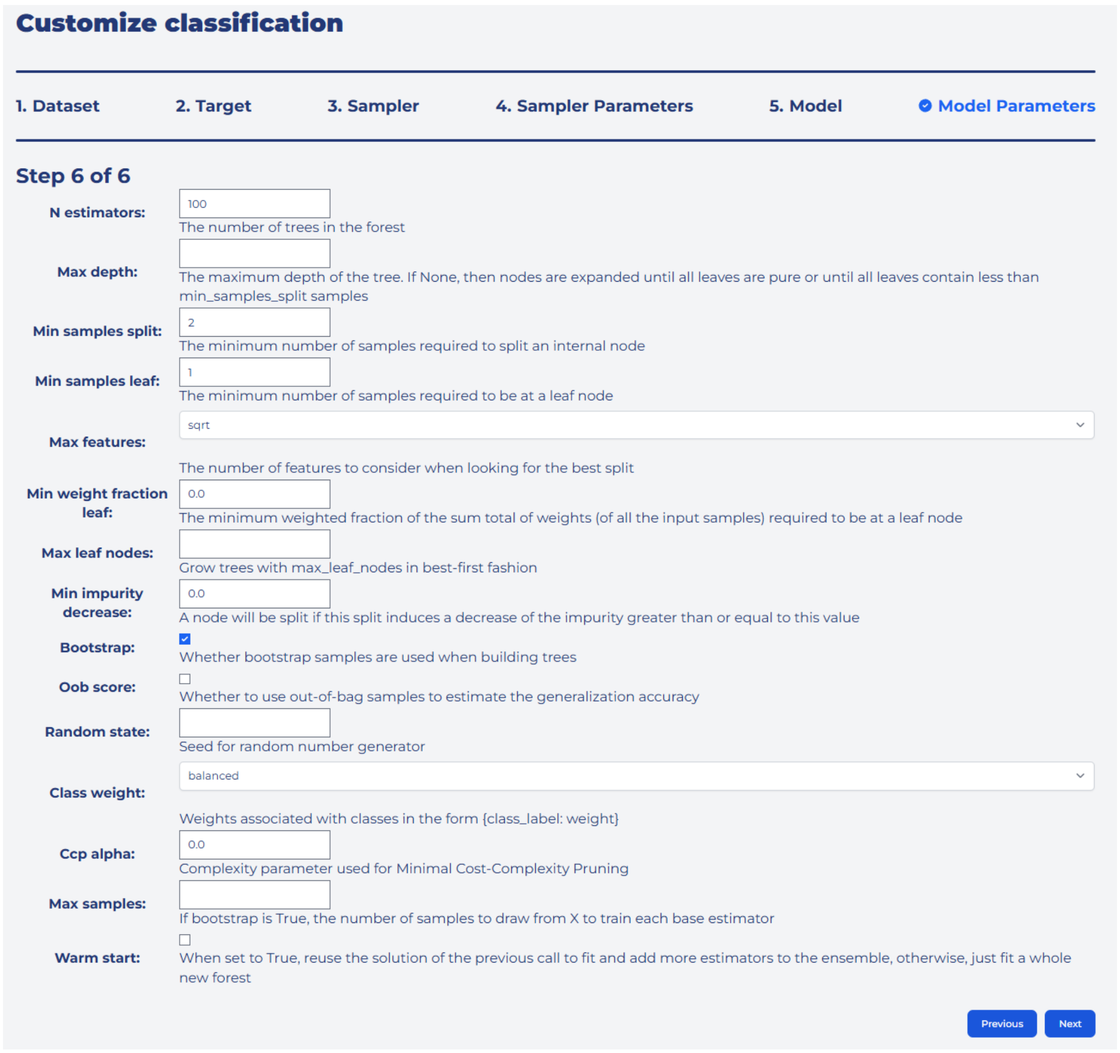
Figure 8.11: Classify-Step 5

Finally, Step 6 is meant to empower consumers over the internal settings of their preferred classification model. Users can affect how the model learns from the data by varying these settings, so affecting the accuracy, precision, recall, and other performance measures. The following key aspects can be adjusted for each model:

- Model-Specific Parameters.

- Customization and Optimization.

In picture 8.12 Random Forest is chosen as example selected model.



Customize classification

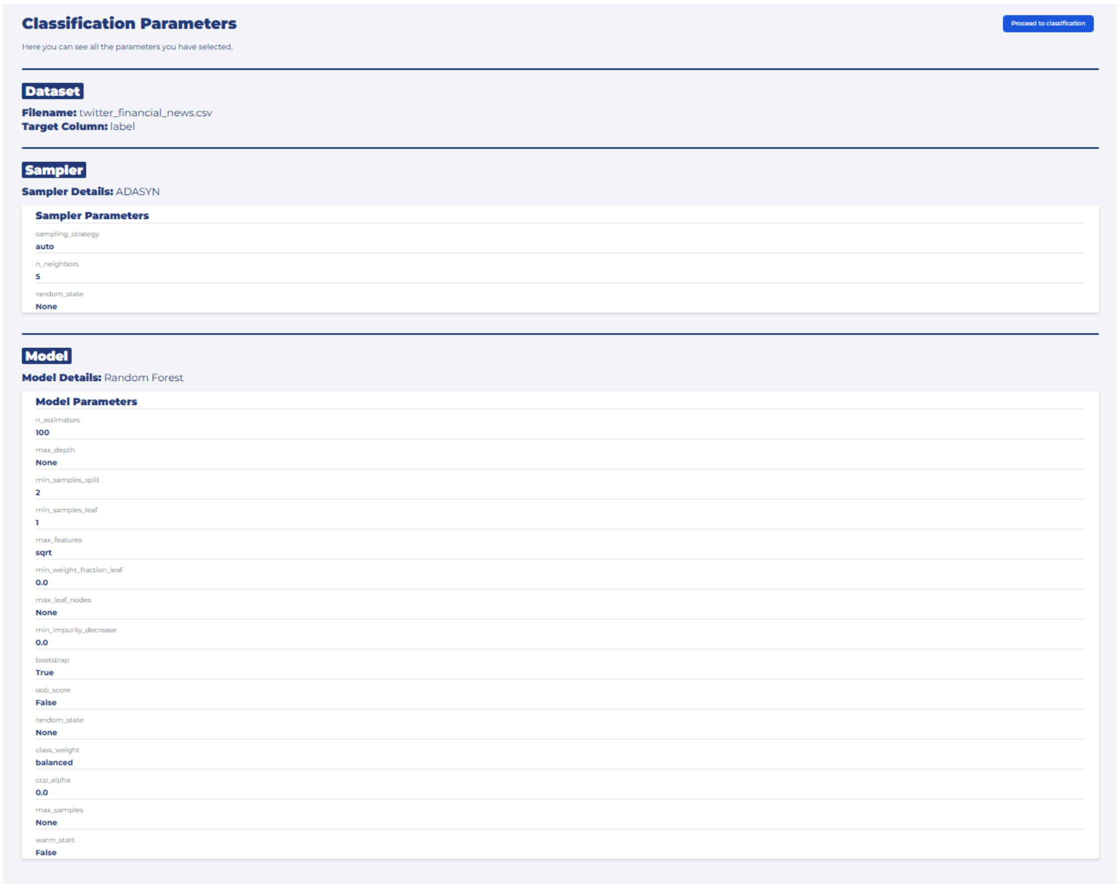
1. Dataset | 2. Target | 3. Sampler | 4. Sampler Parameters | 5. Model | Model Parameters

Step 6 of 6

N estimators:	100 The number of trees in the forest
Max depth:	 The maximum depth of the tree. If None, then nodes are expanded until all leaves are pure or until all leaves contain less than min_samples_split samples
Min samples split:	2 The minimum number of samples required to split an internal node
Min samples leaf:	1 The minimum number of samples required to be at a leaf node
Max features:	sqrt The number of features to consider when looking for the best split
Min weight fraction leaf:	0.0 The minimum weighted fraction of the sum total of weights (of all the input samples) required to be at a leaf node
Max leaf nodes:	 Grow trees with max_leaf_nodes in best-first fashion
Min impurity decrease:	0.0 A node will be split if this split induces a decrease of the impurity greater than or equal to this value
Bootstrap:	☑ Whether bootstrap samples are used when building trees
Oob score:	☐ Whether to use out-of-bag samples to estimate the generalization accuracy
Random state:	 Seed for random number generator
Class weight:	balanced Weights associated with classes in the form {class_label: weight}
Ccp alpha:	0.0 Complexity parameter used for Minimal Cost-Complexity Pruning
Max samples:	 If bootstrap is True, the number of samples to draw from X to train each base estimator
Warm start:	☐ When set to True, reuse the solution of the previous call to fit and add more estimators to the ensemble, otherwise, just fit a whole new forest

Figure 8.12: Classify-Step 6 - Random Forest

After classification customization is completed, an overview of all the parameters and model selected from the user is presented as shown in pic 8.13.



Classification Parameters

Here you can see all the parameters you have selected.

Dataset

Filename: twitter_financial_news.csv
Target Column: label

Sampler

Sampler Details: ADASYN

Sampler Parameters

sampling_strategy
auto

n_neighbors
5

random_state
None

Model

Model Details: Random Forest

Model Parameters

n_estimators
100

max_depth
None

min_samples_split
2

min_samples_leaf
1

max_features
sqrt

min_weight_fraction_leaf
0.0

max_leaf_nodes
None

min_impurity_decrease
0.0

bootstrap
True

oob_score
False

random_state
None

class_weight
balanced

ccp_alpha
0.0

max_samples
None

warm_start
False

Figure 8.13: Classify- Parameter Overview

Once the user has verified his choices, all he has to do is click the button ”Proceed to classification” and wait for the ”Classification Sequence” window to finish processing 8.14

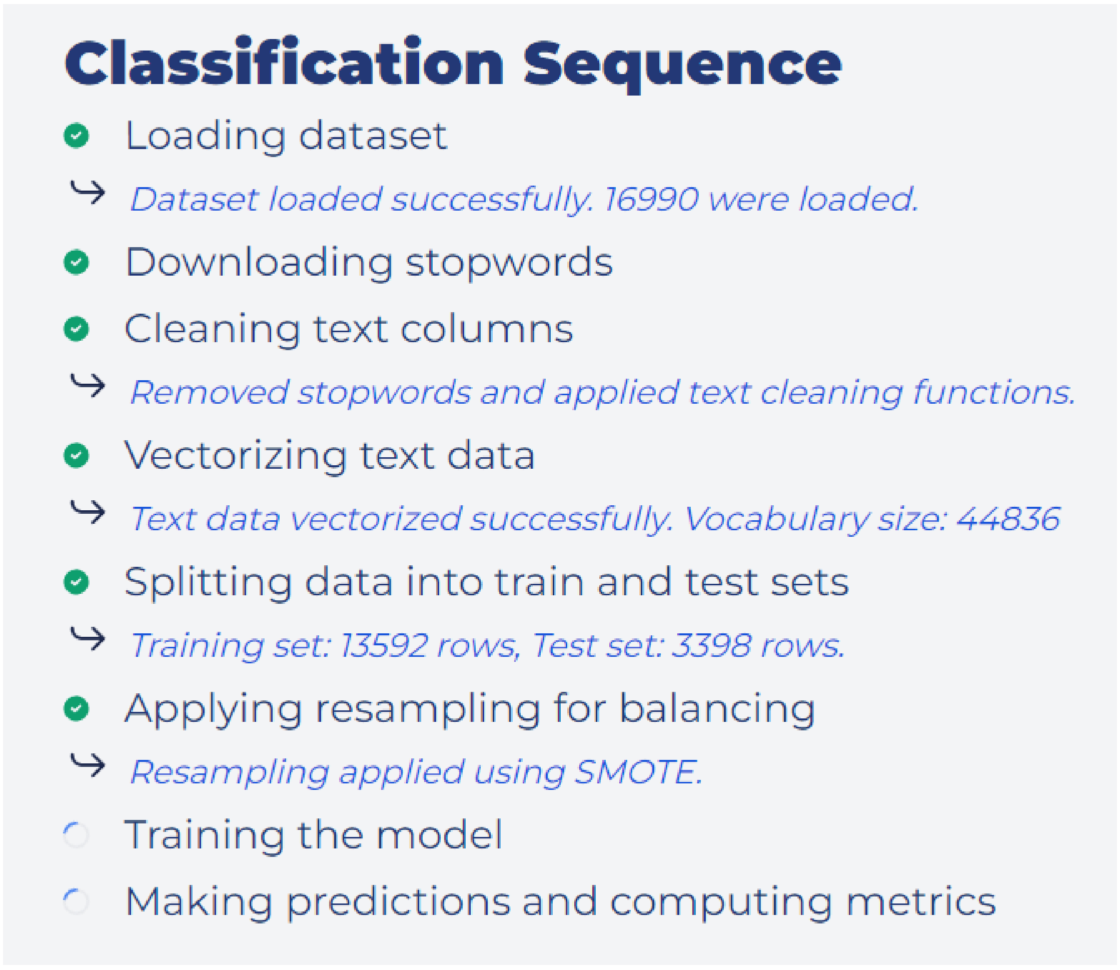
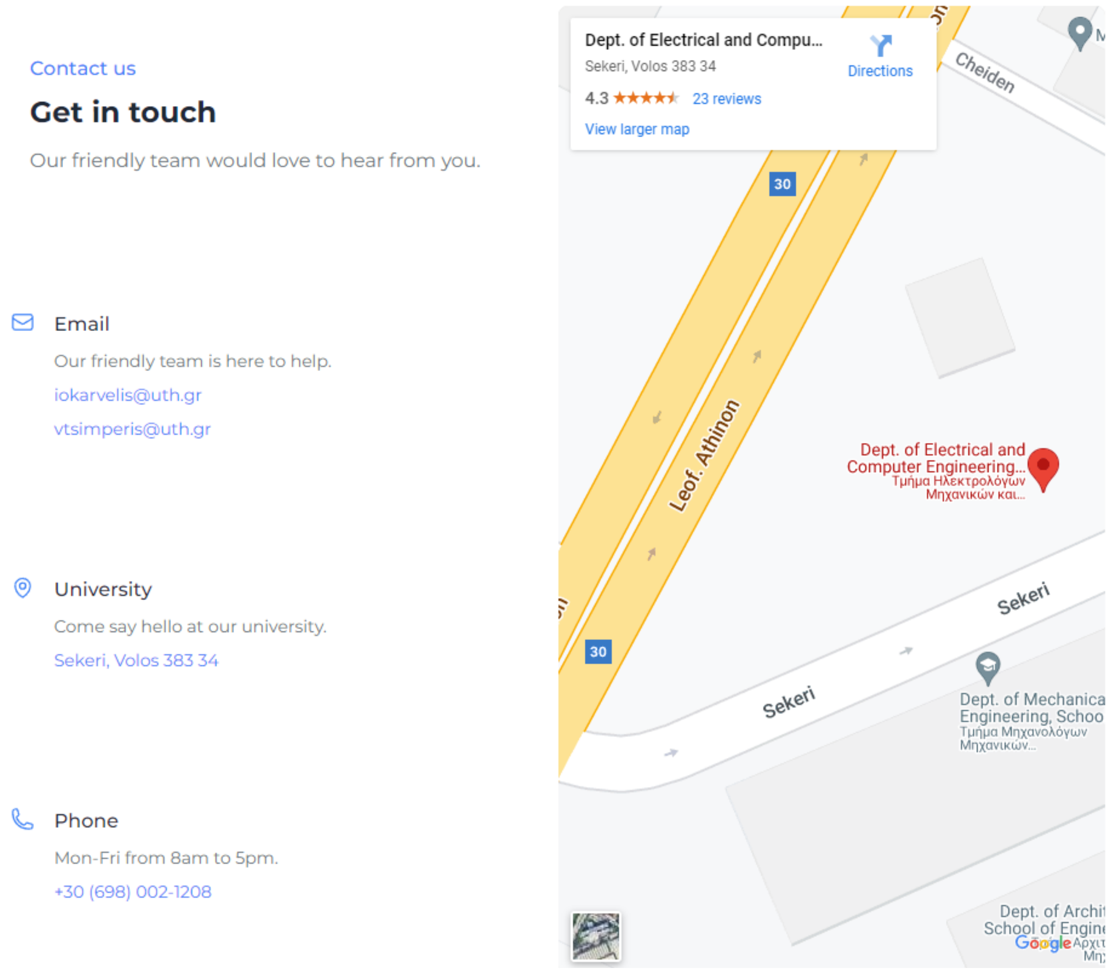

Classification Sequence
- Loading dataset
  ↳ Dataset loaded successfully. 16990 were loaded.
- Downloading stopwords
- Cleaning text columns
  ↳ Removed stopwords and applied text cleaning functions.
- Vectorizing text data
  ↳ Text data vectorized successfully. Vocabulary size: 44836
- Splitting data into train and test sets
  ↳ Training set: 13592 rows, Test set: 3398 rows.
- Applying resampling for balancing
  ↳ Resampling applied using SMOTE.
- Training the model
- Making predictions and computing metrics

Figure 8.14: ”Classification Sequence” window

8.2.3 Contact us

The Contact Us section provides users with a direct line of communication with the developers or support team. Dealing with any questions or problems users could run across on the website depends on this part. The Contact Us part provides a simple approach to get in touch whether users want technical help, have questions about particular features, or wish to provide comments.


Contact us

Get in touch

Our friendly team would love to hear from you.

Email

Our friendly team is here to help.

iokarvelis@uth.gr

vtsimperis@uth.gr

University

Come say hello at our university.

Sekeri, Volos 383 34

Phone

Mon-Fri from 8am to 5pm.

+30 (698) 002-1208

Dept. of Electrical and Compu...
Sekeri, Volos 383 34
Directions
4.3 ★★★★★ 23 reviews
View larger map
Cheiden
30
Leof. Athinon
Dept. of Electrical and Computer Engineering...
Τμήμα Ηλεκτρολόγων Μηχανικών και...
Sekeri
Sekeri
Dept. of Mechanica Engineering, Schoo
Τμήμα Μηχανολόγων Μηχανικών...
Dept. of Archi School of Engin
Google

Figure 8.15: Contact Us

8.2.4 Account Management

Users can access their account management field by clicking on the blue circle icon shown at the top right corner of the web application 8.16. Customizing the user experience and tracking platform activity depend on this part, which also helps to maintain platform integrity. Once clicked, the user’s account information—name and email address among other things—along with several choices for controlling their activity on the site are displayed.

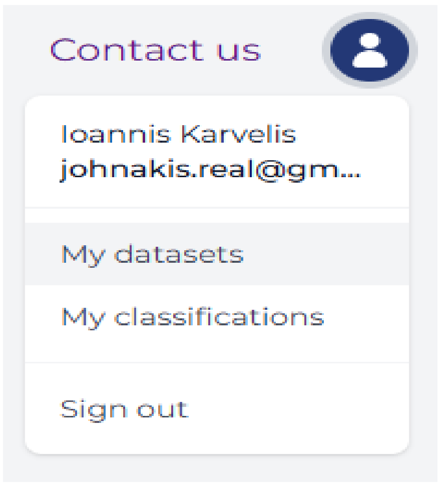
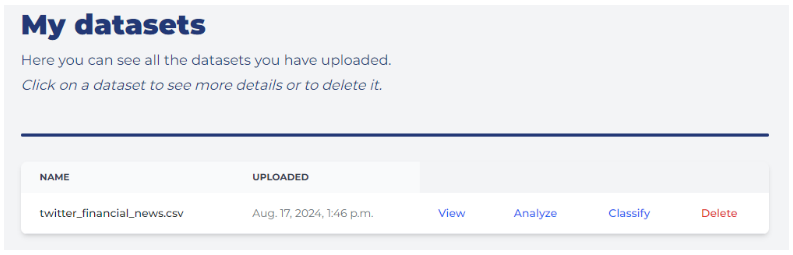


Figure 8.16: Account Management Menu

The following features are available for the user:

- My Datasets: View and manage the datasets you have uploaded. Easy access to all previously examined datasets made possible in this part lets users review or keep working with any dataset at any moment.



My datasets

Here you can see all the datasets you have uploaded.

Click on a dataset to see more details or to delete it.

NAME	UPLOADED				
twitter_financial_news.csv	Aug. 17, 2024, 1:46 p.m.	View	Analyze	Classify	Delete

Figure 8.17: My Datasets page

- Sign out button.

Chapter 9

Summary

This thesis addressed a critical challenge in the field of machine learning: the problem of imbalanced data in multi-label classification. Multi-label classification, where each instance can belong to multiple categories simultaneously, is inherently complex. When combined with imbalanced data, where some classes are significantly underrepresented, it presents a formidable challenge for traditional machine learning algorithms.

Our research methodology encompassed several key areas:

1. **Data Preparation:** We implemented rigorous text cleaning procedures, which involved removing noise, standardizing formats, and handling special characters. This step was crucial in ensuring the quality of input data.

2. **Vectorization:** We explored various text vectorization techniques, including TF-IDF (Term Frequency-Inverse Document Frequency) and word embeddings. These methods transform text into numerical vectors, enabling machine learning models to process the data effectively.

3. **Sampling Strategies:** We focused on ADASYN (Adaptive Synthetic) sampling, a technique designed to address class imbalance by generating synthetic samples for minority classes. ADASYN adaptively creates samples based on the difficulty of learning each minority instance, providing a more nuanced approach than simpler oversampling methods.

4. **Machine Learning Models:** We evaluated several classifiers, with particular emphasis on Random Forest (RF) and Logistic Regression (LR).

The Random Forest classifier, an ensemble method that combines multiple decision trees, demonstrated exceptional performance when coupled with ADASYN. Its ability to handle high-dimensional data and capture complex interactions between features made it particularly suited for our multi-label, imbalanced dataset. The RF-ADASYN combination excelled in:

- Balancing accuracy across both majority and minority classes
- Reducing overfitting through its ensemble nature
- Providing robust performance across various evaluation metrics

Logistic Regression, despite its simplicity, also showed significant improvements when used with ADASYN. Key benefits of the LR-ADASYN combination included:

- Improved handling of rare classes, addressing the imbalance issue effectively
- Maintained interpretability, allowing for easier understanding of feature importance
- Computational efficiency, making it suitable for large-scale applications

A major contribution of this thesis was the development of an interactive website. This platform serves multiple purposes:

- It allows users to upload their own imbalanced multi-label datasets
- Users can experiment with different preprocessing techniques, sampling methods, and machine learning models
- The website provides visualizations of results, making it easier to interpret model performance
- It bridges the gap between advanced machine learning techniques and practical applications, making these methods accessible to a wider audience

The website's user-friendly interface is designed to cater to both researchers and practitioners, potentially accelerating the adoption of these advanced techniques in real-world scenarios.

9.1 Future extensions

Building on the foundations laid by this research, several promising avenues for future work emerge:

9.1.1 Advanced Deep Learning Architectures

Exploring more sophisticated neural network architectures could potentially yield even better results for imbalanced multi-label classification. Specifically:

- Transformer models, known for their ability to capture long-range dependencies in data, could be adapted for multi-label classification tasks.
- Attention mechanisms could be incorporated into existing neural network architectures to help models focus on the most relevant features for each label.
- Graph Neural Networks (GNNs) could be explored to leverage the inherent relationships between labels in multi-label scenarios.

9.1.2 Expanded Dataset Diversity

Applying our methods to a wider range of datasets from various domains (e.g., medical diagnosis, image classification, product categorization) would:

- Validate the generalizability of our findings
- Potentially uncover domain-specific challenges and opportunities
- Lead to the development of more robust and versatile models

9.1.3 Website Enhancements

The current website can be expanded in several ways to increase its utility:

- Implement distributed computing capabilities to handle larger datasets and more complex models
- Add more preprocessing options and sampling techniques
- Incorporate model interpretability tools to help users understand model decisions
- Develop collaborative features, allowing multiple users to work on projects simultaneously and share insights

9.1.4 Advanced Ensemble Methods

Building upon the success of Random Forest, future work could explore more sophisticated ensemble techniques:

- Investigate boosting methods like XGBoost or LightGBM in the context of imbalanced multi-label classification
- Develop custom ensemble methods that are specifically designed for multi-label, imbalanced data scenarios

9.1.5 Theoretical Analysis

While our work demonstrated empirical success, a deeper theoretical understanding could drive further improvements:

- Analyze the mathematical properties of ADASYN in multi-label contexts
- Develop theoretical bounds on the performance of different model-sampling combinations
- Investigate the impact of data characteristics (e.g., label correlations, feature distributions) on model performance

9.1.6 Real-time Learning and Adaptation

Extend the current framework to handle streaming data and concept drift:

- Develop online learning algorithms that can adapt to changing data distributions in real-time
- Implement active learning strategies to intelligently select the most informative instances for labeling in a multi-label context

9.1.7 Fairness and Bias Mitigation

As imbalanced datasets often reflect real-world inequalities, it’s crucial to ensure our models don’t perpetuate or amplify biases:

- Investigate the impact of sampling techniques on model fairness across different demographic groups
- Develop multi-label specific fairness metrics and constraints
- Explore methods to balance performance improvements with fairness considerations

By pursuing these directions, future research can build upon the foundations laid in this thesis, further advancing the field of imbalanced multi-label classification and expanding its practical applications across various domains.

Bibliography

[1] Haibo He and Edwardo A Garcia. Learning from imbalanced data. *IEEE Transactions on Knowledge and Data Engineering*, 21(9):1263–1284, 2009.

[2] Nitesh V Chawla, Kevin W Bowyer, Lawrence O Hall, and W Philip Kegelmeyer. Smote: synthetic minority over-sampling technique. *Journal of Artificial Intelligence Research*, 16:321–357, 2002.

[3] Maciej A Mazurowski, Piotr A Habas, Jacek M Zurada, Joseph Y Lo, Jay A Baker, and Georgia D Tourassi. Training neural network classifiers for medical decision making: The effects of imbalanced datasets on classification performance. *Neural networks*, 21(2-3):427–436, 2008.

[4] Clifton Phua, Damminda Alahakoon, and Vincent Lee. Minority report in fraud detection: classification of skewed data. In *ACM SIGKDD Explorations Newsletter*, volume 6, pages 50–59. ACM, 2004.

[5] Jiahong Pang, Yong Huang, Zuyao Xie, Jianpo Li, and Zhiping Cai. Deep learning approaches for data anomaly detection and classification in environmental monitoring networks. *Science of The Total Environment*, 661:546–555, 2019.

[6] Grigorios Tsoumakas and Ioannis Katakis. Multi-label classification: An overview. *International Journal of Data Warehousing and Mining (IJDWM)*, 3(3):1–13, 2007.

[7] Tom Fawcett. An introduction to roc analysis. *Pattern Recognition Letters*, 27(8):861–874, 2006.

[8] Leo Breiman. Random forests. *Machine Learning*, 45(1):5–32, 2001.

[9] Jerome H Friedman. Greedy function approximation: a gradient boosting machine. *Annals of Statistics*, pages 1189–1232, 2001.

[10] Haibo He, Yang Bai, Edwardo A Garcia, and Shutao Li. Adasyn: Adaptive synthetic sampling approach for imbalanced learning. In *2008 IEEE international joint conference on neural networks (IEEE world congress on computational intelligence)*, pages 1322–1328. IEEE, 2008.

[11] I˜naki Inza Jonathan Ortigosa-Hern´andez and Jose A. Lozano. Measuring the class-imbalance extent of multi-class problems. *PROCEDIA COMPUTER SCIENCE,23rd International Conference on KnowledgeBased and Intelligent Information Engineering Systems*, 2017.

[12] F. Zuo Z. Zhou X. Zhang, Y. Song and X. Wang. Towards imbalanced large scale multi-label classification with partially annotated labels. *IEEE/ACIS 21st International Conference on Software Engineering Research, Management and Applications (SERA), Orlando, FL, USA*, 2023.

[13] Ahmed M. Abdelkhalek and Maggie Ezzat Mashaly. Addressing the class imbalance problem in network intrusion detection systems using data resampling and deep learning. *The Journal of Supercomputing*, pages 1–34, 2023.

[14] Junlan Chen, Ziyuan Pu, Nan Zheng, Xiao Wen, Hongliang Ding, and Xiucheng Guo. A generative deep learning approach for crash severity modeling with imbalanced data, 2024.

[15] Jose G. Moreno-Torresa Francisco Herreraa Victoria Lópeza, Alberto Fernándezb. Expert systems with applications. *ELSEVIER*, 2011.

[16] Dragos D. Margineantu and Thomas G. Dietterich. Bootstrap methods for the cost-sensitive evaluation of classifiers. In *International Conference on Machine Learning*, 2000.

[17] Roweida Mohammed, Jumanah Rawashdeh, and Malak Abdullah. Machine learning with oversampling and undersampling techniques: Overview study and experimental results. In *2020 11th International Conference on Information and Communication Systems (ICICS)*, pages 243–248, 2020.

[18] Piboon Polvimoltham and Krung Sinapiromsaran. Mass ratio variance majority undersampling and minority oversampling technique for class imbalance. *Fuzzy Systems and Data Mining*, 2021.

[19] Zahed Siddique Manjurul Ahsan, Shivakumar Raman. Bsgan: A novel oversampling technique for imbalanced pattern recognitions. 2023.

[20] Priya S and Annie Uthra. Adaptive synthetic oversampling algorithm for handling class imbalance in multi-class data stream classification. *Journal of Computer Science*, 2022.

[21] Erianto Ongko Hartono. Hybrid approach with distance feature for multi-class imbalanced datasets. *INTERNATIONAL JOURNAL ON INFORMATICS VISUALIZATION*, pages 1383–1394, March. 2023.

[22] Joko Lianto Buliali1 Tora Fahrudin and Chastine Fatichah. Enhancing the performance of smote algorithm by using attribute weighting scheme and new selective sampling method for imbalanced data set. *ICIC International 2019 ISSN 1349-4198*, April. 2019.

[23] Muhammad Aksam Iftikhar Zulfiqar Habib Muhammad Arham Tariq, Allah Bux Sargano. Comparing different oversampling methods in predicting multi class educational datasets using machine learning techniques. *BULGARIAN ACADEMY OF SCIENCES,CYBERNETICS AND INFORMATION TECHNOLOGIES*, 2023.

[24] Eréndira Rendón, Roberto Alejo, Carlos Castorena, Frank J. Isidro-Ortega, and Everardo Efrén Granda-Gutiérrez. Data sampling methods to deal with the big data multiclass imbalance problem. *Applied Sciences*, 2020.

[25] Li Yijinga, Guo Haixianga, Li Xiaoa, Li Yanana, and Li Jinlinga. Adapted ensemble classification algorithm based on multiple classifier system and feature selection for classifying multi-class imbalanced data. *ELSEVIER*, 2016.

[26] Neelam Rout, Kuhoo, Debahuti Mishra, and Manas Kumar Mallick. Analysing the multi-class imbalanced datasets using boosting methods and relevant information. *International Journal of Pure and Applied Mathematics*, 2017.

[27] Xiaohui Yuan and Mohamed Abouelenien. A multi-class boosting method for learning from imbalanced data. *Int. J. Granul. Comput. Rough Sets Intell. Syst.*, 4:13–29, 2016.

[28] Negin Samadi Nazila Razzaghi Jafar Tanha, Yousef Abdi and Mohammad Asadpour. Boosting methods for multi□class imbalanced data classifcation: an experimental review. *Journal Of Big Data*, 2020.

[29] Yiwen Zhang Yanping Zhang XZhong Zheng, Yuanting Yan. Combating mutuality with difficulty factors in multi-class imbalanced data: A similarity-based hybrid sampling. *IEEE/ACIS 9th International Conference on Data Science and Advanced Analytics (DSAA)*, 2022.

[30] Swarna Kamlam Ravindran Carlo Tomasi Kelsey Lieberman, Shuai Yuan. Optimizing for roc curves on class-imbalanced data by training over a family of loss functions. *BULGARIAN ACADEMY OF SCIENCES,CYBERNETICS AND INFORMATION TECHNOLOGIES*, 2024.

[31] Christopher M Bishop. *Pattern Recognition and Machine Learning*. Springer, 2006.

[32] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. MIT Press, 2016.

[33] Brian W Matthews. Comparison of the predicted and observed secondary structure of t4 phage lysozyme. *Biochimica et Biophysica Acta (BBA)-Protein Structure*, 405(2):442–451, 1975.

[34] Miroslav Kubat and Stan Matwin. Addressing the curse of imbalanced training sets: one-sided selection. 97:179–186, 1997.

[35] Christopher D Manning, Prabhakar Raghavan, and Hinrich Schütze. *Introduction to Information Retrieval*. Cambridge University Press, 2008.

[36] Isabelle Guyon and André Elisseeff. An introduction to variable and feature selection. *Journal of Machine Learning Research*, 3:1157–1182, 2003.

[37] Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*, 2013.

[38] Jeffrey Pennington, Richard Socher, and Christopher D Manning. Glove: Global vectors for word representation. pages 1532–1543, 2014.

[39] Mihaela Elena Breaban Cristian Padurariu. Dealing with data imbalance in text classification. *PROCEDIA COMPUTER SCIENCE,23rd International Conference on KnowledgeBased and Intelligent Information Engineering Systems*, pages 737–739, March. 2019.

[40] Pandas documentation. `https://pandas.pydata.org/docs/index.html`.

[41] nltk documentation. `https://www.nltk.org/`.

[42] unidecode documentation. `https://docs.python.org/3/howto/unicode.html`.

[43] string documentation. `https://docs.python.org/3/library/string.html`.

[44] scikit-learn documentation. `https://scikit-learn.org/stable/`.

[45] imbalanced-learn documentation. `https://imbalanced-learn.org/stable/`.

[46] Xgb library documentation. `https://xgboost.readthedocs.io/en/stable/`.

[47] Nidhi Agarwal, Rakshit Srivastava, Pratishtha Srivastava, Jassi Sandhu, and Prajesh Pratap Singh. Multiclass classification of different glass types using random forest classifier. *2022 6th International Conference on Intelligent Computing and Control Systems (ICICCS)*, pages 1682–1689, 2022.

[48] Rama Jayapermana, Aradea Aradea, and Neng Ika Kurniati. Implementation of stacking ensemble classifier for multi-class classification of covid-19 vaccines topics on twitter. *Scientific Journal of Informatics*, 2022.

[49] Ricky Renaldo Arisandi, Sumarno Sumarno, and Hamzah Setiawan. Comment sentiment analysis of jne using k-nearest neighbor (knn) method on twitter. *Indonesian Journal of Innovation Studies*, 2023.

[50] Kaushiv Garg, Kanwarpartap Singh Gill, Sonal Malhotra, Swati Devliyal, and G Sunil. Implementing the xgboost classifier for bankruptcy detection and smote analysis for

balancing its data. *2024 2nd International Conference on Computer, Communication and Control (IC4)*, pages 1–5, 2024.

[51] Nagaraja N Poojary, Dr. ShivaKumar G. S, and Akshath Kumar B.H. Speech emotion recognition using mlp classifier. *International Journal of Scientific Research in Science and Technology*, 2021.

[52] Leno Dwi Cahya, Ardytha Luthfiarta, Julius Immanuel Theo Krisna, Sri Winarno, and Adhitya Nugraha. Improving multi-label classification performance on imbalanced datasets through smote technique and data augmentation using indobert model. *Jurnal Nasional Teknologi dan Sistem Informasi*, 2024.

[53] Dwi Intan Af'Idah, Puput Dewi Anggraeni, Sharfina Febbi Handayani, and Dairoh. Imbalanced classes treatment in deep learning multi-label aspect classification using oversampling and under-sampling. In *2023 International Conference on Computer Science, Information Technology and Engineering (ICCoSITE)*, pages 755–760, 2023.

[54] Muhammad Arham Tariq, Allah Bux Sargano, Muhammad Aksam Iftikhar, and Zulfiqar Habib. Comparing different oversampling methods in predicting multi-class educational datasets using machine learning techniques. *Cybernetics and Information Technologies*, 23:199 – 212, 2023.

[55] Debaleena Datta, Pradeep Kumar Mallick, Annapareddy V. N. Reddy, Mazin Abed Mohammed, Mustafa Musa Jaber, Abed Saif Ahmed Alghawli, and Mohammed Abdulaziz Aide Al-qaness. A hybrid classification of imbalanced hyperspectral images using adasyn and enhanced deep subsampled multi-grained cascaded forest. *Remote. Sens.*, 14:4853, 2022.

[56] Miroslav Kubát and Stan Matwin. Addressing the curse of imbalanced training sets: One-sided selection. In *International Conference on Machine Learning*, 1997.

[57] Tianqi Chen and Carlos Guestrin. Xgboost: A scalable tree boosting system. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '16, page 785–794, New York, NY, USA, 2016. Association for Computing Machinery.

[58] Guolin Ke, Qi Meng, Thomas Finley, Taifeng Wang, Wei Chen, Weidong Ma, Qiwei Ye, and Tie-Yan Liu. Lightgbm: A highly efficient gradient boosting decision tree. In *Neural Information Processing Systems*, 2017.

[59] Liudmila Ostroumova, Gleb Gusev, Aleksandr Vorobev, Anna Veronika Dorogush, and Andrey Gulin. Catboost: unbiased boosting with categorical features. In *Neural Information Processing Systems*, 2017.

[60] Yoav Freund and Robert E Schapire. A decision-theoretic generalization of on-line learning and an application to boosting. *Journal of Computer and System Sciences*, 55(1):119–139, 1997.

[61] Paul Viola and Michael Jones. Fast and robust classification using asymmetric adaboost and a detector cascade. *Proceedings of Advances in Neural Information Processing Systems*, 14, 04 2002.

[62] Jerome Friedman. Greedy function approximation: A gradient boosting machine. *The Annals of Statistics*, 29, 11 2000.

[63] Nitesh V. ”Chawla, Aleksandar Lazarevic, Lawrence O. Hall, editor=”Lavrač-Nada Bowyer, Kevin W.”, Dragan Gamberger, Ljupčo Todorovski, and Hendrik” Blockeel. Smoteboost: Improving prediction of the minority class in boosting. In *Knowledge Discovery in Databases: PKDD 2003*, pages 107–119, Berlin, Heidelberg, 2003. Springer Berlin Heidelberg.

[64] Chris Seiffert, Taghi M. Khoshgoftaar, Jason Van Hulse, and Amri Napolitano. Rusboost: A hybrid approach to alleviating class imbalance. *IEEE Transactions on Systems, Man, and Cybernetics - Part A: Systems and Humans*, 40(1):185–197, 2010.

[65] Wei Fan, Salvatore Stolfo, Junxin Zhang, and Philip Chan. Adacost: Misclassification cost-sensitive boosting. *Proceedings of the Sixteenth International Conference on Machine Learning (ICML’99)*, 05 1999.

[66] Zhining Liu, Wei Cao, Zhifeng Gao, Jiang Bian, Hechang Chen, Yi Chang, and Tie-Yan Liu. Self-paced ensemble for highly imbalanced massive data classification. *2020 IEEE 36th International Conference on Data Engineering (ICDE)*, pages 841–852, 2019.

[67] Mikel Galar, Alberto Fernandez, Edurne Barrenechea, Humberto Bustince, and Francisco Herrera. A review on ensembles for the class imbalance problem: Bagging-, boosting-, and hybrid-based approaches. *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, 42(4):463–484, 2012.

[68] Shuo Wang and Xin Yao. Diversity analysis on imbalanced data sets by using ensemble models. In *2009 IEEE Symposium on Computational Intelligence and Data Mining*, pages 324–331, 2009.

[69] Ricardo Barandela, Rosa Maria Valdovinos, and José Salvador Sánchez. New applications of ensembles of classifiers. *Pattern Analysis & Applications*, 6:245–256, 2003.

[70] José A. Sáez, B. Krawczyk, and Michal Wozniak. Analyzing the oversampling of different classes and types of examples in multi-class imbalanced datasets. *Pattern Recognit.*, 57:164–178, 2016.

[71] S. Sathiya Keerthi and Chih-Jen Lin. Asymptotic Behaviors of Support Vector Machines with Gaussian Kernel. *Neural Computation*, 15(7):1667–1689, 07 2003.

[72] Hsuan-Tien Lin. A study on sigmoid kernels for svm and the training of non-psd kernels by smo-type methods. 2005.

[73] Wikipedia contributors. Django (web framework) — Wikipedia, the free encyclopedia. `https://en.wikipedia.org/w/index.php?title=Django_(web_framework)&oldid=1239731424`, 2024. [Online; accessed 15-August-2024].

[74] `https://realpython.com/get-started-with-django-1/`, 30/07/2023. 10/06/2024.