From Supervised Baselines to Positive–Unlabeled Learning: Fake News Detection on the TruthSeeker2023 Ground-Truth

By

Vishnu Rohith Nanduri

Approved by:

Dr Charan Gudla (Major Professor)
Dr Zhiqian Chen (Committee Member)
Dr Jon Woody (Committee Member)
Dr Diegel Amanda (Graduate Coordinator)
David M. Ford (Dean, Bagley College of Engineering)

A Capstone Project
Submitted to the Faculty of
Mississippi State University
in Partial Fulfillment of the Requirements
for the Degree of Master of Science
in Data Science
in the Department of Computer Science and Engineering

Mississippi State, Mississippi

December 2025

Copyright by

Vishnu Rohith Nanduri

2025

Name: Vishnu Rohith Nanduri

Date of Degree: December 31, 2025

Institution: Mississippi State University

Major Field: Data Science

Major Professor: Dr Charan Gudla

Title of Study: From Supervised Baselines to Positive–Unlabeled Learning: Fake News Detection on the TruthSeeker2023 Ground-Truth

Pages in Study: **Error! Bookmark not defined.**

Candidate for Degree of Master of Science

The rapid spread of fake news in the current social media networks has made the automatic identification of fake news a most significant problem in research. In contrast to classical supervised learning cases, most of the real-world fake-news corpora include a large amount of unlabeled posts and a relatively small amount of reliably annotated ones, which encourages the use of Positive-Unlabeled (PU) learning. In this master level capstone project, the detection of fake-news on Twitter using a hybrid feature space is explored and a comparison of standard supervised classifiers with some of the latest state-of-the-art PU learners is carried out.

The corpus consists of approximately 130,000 tweets of binary categories fake and real. A high-dimensional feature representation of each tweet is a combination of TF–IDF bag-of-word representations of the text with user and tweet-level features, such as engagement statistics, bot scores, and elementary linguistic features. In this feature space we train various supervised base models, such as Logistic Regression, Random Forest, Support Vector Machine, XGBoost, LightGBM and a voting ensemble, and evaluate their models on accuracy, precision, recall, F1-score, ROC-AUC and confusion matrices.

In order to enable the identification of fake news in realistic annotation settings in which counterfeit and authentic examples are consistently labeled, we use various positive-unlabeled (PU) learning paradigms

Even though the supervised ensemble results in the better aggregate performance, the PU methods achieve the competitive F1 and ROC-AUC scores, despite being restricted to positive and unlabeled data, thus highlighting their usefulness in the case of limited negative labels or contamination.

Taken together, this exploration indicates that textual and behavioral descriptors large-scale plus PU learning is a feasible research design to identify fabricated news in lowly labelled social media, which provide a complete empirical contrast between the traditional method of supervised and modern PU learning.

Keywords: Positive–Unlabeled (PU) Learning, Fake News Detection, TruthSeeker 2023 Data, TF–IDF Vectorization, High-Dimensional Sparse Features, Semi-Supervised Learning.

DEDICATION

With heartfelt appreciation, I dedicate this project to my beloved family and friends, for all their firm support and encouragement. I'm particularly grateful to my parents for always prioritizing my education over their needs and aspirations .

ACKNOWLEDGEMENTS

I would like to extend my utmost gratitude to my advisor and major professor, Dr. Charan Gudla, for granting me the opportunity to pursue this research and for their invaluable guidance and unwavering support throughout the entirety of this study. I am also deeply appreciative of the encouragement and assistance provided by my project committee members, Dr. Zhiqian Chen and Dr. Jon Woody. Lastly, I am grateful to my friends at Mississippi State University for their constant support throughout my academic journey.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER I

INTRODUCTION

1.1 Background and motivation

Social media networks, Twitter, Facebook, and TikTok, among others, have over the past ten years become the primary place where individuals, particularly the younger group of people, receive their news. A large proportion of adults in the US reportly resort to social media rather than to the ancient newspapers or TV, which indicates a greater movement in the way information is created, distributed, and consumed on the internet. This is much easier to use by anyone, but it also implies that low-quality and deceptive content, which we refer to as fake news, goes viral at a very rapid pace [1,6].

Fake news is simply information that is intentionally misleading and unambiguous and presents itself as real news to deceive individuals either in the name of politics, ideology, or finances [1,6]. There is sufficient evidence of its social influence now. Indicatively, the 2016 US election had fake news all over on social media and individuals feared that they could change opinions and votes [6]. The same has been witnessed in the case of public health. Social media myths about COVID-19 are associated with vaccine hesitancy, those who do not adhere to guidelines, and an infodemic that disrupts communications about the crisis [7,8].

Even a survey indicates that it is common, such as a US survey that 78 percent of adults were certain or uncertain about at least one false COVID claim, and nearly a third agreed with four or more false claims [9]. This has all unleashed a flood of study on fake-news detection, i.e. systems that attempt to identify what news is good or what is bad on social media automatically [1].

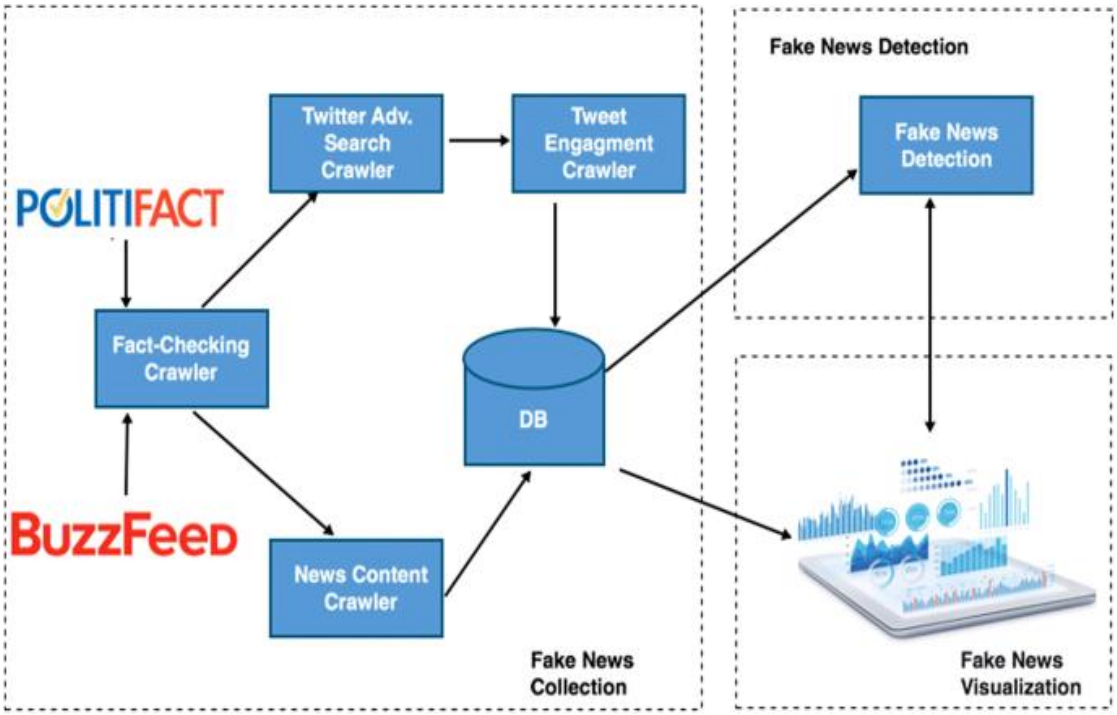

Figure 1.1 High-level View of a Fake News Detection System [10]

In early times, scientists primarily resorted to content features, such as language clues, mood or tone. Recently, they were including additional cues: the person who posted it, how it circulates, who interacts with it, etc. - information incredibly useful in a social media setting [1].

But it is tough, yet it is tough because there is so many posts, bad actors are good and stories keep changing.

1.2 Social media fake news detection

Machine-learning to formulate fake news detection on social media, it is often posed as a supervised text-classification problem: given a collection of posts with labels as fake or real, the goal is to learn a predictor to predict the label of new posts [1].

Classical algorithms (such as Logistic Regression, Support Vector Machines (SVMs), Random Forests, gradient boosted trees, and neural models) have been applied effectively to this task, and they are normally applied to representations such as bag-of-words, TF-IDF, or word embeddings. Besides the textual characteristics, user level and tweet level metadata such as the number of followers, the engagement levels, and the bot scores have been found to boost the performance due to the ability to capture the credibility and propagation dynamics of the source [1].

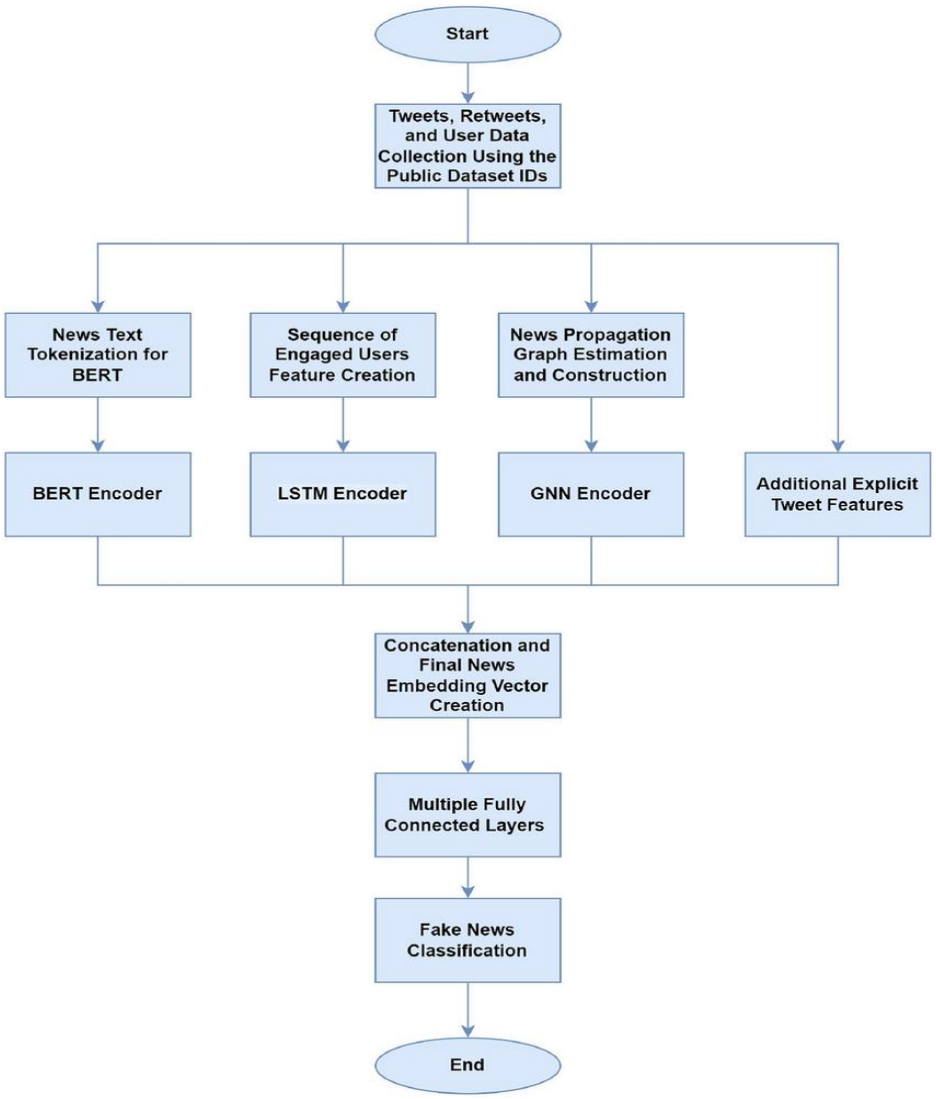

Figure 1.2 a hybrid multi-feature framework for fake news detection on social media [11]

Nonetheless, supervised learning presupposes the availability of sound labels to both classes (fake and real). In reality, such an assumption is often broken in fake news detection: it is expensive and time-consuming to obtain high-quality labels by experts and checking facts, and unlabeled posts are many but noisy. In addition, the labels are usually asymmetric: several datasets have posts that are known (or highly likely to be known), to be misleading, with the rest being considered as unlabeled, rather than trusted as being real.

This imbalance drives approaches capable of utilizing positive and unlabeled (PU) data as opposed to broadly labeled positive and negative data [2,4].

1.3 Positive-Unlabeled learning

Positive-Unlabeled (PU) learning is used in situations where there is unavailability of explicit negative labels and reliable negative labels and only samples of the positive are available and there is a large number of unlabeled samples. The literature proposes a number of PU strategies. Elkan and Noto proposed a probabilistic correction algorithm which approximates the probability of a positive example being one and then makes predictions using a rescaled version of the probabilities when only positive and unlabeled data are available [2]. PU bagging algorithms are based on building ensembles by repeatedly selecting subsets of unlabeled data as pseudo-negatives and combining classifiers that are trained on positive verses sampled unlabeled sets [3]. Two-step algorithms initially select trustworthy negatives (and in some cases trustworthy positives) in the unlabeled set and then train a typical supervised classifier on the they create positive and reliable-negative data sets [4,5]. They have been effectively used in text classification and other similar issues in which the negative labels are not exhaustive or corrupt [4].

PU learning is especially attractive in the fake news detection context since fact-checkers and domain specialists usually label only a fraction of posts as fake, whereas the rest of the posts are not labeled and they can be both authentic and fake. A strongly-constructed PU can thus better represent real-world labeling conditions as compared to an artificially-balanced supervised dataset.

1.4 Problem setting and objectives of this project

The presented Master capstone project focuses on detecting fake news on Twitter in weakly labeled circumstances. The essence of the issue is as follows:

"Having a large set of tweets with some of them marked as fake and real and a significantly larger set of unmarked tweets, come up with models that can correctly perform the task of classifying tweets as fake and real, with special focus on those that can work only with access to positive (fake) and unlabeled data."

To solve this issue, we have a dataset of about 130,000+ tweets that have binary labels which are fake and real content. A hybrid feature space that incorporates represents each tweet.

- Textual characteristics that are developed using TF-IDF representations of preprocessed tweet text.
- User- and tweet-level metadata Follower and friend counts, like status counts, bot ratings, influence scores, as well as basic linguistic markers (e.g., number of words used, use of capitalization and punctuation points, etc.).

On this feature space, we begin by initializing supervised baselines with a number of standard classifiers like Logistic Regression, Random Forest, SVM, XGBoost, LightGBM and a voting ensemble which combines the predictions of these classifiers. These models also presuppose the availability of both positive (fake) and negative (real) labels and are used as an upper-bound performance parameter in an idealized fully labeled environment.

We then explore a family of PU learning approaches that makes use of more realistic assumptions on labeling:

- Elkan–Noto (EN) probability correction.
- PU Bagging based on bagging Logistic Regression as a base model of positive-unlabeled data.
- PU SVM, support-vector based classifier modified to work with PU environments.
- Two-step PU learning, where a first pass is made to remove the unlabeled pool containing reliable negatives, and then a final classifier is trained on the positives and the ones that have been selected due to their reliability.

In all the methods, reliability is assessed based on standard classification measures (accuracy, precision, recall, F1 -score, ROC -AUC) and confusion-matrix analysis, where specific attention is paid to the number of correctly identified fake news/real news.

1.5 Real-world impact of a single false tweet: the 2013 AP Twitter hack

An excellent real-life example of the impact that one tweet of fake news on social media can have on world financial markets is the 2013 hacking of Associated Press (AP) Twitter account. On 23 April 2013, the official Twitter account of the news organization AP was taken over by attackers who sent a fake breaking news tweet that there were two explosions at the white house and that U.S. President Barack Obama was injured [12,15]. Though the tweet was fake, it emerged as a post by one of the most reputable believed news outlets with a verified profile and followed by millions of users, which gave it immediate credit [12,13].

Figure 1.3 Intraday two-minute price chart of the Dow Jones Industrial Average (DJIA) on April 23.

1.5.2 Timeline of the incident

The fake tweet was posted at about 1:07 p.m. Eastern time. In just a few seconds the word started spreading on twitter, being re-tweeted in the thousands in a matter of just a few minutes [12,14]. By that time, the financial markets had already started, and the levels of trading were high. Even trading infrastructures are now more strongly automated, including algorithmic trading systems and high-frequency trading systems which scan news feeds (including Twitter) in search of information which could influence asset prices [12,14].

Figure 1.4 Screenshot of the fraudulent tweet from the hacked Associated Press Twitter.

The tweet appeared to relate an attack on the White House and harm to the incumbent U.S. president which is why it was perceived as an extremely negative and market-relevant news. Human traders and automated trading systems responded immediately. After a matter of two to three minutes of the tweet, the Dow Jones Industrial Average had dropped by an amount of about 140150 points, and the S&P 500 had lost over 130 billion in market value before regaining it in a short period [12,14]. Such a short but sharp drop has been referred to as flash crash and was caused by a single false tweet [12,13]. Soon after the tweet was sent out, AP reporters found out that their account was hacked, posted the statement that the tweet was fake, and terminated the accounts in question, cooperating with Twitter to find out the intrusion [15]. The white house press office also immediately confirmed that President was not hurt and that no such attack

took place [12,15]. The markets recovered their lost gains in minutes after the hoax was revealed and prices reverted to their earlier levels, which were the same [12,13].

1.5.3 Automation and credibility of social media.

The AP incident holds considerable importance due to the temporary loss in market value, as well as the fact which demonstrates how social media, perceived credibility, and automated decision-making interact. The tweet was contributed by a wire service that is old and well-known among the traders and other financial terminals and news organizations and is regarded as the first and highest-trust source [12,13]. This institutional authority made sure that the fake message was acted upon before any confirmation, by human subjects as well as by trading algorithms checking trusted feeds. Event analyses indicate that the systems of algorithmic trading were instrumental in the heightening of the effect of the tweet. They are structured to respond quicker than human traders by analyzing headlines and posts on social media, estimating their probable market effect and making buy or sell deals in milliseconds [12,14]. The implication of a significant geopolitical and security shock in the AP hack seemed to trigger the algorithmic processing of it as a warning of a higher risk, which then led to a rapid sell-off, which, in its turn, led human traders to see the prices drop abruptly [12,14]. The feedback loop which immediately resulted between an unverified social-media statement, algorithmic trading logic, and human response created a brief, yet dramatic shake in the market.

In conclusion, The AP Twitter hack, in the framework of this thesis, is an interesting motivating example that one can use to learn more about fake-news detection on Twitter based on Positive-Unlabeled learning. It shows that (i) one tweet with incorrect information can have very serious real-world effects, (ii) high-trust accounts cannot be entirely trusted, and (iii) automated systems are becoming more dependent on social-media material. This project hopes to make a contribution to the reduction of the risks portrayed by such events by coming up with models that can identify instances of credible and deceptive tweets even to weak labeling assumptions.

1.6 Real-world impact of a single false tweet: "Pizzagate" and the Comet Ping Pong shooting

1.6.1 "Pizzagate" and the Comet Ping Pong incident

The Pizzagate conspiracy can be viewed as one of the brightest examples of online misinformation causing immediate offline damage. At the end of 2016, members of fringe forums and social-media platforms spread a misconceived claim that high-profile Democratic Party officials were involved in a child-trafficking scheme allegedly run out of the Washington, D.C., pizzeria Comet Ping Pong [16]. The story was founded on the wrong interpretation of the leaked emails of John Podesta, the chairman of the presidential campaign of Hillary Clinton. The banal mentions of food, social gatherings, and restaurants were recreated as undercover signs of illegal action. These misimagined items were then fused with images of out-of-context shots of the establishment, its signs, and interior therefore giving the impression of a secret network [16,17]. Even though investigative journalists and law-enforcement agencies were unable to find any supporting evidence, the accusation still spread on the internet.

Figure 1.5 Suspect Surrendering in Pizzagate Incident.

1.6.2 Spread of the conspiracy on social media

Originally found on the forums like 4chan and Reddit, the conspiracy quickly came to the mainstream networks. On Twitter/X and Facebook, users added hashtags such as #Pizzagate to thousands of posts so that they could easily query, monitor, and redistribute the content [16,18]. The assertions were repeated and elaborated in YouTube videos, blog posts, and other news source outlets often as citizen investigations. The discussion was maintained at a high level of interest as it was connected with high-profile political activities and the cases of minors, which created a strong emotional impact and moral outrage. Interactions on social-media in terms of likes, comments, retweets, and recommendations supported the notion that the topic of conversation merits truth by

virtue of the fact that it received a lot of discussion [17,18]. Despite being refuted by numerous fact-checking articles and official words by local policing that no trading activity had been confirmed in Comet Ping Pong, the conspiracy theory still existed among the Internet circles that were predisposed to distrusted media and the government in general [17,18]. Employees and owners of the facility said that they got huge amounts of threatening messages and dangerous calls, which makes the company a primary target of harassment.

1.6.3 The December 4, 2016 shooting

The online conspiracy resulted in a violent event on 4 December 2016. Edgar Maddison Welch, a 28-year-old resident of North Carolina, drove to Washington, D.C., several hundred miles with various guns, including a rifle. Welch later stated that he was out to investigate the Pizzagate allegation and rescue alleged victims. When Welch arrived at Comet Ping Pong, he pulled his gun out as everyone ran away. He fired numerous rounds inside the premises and hit the walls and equipment, but luckily, did not inflict any personal harm. Welch voluntarily gave himself up to police authorities who had surrounded the building after a search of the premises failed to find any evidence of underground rooms or hold captive minors. Welch was arrested, indicted in federal court, and sentenced to several years of prison. What followed later in his declarations was that he was guilty of acting on misleading information he came across on the internet.

For a fake news-detection system trained on Twitter data, Pizzagate is therefore a highly relevant real-world scenario. The core narrative was propagated through tweets, hashtags, and viral sharing exactly the kind of patterns the models in this thesis aim to learn and flag.

1.7 Contributions

1.7.1 Hybrid fake news feature engineering in for Tweets in TruthSeeker 2023 Dataset

Our design is a feature pipeline, which merges text representations that are based on TF -IDF, and user-level, and tweet-level metadata that reflect linguistic patterns and source/engagement properties significant in fake news detection [1].

1.7.2 Supervised and PU models

Overall comparison. On the same large Twitter dataset, we empirically compare a variety of supervised classifiers (Logistic Regression, Random Forest, SVM, XGBoost, LightGBM, and a voting ensemble) with various PU learning strategies (Elkan -Noto, PU Bagging, PU SVM, Two-Step PU) at the same scale, pointing out the trade-offs between fully labeled and weakly labeled regimes. [1,5]

1.7.3 PU learning to simulated realistic fake news labeling

We directly model the context, where a small fraction of the tweets are reliably labeled and the rest are not, which shows that PU algorithms can deliver competitive performance in comparison to supervised baselines and we comment on their implications in the practice of deploying fake news detectors to realistic social-media settings. [2,5]

The rest of this thesis details the dataset and feature building, provides a detailed description of the supervised and PU algorithm, the experimental findings and the limitations and future research of this work.

CHAPTER II

LITERATURE REVIEW

2.1 Evolution of fake news detection

The initial research on the topic of misinformation focused on manual fact-checking, rumour debunking, and domain expert/journalist rating. As social-media channels emerged as the most popular news distributors, scientists started to operationalize the detection of fake news into a supervised text-classification problem: they are given a set of known cases of counterfeited and authentic articles, and a model is trained to make predictions based on a new, unseen article [1]. Primarily bag-of-words or TF-IDF representations and classical classifiers (e.g. Logistic Regression, Naive Bayes, and Support Vector Machines) were used and sometimes supplemented by primitive metadata features (e.g. source domain, publication date).

Shu et al. provide one of the first surveys of fake-news detection on social media and classify methods as content-based and context-based but emphasise the impact of psychological and social theories, including echo chambers and confirmation bias, on misinformation spreading [1]. Follow-up surveys extend this point of view, systematising feature families (content, user, network, temporal), and algorithmic paradigms (traditional machine learning, deep learning, graph -based models) [20]. Altogether, these research reveal that it is more difficult to detect deceptive information on social platforms than on a traditional newswire due to the brevity, informality, and embeddedness of posts in the system of more complicated social interaction.

2.2 Fake news feature engineering: content, user, and network features

The content phenomenon studied through a large body of research has included content-based n-grams, part-of-speech sequences, punctuation and capitalization patterns, sentimentality, subjectivity, and style cues including readability measures or psycholinguistic classes. The purposes of these features are to entrap linguistic differences that characterize deceptive texts, such as a disposition towards increased emotional or moralizing language, increased sentiment extremity, or unique rhetorical forms.

As many studies realise that only content is not always enough, the user-level features and social-contextual features are often combined, including:

- author-profile features (account age, number of followers, number of following)
- engagement (retweets, replies, likes)
- source credibility (historical activity of an account or field)
- propagation patterns (identifying sharers, community affiliation and speed of transmission)

The datasets like FakeNewsNet directly associate news articles with the social context, i.e., user interactions, comments, and network topology, to enable the multi-view modelling [1,20]. As it has consistently been demonstrated in these works, the combination of textual indicator and user and network indicators is much more successful than the use of one or the other modality, especially on the site where the same stories can be reposted and commented on severally. The more recent studies take advantage of graph representations of news diffusion, with each user or post represented as a vertex and the relationship between users or posts as an edge, such as follow, reply, or repost. Graph-based learning algorithms are then used to find out the deceitfulness. The homophily and echo chambers along with unique dynamics of diffusion that distinguish between

fake and real content can be exposed by the network topology. The surveys of graph-based and deep-learning methods confirm that graph neural networks (GNNs) have become one of the attention-seeking tools, which are generally designed to capture the connections between users, news items, and topical entities in a single-layered model [21].

2.3 Deep learning and models based on transformers

Incorporating the achievements of deep learning in natural language processing, fake-news detecting has moved towards the use of neural architectures. Convolutional Neural Networks (CNNs) and Recurrent Neural Networks (RNNs/LSTMs) were initially used as components of early neural models to extract distributed representations of news content directly via word embeddings, occasionally with added attention mechanisms to emphasize important tokens or sentences. These models eliminate the necessity to do manual feature engineering and can learn long-range dependencies in text. The introduction of the pretrained transformer models (e.g., BERT, RoBERTa, GPT-like models) improved the performance even more, by giving contextualised embeddings containing rich semantic and syntactic information. Empirical research shows that transformer models often outperform both classical machine learning and previous neural networks on a range of fake-news datasets, sometimes to very high accuracies of over 95 per cent on test collections [23]. However, these models are data-heavy and require large and high-quality labelled training sets which are expensive to collect in reality. MDPI For fake-news detection, in parallel to the advances of transformers, graph neural networks have been used to form the heterogeneous graph between users, posts, and news entities and learn the relation between them. It has been surveyed that GNN based techniques are skilled at incorporating propagation and community structure, and they generate gains over content based models alone, particularly when labels are accessible on a subset of nodes [21].

2.4 Weak supervision, semi-supervised learning and label scarcity

Although the literature indicates strong performance in completely supervised conditions, one of the notable weaknesses has been the presence of assumptions of plenty labelled data. In practice, fake news needs an expert fact-checking, source cross-reference, and sometimes domain knowledge (e.g. public health, finance, etc.) to annotate. As a result of this, numerous platforms will have large reserves of unlabeled posts but a relatively small number of manually confirmed examples.

To address this limitation, various research directions investigate weak supervision and semi-supervised learning and they include:

- Self-training and bootstrapping, which is a method in which a model trained on a small, labelled set recursively pseudo-labels more data
- Graph-based label propagation, a propagation of labels in a network based on similarity or connectivity.
- Co-training and tri-training, which take advantage of agreement between two or more classifiers.

Although such strategies can use unlabelled data, they also often still need a set of trusted negatives as a baseline, and are often vulnerable to initial mislabeling especially when the positive and negative classes are strongly imbalanced or to distributional change.

2.5 Positive Unlabeled learning: Theory and Algorithm

Positive Unlabeled (PU) learning is a more realistic model which assumes the learner is shown only labelled examples of a single class (usually the positive one), and a large pool of unlabelled examples consisting of a mixture of positives and negatives. The task is to train a binary classifier which differentiates between positives and negatives and where the sampling of the labelled positive is subject to assumptions [2]. Initial PU studies proposed two step methods, the

first stage of which involved selecting trustworthy negatives (and sometimes other trustworthy positives) on the unlabeled set based on heuristics or initial classifiers, followed by training a standard supervised model on the curated dataset [25]. A second avenue of approaches focuses on risk estimation: Elkan and Noto built a model to estimate the probability of an example being labeled, and the probability of the instance being classified as a certain class, which allows correction of conventional probability classifiers in cases where only positive and unlabeled data is available [2]. Later research increased PU learning theory and algorithmic repertoire. Bekker and Davis and other surveys represent methods such as,

- Sample -selection strategies (two-step reliable -negative / reliably -positive schemes),
- Class -prior and propensity estimation schemes (estimating the likelihood of a positive instance being labelled),
- Unbiased risk estimators, which redefine classification risk in terms of the expectation over positive and unlabelled distributions and actively minimise this risk [24].

Such surveys also highlight more realistic issues like estimation bias, overfitting when the imbalances are extreme, and resiliency to breach of the assumption of being selected completely at random when the sample consists of labelled positives [24].

2.6 PU learning in the detection of text and fake news

In the text classification context, positive-unlabeled (PU) learning has been used where only positive examples of data (e.g. documents about a specific subject) are available, as well as a corpus of unlabeled documents. Initial research also considered negative examples as rare or unreliable and followed the PU approaches to generate topic classifiers and sentiment detectors with based on partial annotations [2,25]. Nevertheless, until recently, the research on fake news detection within the framework of a PU has attracted the attention of a small number of studies.

One of the most noticeable research directions entails using network-based PU learning in detecting fake news. De Souza et al. proposed a heterogeneous network representation where the news items, lexical terms, and other attributes form a graph; they then applied PU label propagation to these items and found fake news by only a small subset of news items having the label [27]. This methodology shows that network structure with PU learning can reduce by a significant factor the labeling needs without compromising competition-based predictive performance. Later studies by the authors reviewed the mechanisms of keyword-attention with PU learning to increase fake-news detection under the few-label conditions, thus repeating the advantage of the PU strategies in reducing expert-labeling costs and using rich textual and relational features [28]. Other more modern models such as those that use dynamic graph neural networks to detect fake news acknowledge label scarcity and sometimes adopt PU-style sample selection or re-weighting risks; however, comprehensive, head-to-head comparisons across a variety of canonical PU algorithms on large-scale social-media data are yet to be found [21,23,27,28].

2.7 Gap identification and positioning of this project

Deep supervised architecture is strongly relied upon. Many contemporary studies achieve high accuracy with the aid of transformers or graph neural networks, but they traditionally assume thousands of fake and real samples are available. These assumptions do not coincide with the circumstances on the prevailing platform, where certified marks are rare and costly. Therefore, conservation tests of the performance of classical PU algorithms on large-scale fake-news corpora, provided with purely positive and unlabeled information only, are not extensive.

Weak integration of advanced hand-crafted functionality to PU learning. Most PU-based fake-news models focus on network-level representations, e.g., heterogeneous graphs and label

propagation [27,28], or on text embeddings in general. Much less studies combine high-dimensional TF-IDF vectors and detailed user-and-tweet-level metadata-bot-ness scores, engagement metrics and linguistic indicators- in a PU model, despite the demonstrated effectiveness of such metadata in supervised fake-news setting [1,20].

There is a dearth of comparative analyses between PU strategies. The literature in PU learning spans a range of algorithms, such as probability-correction algorithms (e.g. Elkan-Noto), bagging-based ensemble, SVM-focused PU algorithms, or two-step relevance-negative/relevance-positive (RN/RP) algorithms [2,24,25]. However, in the niche of fake news detection (on Twitter), one is scarcely likely to find multiple PU methods being operationalised and tested in a harmonised experimental design, with equalised feature representations, equal traintest partitions, and equal evaluation measures and compared to strong supervised baselines.

A lack of the in-depth empirical research in realistic labeling regimes. Many studies of fake news report performance in a fully labelled paradigm and conceptualise PU learning in the first place. To that end, quantitative studies of the performance-degradation in switching between fully supervised labels and positiveunlabeled situations and the comparative robustness of different PU strategies at that point are necessary.

This Master capstone project will fill these gaps in the following way:

- It gathers a massive fake-news corpus of Twitter data (about 130,000 tweets), which has a hybrid feature space, integrating TF-IDF textual vectors, user-level and tweet-level numerical features (e.g. bot scores, engagement metrics, linguistic features, etc.).

- It creates strong supervised baselines which include Logistic Regression, random forest, support vector machine, XGBoost, LightGBM and a voting ensemble, thus providing an upper-bound benchmark performance in the fully labelled case.

- It trains and tests various canonical PU learning algorithms Elkan -Noto probability correction, PU Bagging, PU -SVM, and Two-Step PU framework on the same feature representation and train test split used by the supervised models.

- It performs a comparative evaluation of supervised and PU approaches on measures of accuracy, precision, recall, F1-score, ROC-AUC, and confusion matrices, indicating clearly the trade-off between labeling and detection in a realistic positive-unlabeled environment.

Combining extensive feature engineering with a set of PU algorithms and careful analysis, this project attempts to close the gap between academic PU studies and the practical requirements of fake-news detection on social media platforms, providing empirical evidence that is directly relevant to the system that has to operate with very large amounts of unlabeled data and is limited in the amount of expert-label annotation.

CHAPTER III

DATA GATHERING AND PREPARATION

3.1 Overview of the data pipeline

The information used in this study is based on the TruthSeeker 2023 dataset, the author of which is the Canadian Institute of Cybersecurity (CIC) in the University of New Brunswick [20]. This was a resource that provides tweet-leveled ground truth annotations of both real and fake news content with an extensive set of pre-computed features designed across both traditional machine learning and deep learning paradigms [20].

The data pipeline has been outlined in this dissertation as follows:

Raw CSV Data source -> cleaning -> feature engineering -> train/test split -> supervised modelling -> PU data construction -> PU modelling.

The first source is the Features_For_Traditional_ML_Techniques CSV file which includes text messages of the tweet, binary label which indicates the truth of the tweet and over fifty numerical and categorical features that characterize textual, linguistic and user-level features [20]. First stage processing has simple cleaning and consistency checks. The raw columns are then changed to a hybrid feature space. Preprocessing is done to Tweet text such as normalization and token cleansing, followed by encoding using a TF-IDF representation. At the same time, the numeric values derived out of TruthSeeker (such as word statistics, named-entity percentages, follower and engagement counts, bot and credibility score etc.) are organized into a dense numeric

matrix. These pieces are joined horizontally to produce a single representation of features that are compatible with classical classifiers and the PU learning methods.

This is followed by a stratified train/test split on the labeled data, which is used as the standard point of all the supervised models (Logistic Regression, Random Forest, SVM, XGBoost, LightGBM, and a voting ensemble). Subsequently reported performance metrics in the dissertation are always computed on this held out test set. Lastly, the part of the data that has been used in training is reused to create Positive Unlabeled (PU) datasets that can be seen as simulating real-life situations where only a subset of fake tweets are reliably labelled and the rest of the tweets are seen as unlabeled. This chapter explains the meaning of labels and processing feature needed to support both supervised and PU analyses; the elaborated PU algorithms and training programs are discussed in another chapter.

3.2 Data Source and Collection

This paper makes use of the TruthSeeker2023 dataset, which was presented by Dadkhah et.al. as the largest social media ground-truth dataset of real/fake content on Twitter [20]. TruthSeeker was put together through the following steps:

- Pulling of news statements and fact veracity claims in PolitiFact, which is a professional fact-checking organization.
- Keywords query generation based on these statements and then twitter crawling to get related tweets.
- Tweet annotation with Amazon Mechanical Turk, with various validation steps to produce tweet-level truthfulness values.
- Calculation of the additional user- and content-based features, including bot scores, credibility scores and influence measures of each individual user [20].

TruthSeeker consists of over 1,000 different news statements (e.g., 579 real and 479 fake statements) at the statement level and 2009-2022 tweets [20]. The data at the tweet level consists of more than 180,000 labeled data, a large proportion of which is publicly available to academic research. An overall statistical picture shows that 134,198 unique tweets are found, and 68,985 of them are real and 65,213 are false, providing the rates of 0.514 and 0.486, respectively [20].

The set of files included in the TruthSeeker distribution contains a number of CSV files. In this project, we are interested in FeaturesForTraditionalMLTechniques which is a file specifically adapted to classical machine-learning techniques. It sums up more than 50 engineered characteristics that summarize the textual characteristics of each tweet, such as:

- Lexical statistics, e.g. uniquecount (number of unique complex words), totalcount (total length of words), maximum, minimum, and average word length.
- Named-entity percentages (e.g. ORGpercent, NORPpercent, GPEpercent, PERSONpercent, MONEYpercent, TIMEpercent, LOCpercent, LAWpercent, EVENTpercent, WORKofARTpercent), percentages of the text of the tweet [20].
- Verb counts, part-of-speech counts, e.g. counts of present and past-tense verbs.
- User and interaction measures, including follower related (e.g., followerscount, friendscount, favouritescount, statusescount, listedcount), engagement related (e.g., retweets, replies) measures (also in the feature file).
- Behavioral scores (BotScore, BotScoreBinary, cred (credibility score) and normalizeinfluence (influence score)) obtained as part of the original TruthSeeker work to describe the user behavior and its association with truthfulness [20].

Besides these numeric descriptors, the file also has the tweet text and label columns that are necessary in defining the binary target that is being used in this project.

Feature name	Description
unique_count	Count of unique, complex words in the tweet.
total_count	Total number of words in the tweet.
ORG_percent	Share of tokens tagged as organization entities (spaCy ORG).
NORP_percent	Share of tokens tagged as nationalities, religious or political groups (spaCy NORP).
GPE_percent	Share of tokens tagged as countries, cities or states (spaCy GPE).
PERSON_percent	Share of tokens tagged as people names (spaCy PERSON).
MONEY_percent	Share of tokens tagged as monetary values (spaCy MONEY).
DATA_percent	Share of tokens tagged as date expressions (spaCy DATE).
CARDINAL_percent	Share of tokens tagged as cardinal numbers (spaCy CARDINAL).
PERCENT_percent	Share of tokens tagged as percentage expressions (spaCy PERCENT).
ORDINAL_percent	Share of tokens tagged as ordinal numbers (spaCy ORDINAL).
FAC_percent	Share of tokens tagged as facilities (buildings, airports, etc.).
LAW_percent	Share of tokens tagged as legal documents or laws.
PRODUCT_percent	Share of tokens tagged as products or artifacts.
EVENT_percent	Share of tokens tagged as named events (elections, wars, etc.).
TIME_percent	Share of tokens tagged as time-of-day expressions.
LOC_percent	Share of tokens tagged as non-GPE locations (mountains, water bodies, etc.).
WORK_OF_ART_percent	Share of tokens tagged as artistic works (books, songs, films).
QUANTITY_percent	Share of tokens tagged as quantities or measurements.
LANGUAGE_percent	Share of tokens tagged as language names.

Max Word	Length (in characters) of the longest word in the tweet.
Min Word	Length (in characters) of the shortest word in the tweet.
Avg Word Length	Average word length (characters) within the tweet.
present_verb	Number of present-tense verbs in the tweet.
past_verb	Number of past-tense verbs in the tweet.
adjectives	Number of adjectives used.
pronouns	Number of pronouns used.
TO's	Count of the token "to" functioning as a particle/infinitive marker.
determiners	Number of determiners (e.g., "the", "a", "this").
conjunctions	Number of coordinating or subordinating conjunctions.
dots	Number of periods "." in the tweet.
exclamations	Number of exclamation marks "!".
question	Number of question marks "?".
ampersand	Number of ampersand characters "&".
capitals	Total count of uppercase letters.
digits	Total count of numeric digits (0–9).
long_word_freq	Number of long words (above a chosen length threshold).
short_word_freq	Number of short words (below a chosen length threshold).
followers_count	Number of followers the user has.
friends_count	Number of accounts the user follows.
favourites_count	Total likes/favourites the user has given across all tweets.

statuses_count	Total number of tweets (statuses) posted by the user.
listed_count	Number of public lists that include the user.
mentions	Number of times the user has been mentioned.
replies	Number of replies the user has made.
retweets	Number of retweets the user has made.
favourites	Total number of favourites the user's own tweets have received.
hashtags	Total number of hashtags used by the user.
URLs	Indicator of whether the user's profile contains an associated URL.
quotes	Number of quote-tweets the user has made.
BotScoreBinary	Binary indicator of whether the user is classified as a bot.
cred	Credibility score assigned to the user.
normalized_influence	Normalized influence score for the user's impact on the platform.
majority_target	Tweet-level truth label from the TruthSeeker majority-vote process.
statement	Headline/statement from the associated PolitiFact news item.
BinaryNumTarget	Binary encoding of the statement's truth value.
tweet	Raw tweet text related to the statement's keywords.

Table 3.1 Features and their description of the file Features_For_Traditional_ML_Techniques file from the TruthSeeker Dataset

3.2.2 Ethical and Privacy Issues

Even though TruthSeeker is being built on publicly available tweets, it follows a ground-truth building protocol and anonymization, which aim at balancing privacy risk and promoting responsible research conduct [20]. In this investigation:

- - There is no direct use of user identity fields (e.g. usernames or numeric user IDs) as features. Rather, analysis depends on the aggregate user metrics like the number of followers, bot rating, and credibility/influence rating.
- - No additional information is scraped on the live twitter, everything is processed in the stagnant CSV file that is being relayed by TruthSeeker.
- - The study interest does not relate to profiling or the assessment of a particular user, but instead the relationships between the content of the tweet, features derived, and the truthfulness of the tweet.

These choices in design are in line with the current ethical standards in computational social science, as well as the aim of TruthSeeker to be used as a reference point dataset in the field of detecting the real/fake content [20].

3.2.3 TruthSeeker Tweet-labeling Workflow

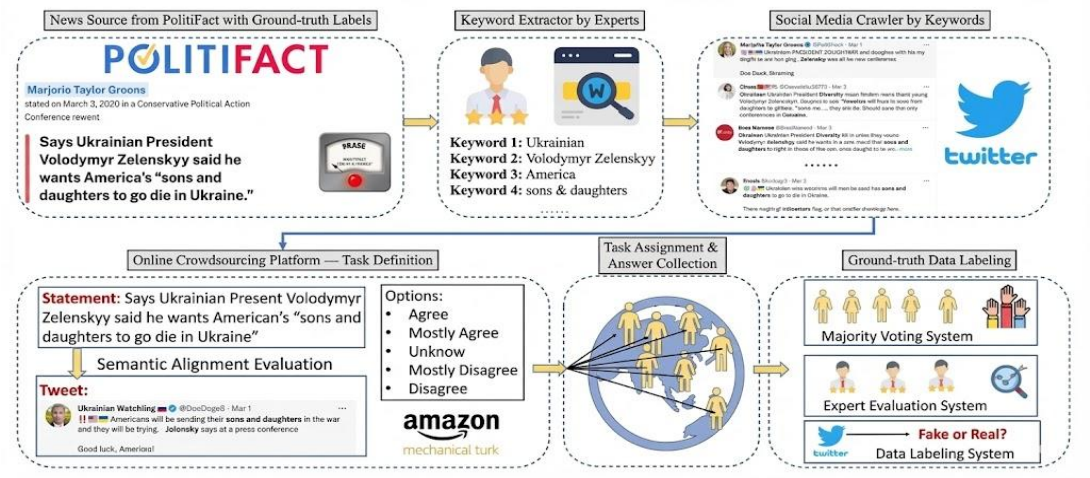

Figure 3.1 Crowdsourced Fake News Data Collection and Labeling Pipeline

The methodology used by the TruthSeeker dataset to produce tweet-level ground-truth labels relying on fact-checked news statements is outlined in Figure 3.1 (configuration chart) [20]. The construction process will consist of several different steps,

- The pipeline starts with a news statement, the source of which is PolitiFact, which is already labeled with an expert-established rating of veracity on a multi-point scale (e.g., True, Mostly True, False, Pants on Fire). In the drawing, the example quote is saying that the Ukrainian President Volodymyr Zelenskyy asked America sons and daughters to die in Ukraine. The statement along with the label of its veracity forms the anchor of all subsequent annotations in tweets.

- Domain experts then do a manual retrieval of a list of keywords out of the statement to represent the essential entities and concepts of the statement (e.g., Ukrainian, Zelenskyy and America, sons and daughters). The keywords will be chosen with the level of specificity to ensure that they will get the relevant tweets but also reduce the noise. They are query terms of the next stage.

- Using the keywords which are defined by experts, an automated crawler is used to harvest Twitter tweets which presumably are interested in the same claim. The right-hand side of the upper row in the figure gives the illustrative tweets obtained by this process. At this point, the tweets are only candidate instances: they are topically consistent with the statement but are not labeled in terms of support and contradiction.

- At the bottom-left part of the figure, all candidate tweets are matched with the original statement and shown to workers on the Amazon Mechanical Turk. The workers conduct a semantic alignment assessment, which is an estimate of the position of the tweet in a relation to the statement on a response scale of categories of Agree, Mostly Agree, Unknown, Mostly Disagree or Disagree. This task is explicitly concerned with meaning, and not topical similarity, and thus, the classification of tweets by echosing, questioning, or contradicting the particular claim is ensured.

- The results are then annotated subsequently by several workers on a per pair basis of tweet-statement. Their decisions are made through a majority vote system as illustrated by the group of icons in the middle part of the bottom row. Voting with the majority nullifies the impact of personal errors, hence forming a stronger crowd measurement of whether each tweet is in line with the statement being considered.

- During the last stage, the failed crowd decisions are reviewed by expert assessors along with the original PolitiFact vericity rating of the statement. According to this confluence, every tweet is labeled with a binary truthfulness indicator, where the tweets that spread or propagate false statement are referred to as fake, and the ones that spread or propagate a true statement (or reject false statement) are referred to as real. The Data Labeling System block is used to effect this dichotomous mapping and gives the binary labels found in the TruthSeeker CSV files.

On the balance, the configuration chart highlighting the fact that the TruthSeeker labels have not been obtained only through the content of tweets or simple matching of keywords. Instead, every tweet is formally linked to a professionally checked fact, which can be evaluated by a number of crowd annotators on a semantic alignment, and then be verified by experts. The result of this multistage process is tweet-level labels that are context-dependent as well as fact-checked externally, making the dataset most suitable to studies on detecting fake-news in the real-world social media.

3.3 Definition of labels and Annotation

TruthSeeker dataset contains tweet-level labels obtained with the help of the multi-stage pipeline outlined in Section 3.2. These labels used Features_For_Traditional_ML_Techniques file mainly in the columns, majority target and BinaryNumTarget along with the raw text of the tweets (tweet) and the fact checked statement (statement) [20]. In modelling, there must be a single and consistent binary target variable

3.3.1 Final binary label

To identify the ultimate label used in every experiment, a new column is introduced with the name of correct target which is denoted as in this work, by a star symbol known as, correct target. The truthseeker ground truth convention is mapped to the Positive class,

but shifted to put the positive class of tweets that were helpful in supporting or spreading a fake or false claim, which is convenient in Positive Unlabeled (PU) learning.

Column name	Meaning in TruthSeeker	Usage in this project
majority_target	Truth value of the tweet (binary real/fake)	Used as primary input label
BinaryNumTarget	Binary representation of statement truth (1 = true, 0 = false)	Used for consistency checks
tweet	Tweet text associated with the statement	Used as input text
correct_target	**New column:** 1 = real, 0 = fake	Final target for all experiments

Table 3.2 Label and text fields used in this study

The conversion of the native labels of TruthSeeker to the labels of correct target is done during the preprocessing phase and it is done by a simple mapping (i.e. mapping the false category to 0 and true category to 1) after a consistency check between majority target and BinaryNumTarget. This mapping is provided in a concise form (Listing 3.1 in the thesis) thus, making the transparent definition of the labels. Table 3.2 summarizes all the columns relating to labels and text, and states that there is only one target, which is called correct target, and the rest of the thesis involves this target.

3.3.2 Class Distribution

Once the construction of the correct target is completed, the total distribution of classes is studied. On the corpus level, TruthSeeker is roughly equal, It has 68985 real tweets and 65213 fake tweets which constitute 0.514 and 0.486 respectively [20]. The subsample that was used in this project, the entire Features for Traditional ML Techniques file, demonstrates a comparable close-to-balanced distribution.

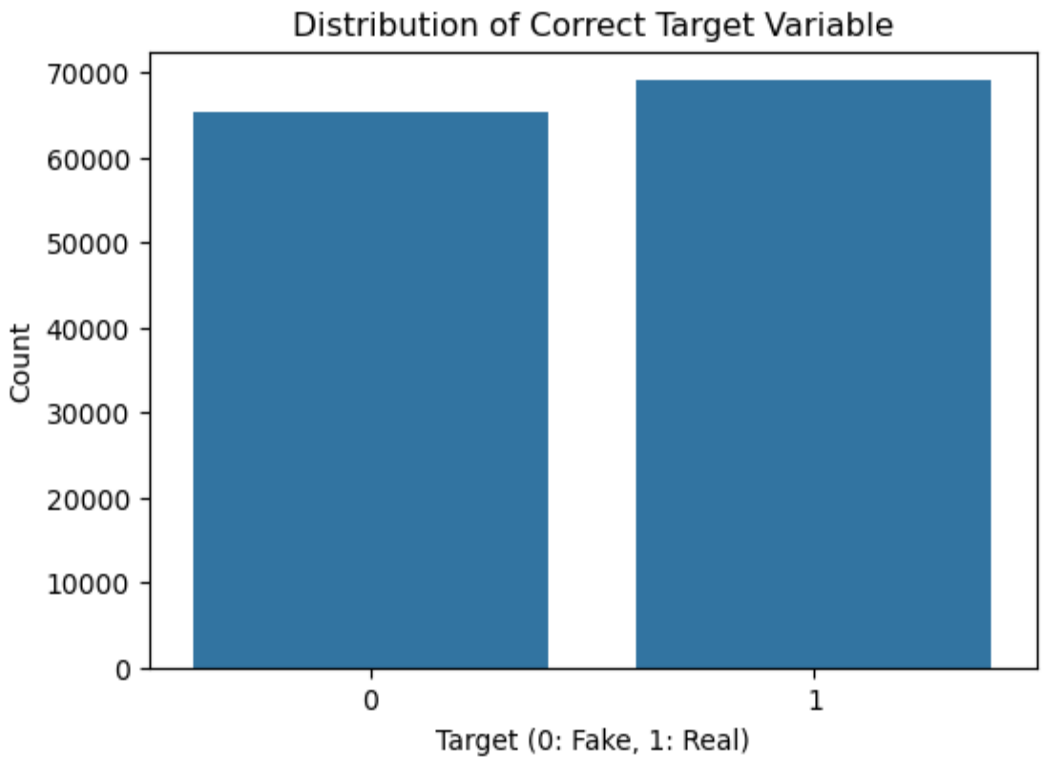

Figure 3.2 Representation of the Class Distribution of the result of a tweet.

Figure 3.2 shows a bar chart of the number of times of the correct target 0 (real) and the number of times of the correct target 1 (fake). The height of the two bars is similar, which proves the lack of the severe imbalance of the classes at the dataset level. The balance is beneficial as it decreases the possibility of the classifiers defaulting to predicting the majority class and allows measures like accuracy and F1-score to be understood in a much simpler way.

3.3.3 Label reliability and motivation for PU learning

Large-scale datasets, despite being designed to be thoroughly annotated, do not suffer from label noise to any degree. Single tweets are sometimes sarcastic, ambiguous or

merely loosely connected to the message behind it and these can be sometimes misinterpreted by the crowd workers. Furthermore, TruthSeeker is an ideal environment in terms of research; in real-world implementations, a very small fraction of unambiguously false or authentic material is labeled, and the majority of tweets are not labeled. This observation is an incentive to the Positive Unlabeled (PU) view taken in the later stages of the thesis. Under this framing:

- Labelled fake tweets (with correct target) in the training set are treated as a sample of true positive class.
- All the rest of the tweets in the training set (real and unlabeled fake) are considered to be belonging to an unlabeled pool, denoted U.
- PU learning aims at recovering a classifier that classifies fake and real tweets by using labelled positives and unlabeled data only.

3.4 Primary Data inspection and Cleaning

Once the final label becomes determined, one should inspect and clean the raw CSV file and then feature engineer it. This step is taken to make sure that the data is found to be data and without any apparent inconsistencies that may lead to bias in later analyses.

3.4.1 Data loading and structural checks

The Features For Traditional Ml techniques CSV is loaded into a pandas DataFrame. Columns with technical indices added in previous preprocessing (e.g. Unnamed: 0) are dropped. Some fundamental structural inspections are then carried out,

- number of rows and columns
- data types of all features
- presence of duplicate rows

There are any exact duplicate rows of tweet-feature that are eliminated to ensure that there is no over-representation of the same content. Practically, the number of duplicates, even including very small numbers, is extremely minimal, therefore this step does not have a significant effect on the size of the dataset.

3.4.2 Tweet content exploratory analysis

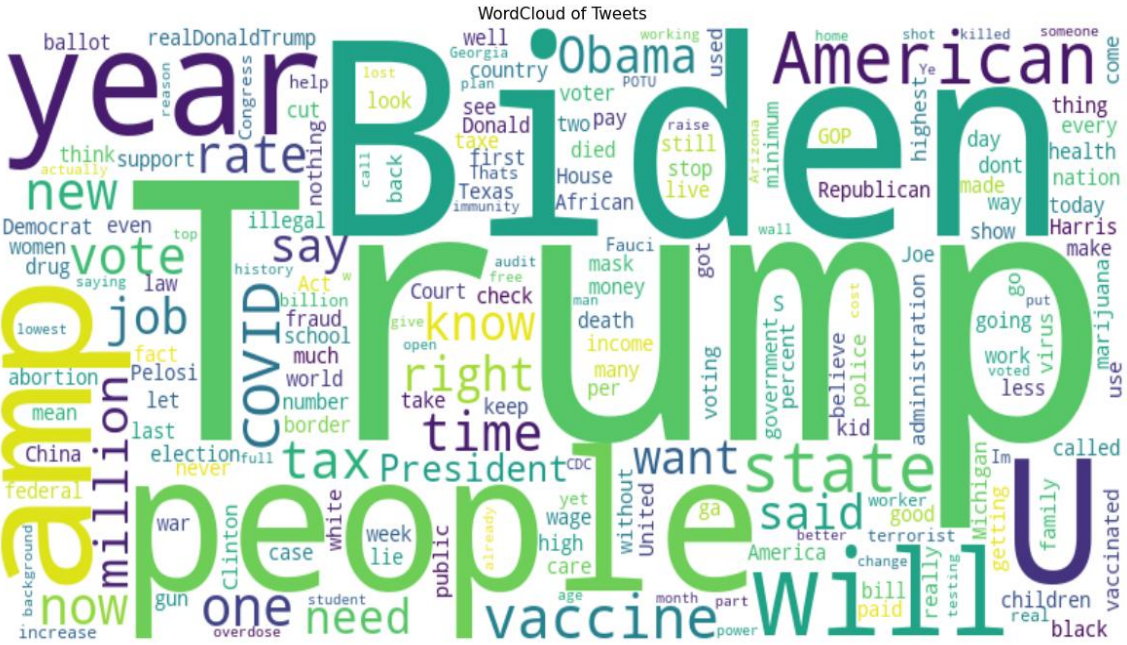

Figure 3.3 Word Cloud representing prominent words with a weightage that result in a tweet to be a Fake or Real.

A word cloud generated out of the raw text of the tweets is created to form a preliminary form of qualitative exploration before preprocessing. The visualization is presented in figure 3.3. The themes that are prominent in the corpus, which are U.S. politics, elections, taxation, vote, COVID, American, and vaccine, are accentuated by large words in the cloud, i.e., by the word Biden, Trump, people, tax, vote, COVID, and American. These areas are the ones that are specifically prone to misinformation and

partisan spin and thus support the applicability of TruthSeeker to the study of fake news detection. This exploratory move also serves to confirm that the data is suitable to its stated purpose, which is political and policy-related claims, prior to more rigorous feature engineering and modelling activities.

3.5 Text Preprocessing

The raw text in the tweet column includes URLs, user mentions, hashtags, emojis and other platform-specific artifacts that cannot be easily used in bag-of-words style models. A special text cleaning pipeline is hence used to generate a filtered copy of the text, which is stored in a new column clean tweet. The resultant processed text then gets into the TF-IDF vectorizer.

3.5.1 Normalization and Token cleaning

Preprocessing pipeline follows the following steps in the order:

- Lowercasing Conversion to lower case is done on all characters so that such tokens like President and president are considered identical lexical items.
- URL removal, Regular expressions are used to eliminate the hyperlinks (e.g., http://… or https://…). These links are usually long, unique, strings that provide insignificant semantic information to use in content classification.
- User mentions and hashtags Tweets containing twitter-specific mentions, which start with the character (e.g. @username) are also deleted, they are mostly user identification, not content identification. Hashtags that start with a hash are either stripped of the hash mark (leaving the underlying word) or deleted altogether, according to whether or not the text in the hashtag contains substantive content (e.g. #COVID19).
- Unalphabetic characters and empty space, All punctuations, remnant symbols and digits are eliminated in the text, unless their number is explicitly maintained as distinct numeric properties (see Section 3.6). Several consecutive spaces are reduced and merged into one space and whitespace before and after the words are removed.

3.5.2 Stopword processing and token filtering

Stopwords used in common English (e.g., the, and, is) are commonly handled at the TF-IDF step with the use of a vectorizer with stopwords=english. This is to avoid the explicit stopword removal pass and assure that only one stopword list is used in the process of vectorization. The main pipeline does not carry out any stemming or lemmatization, since initial tests of simple lemmatization had not yielded significant improvement in performance, and it added another code processing burden.

Version	Example text	Description
Raw tweet	"BREAKING: Biden says Americans will pay more tax in 2023!! Read: https://t.co/xyz #politics"	Original tweet as collected from TruthSeeker, including capitalization, punctuation, URL, hashtag, and year.
Cleaned tweet	"breaking biden says americans will pay more tax in"	Preprocessed tweet after lowercasing and removing the URL, hashtag, hashtag label, punctuation, numbers (e.g., 2023), and trailing noise words, ready for feature extraction.

Table 3.3 Example tweet before and after preprocessing

The cleaned version eliminates the URL, hashtag, numbers and excess punctuations but retains the main lexical information that might be pertinent to the classifier. One or two of such examples are provided in the thesis to help the reader to comprehend the difference between cleantweet and the original tweet.

3.6 Numeric feature construction

Besides text, TruthSeeker also provides a rich set of both numerical and categorical attributes that can be used in traditional machine-learning [20]. A summary of these features is presented in the table presented in Section 3.2. They are collected into three major categories and used in this study with text features TF-IDF.

3.6.1 Lexical and linguistic characteristics

- Statistics of length like uniquecount, totalcount, Max Word, Min Word and Avg Word Length measure the richness of vocabulary and word-length distribution.

- Part-of- speech counts (e.g. presentverb, pastverb, adjectives, pronouns, determiners, conjunctions) are quantitative measurements of grammatical patterns that might vary between fake and real narrative.

- Measurements of punctuation and character usage (i.e., dots, exclamations, question, ampersand, capitals, digits, longwordfreq, shortwordfreq) are useful measures of the stylistic options (favoring too many dots or a capital).

- NER features, including PERSONpercent, ORGpercent, GPEpercent, and others, represent the percentage of tokens that designate people, organizations, places, dates, and numbers among others. Such characteristics indicate the presence of specific actors in a tweet or vague language that can be used to judge the credibility.

3.6.2 User and interaction characteristics.

- The metrics of user popularity and activity such as followerscount, friendscount, favouritescount, statusescount, and listedcount

- Interaction numbers, including mentions, replies, retweets, favourites, hashtags, URLs and quotes.

These variables present a rough description of the reach and the engagement of each user. An example would be that tweets that are sent by an account with very few followers or where the interactions are very skewed are more likely to be attributed to low-credibility content or robot behavior.

3.6.3 Bot, Credibility, and Influence: Behavioral Scores

- BotScoreBinary: binary value of account being a bot;

- Cred: credibility rating of the user calculated according to past conduct and contents;

- Influence: normalized value of the user influence on the platform (e.g. judged by the number of followers and the interaction) [20].

Such characteristics prove especially useful since they can condense large amounts of behavioral information into readable indicators of reliability or authority, which can be associated with the probability of transmitting fake news.

3.7 Vectorization using TF-IDF

To use machine learning methods on textual information, every piece of a tweet has to be converted into a numerical vector. In the current study, the cleaned tweet text (clean_tweet) is modeled with the help of a Term Frequency Inverse Document Frequency (TFIDF) scheme, which is one of the most popular representations of the scope of text classification and information retrieval [1, 24]. TF-IDF uses basic bag-of-words representation and then deemphasizes the high-frequency words and emphasizes those terms that are discriminative of specific documents.

On a conceptual level, the process is represented in Figure 3.Z (the original TF–IDF diagram). The tweets have been cleansed are fed to a TFIDF vectoriser that initially builds a vocabulary of unique lexical elements and thereafter calculates, on a document-by-document basis, a vector of TFIDF weights. The resulting term-document matrix represents individual vocabulary words (e.g. this, good, bad, awesome) in each column, and the individual documents in each row. These entries take the value of 0 when the term is not found in the document and positive values when it is found in the document and positive values of no value terms are set in a way such that frequent terms in the document and relatively rare terms in the corpus will have larger values.

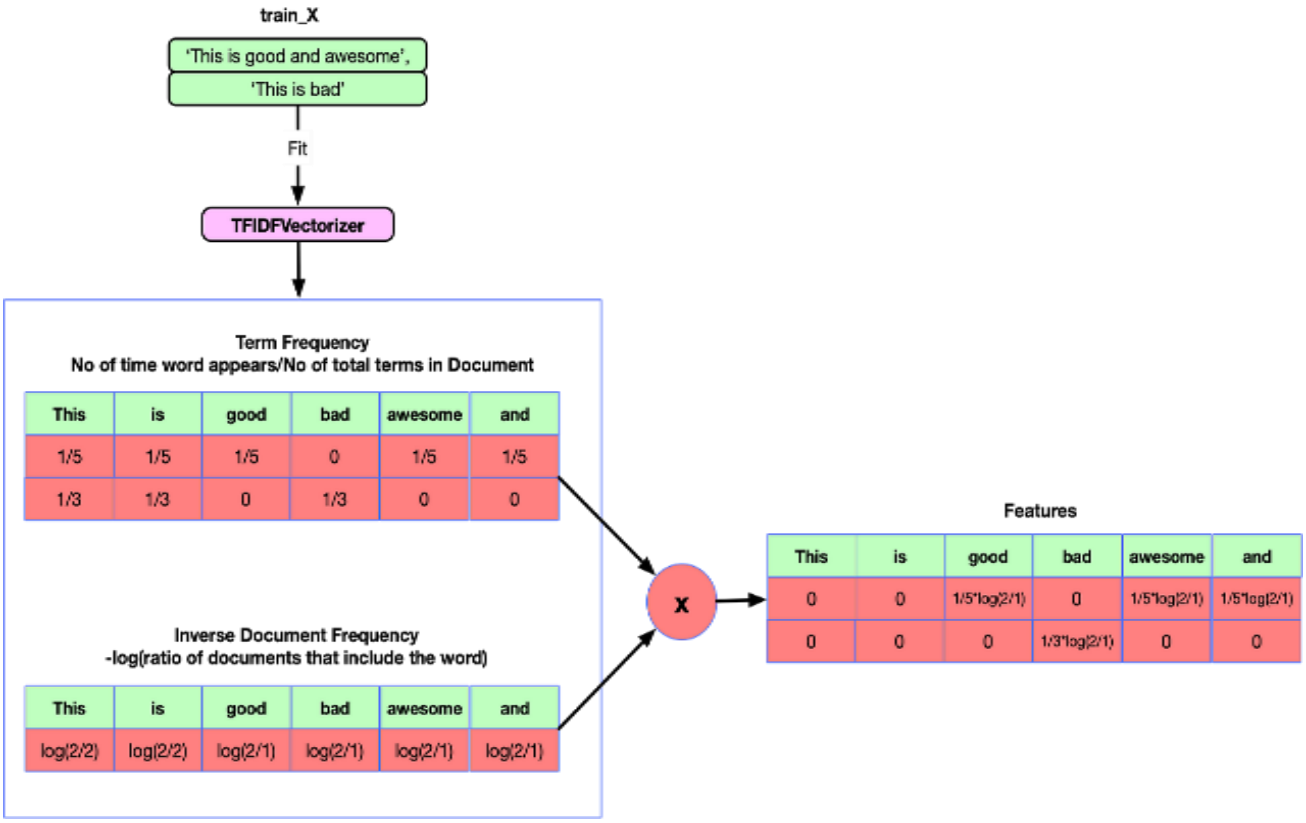

Figure 3.4 Flow diagram of a text into a table of TF-IDF vectors

Formally, for a term t in document d, the TF–IDF weight is defined as

$$\mathbf{TFIDF(t,\ d)\ =\ tf(t,\ d)\ \times\ idf(t)}$$

Where,

$$\mathbf{TF}(t,d) = \frac{\mathbf{count}(t,d)}{\sum_{t'} \mathbf{count}(t',d)}$$

is the **term frequency** of t in d, and

$$\mathbf{IDF}(t) = \log\left(\frac{N}{\mathrm{df}(t)}\right)$$

is the **inverse document frequency** of t. Here N is the total number of documents in the corpus and $\mathrm{df}(t)$ is the number of documents that contain t.

Any logarithmic base can be used, although changing the base only multiplies all TF weights by a constant multiplicative factor and then normalizes all TF weights by the base.

Intuitively, the importance of a word in a document is reflected by tf(t,d) and prevents the significance of words belonging to many documents (e.g., the) and increases the significance of more specific words that belong to only a single document (e.g., vaccine or border). Therefore, TF-IDF gives fairly small weights to ubiquitous presence words, but large weights to content words that contribute to the differentiation of fake and real tweets.

Word	**TF (Doc 1)**	**TF (Doc 2)**	**IDF**	**TF×IDF (Doc 1)**	**TF×IDF (Doc 2)**
The	1/6	0	$log(2/1) \approx 0.30$	0.05	0
Beautiful	1/6	1/5	$log(2/2) = 0$	0	0
Cherry	1/6	0	$log(2/1) \approx 0.30$	0.05	0
Blossoms	1/6	0	$log(2/1) \approx 0.30$	0.05	0
In	1/6	1/5	$log(2/2) = 0$	0	0
Japan	1/6	1/5	$log(2/2) = 0$	0	0
Is	0	1/5	$log(2/1) \approx 0.30$	0	0.06
Very	0	0	$log(2/1) \approx 0.30$	0	0
Spring	0	1/5	$log(2/1) \approx 0.30$	0	0.06

Table 3.4 Example TF–IDF computation for two simple sentences.

In a bid to make the concept practical, Table 3.4 introduces a simple example consisting of two short sentences:

- Sentence 1: “The beautiful cherry blossoms in Japan.”
- Sentence 2: “Japan is beautiful in spring.”

On each unique word in the two sentences, we calculate:

- A term frequency of both documents
- An inverse document frequency between the two documents
- TFIDF weight of the term in each document.

In this toy corpus of only two documents, the word can appear both in the two sentences (e.g., beautiful, in, or Japan) has $df(t) = 2$ and $idf(t) = log(2/2) = 0$, and hence a TF-IDF weight of 0 in both documents. One-line words (e.g., cherry, blossoms, is, spring, etc.) have $df(t) = 1$, and thus $the\ idf(t) = log(2/1) = 0.30$, and so they are not zero TF-IDF weights. This, as shown in table 3.4, is the underlying principle according to which non-informative or common words are likely to receive closer values to TF -IDF values of near-zero, as compared with more distinctive words, which receive higher TF -IDF values and therefore contribute more to the document vector.

Even more frequent stopwords, such as the and is, would have even lower IDF values and virtually disappear in the representation in a real corpus of thousands of tweets, but topic-bearing words such as tax, Biden, fraud, vaccine, or border would still have high weight. In the current project, the tweets texts have been cleansed and then vectorized with a scikit-learn TfidfVectorizer with settings of only using unigrams, to remove English stopwords, and to only keep the top 5,000

most frequent words. The resulting matrix is sparse, with rows forming the TF-IDF vectors of a particular tweet and N is the total number of tweets.

Lastly, this TF -IDF matrix is merged with the numeric features described in Section 3.6 (lexical statistics, bot scores, user metadata, etc.). represent the matrix of numeric features by M. The last representation used by all supervised and PU models is the horizontally concatenated matrix,

$$Xcombined = [Xtext \mid Xnum]$$

which is realised through the sparse matrix operations in such a way that the high-dimensional text features do not consume a lot of memory. The resulting hybrid representation allows the classifiers to use both the textual representation (TF-IDF vectors) and the contextual metadata (numeric features), which is specifically important in the process of identifying fake news on Twitter.

3.8 Train test split for Supervised models

To test the supervised classification models in circumstances that are close to realistic use of the model, the current study breaks down the dataset into the training and testing set. The stratified sampling scheme is utilized to make sure that there are no differences between the relative frequencies of incorrect ($correct\_target = 1$) and valid ($correct\_target = 0$) tweet. Particularly, the training subset (80 percent) and the held-out test subset (20 percent) each form half of the observations used, with the rest being of the test subset used during the final evaluation of the model performance. The partitioning process uses a predetermined random seed to ensure that the results can be reproducible.

Each of the investigated supervised learning algorithms, including Logistic Regression, the Random Forest, Support Vector Machine, XGBoost, LightGBM, and a voting ensemble are trained on the training subset and evaluated on the held-out test subset. Notably, any ancillary information based on the test subset is never included in feature selection, hyperparameter optimization or the creation of PU datasets and thus the label information is never accidentally leaked. A parsimonious tabular list, which is given in the experimental results section, lists the number of legitimate and fraudulent tweets in every partition, thus supporting the fact that the stratified sampling plan maintains the original distribution of the classes.

3.9 Chapter Summary

This chapter outlines how TruthSeeker 2023 data can be processed to convert it into a raw CSV file to a model-ready feature matrix to be used in supervised and unsupervised fake-news detection. Starting with the Features_For_Traditional_ML_Techniques file, the file first describes how the definitive binary target variable, correct_target, would be constructed in which tweet labels would be real (0) and fake (1) according to the ground truth of TruthSeeker [20]. The resulting allocation of labels has been shown to be approximatively even, hence giving both the training and appraisal stages of classification algorithms an added benefit. Thereafter, this is followed by the description of preliminary data inspection and data cleansing steps, including the deletion of extraneous index columns, the treatment of missing values, deletion of records with empty texts or labelling defects beyond the set of 0 and 1, and crude exploratory data analysis. An example of a word cloud created with the raw text of the tweets highlighted the prevalence of the political and public-policy themes, including elections, taxation, and COVID-19, thus confirming that the corpus is appropriate to study misinformation. After this, the chapter provides a detailed description of the feature engineering pipeline. The text of the original tweets is normalized by

lowercasing, placement of URLs, mentions, hashtags, and additional unrelated symbols to produce the field of clean tweets (clean_tweet). The large set of numeric metrics provided by TruthSeeker is then divided into logical groupings: lexical and linguistic statistics (length measures, part-of-speech counts, punctuation and character counts and named-entity percentages); user and interaction metrics (friend numbers, follower numbers, time on platform, likes, replies, retweets, hashtags and URLs); behavioral scores (bot measures, credibility, and influence measures). Specific scaling and transformations are implemented to reduce the effect of extreme values, hence improving the features to be used in linear modeling methods.

Lastly, the chapter elaborates the design of stratified training test partition. A fifth of the dataset will be held out as a test set, and stratification will ensure that the real/fake proportions are maintained in both subsets. This stratified division forms the core dataset of all the supervised modelling ventures that are examined in the next chapter.

CHAPTER IV

MACHINE LEARNING MODELS

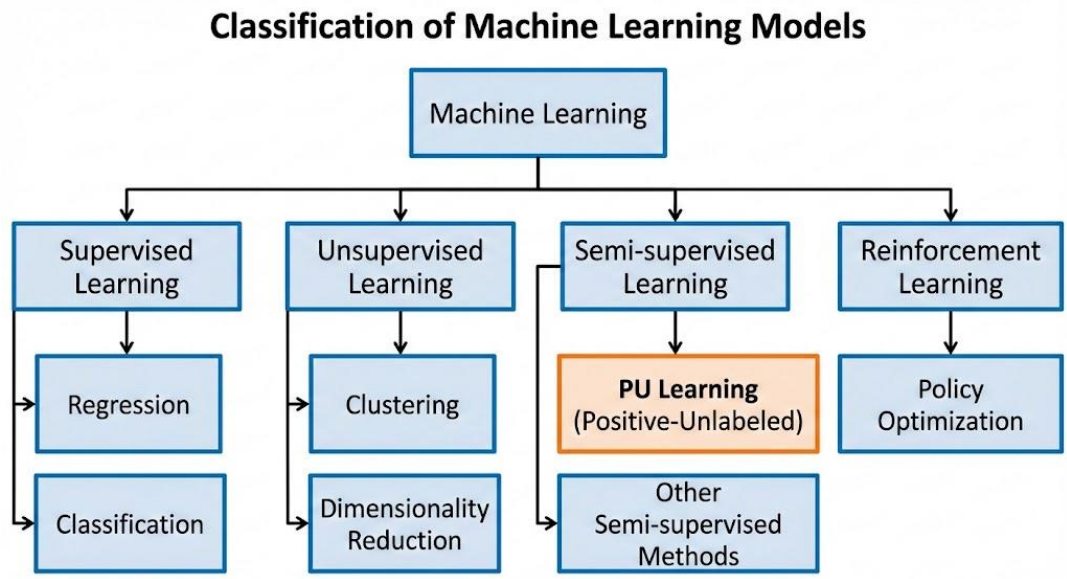

Figure 4.1 Classification of Machine Learning Models

The current chapter defines the machine-learning models that will be used in the current study, these are categorized into supervised baselines and Positive-Unlabeled (PU) learning methods. Each of the models is instantiated on the hybrid feature representation presented in Chapter 3 whereby each tweet is modeled using a TF-IDF vector based on the concatenated text of the tweet (after cleaning) and numeric features capturing linguistic statistics, user metadata, and behavioral scores. The TruthSeeker - FeaturesForTraditionalMLTechniques file is the only piece of data used in all the experimental assessments. This chapter is aimed at explaining the chosen models, justifying the choice, explaining the conceptual structure, and specifying the creation of PU learning environment. The quantitative outcomes are held to Chapter 5.

4.2 Overview of the Modelling Strategy

Supervised Learning Baselines: On the completely labelled dataset (where the correct_target value is 0 in fake and 1 in real tweets), a variety of canonical classifier is trained logistic regression, Support Vector Machine (SVM), Random Forest, XGBoost -(gradient-boosted decision trees), LightGBM (leaf-wise, gradient boosting), Voting Classifier (Ensemble, It combines the single models by a soft-voting group).

Positive-Unlabeled Learning: The identical training data is reconceptualised to constitute PU setting a small labelled positive set P (real tweets) and a large unlabelled pool U, which is composed of real and hidden fake tweets. A number of PU algorithms are then trained on (P, U) this set up and tested on the same fully labelled test set used by the supervised models, These PU learning techniques are Elkan -Noto logistic regression (propensity -corrected probabilities) [34], PU Bagging, PU SVM (asymmetric misclassification cost biasial SVM), Two-step PU learning (reliable -negative selection followed by the calibrated classifier). This design allows you to directly compare fully supervised and PU methodologies on the same downstream fake-news classification problem.

4.3 Supervised Learning Models

Supervised learning assumes that there are labelled examples of both classes. Let combined refers to the end feature array and $y \in \{0,1\}^N$the binary label array (0=real, 1= fake). All the models trained are trained on 80 per cent stratified training split and tested on 20 per cent held-out test split, as outlined in Chapter 3. To reduce small class imbalance, most of the models have a set of equal weights of classes or similar functions. Hyperparameters are chosen through manual hyperparameter tuning of the training set (and internal validation where necessary) and are chosen

to be more stable and with reasonable training times as opposed to choosing hyperparameters that optimize aggressively.

4.3.1 Logistic Regression

The Logistic Regression estimates the log-odds of positive class, which is an affine form of features [29].

In this project,

- It uses $L2$ regularization to avoid overfitting the large, sparse TF -IDF space.
- The SAGA optimiser (designed to handle large, sparse data)
- "Class_weight=balanced" to mitigate the effects of remaining class imbalance
- Feature scaling using StandardScaler (mean=False to maintain sparsity).

Logistic Regression is a powerful linear baseline and directly takes the advantage of the sparse TF-IDF representation.

4.3.2 Support Vector Machine (SVM)

SVMs aim at finding a maximum-margin separating hyperplane between the positive and negative classes commonly using a kernel function to represent non-linear relationships [30].

The linear Support Vector Machine aims at maximally separating the two classes by finding a hyperplane by using the solution,

$$w, bmin21 \parallel w \parallel 22 + C \sum_{N}^{i=1} max\left(0{,}1 - yi'(w\top xi + b)\right)$$

categorizes the classes and the hinge loss punishes the samples that fall on the other side of the margin [25]. The parameter is a trade-off parameter between the margin size and training error. Linear SVMs are also known to be efficient in high-dimensional problems that are linearly separable or near separable like in text classification using TF-IDF. In the current paper, the SVM implementation is linear, and it is trained in the same feature space as that of the Logistic Regression. Though the SVM objective does not directly optimize probabilities, probabilities which are calibrated can be obtained later using Platt scaling or isotonic regression when its use is required by PU methods.

4.2.3 Random Forest

Random Forests are ensemble algorithms that are comprised of decision trees that are jointly trained on a bootstrap sample of the data with random sub-sample of features at each split [26]. Growing of individual trees is carried to large depth and aggregate predictions can be produced by majority voting, in the case of class labels, or by averaging, in the case of probability estimates. This approach reduces the range of variance but maintains a low level of bias and the results provide inherent quality of feature significance [26].

In the given work, the main functionality of the Random Forest is based on the numeric feature subset because tree-based algorithms do not necessarily use extremely high-dimensional, sparse TF-IDF matrices. However, the TF–IDF attributes still can

possibly indirectly affect the model by using the dimensionality-reduced representations or focusing on a finite set of informative terms.

Gradient-Boosted Tree: XGBoost and LightGBM

Gradient-boosted decision tree (GBDT) is a model that builds upon the previous tree, sequentially, in such a way that each new tree is trained on the error that the existing ensemble has made [27,28]. Two state of the art implementations are used,

- XGBoost is a scalable tree-boosting framework with sparsity-sensitive algorithms and complex regularization conditions that are used to control the complexity of the models [27].
- LightGBM also uses other methods like Gradient-based One-Side Sampling (GOSS) and Exclusive Feature Bundling (EFB), to train faster and to effectively handle the high-dimensional feature space [28].

The two models can describe complicated non-linear interactions of numeric features and have the strength of being scale and distribution-heterogeneous. They are also consistent with Random Forests and are typically used on the block of numeric features, which contains user-level and behavioural features, topped with salient text-derived statistics.

4.2.5 Voting Classifier

A soft-voting ensemble with every single individual supervised models (LightGBM, XGBoost, Logistic Regression, etc.) is created to utilize the advantages of the related models. The ensemble: is fed an input of the calibrated class probabilities of each base model, sums up these probabilities (equally or optimally weighted), and is the one which predicts the class with the largest aggregated probability. The ensemble method is usually more stable than any of the

models and is used as an overall supervised guide to assess the PU learning techniques. The Final model in my supervised learning implementation is the Voting Classifier.

4.3 PU learning Models

The current paper uses some classical PU algorithms of learning based on a feature representation which is the same as used in supervised models i.e. TFIDF text and numeric descriptors. Conceptually, PU learning assumes that there is a collection of labeled positives P (in this case, fake-news tweets, which denote the class of interest) and a collection of unlabeled examples U (which is an unknown mixture of positives and negatives).

4.3.1 Elkan-Noto PU

ElkanNoto methodology is a methodology that shows how any probabilistic classifier may be adapted into a PU classifier by considering the fact that only a small fraction of positives is labeled. The key idea is that for an example with features x:

a standard model trained to separate P from U estimates,

$$s(x) \approx P(s = 1 \mid x)$$

Where $s = 1$, (indicates as an example of labelled positive).

We can assume that every positive is labelled and the probability labeling a positive with probability $c = P(s = 1 \mid y = 1)$, we now have,

$$P(s = 1 \mid x) = c\,P(y = 1 \mid x)$$

which is the probability of being a positive, and the probability of being in the positive class is, then, corrected, such that, the probability of being in the class of y is,

$$P(y = 1 \mid x) \approx \frac{P(s = 1 \mid x)}{c}$$

In this paper, logistic regression will be used as the base classifier; once the values of c are estimated using the labelled positive data, the rescaled values of the predicted probabilities will be used, by dividing by c to give $P(y = 1|x)$ and used to classify the tweets, i.e. fake and real.

4.3.2 PU Bagging

PU Bagging is a modification of the bagging ensemble principle to the PU setting. It builds classifiers repeatedly on varying pseudo-negative samples sampled over the unlabeled pool of data, instead of assuming a fixed negative set. At every iteration, a base classifier (usually a linear model) is trained where all labelled positives are used as the positive class, a random sample of is used as the negative class, temporarily, on the sample that is being trained. The same process can be repeated several times and the results combined to obtain an ensemble that is more resilient to the fact that some of the negatives are positives. It is an intuition that every base model testifies to a different hypothesis of what it takes to be negative; therefore, positives that always test positive in a large sample of such models receive high end scores

4.3.3 PU SVM

The PU SVM is a biased PU specific support vector machine. The strategy assumes that all the labelled positives in the set of as positive in a set called P are considered positive and a high penalty is posed to any such misclassification whereas the unlabeled points in a set known as U are considered negative but with a lower penalty since some of them might be positive. It is this

bias that leads the SVM to make highly confident decisions on all the labelled positives, and to accept any mistakes on the unlabeled examples, which are just assumed to be negative.

4.3.4 Two Step PU

The two-step PU approach in this case is based on a generally accepted template: first, reliable negatives (RN) are found; a preliminary classifier is scored on all unlabeled examples and reliable negative tweets, denoted by the character of the set, are those with the lowest estimated chance of being positive, i.e. , are identified, denoted. Second, the last supervised classifier is trained on $P\ U\ RN$. After a negative set of sufficient cleanliness is to be obtained, any standard supervised model, including a random forest or any other robust classifier, is trained on the combined labelled positives and on the chosen trustworthy negatives. Since the last step is a standard supervised binary problem, such an approach allows us to use powerful off-the-shelf classifiers without altering the PU framework. This two steps process, as applied in the current project, is one of the main PU baselines on which the Elkan- Noto, PU Bagging and the PU SVM methods are compared.

4.4 Learning/Training Procedures (Supervised and PU)

This part outlines the training processes that were used on the models in this project, and they include the fully supervised learning and Positive-Unlabeled (PU) paradigms. All along, the feature matrix Xcombined, comprised of TF-IDF features as well as number features, and the final label variable correcttarget are continuously used. The correcttarget label variable can be defined in the following way: correct_target = 1 when it is a real/truthful tweet, and 0 in case of fake/misleading tweets.

Let $X \in \mathbb{R}^{N \times d}$ denote the final feature matrix and $y \in \{0,1\}^N$ the label vector.

4.4.1 Supervised Training

In the case of supervised learning, the models assume that labels of authentic and counterfeited tweets are available. The data is subjected to a stratified hold-out scheme where 80 percent of the instances will be included in the training set ($Xtrain, ytrain$) and 20 percent of the cases in the test set ($Xtest, ytest$). Stratification ensures that the ratio of the number of legitimate (1) and invented (0) tweets is more or less equal in both subsets. Reproducibility is done with a fixed random seed ($randomstate = 42$).

Any preprocessing step calculating parameters based on data is only fitted on the training set: the TF-IDF vectorizer is trained using $clean\_tweet$ with the training dataset; numerical scaling (e.g., StandardScaler on some of the models) is fitted using $Xtrain$. The trainable transformers are then used on the test set, thus avoiding information leakage of $ytest$, or the test distribution to training.

All the supervised models include Logistic Regression, SVM, Random Forest, XGBoost, LightGBM and ensemble and are trained on ($Xtrain, ytrain$). Where necessary class weights or equivalent mechanisms are used to counter minor class imbalance. The test set that is held out is only used in final evaluation and not in hyper-parameter optimization.

4.4.2 PU Setup and Learning

In this thesis, the PU setup is aimed at initializing with a fully annotated fake-news corpus and deliberately obfuscating most of the labels such that the training data reflects the real-world Positive-Unlabeled set. All of their binary labelled tweets (1= positive, 0 = negative) are firstly stratified in a 80/20 split between training and test sets. Importantly, the test set of 20 percent is left intact: it still has the positive and negative labels and is only

utilized on the conclusion of the study to assess all the models. This is the typical experimental design of PU learning: training should be designed to be like the actual PU environment, but performance comparison is maintained on a clean, full-labelled test set.

The main conversion step is the part of the function buildpusets, in which the supervised training split is re-cast to a miniaturised labelled positive cohort and a large pool. Based on the training labels, the code selects the indices in the positive and negative classes by means of the code. Then, a constant proportion of the positive examples (30)is kept as labelled positives, and the rest of the positives (70) are de-labeled on purpose. All these "positives are then combined with all the negatives to form the unlabeled pool. As a result, the unlabeled set is explicitly built as a realistic contamination: there are many true negatives, and an unspecified number of positives, just the situation that PU algorithms are meant to solve

Such a choice protocol is consistent with the popular Selected Completely At Random (SCAR) assumption of PU studies. In SCAR, the probability of the authentic positive being labelled is equal, and the probability of achieving the label is currently 30% in the implementation. Since the sampled positives are uniformly sampled, they represent an unbiased sample of the overall positive population, a fact that is fundamental to the application of such methods like Elkan -Noto, and many current risk-based PU algorithms, which assume that labelled positives reflect the overall population of positives and can thus be used to predict prevalence of positives of the unlabeled pool [24].

In implementation, the function builds three main parts, which include $Xpos, Xunlabeled$ and the aggregated matrix $Xpu. Xpos$ are simply the 30% of training cases are still labelled as positive. $Xunlabeled$ is all the negatives mixed with the other

positives whose labels have been marked out. The two matrices are combined into Xpu and a label vector ypu is generated with a value of 1 assigned to the explicitly labelled positives and a temporary value of 0 assigned to all the unlabeled [24].

The 0 value is to be viewed as unlabeled standby as opposed to an absolute negative name. This 0 is then processed by later PU algorithms as being distinctly different than conventional supervised negatives; to give only one example, by approximating the probability that a given instance is a genuine negative, or by modifying risk formulations to have unlabeled instances count as positives in part and negatives in part.

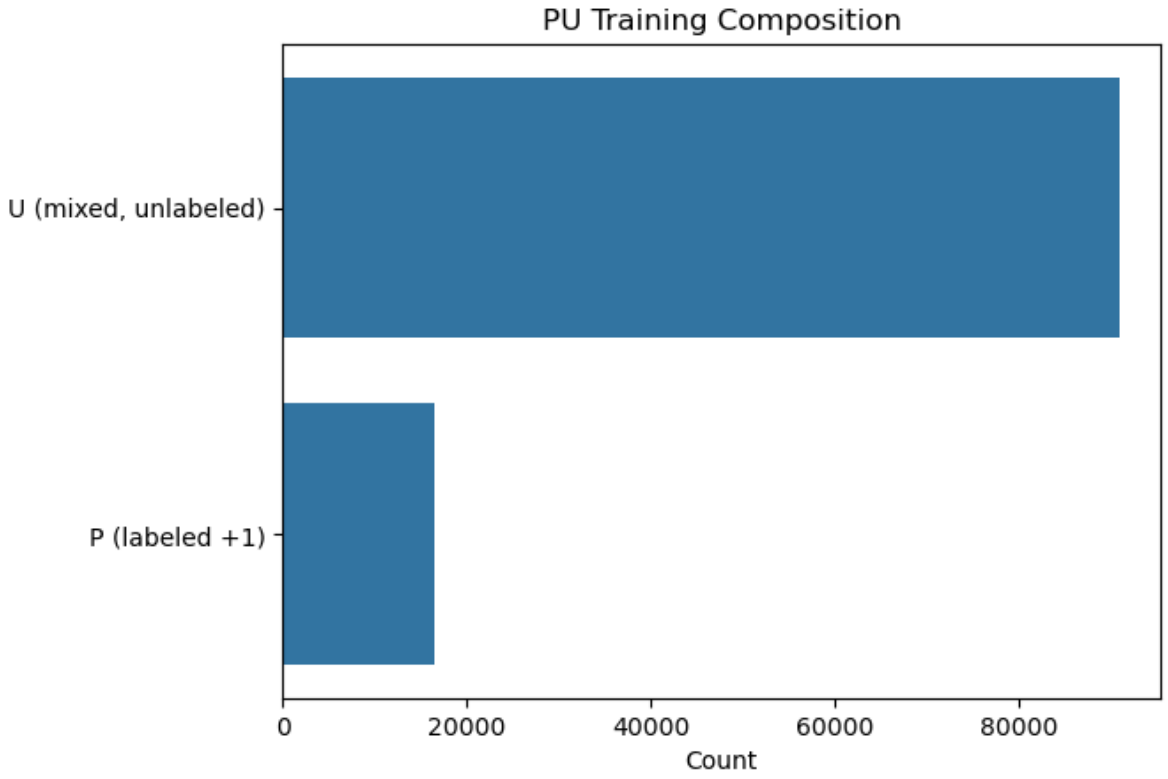

Figure 4.2 PU Training Data Composition

This construction is briefly represented in the bar chart titled PU Training Composition. The bottom bar has the label of P which shows the number of cases that are still explicitly labelled as positive after the 30% sampling. The larger upper bar marked U

(mixed, unlabeled) is in fact much larger, indicating the consideration of all the negatives and the hidden positives. This number then conveys two facts that are salient about the PU training data: the labels are sparse, only a small fraction of the positives is known; and the unlabeled pool is contaminated, which contains a very large number of negatives and an unspecified percentage of positives. PU learning methods have been specifically constructed to work within these two constraints that are often found in real-life applications, including medical diagnosis, recommendation systems, and information retrieval.

Finally, the dictionary generated by build PU Sets is structured in a manner that it interfaces with the next PU models without any problems. Advanced PU algorithms, which use a two-step paradigm, include isolating reliable negatives and then training a more standard classifier on positives versus those reliable negatives, can $use\,Xpos$ and $Xunlabeled$ directly. In addition, the existence of the joint Xpu and ypu representation makes it easy to train support auxiliary classifiers that can differentiate between labelled and unlabelled instances as required by methods like Elkan-Noto[21,25].

4.4.3 Workflow of the PU models

Elkan Noto

The ElkanNoto part of your notebook starts with a call to the PU helper function (such as $buildpusets\,(\ldots)$ that creates two matrices, one, $Xpos$, containing only the labelled instances you have chosen to keep with $keepfrac$, and another, $Xunlabeled$, containing all the other instances (the rest of the positive class and the reverse). These matrices are vertically concatenated to become Xpu and a temporary label vector is created, rows of which come out of $Xpos$ are greatened the label 1 (labelled) and those

which come out of $Xunlabeled$ are given the label 0 (unlabelled). This tentative label does not relate to the actual or false ground truth; it is simply used to differentiate between the items of the labelled set and the items of the unlabelled pool to facilitate the PU learning. The code also uses the routine called $scalenumerictail(\dots)$ before the actual training and is used to maintain the sparse TF-IDF component but standardize only the last numeric columns. The process is like that of preprocessing of supervised logistic regression and helps in avoiding the large numeric figures (followers, statuses, etc.) taking over the inner products.

Then on top of this PU dataset, the $CalibratedClassifierCV$ is fitted with a calibrated logistic regression model. The model is first trained to distinguish between labelled and unlabeled cases, which in turn produce a conditional probability $p\_s = P(s = 1|x)$ After that, on the labelled positives $(Xpos)$ the statistical average of these probabilities is used to approximate the constant c, i.e. $c = P(s = 1 \mid y = 1)$ the probability that a true positive would be in the labelled set. The results of the model are raw scores which at the time of prediction are rescaled by this constant c to approximate the true class probability. This is followed by a validation split that is used to search over thresholds in precision-recall curve as well as F1 score to find the best cut-off as opposed to the default 0.5. Since this pipeline uses a controlled linear model on TF-IDF vectors, estimates probabilities and explicitly optimizes an F1-based decision threshold, the adopted methodology has a substantial degree of stability. Thus, in the experimental findings, Elkan–Noto often turns out to be better and more robust than simpler PU techniques; it uses the whole unlabeled population, it removes label bias, and calibrated scores are converted into a decision that is tuned more towards F1 than default parameters.

PU Bagging

The same is true of your PU Bagging implementation, except that you do not fit one calibrated model, but rather, you fit multiple lightweight models on different views of the unlabeled data. In code, this is achieved by repeating a fixed level of bagging repetitions (e.g. 20 or 50). In the first step, $Xunlabeled$ is sampled by means of bootstrapping, which receives the label 0 (pseudo negative) temporarily and all the rows of $Xpos$ have the label 1 (positive). This synthetic supervised problem is trained on a base learner (which in this case is generally another linear model, e.g. logistic regression or a linear SVM). Once all the base models are trained, each of them is implemented to the test or evaluation data; their respective predicted probabilities are summed up to create a single score per tweet. The intuition behind this is that every bootstrap sample of the unlabeled pool has a slightly different set of true and hidden positives; that averaging over many such classifiers should make it less sensitive to the presence of a specific pseudo-negative sample.

Practically, on this data, this procedure was weaker. The base learner is conditioned to learn over small, noisy negative samples which have, by definition, a fraction of positives mislabeled as negative. When the base model is not sufficiently regularised, or when bagging is run only a small number of times, some runs will be able to overfit to artefacts due to that particular sample. Also, the PU Bagging workflow has no such step as extensive probability calibration and threshold tuning, as in the Elkan Noto workflow, using a separate validation set. An average of the raw scores is usually used to get the final decision with a simple threshold (usually 0.5), which is not necessarily the most optimal when the class prior is changed. All these reasons can explain why the PU Bagging model

used in your experiments usually provided less stable or even slightly worse measures than Elkan-Noto and the better two-step model.

PU SVM

In the PU SVM component, the notebook once more builds a PU training matrix whereby $Xpos$ is piled up with $Xunlabeled$ and a binary label row is used to label rows by $Xpos$ as 1 and the rows by $Xunlabeled$ as 0. This information is entered into a SVM model (with either a linear or RBF kernel, depending on configuration) with asymmetric class-weight or cost parameters often having a higher penalty on misclassifying labelled positives than on misclassifying unlabeled ones. In theory, the code directs the SVM to place greater emphasis on the labelled positives and is less strict on the pseudo-negative pool, which is aware that the pseudo-negative pool may have real positives. The same scaled feature matrix, a mix of TF -IDF vectors and normalized numeric features, is re-used before training so that SVM uses the same feature space as logistic regression.

SVM, however, is very sensitive when it comes to the selection of regularizations parameter C, the kernel and one configuration of the class weight. With the unlabeled in the PU setting, the unlabeled set is extensively polluted with actual and false negatives, and a biased SVM may swing to being over-conservative (pushing the decision boundary out and missing large portions of the positive population), or over-aggressive (classifying a large percentage of the unlabeled population as negative and conditioning on the noisy pseudo-labels). No in-depth threshold search and probability calibration are in-built in the code around the SVM; the code usually uses the native decision function or crudely derived probability estimates. Combined with high dimensional TF-IDF feature space and noisy unlabeled pool, these are the features that make PU SVM results relatively unstable, and

this is the reason why empirically, the SVM-based PU model was more unstable than the Elkan-Noto and the two-step approach, especially in achieving F1 and recall.

Two Step PU

The two-step PU method used in your notebook is better, and it is more refined but efficient intuitively. During the former stage, a temporary classifier (typically, a tree-based model like the Random Forest) is trained on the identical PU data considered above, with $Xpos$ labeled 1 and $Xunlabeled$ labeled 0. The classifier should not be used as the ultimate predictor; its only goal is to compute a scoring function as the measure of the likelihood that unlabeled tweet should be in the positive class. After the model has been trained, all rows in $Xunlabeled$ are applied to acquire the probability scores. The algorithm then picks a set of unlabeled instances having the lowest predicted hips- presumably good negative instances- often the bottom 1020 percent of the score distribution. This subcategory is referred to as RN set of high confidence negative examples. At the second stage, all the PU labels are set aside, and the standard supervised learning problem is re-initiated. A fresh data set is made of all labelled positives ($Xpos$) whose label is 1 and all reliable negatives (RN) whose label is 0. Another powerful tree-based model (such as Random Forest or boosted ensemble) is then trained on this cleaner fully annotated dataset. The classifier is enclosed by the calibration step (with $CalibratedClassifierCV$) in order to match the predicted probabilities with the empirical frequencies.

Lastly, a validation split is selected on which a threshold is selected to trade off precision and recall, and then the model is tested on the held-out test set. Since the final model is trained using only labelled positives and valid negatives with explicit class labels its behavior is very similar to that of the supervised baselines although using fewer and

more carefully chosen negatives. The two-step design explains the reason why the approach achieves results close to the supervised baselines as well as better results than PU Bagging and PU SVM in your experiments. The first step uses the whole unlabeled pool to cut a sub of the negatives that the model is very sure about without considering the rest of the noisy pool as negative or constant resampling. The next step is a powerful exploitative supervised learner with all its calibration and threshold tuning with all-time analogous best supervised pipelines. It is essentially a way of converting a problem of PU that is difficult to handle into a smaller, cleaner supervised problem, that the existing code is adept at handling. Therefore, the reliability-negative selection, and a solid final model explain why, together with Elkan-Noto, the enhanced two-step PU strategy produced the best performance of the PU strategies.

4.4.4 Cross-validation strategy

To measure model stability, as well as to optimize hyper-parameters without polluting the test set, the project uses k-fold cross-validation on the training data mostly with $k = 3$. The training set is divided into three folds which are roughly equal. In both models, the training validation cycle is employed three time, In both repetitions, two folds are trained and the rest of the fold is validated, Then the validation scores are averaged to give an out of sample performance estimation.

The following procedure is accomplished through the $cross\_val\_score$ function:

- In the case of Logistic Regression: $cv = 3$, $scoring =$ "$accuracy$" on the scaled training features.
- In the case of tree-based and boosting models (Random Forest, XGBoost, LightGBM), $cv = 3, scoring = f1$, often $n\_jobs = -1$ so that the models can be run in parallel.

The mean cross-validated F1 score is also compared to the test-set F1 score (after training on the entire training set) in the notebook to determine any significant differences that can be an indication of over-fitting or data leakage. Cross-validation is implemented only to supervised models. PU models are built (algorithms) by using internal validation or threshold-selection algorithms (e.g., they have a validation set to tune threshold in the Elkan-Noto or calibrated two-step PU methods) which are always based on the training data.

4.4.5 Evaluation Metrics

The models are both supervised and PU models evaluated on the same held-out test set with a consistent set of metrics.

- TP = true positive (real tweets which were predicted correctly)
- TN = true negatives (correctly predicted false tweets)
- FP = false positives (tweets that are not real and forecasted as such)
- FN = false negatives (actual tweets which are forecasted to be fake)

The principal metrics are:

Accuracy,

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

Precision,

$$\text{Precision} = \frac{\text{TP}}{\text{TP} + \text{FP}},$$

Recall,

$$\text{Recall} = \frac{\text{TP}}{\text{TP} + \text{FN}}$$

F1,

$$\mathrm{F1} = 2 \cdot \frac{\mathrm{Precision} \times \mathrm{Recall}}{\mathrm{Precision} + \mathrm{Recall}}$$

In addition, the project computes, ROC–AUC, Precision–Recall curve. Confusion matrices and full classification reports are also given to give a more in-depth picture of the error patterns (e.g. how often real tweets are falsely classified as fake and vice versa). Using the same measures and testing set on both the supervised and PU models allows one to make a fair comparison of how much the performance will be degraded when it is necessary to move to more realistic Positive Unlabeled situation.

CHAPTER V

RESULTS AND DISCUSSION

The chapter outlines and discusses the findings of the experiment on the fake news detectors that have been constructed in the framework of this project. Based on the TruthSeeker2023 dataset and the feature representation described in Chapter 3, a set of supervised models and PositiveUnlabeled (PU) models was trained as discussed in Chapter 4. The further discussion will focus on the performance achieved by each of the models, the discrepancies noted in their behaviour, and the implication of the discrepancies on the practical possibility of detecting fake news on Twitter.

The chapter has been divided into three major parts. The models of the supervised learning are first reported using a held-out test set. The summary of the main evaluation metrics, such as accuracy, precision, recall, F1 -score, ROC -AUC, and Matthews correlation coefficient (MCC), are presented in the case of each supervised model, i.e., the Logistic Regression, SVM, Random Forest, XGBoost, and LightGBM. Confusion matrices and complementary visualisations are used to show both the discriminative ability of each model between real and fake tweets as well as to outline common patterns of errors, e.g. a tendency to be overly conservative (incorrectly labeling tweets as real) or overly aggressive (incorrectly labeling tweets as fake). The results are as interpreted where suitable in relation to the properties of a model-based result, e.g. why linear models would not perform similarly to tree-based ensembles in the TF-IDF feature space.

Second, there are the findings regarding the PU learning models, that is, Elkan-Noto logistic regression, PU Bagging, PU SVM, and the improved two-step PU method. Such models were tested using the same full labelled test set, using the same set of metrics as the supervised models hence allowing direct comparison. In each case of the PU approaches, the analysis is

focused on how much it manages to recover the real vs. fake separation on labelled positives and unlabeled data during the training, and why certain methodologies (e.g., Elkan-Noto and two-step approach) perform better and more consistently than others. The behaviour of the PU models is viewed according to the training pipelines of the models, the quality of the unlabeled pool and the effect of threshold selection and calibration. Lastly, a comparative study of supervised and PU models is provided. Aggregate tables and visualisations are used to compare all the models thus explaining the trade-offs of accuracy, recall on fake tweets, robustness, and practical deployability in the absence of complete labels. The discussion then relates to the overall goals of the project to quantify the performance loss that occurs during transition to a less idealized Positive-Unlabeled scenario, and what PU strategies provide the best balance between quality of detections and labeling cost.

The narrative in the subsequent sections is done methodically with the beginning of the metrics and visualisations of the supervised learning followed by the PU learning outcomes and finally a synthesis of the findings in a complete comparison.

5.1 Supervised Learning Results

The Table 5.1 shows a brief summary of performance metrics of all the supervised models tested on the held-out test set. In general, the models are characterized by high performance, with accuracy around 9194% and F1-score of more than 9194% of most algorithms. The measures are calculated with the actual class (correct target = 1) being the positive class hence making precision and recall be indicators of how well the models identify truthful tweets when they can be trusted to do so. Imperatively, they also depict the extent to which the false tweets are rightly considered as the negative category.

Model	Accuracy (%)	Precision (%)	Recall (%)	F1-Score (%)	ROC-AUC (%)	PR-AUC (%)
Logistic Regression	92.83	93.08	92.96	93.02	95.12	94.23
Random Forest	91.06	90.67	92.09	91.37	94.69	93.89
XGBoost	90.69	88.68	93.88	91.20	94.50	93.75
SVM	66.38	66.67	69.17	67.90	71.87	72.69
LightGBM	92.60	92.64	93.00	92.82	95.11	94.20
Ensemble	93.73	93.51	94.35	93.93	95.39	94.65

Table 5.1 Supervised Learning Metrics

The results of the Logistic Regression are very strong, as the accuracy is 92.83, the precision is 93.08, the recall is 92.96 and the F1-score is 93.02. The fact that its ROC-AUC of 95.12 and PR-AUC of 94.23 are significantly greater than one usually indicates that at all levels of decision-making, the algorithm is consistently inclined to give priority to real tweets as opposed to fake ones. This supports the fact that a linear decision boundary placed in the TF idf plus numeric feature space is very effective on this data set which is in line with the existing belief that text classification problems at a large scale have approximately linear separability.

There is a mild difference in the trade-offs illustrated in the tree based models. Random Forest attains a 91.06 percent accuracy and an F1-score of 91.37 with the ROC-AUC and PR-AUC being 94.69 and 93.89 respectively, slightly lower than the results of Logistic Regression.

XGBoost also has a similar overall accuracy of 90.69 per cent but with a more recall-oriented behaviour; the maximum recall of 93.88 per cent and the minimum precision of 88.68 per cent, which results in an F1-score of 91.20. These results imply that increased trees are more violent in including tweets as real (class 1), and thus obtaining more true positive at the cost of higher number of false positive. This behaviour can be good or bad depending on the tolerance of false alarms with the application.

SVM baseline has significantly lower performance than the rest of the supervised models with an accuracy of 66.38, F1-score of 67.90, and significantly lower ROC-AUC (71.87) and PR-AUC (72.69). This finding in a high-dimensional sparse TF-IDF feature space, suggests that the specific SVM setup used (both in terms of what kernel to use as well as regularisation) is not adapting appropriately: the model is underfitting the complicated decision boundary, or overfitting the limited set of features, undermining generalisation. The fact that this contrast is not all conventional text classifiers working out of the box achieve their best factor on this data set highlights that kernel and hyper-parameter tuning is required to attain closer SVM performance to that of the other baselines.

LightGBM, with respectable regularization choices, alleviates the significant part of the gap between tree-based and linear models. It achieves an accuracy of 92.60 0 -92.82 and ROC-AUC and PR-AUC of 95.11 and 94.20 respectively, which, by all criteria are virtually identical to those of Logistic Regression. This shows that properly regularised tree ensembles can outperform linear models, and at the same time they can learn non-linear interactions between textual and

numeric variables; however, they do not differ significantly with the already high-performing linear baseline on this task.

The Ensemble model is the best performing overall model which combines the best individual performers by way of soft voting. It reports accuracy of 93.73% precision of 93.51% recall 94.35% and the best F1-score (93.93), and the best ROC-AUC (95.39) and PR-AUC (94.65). The ensemble exhibits small improvements in precision and recall compared to the constituent models: this implies that it enjoys the drawbacks of being complementary: namely, it can average the conservative predictions of the Logistic Regression with the more recall-oriented predictions of the XGBoost and LightGBM constituents. The fact that the different metrics have been showing consistent, although small, gains suggests that ensembling is a good approach in stabilizing predictions and minimizing variance without overfitting.

These results taken altogether suggest that trained models that are monitored on the full set of labels are able to distinguish between real and fake tweets with high accuracy. The ensemble is a slight but significant improvement over a strong baseline of linear models based on TF-IDF features (Logistic Regression) and highly-configured gradient boosting algorithms (LightGBM, XGBoost). Such relatively low SVM performance is a warning example that more traditional text classifiers might not necessarily be well-adapted to this feature space without a priori optimization.

The SVM baseline has a significantly worse performance, having accuracy and F1 -score in the range of 66-68 -75% and considerably lower ROC-AUC and PR-AUC than other supervised models. This difference can mainly be explained by high sensitivity of SVMs to their

hyperparameters which is very sensitive in high-dimensional, low-density representation like TFIDF. The model used in this research is based on a large set of features (textual features) of more than 5,000 features along with numeric variables; the chosen kernel and regularization allow one to either fail to reflect the real decision boundary, hence underfitting, or being over-constrained, thus, limiting its discriminative ability between classes. As a result, the specific SVM setup is not well-adapted to the structure of this dataset and the margin that it learns is not consistent with the true separation between authentic and fake tweets.

On the contrary, the Logistic Regression and tree-based models (Random Forest, XGBoost, LightGBM) are more appropriate models to fit this feature space and are better-tuned. Logistic Regression is a linear classifier designed to maximize a probabilistic loss directly with sparse TF -IDF features using L2 regularization and class equalisation; it has been shown to work well with large-scale text classification problems, as here with greater than 92 -percent accuracy and F1 -score. The tree-based ensembles also represent the non-linear interactions between the lexical and numeric variables and with an adequate depth and regularization, they achieve similar or even better performances. The ensemble model also derives advantages related to averaging in these strong but diverse decision boundaries thus increasing prediction stability. All these together can explain the fact that Logistic Regression and the tree-based models can provide a strong and stable result, in this particular setting, the SVM is not performing well.

5.1.1 Confusion Matrices for Supervised learning

The confusion matrix is a 2 by 2 contingency table, which compares the predicted labels with the actual labels. The rows in the current research are the actual class labels (0 0 - fake), and the columns are the predicted class labels. The upper-left and bottom-right diagonal values sum up to the total amount of true positives (you are a fake) and negatives (you are a real), respectively, and the off-diagonal values represent the false classification (you are a fake that are labelled as a real) and the true classification (you are a real that are labelled as a fake). This representation is desirable since it not only measures the total model accuracy, but also explains possible class-specific bias in predictive accuracy.

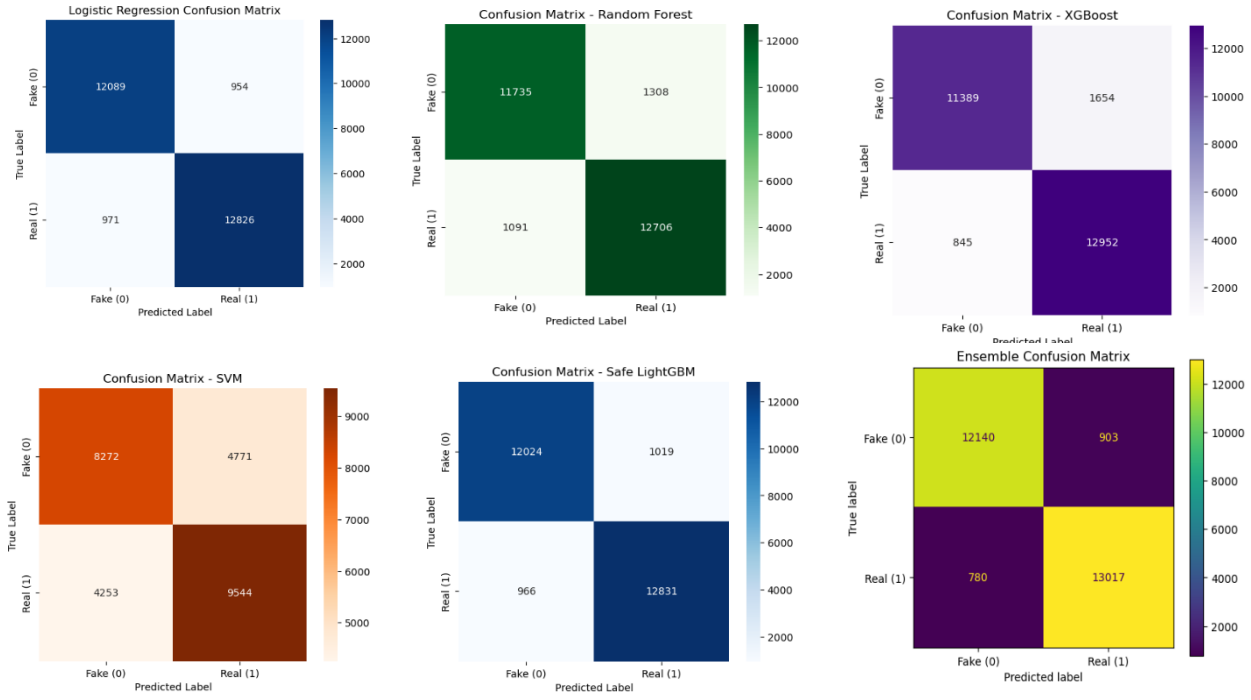

Figure 5.1 Confusion Matrices of Supervised Models

Figure 5.1 shows that the most popular supervised algorithms (Logistic Regression, LightGBM, XGBoost, and the Ensemble) have strong diagonal values and relatively weak off-

diagonal values, which means that the overall rate of misclassification is low. Symmetric profiles of error are also evident in Logistic Regression and LightGBM, wherein the count of fake -real and real-fake misclassifications are close and relatively small. Nonetheless, XGBoost has a small bias towards protecting real content: it has an even lesser number of real to fake errors and slightly higher rates of fake tweets being set as real.

Such behaviour is in line with its superior recollection of the actual class. The ensemble strategy has the most desirable confusion matrix, and it also reduces both types of errors compared to the individual base learners, across the board. The ensemble has the lowest rates of fake to real and real to fake misclassifications as it combines predictions of several high-performing models, which supports its achievement of the highest F1-score and AUC values in the summary table. On the other hand, the confusion matrix obtained with Support Vector Machine shows the significant off-diagonal values in both directions, which means that the Support Vector Machine often misclassifies both fake and actual tweets. The given visual pattern is in agreement with the highly worse performance measures which were reported in Table 5.X, proving that, in the current hyperparameter setting, the SVM does not learn a suitable decision boundary in this high-dimensional feature space.

5.1.2 ROC-AUC and PR-AUC Curves

The definition of the Receiver Operating Characteristic (ROC) curve shows that it is a blueprint of the True Positive Rate (TPR) versus the False Positive Rate (FPR) where the decision threshold is varied in a systematic way. An idealized model where a positive class, namely, real tweets with correct target = 1, was ranked perfectly discriminative would result in a curve that fits closely to the lower-left corner of the plot, the ROC-AUC of which is 1.0. Random chance, on the other hand, takes the shape of a diagonal line at which AUC takes a value of about 0.5. Figure 5

Y shows that the ROC curves of the Logistic Regression, the random forest, the XGBoost, and LightGBM, and the Ensemble all tend to establish themselves on the inner-left corner, with the respective AUCs of approximately 0.95. The rates of these findings suggest that the models continue to score higher with real tweets than the fake twets through an extensive range of threshold and not just at a single operating point which is depicted in the confusion matrices. Although Ensemble achieves the best ROC-AUC, the improvement compared to the Logistic regression and LightGBM is relatively lower implying that the ranking performance of the three algorithms is equally robust.

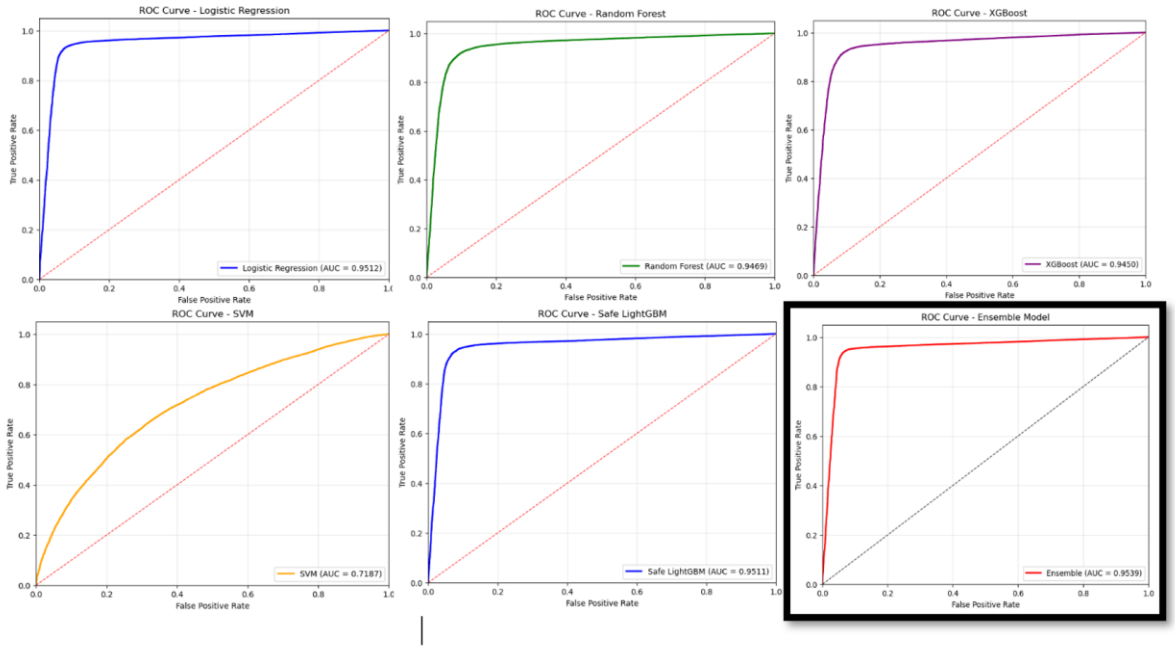

Figure 5.2 ROC-AUC Curves of Supervised Models

In comparison, the ROC curve of the support vector machine (SVM) is significantly more similar to the diagonal, and its AUC is around 0.72. This stance implies that the SVM cannot discriminate between genuine and falsified tweets even in the situation when the amount of the decision threshold is set at will, since its score distribution does not coincide with the actual values

satisfactorily. This has been supported by the fact that the confusion matrix and F1 -score are poor, thus supporting the conclusion that the chosen SVM setting is not applicable to the eature space offered.

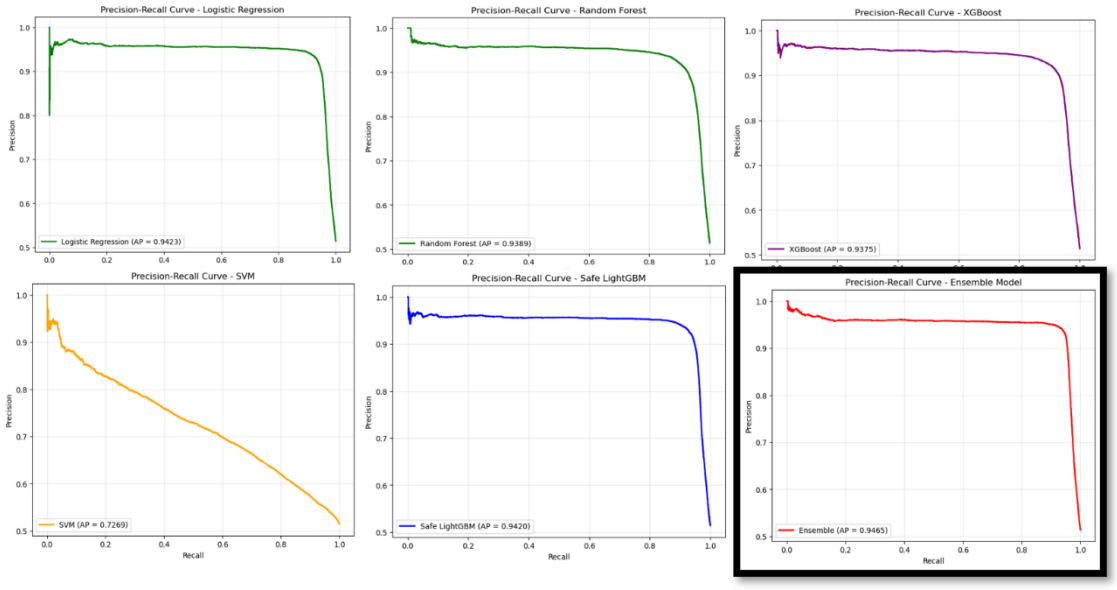

Figure 5.3 PR-AUC Curves of Supervised Models

Figure 5.3 shows the Precision Recall (PR) curves, which can provide a complementary view to highlight the quality of the predictions in the positive class and in which the quality is represented by precision, plotted on the y-axis, and recall, plotted on the x-axis. Curves which concentrate around the upper-right corner indicate models which maintain high precision in high recall rates. All models show PR curves that are fairly level off around a precision range of 0.93–0.95 with most of the recall range, which attain an Average Precision (AP) of about 0.94095. The Ensemble marginally outperforms the individual models in AP; but the difference is small, meaning that both approaches can retrieve the enormous majority of true tweets and still have a small rate of false alarms of fake tweets. Contrary to this, the SVM PR curve will exhibit a stiffer decrease: it has a worse precision below 0.8 with a recall, and a worse AP of approximately 0.73.

The implication of this trend is that, as the SVM is configured to increase the rate of recall, it decreases this rate at the cost of a significant number of false positives; or the reverse, as constraining the rate of recall results in the false to genuine tweet ratio increasing significantly. Together, the ROC and PR analyses confirm the story told by the metric tables and confusion matrices: the linear and boosted-tree models, and the Ensemble, in particular, providing uniformly high discrimination at all thresholds, and the SVM, being a worse model, being a worse baseline even with the threshold selection removed.

5.2 Positive Unlabled Results

Model	Accuracy (%)	Precision (%)	Recall (%)	F1-Score	ROC-AUC	PR-AUC
Elkan–Noto PU	84.53	88.41	87.53	85.33	90.40	88.41
PU Bagging	64.88	70.39	64.59	64.76	71.16	70.39
PU SVM	58.27	61.99	48.66	54.53	61.92	62.80
Two-Step PU	76.69	71.97	89.50	79.79	83.75	81.98

Table 5.2 PU learning Results

Table 5.2 gives the performance of the four positiveunlabeled (PU) models with the same held-out test set as used in the supervised baselines. The computations of metrics are done with the real class (correct_target = 1) as the positive class; thus, precision and recall are the measures of how much the models recover the real tweets when trained on labeled positives and an unlabeled pool only.

Elkan Noto PU Logistic Regression comes out the best PU method. It has a 84.53% accuracy, 88.41% precision, 87.53% recall and an F1-score of 85.33 and ROC-AUC and PR-AUC of about 0.90 and 0.88, respectively. These values are not as good as the fully supervised Logistic Regression and Ensemble models, but they do show that with no explicit negative labels in training, Elkan-Noto is to an extent retrieving the underlying signal. This fact is in line with its design: the approach uses all the unlabeled data, corrects the likelihood of a sample being labeled, and uses a calibrated linear model, which fits the TF-IDF feature space well.

The second-strongest group is the enhanced Two-step PU method. It is somewhat less accurate (76.69‘%), but the accuracy provides a high recall of 89.50 and a precision of 71.97, respectively (F1 -score = 79.79 and ROC -AUC/PR -AUC = 83.75 and 81.98). This trend is indicative of its training approach: by first identifying a pool of trusted negatives in the unlabeled pool, and, then, training a supervised classifier on positives versus these negatives, the approach concentrates its efforts on the retrieval of real tweets, against which the false positives are going higher. This high-recall behaviour could also be desirable in situations, when it is the missed real content in which this kind of behaviour is particularly unwanted.

PU Bagging and PU SVM show significantly poor performance. PU Bagging has an accuracy of 65 about and an F1-score of 64.8, and ROC-AUC and PR-AUC of 0.71. Since all the base learners are trained on a different bootstrap sample of the unlabeled pool, which functions as negative, the ensemble is strongly influenced by the label noise: any sample of the negatives has a composition of true and non-true tweets, and an average of such contaminated models is not sufficient to remove the noise. PU SVM is the weakest and is achieved with an accuracy of 58.27,

F1 -score of 54.53, and AUC values a bit above 0.6. Such findings can be explained by the fact that, in this setup, a biased SVM is extremely sensitive to the significant pollution of the unlabeled category and does not find a stable separating hyperplane in the high dimensional TF-IDF space.

Overall, the performance of all PU models is lower than that of their fully supervised versions, as is expected, in the supervised scenario, the algorithms can observe clean labels on both real and fake tweets, whereas in the PU scenario they have to learn the negative class indirectly by observing a noisy unlabeled pool. Elkan-Noto and Two-step method have the smallest performance gap, which explicitly models the probability of being labeled or construct a well-filtered negative set. In contrast, PU Bagging and PU SVM, which are more directly based on using large proportions of the unlabeled pool as negative label noise, are affected more by the label noise which leads to a significant decrease in accuracy and discriminative power compared to the supervised baselines.

7.2.1 Confusion Matrix for PU learning

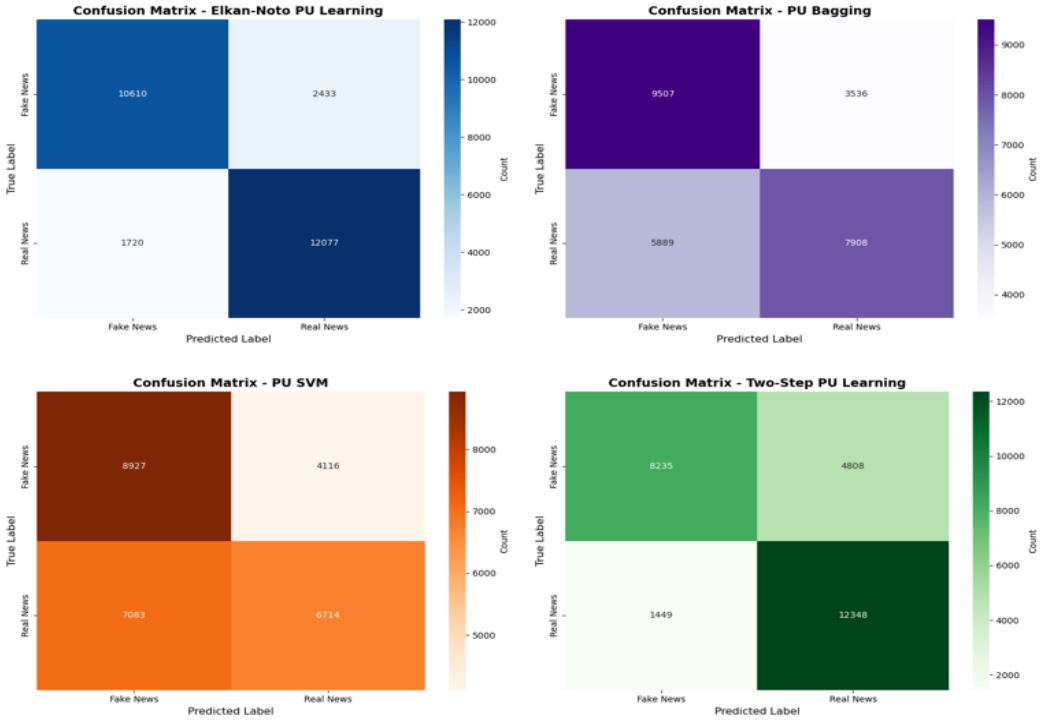

Figure 5.4 Confusion Matrices of Pu learning models

confusion matrices of four PU models are shown in Figure 5.4. False news and real news are represented as the true label in each row and the predicted label in each column respectively. Compared to the supervised confusion matrices, each PU model has reduced diagonal values and improved off-diagonal values, which is also expected given that there is no explicit negative supervision during training.

Noto PU model offers the cleanest model: it hits the correct labeling of most fake tweets (10,610) and actual tweets (12,077) with the mislabeling (about 2,433 fake -real and 1,720 real -fake) being moderate. This is in line with its relatively good F1-

score and AUC values; hence, suggesting that probability-correction and calibration processes are able to restore a significant part of the underlying data structure.

On the other hand, bilateral PU model achieves a different trade-off. It captures a high figure of correctly identified real tweets (12,348) and a rather small number of real 1449 of real to fake errors showing that real news is well recalled. However, it has a larger percentage of incorrectly classified fake tweets (4,808 fake 4,808 real) and this explains its low precision but high recall as indicated in the metric table.

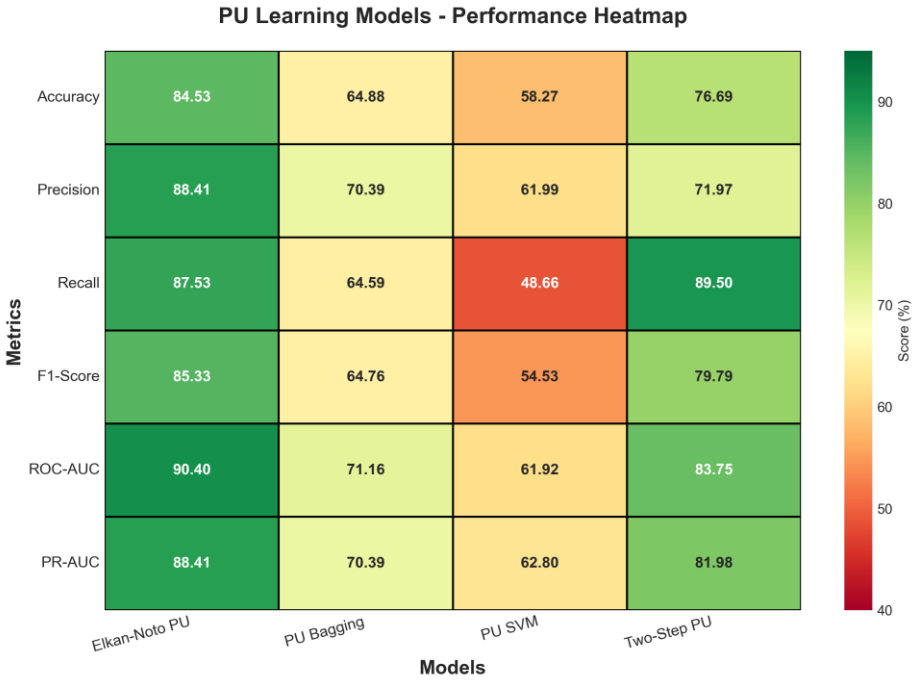

Figure 5.5 Heatmap for the metrics of PU models

The PU Bagging and PU SVM methods produce a lot more noisy confusion matrices. Besides a non-negligible fraction of fake tweets being classified as real (3,536), PU Bagging on average misstages a large fraction of real tweets (5,889 real into fake) as

well as negatively (1). PU SVM has the worst overall results, with high scores in both directions (4,116 fake[SEP]″real and 7,083 real ″fake), thus proving the assumption that the biased SVM model is not able to draw a strong decision line when the class of interest (the negative one) is significantly contaminated with unlabeled positives.

5.2.2 ROC-AUC and PR Curves

The ROC curves of the four PositiveUnlabelled (PU) classification models being investigated are shown in Figure 5.5. Elkan Noto PU model is the one that has the most favourable ROC trajectory, as it is still located close to the upper-left corner of the plot, with the area under the curve (AUC) of 0.90. This empirical finding suggests that, despite its training regime consisting of only labeled positives and unlabeled observations, the model at all classification thresholds constantly places authentic tweets over fake ones, hence supporting both its strong F1 -score and balanced confusion matrix structure.

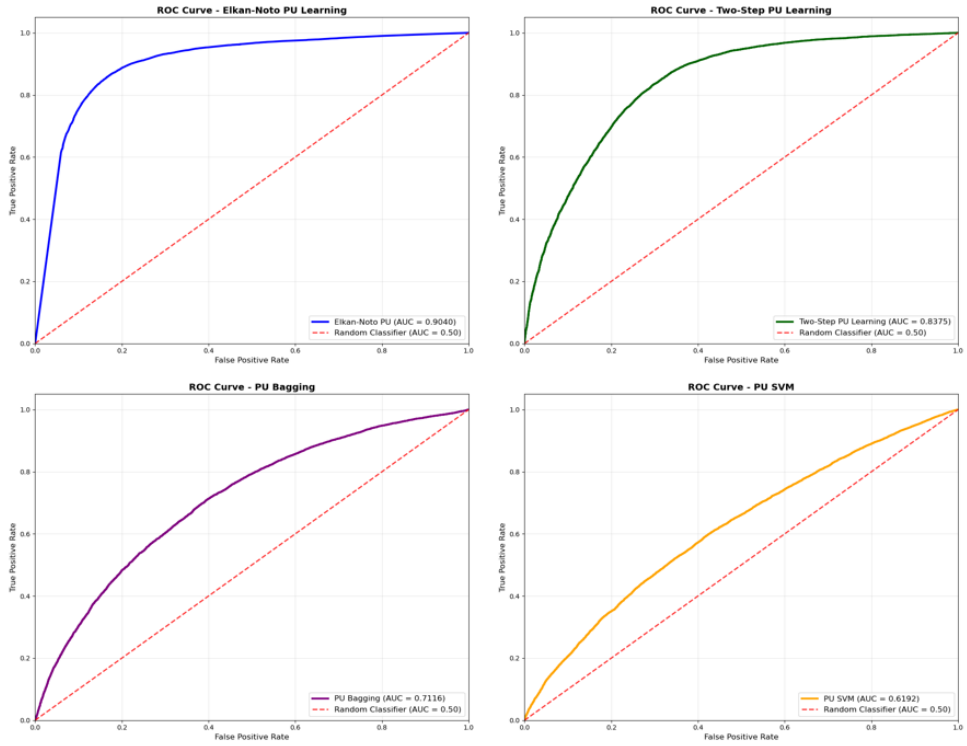

Figure 5.6 ROC Curves of PU models

The second-best ROC progression achieved by the Two-step PU is 0.84. The resulting curve rises quickly when the false positive values are low, then levels off, representing the ability of this model to achieve moderate true positive values, at the cost of moderately damping false positives, though not as aggressively as the Elkan-Noto scheme. This performance profile fits into the architectural design of the model: once careful decisions have been made regarding the use of credible negative examples, the following supervised classifier provides effectiveness, but it is limited by the lack of complete label information as compared to fully supervised learning settings. Both PU Bagging and PU Support Vector Machine (SVM) models share relatively flatter ROC curves, whose AUCs are around 0.71 and 0.62, respectively, only slightly higher than in the case of random chance. Their discriminative values between authentic and fraudulent tweets do not generate a significant difference, and threshold changes are too little to restore the optimal results. This weakness can be explained by the fact that it is especially easy to fall prey to label noise when using large scale training on the unlabeled pool (when using the PU Bagging technique) and also due to the bias that SVM model is biased towards mislabeled data.

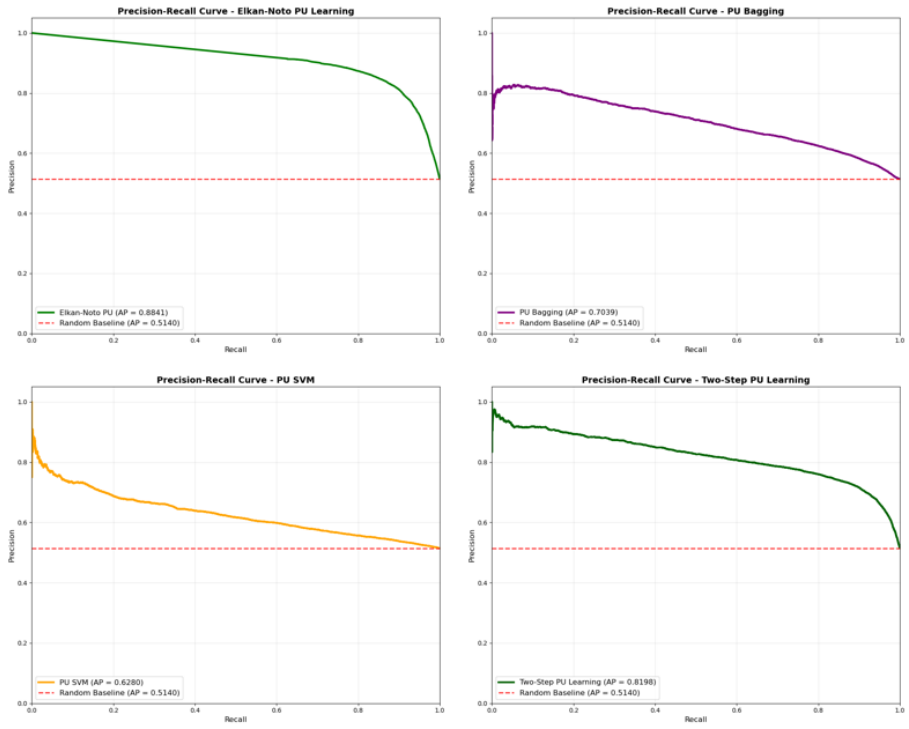

Figure 5.7 PR Curves of PU models

The Precision Recall (PR) curves of the PU models have been plotted in figure 5.6 with a random baseline (average precision, AP, 0.51) plotted as a dashed reference line. In this context, ElkanNoto PU model achieves the strongest PR profile: it has its own level of precision, namely about 0.90 over a very broad range of recall values, with the sharp decline at the end of the curve, resulting in almost an AP of 0.88. This outcome means that in an optimization of the decision threshold, most of the tweets that are considered authentic are actually authentic and at the same time, the model has a high recall of authentic tweets. The strongest performer that follows the Two- Step PU methodology has an AP of approximately 0.82. Even though it has a noticeably lower precision profile compared to ElkanNoto, it still has a very high recall rate of genuine tweets with only a

slight decrease in precision, indicating its high recall of genuine tweets at the cost of a slight decrease in precision.

On the other hand, PU Bagging and PU SVM have significantly subdued PR behaviour. PU Bagging starts at a high level of precision but decreases with increased recall producing an AP of about 0.70, which is an effect due to the model attempting to include more authentic tweets, which is also in line with label noise also being produced by its pseudo-negative samples. PU SVM shows the flattest curve and it has an AP value of about 0.63 just above the random baseline. These measures prove that the ranking scores obtained by this model do not give clear demarcation between genuine and fake tweets: to obtain high recall, significant trade-offs in precision are required. Taken all together, the PR curves support the findings of the ROC analysis: Elkan-Noto and Two-Step PU models are the only models that maintain a favorable trade-off between the precision and recall, with PU Bagging and PU SVM significantly less effective.

5.3 Supervised vs Pu Learning

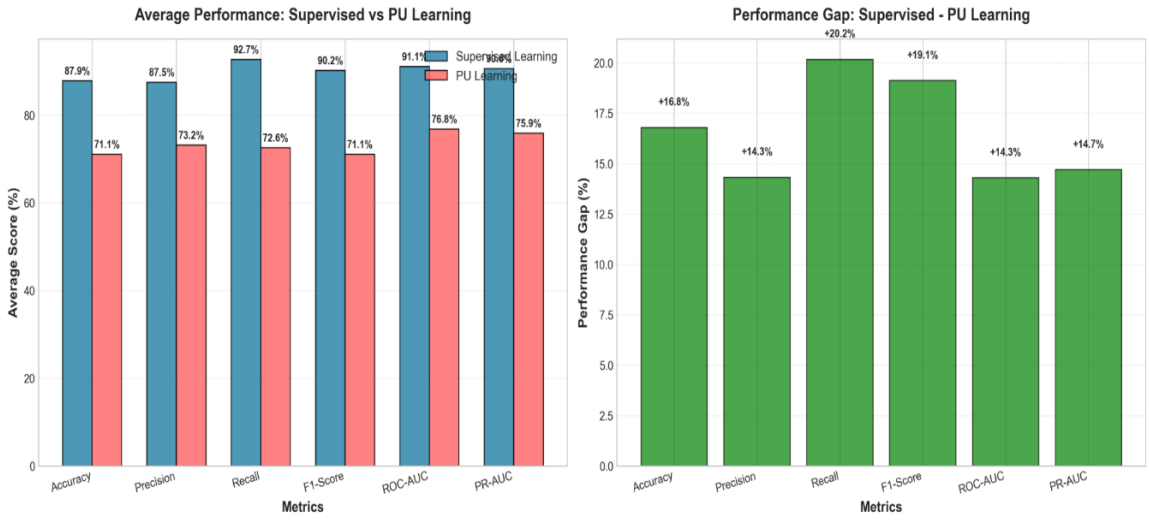

Figure 5.8 Average performance and performance gap

Figure 5.8 (left panel) combines the models by averaging each measure across all the supervised methods and each separately across all PU methods. The collection of supervised learning methods averages 8893 percent in the six measures, compared to the PU learning group that averages around 7177 percent. This affirms that there is a significant, but not disastrous, performance decrease in the move to the PU setting. This observation is then translated into an explicit performance gap by the right-hand panel, on average, the performance of supervised models can be likened to PU models by about 1420 percentage points per metric, with the largest differences recorded in recall and F1-score. As a result, the total error rate of PU models is larger, and there are also problems with achieving high precision and high recall at the same time.

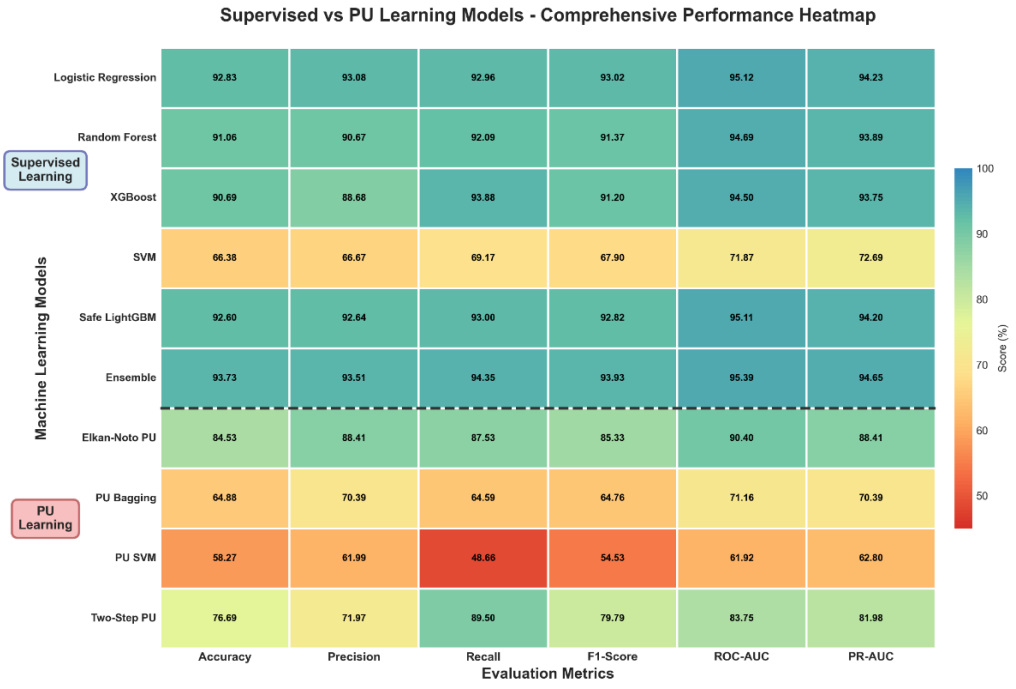

Figure 5.9 Supervised vs PU learning: performance heatmap

Figure 5.9 compares the results of all the supervised models and positive-unlabeled (PU) models on six evaluation metrics. The supervised models, which are the upper half, have consistently high scores: Logistic Regression, LightGBM, and Random Forest have scores that are close to 91-93 packages on most measures and the ensemble method is above that with values that are near to 94-95 packages. The PU models in the second half of the performance space, in turn, occupy a much cooler region of the performance space. Elkan-Noto PU has the best scores with a mid 80 percent score on most metrics; Two-Step PU scores in the high-70 percent to low-80 percent range; and PU Bagging and PU Support Vector Magazine scores in the 60 percent range. The strong change in colour at the dashed boundary not only highlights the performance degradation to be expected with the replacement of fully labelled data with positive-unlabeled data, but also shows that Elkan-Noto and Two-Step PU are the only PU methods that come anywhere close to the supervised performance band.

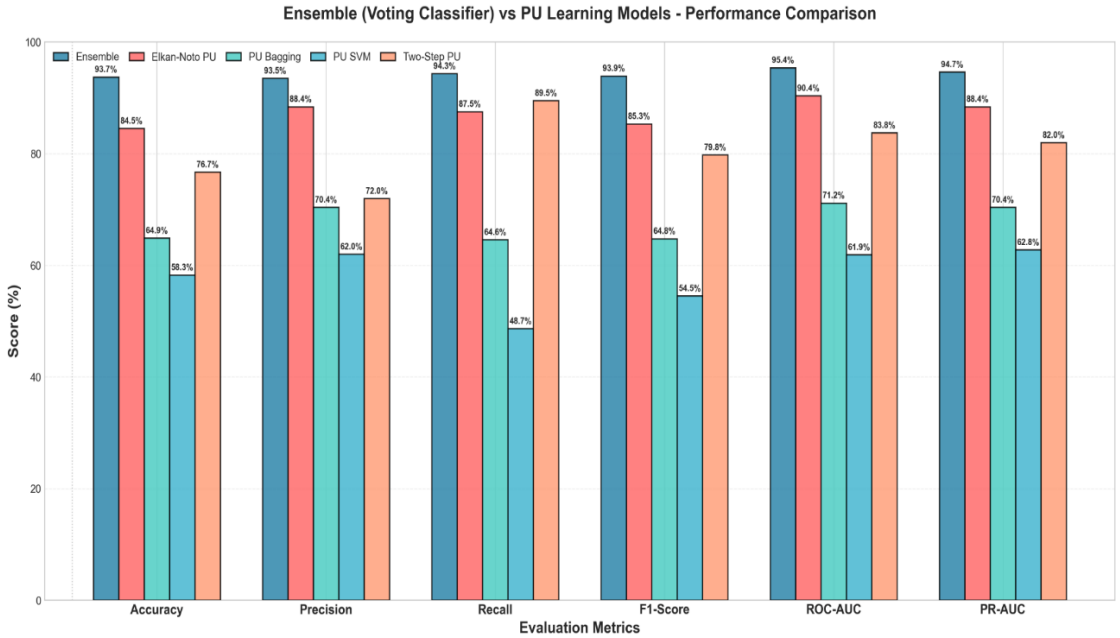

Figure 5.10 Ensemble vs all PU models

Figure 5.10 gives a direct contrast between the best supervised model (the ensemble) and each of the PU methods individually on all metrics. The ensemble scores the highest on every evaluation criterion, which ranges between the mid-90s and upper-90s in ROC-AUC and PR-AUC and a little below in terms of accuracy, precision, recall, and F1. Elkan Noto PU is the closest competitor that is always within 5-10 percentage points of the ensemble, hence justifying its position as the most competitive PU strategy. Two-step PU lags a bit higher but still is within a reasonable range especially with recall. PU Bagging and PU Support Vector Machine are significantly weaker on the whole, and makes it clear that naive methods that hold unlabeled data as negatives lose significantly more information than the more principled PU methods.

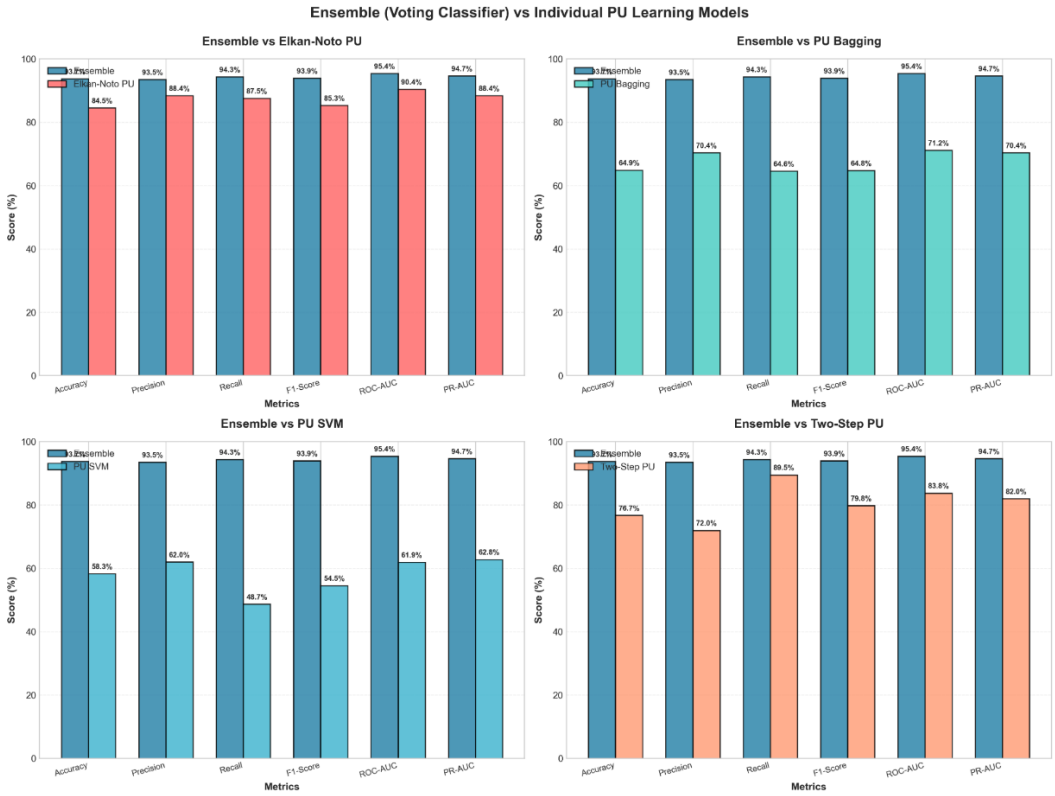

Figure 5.11 Ensemble vs individual PU models (per-panel)

The previous comparison has been broken down into four small multiples, each devoted to one PU method, in figure 5.W. The ensemble method, in all the panels, provides the best results on all metrics compared to the corresponding PU model but the difference in results is varied. Compared to Elkan-Noto PU, these differences are quite small, particularly between ROC-AUC and PR-AUC, which means that a PU model can regain a significant part of the ranking performance of a completely supervised ensemble when it is carefully probabilistically corrected. The commonality gaps are even further apparent when compared to PU Bagging and, more significantly, to PU Support Vector Machine, in which accuracy and F1 decreases by 2030 percentage points, which clearly shows the strong influence of label noise on the unlabeled set.

CHAPTER VI

CONCLUSION AND FUTURE WORK

6.1 Conclusion

The project has been conducted in order to develop and analyse a framework to identify fake news on Twitter/X, focusing especially on the predictive performance as a paradigm shift is made to a realistic Positive Unlabeled (PU) scenario. Using the TruthSeeker-2023 dataset, the investigation involved the entire sequence of processing raw data to analysis, which included such steps as text cleaning, feature engineering, supervised learning, PU set building, and PU learning. The same conventions were used on the labelling: correcttarget = 1 on authentic and correcttarget = 0 on deceptive tweets.

6.1.1 Summary of Achievement

- A complete automated data processing pipeline on the TruthSeeker-2023 Twitter data
- A composite feature space which is a combination of TF-IDF text representations with about 5,000 sparse text variables and 20 numeric/metadata variables which capture user- and tweet-level cues.
- A collection of supervised base models, such as Logistic Regression, Random Forest, XGBoost, Support Vector Machine, a safe variant of LightGBM and a soft-voting Ensemble.
- A moralistic PU learning setup, in which only a non-uniform fraction of genuine tweets are supplied as labelled positives, and the rest of the corpus forms a heterogeneous unlabeled pool.
- Four PU learning schemes based on this setup Elkan Noto PU Logistic Regression PU Bagging PU SVM and a PU Two-Step improvement strategy, using reliable negative examples.
- A deeper analysis and comparison of all the models, using various performance measures, such as accuracy, precision, recall, F1-score, ROC-AUC, PR-AUC and confusion matrices, ROC and PR curve.

6.1.2 Supervised Learning: Decent Fake News Detectors

The supervised models with complete label information of both authentic and fake tweets verified that the features that are picked by TruthSeeker-2023 are highly discriminative. The best single learners turned out to be Logistic Regression and the so-called safe but strong LightGBM model with the accuracy and F1-score of approximately 9293 and the ROC-AUC and PR-AUC of about 0.95. This observation supports two important arguments: Linear decision boundary on TF-IDF features regularised with an L2 penalty and class balancing is enough to capture most of the separability between actual and fake tweets. The performance of well-regularized gradient-boosted trees can be equal to this and at the same time they model nonlinear interactions between textual and numeric signals. Random Forest and XGBoost have continued to perform competitively though with minor trade-offs between recall and precision. Conversely, the SVM baseline produced significantly worse scores, and F1 and AUC scores were in the high 60s to low 70s. This finding highlights the fact that standard text classifiers cannot always be used with high-dimensional sparse data, unless great care is taken in the choice and regularization of kernels, an SVM decision surface can shift significantly away in practice, and can have little to no relationship with the actual boundary.

The highest overall metrics were obtained with the ensemble that used the soft vote between the Logistic Regression and LightGBM and XGBoost with a large number of models: the F1 -score of nearly 94 -percent and the largest ROC -AUC and PR -AUC. Confusion matrices showed that ensembling also reduced false-positive and false-negative rates, and is therefore also shown to be an easy and effective method to stabilize predictions

and exploit complementary advantages, and in effect give an upper bound on the performance of fully supervised on this dataset.

6.1.3 Positive Unlabeled Learning: Real but Difficult

- The next step of the work touched upon an even more realistic limitation: on work platforms, a restricted set of existing verified authentic tweets is usually present, as opposed to an enormous amount of unlabeled material, and no cleanly labelled false tweets. In order to replicate this situation, the training set under supervision was restructured to resemble a real PU environment by keeping only a fraction of genuine tweets as labeled positive and forcing all the rest of the tweets (both fake and unlabeled genuine) to the unlabeled pool.

In this set-up four PU techniques were adopted and tested:

Elkan- Noto PU Logistic Regression.

- Obtained a precision of about 84.5, F1 -score of about 85.3 and ROC -AUC of about 0.90. The approach restores a significant part of the supervised logistic regression result by acquiring a probabilistic model of labeled versus unlabeled and reconditioning by the probability of being labeled.

PU Learning Two-step with trustworthy negatives.

- Achieved a recall of approximately 89.5, and an F1-score of 79.8, and reached a significantly high accuracy of approximately 76.7 per cent on authentic tweets. The first one is to use a temporary classifier to determine the good negative examples in the unlabeled pool, and then learn the final supervised model using the good positives and the detected negatives. This design inherently favours high recall on the positive class which works well in the context where missing authentic content is very costly.

PU Bagging

- Performed moderately with an accuracy and F1 -score range of mid-60%. Through repeated re-sampling of the unlabeled data as pseudo-negatives, every base classifier is trained on a heavily contaminated negative class; ensemble averaging fails to entirely counter the noise induced.

PU SVM

- Gave the worst performance, with accuracy of approximately 58 03 and F1 of about 55 03, and ROC -AUC of slightly over 0.6. The biased SVM is very vulnerable to the label noise in the unlabeled pool and finds it difficult to determine a stable separating hyperplane in the TF feature space of the IDF feature space.

These conclusions were supported by confusion matrices, ROC curves, and PR curves: Elkan-Noto had the best separation of classes, Two-step PU sacrificed some of its accuracy to very high recall, whereas PU Bagging and PU SVM had very strong off-diagonal errors and lower ranking.

6.1.4 Supervised versus PU: quantified trade-offs

The main implication of this study is one to draw a direct, quantitative comparison between fully supervised learning and PU learning when applied on the same dataset and feature representation. An analysis of the results of all models revealed that: Supervised models outperformed PU models by an average of 1520 percentage points in terms of accuracy, precision, recall, F1-score, ROC-AUC and PR-AUC. The difference in performance was minimal between ElkanNoto and Two Step PU, which were close to the performance of the ensemble especially in terms of the ranking (ROC-AUC and PR-AUC). Nonetheless, even with the gap, PU models remained recorded to reach practical meaningful performance particularly the Elkan-Noto models with high precision and recall when decision thresholds were optimised. These results define a realistic trade-off: in cases when detailed labels are available, self-supervised groups are the best option and it has a high performance limit. Nevertheless, in the environments where the cost of obtaining credible negative labels is high or uncertain, i.e. the situation that one faces when fake-

news detection is considered, the best PU techniques can provide a strong fall-back mechanism, offering decent detection performance with significantly worse supervision.

6.1.5 Key takeaways

Overall, this research study indicates that:

- A carefully designed feature estimation that combines TFIDF text vectors with metadata produces discriminatory functions of a high degree of reliability between authentic and misleading political tweets.
- An assembly of strong supervised learners provides a reliable detector that reduces both occurrences of misclassification.
- In the case of limited labelled data, Positive Unlabeled learning provides a conceptually relevant way to utilize an unlabeled stream with Elkan-Noto and Two-step PU achieving significant superiority over naive PU Bagging or biased SVM strategies.
- In addition to giving quantitative performance data regarding the TruthSeeker-2023 dataset, the research clarifies the practical gap between supervised and PU learning when it comes to fake-news detection, and provides clear evidence on the amount of information that is lost, and the levels to which it can be retrieved under the limited conditions of supervision.

6.2 Future Work

In spite of the fact that, despite providing a complete and efficient pipeline to detect fake news on Twitter based on the use of TruthSeeker-2023, this project has a number of opportunities towards extension and strengthening the work.

6.2.1 Richer Representations and Multi Modal Signals

The existing system is based on TFIDF characteristics besides twenty numeric or metadata features. The next obvious step is to move to contextual embeddings based on transformer models like BERT, RoBERTa or domain specific variants which have continued to perform better in fake-news detection and misinformation tasks compared to

the traditional bag-of-words methods. Future studies would be to substitute TF–IDF with sentence-level embeddings forming after a fine-tuned transformer without modifying the current numeric features. Further, investigating the multimodal fake-news-detection by adding images, URLs (linked articles), and user-level information (network structure, profile descriptions), as the latest multimodal FND models do, would see whether the PU and supervised pipelines would hold similar performance when the feature space is significantly enriched and expressed.

6.2.2 Bias in Dataset, Robustness, and Generalisation

Recent studies have found that TruthSeeker-2023 can include spurious correlations, e.g. the name of a particular politician or other political term being highly correlated to a certain label. This should be explicitly tackled in the future by using any of the debiasing strategies, including rebalancing, adversarial training, or counterfactual data augmentation, to minimise the use of artefacts. The models ought to be tested in cross-dataset and cross-domain (e.g. training on TruthSeeker-2023 and testing on other Twitter or news datasets) to determine the extent to which the learned decision boundary can be generalised to other collections. Moreover, resilience ought to be investigated in topic shift and concept drift (e.g., new political events or health crises) with time-based division or streaming verification.

6.2.3 More Sophisticated PU Learning Algorithms

This thesis applied classical PU strategies (Elkan -Noto, PU Bagging, PU SVM, Two-Step PU). In the modern PU literature, there are some extensions, which would easily be compatible with the existing configuration. Non negative PU (nnPU) learning presents

a non negative risk estimator that allows flexibility of deep models, and reduces overfitting. PU with biased negatives (PUbN) uses a small non-representative sample of known fake tweets as biased negatives in order to improve the classifier. The latest approaches that use learned negative selectors like PULNS automatically detect effective negatives in the unlabeled pool. PU models that include instance-dependent and label-mechanism-aware are instance-dependent and explicitly model the annotation process of tweets. Comparison of these methods with Elkan -Noto and Two-Step PU on TruthSeeker-2023 would further help understand which PU assumptions are most consistent with social-media fake news.

6.2.4 Online, Early Detection and Temporal

The goal of fake news spreading is extremely high, and early detection is sometimes more important than post-hoc detection that is correct. The extension in the future can use the tweet stream as a time-series where the early periods are trained and then tested on the later periods as a way of evaluating performance in the face of temporal drift. Online or incremental learning methods might be used to update models with new labelled data instead of retraining them all new data. A combination of temporal features with transformer-based fake-news detectors would be aimed at flagging fake information early, as it occurs.

6.2.5 Interpretability and Human-in-the-loop Systems

Finally, a practical implementation of the framework would have better interpretability and human interaction. Explainability tools (e.g. SHAP values or attention visualisations) can provide information about which words, phrases or metadata fields cause any prediction, helping an analyst understand why a tweet is flagged. There are

different operating points (high-precision vs high-recall mode) that the models should be calibrated to and made optional depending on whether the system is used in automated filtering mode or decision support mode. Both supervised and PU models can be directly improved by integrating active learning, in which the model chooses the uncertain tweets to be viewed by hand, adding feedback in the form of manual labels as new labelled positives or, in case of curation, biased negatives.

In general, the present study forms a solid foundation in fake-news detection on Twitter with full-label and positive-no-label whatsoever. The steps above give a guide on how to make this thesis more substantial, more robust, and more realistic so that it can respond to emerging subjects and surfaces and in more intricate ways of misinformation.

CHAPTER VII

REFERENCES

[1] **Shu, K., Sliva, A., Wang, S., Tang, J., & Liu, H.** (2017). Fake News Detection on Social Media: A Data Mining Perspective. *ACM SIGKDD Explorations Newsletter*, 19(1), 22–36.

[2] **Elkan, C., & Noto, K.** (2008). Learning classifiers from only positive and unlabeled data. In *Proceedings of the 14th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD 2008)*.

[3] **Mordelet, F., & Vert, J.-P.** (2014). A bagging SVM to learn from positive and unlabeled examples. *Pattern Recognition Letters*, 37, 201–209.

[4] **Liu, B., Lee, W. S., Yu, P. S., & Li, X.** (2003). Building text classifiers using positive and unlabeled examples. In *Proceedings of the 3rd IEEE International Conference on Data Mining (ICDM 2003)*.

[5] **Kaboutari, A., & Bagherzadeh, J.** (2014). An Evaluation of Two-Step Techniques for Positive–Unlabeled Learning in Text Classification.

[6] **Allcott, H., & Gentzkow, M.** (2017). Social Media and Fake News in the 2016 Election. *Journal of Economic Perspectives*, 31(2), 211–236.

[7] **Gisondi, M. A., Barber, R., Faust, J. S., Raja, A., Sawyer, K. N., & Centor, R. M.** (2022). A Deadly Infodemic: Social Media and the Power of COVID-19 Misinformation. *Journal of Medical Internet Research*, 24(2), e35552.

[8] **Denniss, E., et al.** (2025). Social media and the spread of misinformation. *Health Promotion International*, 40(2), daaf023.

[9] **Kaiser Family Foundation (KFF).** (2021). *COVID-19 Misinformation is Ubiquitous: 78% of the Public Believes or is Unsure About at Least One False Statement…* KFF Report.

[10] **Shu, K., Mahudeswaran, D., & Liu, H.** (2019). FakeNewsTracker: a tool for fake news collection, detection, and visualization. *Computational and Mathematical Organization Theory*, 25, 60–71.

[11] **Chalehchaleh, R., Salehi, M., Farahbakhsh, R., & Crespi, N.** (2024). BRaG: a hybrid multi-feature framework for fake news detection on social media. *Social Network Analysis and Mining*, 14.

[12] **Selyukh, A.** (2013). Hackers send fake market-moving AP tweet on White House explosions. *Reuters*.

[13] **Moore, H., & Roberts, D.** (2013). AP Twitter hack causes panic on Wall Street and sends Dow plunging. *The Guardian*.

[14] **Karppi, T., & Crawford, K.** (2015). With one false tweet, computer-based Hack Crash led to real panic. *Theory, Culture & Society*.

[15] **Associated Press.** (2013). Hackers compromise AP Twitter account. *AP News*.

[16] **Background reporting on the "Pizzagate" conspiracy** and its spread across online platforms, 2016–2017. (Various news and fact-checking sources).

[17] **Debies-Carl, J. S.** (2017). Pizzagate and Beyond: Using Social Research to Understand Conspiracy Legends. *Sociology Faculty Publications*.

[18] **Analyses by major news and fact-checking organizations** documenting the evolution and debunking of the Pizzagate conspiracy, 2016–2018.

[19] **Court documents and news reports** on the prosecution and sentencing of Edgar Maddison Welch in connection with the Comet Ping Pong incident, 2017.

[20] **Shen, Y., et al.** (2023). Fake News Detection on Social Networks: A Survey.

[21] **Gong, S., et al.** (2023). Fake News Detection Through Graph-based and Deep Learning Techniques: A Systematic Review.

[22] **Kuntur, S., et al.** (2024). Comparative Analysis of Graph Neural Networks and Transformer Models for Fake News Detection.

[23] **Bekker, J., & Davis, J.** (2020). Learning from Positive and Unlabeled Data: A Survey.

[24] **Liu, B., et al.** (2003/2005). Learning from Positive and Unlabeled Examples: A Survey.

[25] **de Souza, M. C., et al.** (2022). A Network-Based Positive and Unlabeled Learning Approach to Detecting Fake News.

[26] **de Souza, M. C., et al.** (2024). Keywords Attention for Fake News Detection Using Few Labels and Positive–Unlabeled Learning.

[27] **Shu, K., et al.** (Year). Additional works by Shu et al. on fake news detection and social media mining