ADDRESSING CHALLENGES OF TWITTER FEATURE ENGINEERING FOR MACHINE LEARNING IN DIFFERENT DOMAINS

by

Ahmed Balfagih

Submitted in partial fulfillment of the requirements
for the degree of Doctor of Philosophy

at

Dalhousie University
Halifax, Nova Scotia
July 2023

Table of Contents

List of Tables

List of Figures

Abstract

Feature engineering is one of the essential steps in machine learning. It helps to create a better data model and provides reasonable learning results. In natural language processing, feature engineering becomes even more critical if we can generate many features and feed them to our model. Twitter data became an important source of written natural language used to train and test machine learning models, and preparing features for these models is an important step. In this study, we examine different feature engineering approaches to three tasks on Twitter. In all these three studies, we compared the efficiency of generating word embedding features, with the generated n-grams of the Tweets, by applying machine learning algorithms and noticing the experiment's accuracy. First, we generate features of tweets for financial forecasting; second, we generate features for spam recognition on tweets; and finally, we generate features to learn different dialects of a language. In the financial forecasting task, we explore the relationship between the Twitter feed on Bitcoin, its sentiment analysis, and its price prediction. Two approaches are used to generate features from tweets: tweet embedding and n-gram modeling. Using a time-series approach, we found a partial correlation between Bitcoin price fluctuation and sentiment class accuracy fluctuations using different machine learning algorithms. In the spam recognition task, we apply machine learning techniques to identify spam on tweets in the Arabic language. We use two feature generation techniques: n-grams and Word2Vec embeddings. The experimental results show improvement from using Word2Vec embeddings over n-grams in more balanced datasets versus the more unbalanced ones. Finally, in the dialects recognition task, we used tf-idf n-grams, AraVec feature embedding, and BERT fine-tuned generated features, on Arabic tweets to locate where the tweet came from in the Arabian region based on dialect learning. The results showed that BERT-Base pretrained language models are more powerful for this study than other methods, specifically ARABERT and MARBERT.

List of Abbreviations and Symbols Used

AI	Artificial Intelligence
ARABERT	aubmindlab/bert-base-arabert
ARABERTV2T	aubmindlab/bert-base-arabertv02-twitter
ARBERT	UBC-NLP/arbert
ASAFAYA	asafaya/bert-base-arabic
ASDL	Arabic Spam Detecting Lexicon
BERT	Bidirectional Encoder Representations from Transformers
Bi-LSTM	Bi-directional Long Short-Term Memory
BoW	Bag of Words
BTC	Bitcoin
CBOW	Continuous Bag-of-words
CNN	Convolutional Neural Networks
DA	Dialectal Arabic
DL	Deep Learning
DT	Decision Tree
GPT	Generative Pre-trained Transformer
idf	inverse document frequency
KNN	k-Neareast Neighbours
LASER	Language-Agnostic SEntence Representations
LR	Logistic Regression

LSTM	Long Short-Term Memory
MADAR	Multi-Arabic Dialect Applications and Resources
MARBERT	UBC-NLP/marbert
MCXENT	multi-class cross-entropy
ML	Machine Learning
MLM	Masked Language Modeling
MLP	Multilayer Perceptron
MSA	Modern Standard Arabic
MULTIDB	bashar-talafha/multi-dialect-bert-base-arabic
MultinominalNB	Multinominal Naive Bayes
NADI	Nuanced Arabic Dialect Identification
NB	Naïve Bayes
NLP	Natural Language Processing
RATS	R-Auto Tweet Sentiment
RBF	Radial Basis Function
RBFN	Radial Basis Function Network
ReLU	Rectified Linear Unit
RF	Random Forest
RNN	Recurrent Neural Network
RoBERTa	Robustly Optimized BERT Approach
SG	skip-grams
SSWE	Sentiment-Specific Word Embeddings
SVM	Support Vector Machine
T5	Text-to-Teaxt Transfer Transformer
tf-idf	term frequency-inverse document frequency

TP	True Positive
Weka	Waikato Environment for Knowledge Analysis
WiSARD	Weightless Neural Network
Word2Vec	Word-to-Vector Embedding
WTFM	Weighted Text Feature Model

Acknowledgements

I would like to express my gratitude to my supervisor Dr. Vlado Keselj of the Computer Science Department at Dalhousie University. The door to Dr. Keselj's office was always open whenever I had a question about my research or writing, and for his understanding, elegant treatment, useful comments, remarks, engagement through the learning process, and support of this Ph.D. thesis.

I would also like to acknowledge my thesis defense committee Starting from Dr. Srinivas Sampalli, Dr. Qigang Gao, and Dr. Malcolm Heywood for being my internal readers of this thesis, and Dr. Malek Mouhoub for being the external reader of this thesis, and I am gratefully indebted to them for their very valuable comments on this thesis.

I must express my very profound gratitude to my parents, Mustafa and Maha Balfagih, for providing me with unfailing support, and continuous encouragement throughout my years of study, and through the process of researching and writing this thesis, with their patience. This accomplishment would not have been possible without them. I dedicate my success to them and I hope they are always proud of me.

Finally, I would not forget to thank my university, King Abdulaziz University, which gave me the opportunity of conducting my graduate studies in Canada, with their unlimited material and moral support.

Author
Ahmed Mustafa Balfagih

Chapter 1

Introduction

1.1 Background

In big data analytics, feature engineering is essential. Data is required for machine learning and data mining algorithms to enhance performance. When there are few features to represent the underlying data objects, little can be accomplished, and the quality of the available features primarily determines the quality of the results of those algorithms. In Natural Language Processing (NLP), feature engineering techniques are essential for text data learning. In machine learning, feature engineering is the process of transforming raw data into a more understandable format, and extracting, generating, or re-forming features from existing features [75]. The area of feature engineering encompasses a wide range of challenges and tasks. The most common challenges and activities are feature transformation, feature creation and extraction, feature selection, automatic feature engineering, and feature analysis and evaluation [56].

- **Feature transformation** is concerned with creating new features from existing ones; this is frequently accomplished via the use of mathematical mappings. The BMI index, for example, is a feature created by transforming features using a mathematical formula.

- **Feature generation** is concerned with the creation of new features that are not always the result of feature transformation. The feature generation category also includes several domain-specific approaches to defining features. Feature generation methods can be generic automatic ones, in addition to domain-specific ones. Patterns mined from data may also be used to generate new features.

- **Feature selection** is about picking a small number of features from a significant number of options of them. Because of the reduced feature set, some machine learning and data analysis techniques are computationally possible. Feature selection might potentially increase the quality of the algorithms' results.

- **Automatic feature engineering** is about generic techniques for automatically creating a large number of features and then choosing an effective subset of those features.

- **Feature analysis and evaluation** is concerned with the analysis and evaluation of the usefulness of features and feature sets. This is usually a part of the feature selection process.

By looking at what has been done with text data, we may better understand feature engineering. For text data, the previous research has been mostly organized around a specific set of topics. Beyond single words, structural representation of text, semantic structure features, latent semantic representation, explicit semantic representation, word embeddings for text representation, and context-sensitive text representation are all examples of topics of feature generation for text as strings. Automatic feature creation is a common method, with Word2vec and DeepLearning-based approaches as examples [56]. Feature Engineering has always been a part of Machine Learning. It is one of the most important tasks for those working in data science and natural language processing. Because text data differs from structured tabular data, machine learning requires developing feature techniques.

1.2 Feature Generation

Feature generation is part of the data preprocessing phase and is the act of developing new features from one or more existing features, or text data, for use in statistical analysis or as predictors based on domain knowledge. Feature construction, feature extraction, and feature engineering are all terms used to describe feature generation [54]. This process adds new information that can be accessed during model development, resulting in a more accurate model [51]. Unless otherwise noted, feature generation and any synonyms such as feature construction from raw data [86], creating a mapping to convert original features to new features [96], and building new features from one or more features [101], shall be considered in the context of this thesis to *create* new features.

Feature generation of text data has recently become a significant challenge in many studies, particularly where articles or paragraphs must be classified based on their size, writing style, author, sentiment analysis, or other text corpus characteristics. These "fundamental features" do not always return the best classification accuracy or the desired results of the machine learning task. To address this, a variety of feature generation approaches have been created.

The Word2vec feature generation algorithm, for example, is a collection of algorithms for generating numerical feature vectors from words. They reconstitute linguistic contexts using shallow, two-layer neural networks. Word2vec takes a massive corpus of text as input and generates a vector representation for each unique word in the corpus, generally with several hundred dimensions. When words are used in comparable situations, their word vectors are similar. The Continuous Bag-of-words (CBOW) architecture trains a model to predict the current word based on the Bag of Words' context, whereas the Skip-gram architecture trains a model to predict surrounding words given the current word. Deep Learning approaches turn high-dimensional data into a low-dimensional representation by training a multilayer neural network to rebuild high-dimensional input vectors with a tiny center layer [56].

Feature generation can enhance machine learning algorithms for text categorization based on domain-specific and common-sense knowledge. This knowledge is represented using publicly available ontologies that contain hundreds of thousands of concepts, such as the Open Directory. These ontologies are further enriched by several orders of magnitude through controlled Web crawling. Evgeniy Gabrilovich and Shaul Markovitch in their study, prior to text categorization, used a feature generator to analyze the documents and map them onto appropriate ontology concepts. This created a set of generated features that augment the standard bag-of-words [62]. Feature generation is then done through contextual analysis of document text, implicitly performing word sense disambiguation. Coupled with the ability to generalize concepts using an ontology, this approach addresses two important challenges of natural language processing—synonymy and polysemy. Categorizing documents with the aid of knowledge-based features leverages information that cannot be deduced from the documents alone. Experimental results confirm improved performance, breaking through the plateau previously reached in the field [62].

Most classification algorithms receive as input a set of attributes of the classified objects. In many cases, however, the supplied set of attributes is not sufficient for creating an accurate, succinct, and comprehensible representation of the target concept. To overcome this problem, researchers have proposed algorithms for the automatic construction of features. The majority of these algorithms use a limited predefined set of operators for building new features. In a study made by Markovitch and Rosenstein, they proposed a generalized and flexible framework that is capable of generating features from any given set of constructor functions. These can be domain-independent functions such as arithmetic and logic operators, or domain-dependent operators that rely on partial knowledge on the part of the user [101]. Markovitch and Rosenstein described an algorithm that receives as input a set of classified objects, a set of attributes, and a specification for a set of constructor functions that contains their domains, ranges, and properties. The algorithm returns a set of generated features that can be used by standard concept learners to create improved classifiers. The algorithm maintains a set of its best-generated features and improves this set iteratively. During each iteration, the algorithm performs a beam search over its defined feature space and constructs new features by applying constructor functions to the members of its current feature set. The search is guided by general heuristic measures that are not confined to a specific feature representation. The algorithm was applied to a variety of classification problems and was able to generate features that were strongly related to the underlying target concepts. These features also significantly improved the accuracy achieved by standard concept learners, for a variety of classification problems [101].

1.3 Problem Statement

One of the most important problems in machine learning for Twitter data is how to generate features: the number of features related to the tweet is limited due to text size and the fact that Twitter only has the “like” feature. In comparison, Facebook’s text is not limited in length and it has more sentiment-related features that may help analyze the text such as like, love, laugh, anger, care, sad, and wow — all of which evoke specific sentiments.

Therefore, generating features on Twitter data is non-trivial; Many experiments are needed to get a better understanding of how to generate quality features for different machine-learning tasks.

In our study, we propose applying three different tasks of feature generation for three areas. The first study examines feature generation for financial forecasting using English tweets. The second task is about generating features of tweets for spam recognition, and this study is on tweets in Arabic. The third task explores generating features for language and dialect recognition using Arabic tweets.

The three case studies share a central theme of tackling the challenges of feature engineering in various domains using Twitter data. While each study focuses on a specific domain, they all examine the effectiveness of feature engineering techniques when comparing n-gram features quality with word embedding vector features derived from tweets.

In the first study on feature engineering for financial forecasting, the researchers utilize Twitter data related to Bitcoin markets and aim to predict market trends or stock prices. They explore how different feature engineering approaches, including n-grams and tweet embeddings, can be used to capture relevant financial information and improve forecasting accuracy using time-series.

Similarly, the second study on feature engineering for spam recognition focuses on detecting and filtering spam messages on Twitter in Saudi Arabian trend, in Arabic language. By comparing the performance of different feature engineering methods, including n-grams and word embeddings, to identify the most effective approach for accurately classifying spam tweets.

The third study addresses the challenge of feature engineering for Arabic dialect recognition on Twitter. It involves distinguishing between different dialects of the Arabic language in tweets. Here, the researchers evaluate the performance of n-gram features, AraVec word embeddings, and pretrained language models (BERT) features, to determine which approach yields superior results in accurately identifying the Arabic dialect used in each tweet.

Although the studies cover different domains, they share a common focus on efficiently extracting meaningful features from Twitter data. By comparing the quality and performance of n-grams and word embeddings as features, the studies aim to contribute to the understanding of which approach is more effective in their respective domains. This central theme of feature engineering and the comparison of n-gram features with word embedding vectors across different domains unifies the studies and lends to a broader understanding of the importance of feature selection and representation in Twitter-based machine learning applications.

1.4 Contributions

Twitter data is considered a key data source due to its short text and public availability. Hundreds of millions of tweets are sent on the platform each day. The main contribution of this thesis is comparing the results of feature engineering approaches by applying different methods to each study. Another contribution is highlighting the differences in applying feature engineering to diverse tasks ranging from financial forecasting to spam recognition to language dialect identification. As well, this research uses the diversity of natural language of English and Arabic. Research on these topics were published with the following conferences and journals:

- **Evaluating Sentiment Classifiers for Bitcoin Tweets in Price Prediction Task** paper that is published in 2019 IEEE International Conference on Big Data (Big Data) [33].
- **N-gram and Word2Vec Feature Engineering Approaches for Spam Recognition on Some Influential Twitter Topics in Saudi Arabia** paper is published at the ICISDM 2022: the 6th International Conference on Information System and Data Mining and published by Association for Computing Machinery (ACM) [31].
- Journal article **N-gram and Word2Vec Feature Engineering Approaches for Spam Recognition on Some Influential Twitter Topics in Saudi Arabia paper** is published in JAIT: Journal of Advances in Information Technology [32].
- **Twitter Arabic Dialect Identification using AraBERT** is a short paper and presentation that is published at the 39th IBIMA conference as an in-proceeding short paper, and the presentation is also published as a YouTube video on IBIMA's official channel as presented at their virtual conference [29].

- **Twitter Arabic Dialect Identification using Different Feature Engineering Models** is published at the 40th IBIMA conference as a full paper, with an invitation for journal submission of the paper, and published YouTube video on IBIMA's official channel as presented at their virtual conference [30].
- Journal article **Twitter Arabic Dialect Identification using Different Feature Engineering Models** was accepted to be published at "The Communications of the IBIMA", by IBIMA Publishing.

1.5 Outline

The rest of the thesis is organized as follows:

- **Chapter 2** presents the general background and literature review on related and common readings among the three thesis studies.
- **Chapter 3** presents the first study about "Feature Engineering for Financial Forecasting of Bitcoin Price", including its specific literature review and related work, methodology, machine learning methods, price fluctuation prediction methodology, time-series, proposed framework, experiment results with a discussion, and conclusion.
- **Chapter 4** presents the second study about "Feature Engineering for Spam Recognition of Arabic Tweets", including its specific background and related work, methodology and data modeling, discussing the experimental results, and conclusion.
- **Chapter 5** presents the third study about "Feature Engineering for Arabic Dialect Recognition", including its specific background and related work, data description and modeling, proposed system, discussion of the experimental results, and conclusion.
- **Chapter 6** presents the conclusions and future work directions.

Chapter 2

Literature Review

2.1 Social Media

Social media technology is a phenomenal development on the Internet accompanied by the emergence of many Web 2.0 technologies. Generally, social media represent a big leap from the past when it was limited to communicating very small amounts of information and with greater control of data managers to the present of connecting through the web interactively. Social media is becoming more important in data analysis. Social media also provides many opportunities, including information sharing among all subscribers of the network with the possibility of free and direct interaction on social networking sites. Furthermore, it allows participants to determine what they like or dislike, or what information they are sharing on their personal profiles, which gives an effective measurement of the popularity of products, activities, opinions, and trends and gives useful data to make statistical studies related to this data.

The term social media can be described as the use of the Internet and mobile technologies to turn communication into more interactive dialogue. Andreas Kaplan and Michael Haenlein defined social media as "a group of Internet applications that build on the foundations of ideology and technology from Web 2.0, which allows the creation and exchange of data" [87]. Trisha Baruah listed many different forms of social media technology including Internet forums, social blogs, microblogging, wikis, social networks, podcasts, photographs or pictures, videos, rating, and social bookmarking. Technologies include activities such as blogging, wall-posting, picture-sharing, vlogs, music-sharing, crowdsourcing and voice-over IP, rating, liking, and voting. Social networks also can integrate many platforms [34].

Facebook is a useful and effective social media tool. This social network can be accessed free of charge and allows users to join networks organized by geographical place or point of work, educational background, or region, in order to communicate with others and interact with them. Users can add friends to their friends' list and send them messages, also update their profiles and define their own friends into different categories, each category can have different privacy preferences. This social network became an essential application that users use either on their PCs or their smartphones for its features of making friends, finding friends, and figuring out friends' preferences. These features made Facebook one of the indispensable stations for e-commerce advocates and affiliated companies due to the huge number of participants on the website (over one billion users around the world) [60]. Alexa's website indicates that Facebook is the second most popular website in the world [16].

Twitter, which is considered a microblogging social media website is another common social networking website. Twitter is a good tool to express short ideas, give short feedback, and share important activities such as retweets and hashtags, which are two of the most interesting tools to measure trends and the popularity of opinions. That's why companies and organizations are eager to have an account on Twitter, to know the number of followers, and retweets for each tweet, and to spread the news on the high trend hashtags [157].

Instagram, another important free photo-sharing social network, allows users to capture an image and add a digital filter to appear like professional captures. A user can share in a variety of social networking services, as well as the Instagram network itself. In the beginning, Instagram was only supported on iOS devices: iPhone, iPad, and iPod Touch. Now, Instagram supports Android smartphones. The application provides the features of comments and likes and now has developed shooting 15-second intermittent videos [82].

YouTube is a well-known website for videos, allowing users to upload, watch, share, and rate video clips for free. YouTube has now become the third-ranked website in the world by Alexa website rating. Because of the high demand on the website and the high number of users, YouTube has become a good target for companies to make their own channels and spread their advertisements before users can stream a video, and has encouraged people to subscribe to their channels to inform subscribers about new videos uploaded [167].

Analyzing social media content can be a valuable tool for businesses, organizations, and individuals. It has many benefits, such as understanding audience sentiment, by analyzing conversations and posts, businesses and organizations can gain insights into how their audience feels about their products, services, and brands. Another benefit is the ability to identify trends, it can help businesses identify emerging trends among their audience, which can inform their marketing and business strategies. Also, they can measure their social media engagement, such as the number of likes, shares, and comments on their posts. Social media analytics can provide businesses with insight into what their competitors are doing, such as their engagement rate, the types of content they share, and their messaging. Furthermore, by monitoring social media conversations and responding promptly to customer inquiries and complaints, organizations can improve their customer services and build stronger relationships with their customers.

In summary, social media is the information systems treasure of this century. The richness of data is growing rapidly, in many fields and topics. Social media data can be related to finance, business, health, politics, or even entertainment that can be used in data analysis to discover knowledge. Choosing the right social media to be analyzed relies on the goals of the study, and applying machine learning techniques to the analyzed social media text data can help to predict future events, such as financial forecasting, or help in detecting language features to be used in classifying the text type, such as identifying the dialects or recognizing the spams.

2.1.1 Twitter

Twitter is an American social networking site that provides a micro-blogging service that allows its users to send “tweets” that will get retweets and/or likes from other Twitter users, with a maximum length of 280 characters per message. This is done directly through Twitter, or by sending a short text message, instant chat programs, or applications provided by developers such as Facebook and others. These updates appear on the user’s page, and the user’s followers can read them directly from their timeline or visit the user’s profile [100].

Twitter was founded in March 2006 by Jack Dorsey, Noah Glass, Biz Stone, and Evan Williams and was actually launched in July of the same year [147]. The site was very popular all over the world. By 2012, the site had more than 100 million users, posting more than 340 million tweets per day, while the search queries reached 1.6 billion per day [124]. In 2013, Twitter was one of the top ten most visited websites in the world [2], and as of 2016, Twitter has more than 319 million monthly active users [4]. Its popularity skyrocketed during the 2016 US Presidential Election as the site truly proved to be the source for breaking news with more than 40 million election-related tweets posted as of 10pm ET [109].

The site was distinguished thanks to the 140 characters that it allows during the publication of the tweet, which is a service almost similar to the SMS service, and it also helps to use the method of shortening the message. At the same time, other sites appeared that help in shortening links, such as Bitly and many others, in addition to hosting and accommodating multimedia, especially those whose address exceeds 140 characters [125]. Since June 2011, Twitter users have been relying on URL shortening services when posting tweets to avoid going beyond the 140-character limit [98]. In 2016, Twitter announced that media such as photos and videos can now be shared without affecting the 140-character limit. Furthermore, attachments and links became also no longer part of the minimum character limit [125]. By 2017, Twitter has increased the Tweets to 280 characters [5].

Twitter trends or “Trending Topics” refer to those topics that are trending more than others, or rather, the word that appears frequently in users’ tweets [11]. Some topics become popular for two reasons. The first is related to the event, which prompts people to talk about that topic and give their opinion on it, while the second is related to the concerted efforts of dozens of users to bring the topic to the list of popular topics in the world. These trending topics help site users understand what is happening in the world and what people think about it. Figure 2.1 shows the anatomy of Twitter tweet.

Twitter has a Developer Platform that enables developers to harness the power of Twitter’s open, global, real-time, and historical platform within their own applications. The platform provides tools, resources, data, and API products for you to integrate, and expand Twitter’s impact through research, solutions, and more. The Twitter API makes it possible for programs to access Twitter in unique and advanced ways. Use Twitter’s fundamental features, including Tweets, Direct Messages, Spaces, Lists, users, and more. Using API, Twitter data can be collected to be used for Natural Language Processing, and text analytic applications and studies [3].

At the end of 2022, Twitter is acquired by the American businessman Elon Musk,

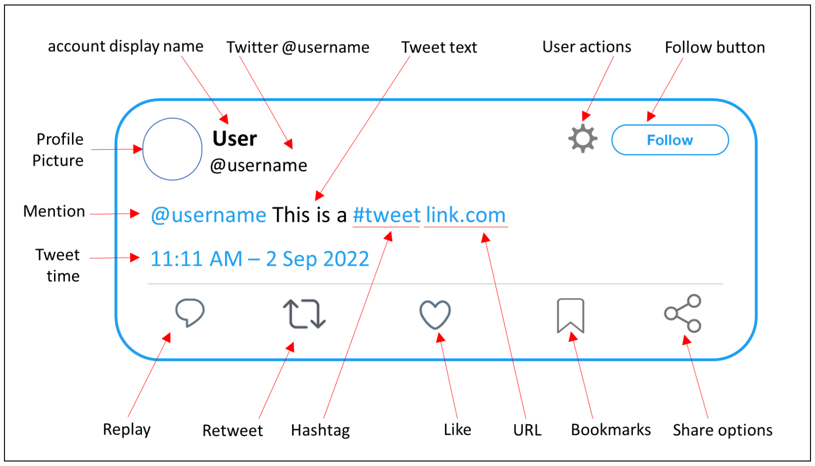


Figure 2.1: The anatomy of tweet

who is leading a process of changing the properties of Twitter that it's known for. Musk recently announced that Twitter would once more raise the character limit to 4,000, which sparked speculations regarding the nature of Twitter and whether it will remain a micro-blogging platform. While Twitter now also supports longer-form content through its threads feature, by subscribing to Twitter Blue paid membership, the platform's main focus is still on micro-blogging [50].

Feature Generation in Twitter

With the rise of Big Data, there has been a surge in demand for organizations and data scientists to extract information from non-traditional data sources. According to research, roughly 80% of data is unstructured text data; therefore, text analytics is critical for analyzing the quantity of information accessible on chat transcripts, social media postings, reviews, news feeds, etc. [146].

Twitter is one of these non-traditional data sources, and it is a social networking service based on micro-blogging. We can read a brief of 280 characters (or fewer) based on communications known as tweets. Unlike other social networking services, Twitter has a large and diversified user base. Nowadays, Twitter is regarded as a person's voice or speech. Tweets are considered as statements and are parts of the news bulletin, etc. [48].

Text analytics as a field that apply natural language processing techniques is the practice of combining unstructured data in order to find patterns and make decisions, and “tweets” are a rich source of information on a large set of topics that can be analyzed as text. This information may be used to identify patterns connected to a given phrase, assess brand sentiment, or collect feedback on new goods and services [49]. Tweets text analytics starts with collecting the tweets, passing through the preprocessing phase, which includes data modeling and feature engineering, then applying a proper sentiment analysis, and finally, classifying the tweets to make them ready for machine learning techniques.

Since the “tweet” itself is up to 280 characters, many researchers applied feature generation techniques to help in the machine learning process. In a broad study, Lin and Kolcz from Twitter, Inc. applied common machine learning tasks such as data sampling, feature generation, training, and testing to be accomplished directly in an existing Hadoop-based, Pig-centric analytics platform [94]. They mentioned that data, features generated from the data, and the model are the three key components of a machine-learning solution.

As an example of a related study, Kolchyna et al. compare several lexicon combinations and show that adding emoticons, abbreviations, and social-media slang terms to sentiment lexicons improves the accuracy of lexicon-based classification for Twitter. For machine learning sentiment classification, they highlight the necessity of feature generation and feature selection methods [90]. They applied the presence and frequency of n-grams extracted during the pre-processing step. They added additional features to the n-grams to improve the overall quality of text classification, such as the number of words with positive/negative sentiment, number of negations, length of a message, number of exclamation marks, number of different parts-of-speech in a text, such as, for example, number of nouns, adjectives, verbs, and number of comparative and superlative adjectives.

Word embedding is one of the recognizable feature generation approaches. Li et al., in their study, offer a novel method that combines Sentiment-Specific Word Embeddings (SSWE) with a Weighted Text Feature Model (WTFM) [93]. Text negation, a tf-idf weighting scheme and a Rocchio text classification approach are used to create WTFM features. They found that WTFM is easy to build, simple, and effective compared to other Twitter sentiment feature-generating algorithms. In another study, Punyajoy et al. experimented with the pre-trained sentence embedding models to detect hate speech over Twitter [136]. They propose HateMonitor, a machine-learning model designed for identifying hate speech and offensive content in Indo-European languages. To build the system language agnostic, they applied the Gradient Boosting model, as well as Bidirectional Encoder Representations from Transformers (BERT) and Language-Agnostic SEntence Representations (LASER) embeddings.

Having a large scale of tweets makes the applied feature generation approach more helpful. As Nagarajan and Gandhi mentioned in their study, they collected 600 million public tweets using URL-based security tools, and feature generation is applied for sentiment analysis [90].

Much research is done using automatic feature-generation tools. These tools can automate the feature engineering process and generate many features for relational and non-relational data. Some of these recognized tools are: FeatureTools, AutoFeat, TsFresh, Cognito, OneBM, ExploreKit, and PyFeat [83].

2.2 Natural Language Processing

Natural Language Processing (NLP) is the field of computer science and linguistics for interactions between computer science and linguistics. NLP started as a branch of artificial intelligence. It enables computers to understand human language [81]. NLP is usually divided into sevel levels of language processing:

- **Phonetics:** which is the computer processing of physical sounds.
- **Phonology:** which is the linguistic processing of the sounds of a spoken language.
- **Morphology:** which is the level of the word structure processing in a spoken language.
- **Syntax:** which processes the inter-word structure up to sentence structure.
- **Semantic:** which processes the meaning up to sentence level
- **Pragmatics:** which is the process of interpreting the speaker's meaning.
- **Discourse:** which processes the inter-sentence meaning up to larger units [127].

For text analysis, NLP also has many tasks related to the level of the application that it is used for, such as sentiment analysis, question answering, machine translation, and information extraction.

Also, NLP is using different computational language modeling that helps in the feature generation of the analyzed text to support the task of the processed natural language [127].

2.2.1 Sentiment Analysis

Sentiment analysis is the use of NLP, computational linguistics, and textual analysis, in order to identify affective states and subjective information, whether positive, negative, or neutral toward the topic of the text [68]. Fields of marketing, finance, customer service, and other areas, are considered the most common fields that are using sentiment analysis. Sentiment analysis aims in general to determine the feelings of the speaker or writer towards a subject or identify the dominant feeling on a document. These feelings can express the author's opinion (i.e., appraisal) or emotional state (i.e., the case of conscience of the author during the writing) or a deliberate feeling to be delivered (i.e. the influenced conscience that the author wants to deliver to the reader) [19].

The polarity of opinion classification is considered a fundamental process in the analysis of sentiments in texts at the level of the entire document or sentence. The polarity classification refers to the determination of whether the opinion expressed in the document or sentence is positive, negative, or neutral. Other systems have evolved to look at specific emotional states such as anger, sadness, happiness, fear, etc. Sentiment analysis is widely used to evaluate the customer materials such as reviews, survey responses, social media, and materials for applications in many fields [105]. Social-Sentiment Analysis, which is also called "Opinion Mining" is a type of data extraction that measures the tendency or direction of individual opinions, which are used to extract and analyze personal information from websites. The analyzed data is used to determine the public's feelings or reactions to certain products, people, or ideas. The term refers to the process by which consumers' feelings and opinions are analyzed on social media such as Facebook, Instagram, Twitter, and others, where individuals use these platforms to communicate with their friends, colleagues, and family and share their opinions, experiences, and feelings on a particular topic.

The rise of social media such as blogs and social networks has increased interest in sentiment analysis. Social media users share their views on a range of products, services, or news about an event or a brand. These views may enhance the event or brand's reputation or diminish it, and they may help consumers to take a position or make a decision that relies on the brand's reputation. Most sentiment analysis algorithms use simple terms to capture sentiment about a product or service. However, cultural factors, linguistic affairs, and a variety of contexts make it extremely difficult to turn a string of written text into a simple polarity sentiment. The shorter the string of text, the harder it becomes. However, sentiment analysis within microblogging such as Twitter has shown that it can be a valid online indicator of political and economic sentiment [19].

2.2.2 Language Modeling-based Approaches to Features Generation

Language modeling approaches can be defined as probabilistic distribution modeling over a sequence of words to provide context that helps in the natural language processing tasks. It has many modeling types, such as Unigram and n-gram modeling, exponential modeling, neural modeling, and word embedding modeling [106]. In this section, we discuss language modeling-based approaches to feature generation, which are used further in our methodology.

N-grams

In the fields of computational linguistics and probability, an n-gram is a contiguous sequence of n items from a given sample of text or speech. The items can be phonemes, syllables, letters, words, or base pairs according to the application. The concept of n-grams in natural language processing can be traced back to the work of Claude Shannon in the 1940s [144]. He was a pioneer in the field of information theory and he used n-grams as a tool to analyze the frequency and distribution of letters and words in written language. In the 1950s, n-grams gained popularity in computer science and linguistics, particularly in the context of machine translation. Researchers found that by analyzing the frequency and distribution of n-grams in parallel text, they could improve the accuracy of the machine translation system. In 1960s and

1970s, n-grams continued to evolve exploring their efficiency in speech recognition, text classification, and information retrieval. It was also during this time that the term “n-gram” was first coined by J. A. Firth, who used it to describe sequences of words of any length [53]. In the 1980s and 1990s, n-grams became widely used in the field of computational linguistics, particularly in the development of probabilistic language modeling. Researchers found that by modeling the probability of the n-grams in a given text or corpus, they could improve the accuracy of language modeling and enable more effective language processing. When the items are words, n-grams may also be called shingles [43]. N-grams are basically set of occurring words within a given window so when

- $n = 1$ they are called unigrams,
- $n = 2$ they are called bigrams,
- $n = 3$ they are called trigrams, and so on.

The generated n-grams can be vectorized by token count technique [158] or term frequency and inverse document frequency techniques, which is known as tf-idf [63]. Count Vectorizer is a way to convert a given set of strings into a frequency representation, and frequencies are calculated as 0 while other words are present once hence their frequencies are equal to 1. Count Vectors can be helpful in understanding the type of text by the frequency of token n-gram in it. But it cannot identify more important and less important words for analysis. It just consider tokens that are abundant in a corpus as the most statistically significant word It also doesn’t identify the relationships between words such as linguistic similarity between words. On the other hand, tf-idf is a static that based on the frequency of a token in the corpus but it also provides a numerical representation of how important a word is for statistical analysis. Using tf-idf we can then remove the tokens that are less important for analysis, hence making the model building less complex by reducing the input dimensions.

For the term i in document j:

$$tfidf_{i,j} = tfi,j \times log(\frac{N}{df_i})$$

where:

- $tf_{i,j}$ is the number of occurrences of i in j
- df_i is the number of documents containing i,
- N is the total number of documents
- tf is the count of a token in a sentence.
- df is the number of documents that the token is present within corpus.
- And $log(\frac{N}{df_i})$ represent the inverse document frequency (idf).

tf-idf is based on the logic that token that are too abundant in a corpus and token that are too rare are both not statistically important for finding a pattern. The Logarithmic factor in tf-idf mathematically penalizes the words that are too abundant or too rare in the corpus by giving them low tf-idf scores [138].

Word Embedding

Word embedding is one of the most common text vocabulary classifications that is used in natural language processing to represent words as vectors of numeric values. It can capture the context of a word in a document, semantic and syntactic similarity, and relation with other words in a way that can be used by machine learning algorithms to process natural language. The idea of this technique can be traced back to the 1960s, when researchers such as J. R. Firth and Zellig Harris introduced the concept of distributional semantics [107]. However, it was not until the middle of 2000s that word embedding as we know it today began to emerge. The first major breakthrough came in 2003 when Yoshua Bengio and his colleagues introduced the idea of neural language modeling [99]. This approach used a neural network to predict the next word in the sentence based on the vector representation of the preceding words. this led to the development of neural language models such as the Continuous Bag-of-words (CBOW) and skip-grams (SG) models. The most commonly used word embedding model is Word2Vec [106], which is introduced in 2013, by Tomas Mikolov from Google, and generates vector representations of words based on their co-occurrence with other words in a large corpus of text. Other embedding models include GloVe [119] and FastText [108]. Word embedding is commonly used in tasks such as sentiment analysis, document clustering, and machine translation.

Word-to-Vector Embedding (Word2Vec): Word2Vec is a neural network-based model for Natural Language Processing (NLP). Specifically, it is a shallow neural network that is used to generate word embeddings - continuous vector representations of words in a high-dimensional vector space [28].

Word2Vec is not a singular algorithm, rather, it is a family of model architectures and optimizations that can be used to learn word embeddings from large datasets. Embeddings learned through word2vec have proven to be successful on a variety of downstream natural language processing tasks. Word2Vec uses two methods: Continuous Bag-of-words (CBOW) model that predicts the middle word, or the target word, based on surrounding context words; and continuous skip-grams (SG) model that predicts the surrounding words within a certain range before and after the current word, or the given target word, in the same sentence [128]. Figure 2.2 shows the difference between Word2Vec embedding models, CBOW and skip-grams.

Word2Vec model consists of an input layer, a hidden layer, and an output layer. The input layer is one-hot encoded vector representing a specific word in the vocabulary, which is fed into the model. in the SG model it will be one input neuron that represent the target word, while in CBOW it will be the number of context word neurons. The hidden layer is used to calculate the continuous vector representation of the word, based on its context in the input text. It consists of a set of neurons that perform a linear operation on the input vector, followed by a non-linear activation function (usually the sigmoid [133] or ReLU function [12]). In SG model, the output layer produces the probability distribution of the nearby words, given the input word, and based on the vector representation generated by the hidden layer. It also consists of a set of neurons, each representing a particular context word. The output of each neuron is a probability distribution over the entire vocabulary, indicating the likelihood of the corresponding context word appearing near the target word. The softmax function [45] is usually applied to convert the output neuron values into probabilities. On the other hand, the output layer consists of a single neuron, which predicts the probability distribution of the target word given the context words.

In Word2Vec model, there are some important parameters that need to be considered, such as window size and negative sampling. Window size refers to the number of context words used to predict the target word in a given input sequence. The

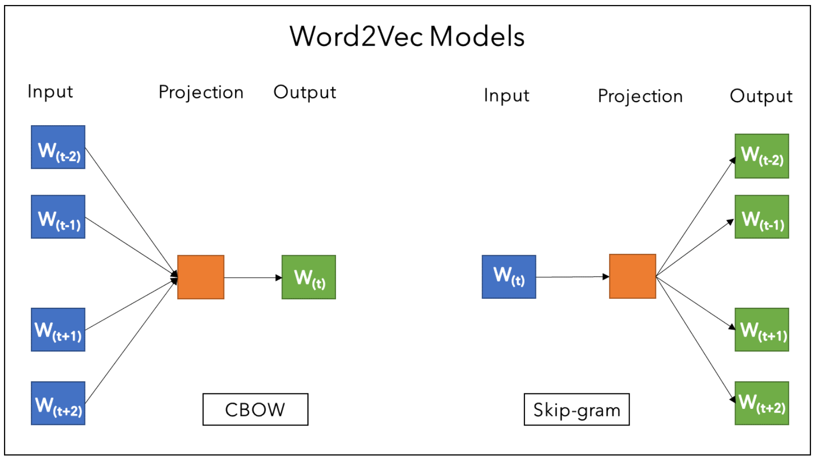


Figure 2.2: The difference between CBOW and SG models in Word2Vec embedding method

window size determines the range of words around a given target word that the Word2Vec model considers when training. The choice of window size is important hyper-parameter that effect the quality of learned word embeddings. A smaller window size will capture more local context and result in word embeddings that reflect specific usage scenarios, while a larger window size will capture more global context and result in more general-purpose word embedding [47].

Negative sampling is a technique used in Word2Vec model to speed up the training process and improve the quality of the learned word embedding [95]. The basic idea of negative sampling is to replace the original softmax layer [45] of the Word2Vec model with a binary Logistic Regression classifier. Rather than attempting to predict the propabilities of all possible words in the vocabulary, negative sampling only updates a small subset of the weights in the model during training. The binary Logistic Regression classifier then predicts weather each of sampled words is a positive or negative example for the given target word. Negative sampling can help the Word2Vec model to better capture the meaningful relationships between words and produce haigher-quality word embedding.

During training, the model adjusts the weights of the connections between the input, hidden, and output layers, using an optimization algorithm like stochastic gradient descent, to minimize the error between predicted and actual context words. Once trained, the hidden layer of the Word2Vec model can be extracted to obtain the vector representation (embedding) of each word in the vocabulary. These embeddings can then be used for various NLP tasks, such as sentiment analysis, text classification, and machine translation [104].

Word2Vec models can be pre-trained on large text corpora using unsupervised learning techniques before being fine-tuned [1] for specific downstream NLP tasks. There are several pre-trained Word2Vec models available that can be used for various NLP applications, such as; Google News Word2Vec Model [1], which was trained by Google on a huge corpus of Google News articles and consists of 300-dimensional word embeddings for over three million words and phrases. It is available for download and can be used for a wide range of NLP tasks. Another example of pre-trained Word2Vec models is ConceptNet Numberbatch [151]. This is a state-of-the-art open-source semantic embeddings model that combines several existing models, including GloVe and Word2Vec, into a single unified model. It contains high-quality word embeddings in 17 languages and has been trained on several large text corpora and knowledge graphs. A final example of pre-trained Word2Vec models is AraVec [148], which is a is a pre-trained word embedding model for Arabic language. It is trained on a large corpus of Arabic text data with over 10 billion tokens, using the continuous skip-gram algorithm.

Pre-trained language Models

Is a machine learning model that has already been trained on a large corpus of text data that can be used to generate feature vectors for text data. Feature vectors are numerical are numerical representations of features that can be used as input for machine learning models. A pre-trained model can assign high-dimensional vector to each word in the text, which can be averaged or concatenated to create vector for the entire text. The evolution of word embeddings and neural language models, paved the way for more advanced pre-trained language models like BERT (Bidirectional Encoder Representations from Transformers) in 2018 [55]. BERT was developed by researchers at Google and is considered a major breakthrough in NLP. Since the release of BERT many other pretrained language models have been developed, such as

[1]Fine-tuning in Natural Language Processing (NLP) refers to the process of taking a pre-trained language model and adjusting its parameters on a specific task or domain.

GPT (Generative Pre-trained Transformer) [123] in 2018, RoBERTa (Robustly Optimized BERT Approach) [97] in 2019, and T5 (Text-to-Teaxt Transfer Transformer) [129] in 2020.

Bidirectional Encoder Representations from Transformers (BERT): BERT is a pretrained language model that can be used to generate features for natural language processing tasks [55]. BERT is trained on a large corpus of text, and as a result, it learned a lot about structure of human language. To generate features using BERT , fine-tuning the model on a specific task is required. Fine-tuning is the process updating the parameters of the pre-trained model on new task, such as sentiment analysis or named entity recognition. After fine-tuning, the model can be used to generate features for new text data related to that task [155]. The structure of BERT is based on the Transformer architecture, which is a type of neural network used for sequence-to-sequence modeling. The Transformer architecture consists of an encoder and decoder, both of which are made up of multiple layers of attention and multi-head self-attention mechanism [52].

The BERT model has only encoders, which is why it is referred to as an encoder-only architecture.

The encoder layer of BERT is made up of Transformer blocks, each which contain multiple layers of self-attention mechanism and feed-forward neural network. Each transformer encoder layer in BERT has several sub-layers that perform specific functions:

- **Multi-Head Self-Attention layer:** This sub-layer computes a series of attention weights for each token in the input text sequence based on its relationship to the other tokens in the sequence. In this stage, BERT takes as input the embeddings of all tokens in the input sequence along with some additional information such as position and segment embeddings. BERT then computes a set of attention weights that capture the relationship between each input token and all other tokens in the sequence. These attention weights are calculated using a dot product between the embeddings of each pair of tokens, and a softmax function is applied to normalize the weights. Once the attention weights have been calculated, BERT computes a weighted sum of all token embeddings, using the attention weights as coefficients. This weighted sum produces a context vector for each token in the sequence, which captures its context and relationships with other tokens in the sequence [159].

- **Layer Normalization:** This sub-layer normalizes the output from the multi-head self-attention layer [159].

- **Feed-Forward Neural Network layer:** This sub-layer applies a non-linear transformation to each token embedding based on its contextualized representation produced by the self-attention layer. Specifically, the feed-forward layer consists of two linear transformations (a fully connected layer) with a ReLU activation function in between them. The goal of this layer is to help BERT model to enhance the learned features and capture more complex interactions between the input tokens [64].

- **Another Layer Normalization:** The output of the feed-forward neural network layer is also normalized.

BERT is based on stacked layers of encoders. The difference between BERT base and BERT large is on the number of encoder layers. BERT base model has 12 encoder layers stacked on top of each other whereas BERT large has 24 layers of encoders stacked on top of each other.

In BERT base, there are 12 hidden layers that are used for encoding text input into contextualized token representation. Each of these layers has specific role in the coding process and contributes to the overall quality of the model prediction, in addition to two layers for input and output. Here is an overview of the role of each layer in the BERT model:

- **Input embedding layer:** This layer takes raw text inputs (such as a sentence or document) and converts them into a series of token embeddings. Each token embedding is an initial vector representation of the token that is used as input to the subsequent layers.

- **12-Transformer encoder layers:** In this section, there are 12 similar transformer encoder blocks that take the token embeddings produced by the input embedding layer and refine them through a series of attention-based operations. More specifically, the first few layers in BERT primarily focus on identifying basic word-level features such as individual words, punctuation marks, and basic syntax. As the layers progress, the model starts to extract more complex features such as named entities, idiomatic expressions, and other subtle linguistic nuances. The final few layers in BERT are typically the most important, as they are responsible for producing the final contextualized word embeddings that are used to make predictions for downstream NLP tasks such as sentiment analysis, text classification, and named entity recognition.

- **Final output layer:** After the token embeddings have been processed by the 12 transformer layers, the final output layer produces a set of contextualized token embeddings that are capable of capturing rich contextual information about each token in the input text [64] [159].

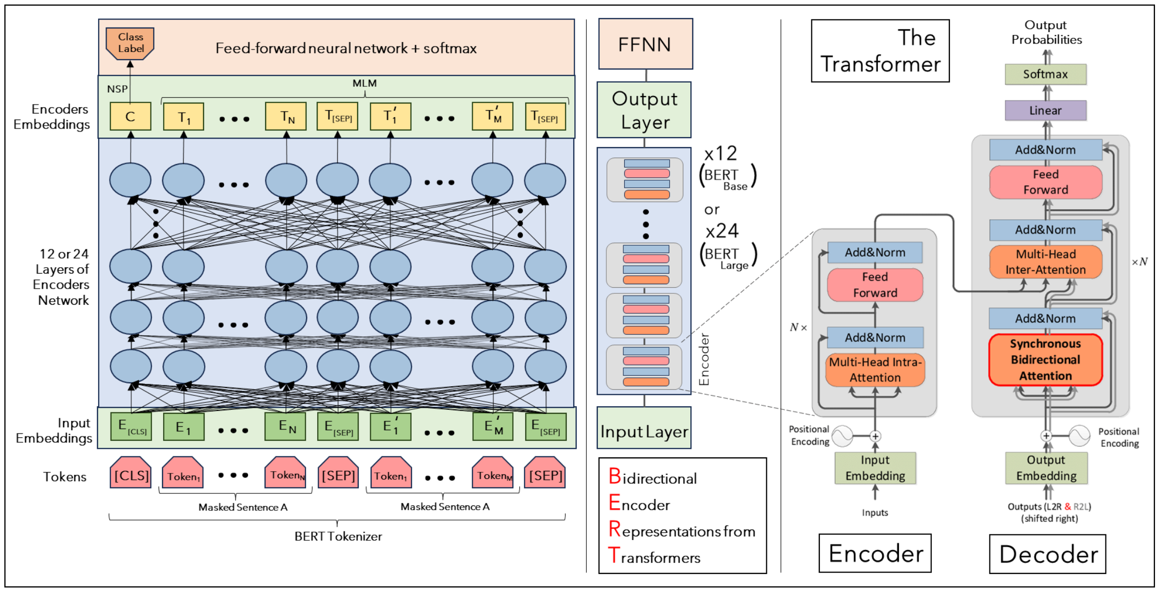


Figure 2.3: The detailed structure of BERT

In BERT, the softmax function is used in the final layer of the model for classification tasks. It is applied to the output of the classification layer, which consists of a single dense layer followed by a softmax activation [64]. The dense layer computes a weighted sum of the contextualized word representations and produces a fixed-length vector representation of the input sequence. After the dense layer, the softmax function is applied to normalize the output vector into a probability distribution over all possible class labels. Each element in the output vector represents the probability of the input sequence belonging to a specific class. Instead of softmax, the argmax function [44] can be used to find the index of the maximum value in a tensor or a vector. In the context of BERT, argmax is commonly used to find the index or indices with the highest scores in the output of the model.

The key innovation in BERT is that it is a bidirectional model, meaning it can into account both the left and the right context of word in a sentence. This is achieved by using a technique called Masked Language Modeling (MLM), where some of the word in sentence are randomly masked and the model is trained to be predict the original word from its context [42]. Additionally, BERT is pretrained on a large corpus of raw text, such as Wikipedia, BookCorpus, and the Common Crawl. Figure 2.3 shows the structure of BERT pre-trained language model.

In MLM, a percentage of the input tokens in a sentence are masked or replaced with a special [MASK] token, and the goal of the model is to predict the original words from the masked ones. Specifically, during pre-training, BERT randomly masks 15% of the tokens in each input sentence, where 80% of them are replaced with the special [MASK] token, 10% are replaced by a random token, and the remaining 10% are left unchanged. The model is then trained to predict the original tokens that were replaced with [MASK]. By performing MLM as a pre-training task, BERT is able to learn rich, context-sensitive word representations that capture both the left and right context of each token, making it truly bidirectional. This is in contrast to traditional left-to-right or right-to-left language models that only use a fixed context to predict the next token. Once trained on the MLM task, the learned parameters of the BERT model, including the word embeddings and the transformer layers, can be fine-tuned on downstream NLP tasks such as text classification, question answering, and named entity recognition. This fine-tuning further enhances the contextualized word representations learned by the model during pre-training, resulting in improved performance on the task at hand [42].

2.3 Machine Learning and Deep Learning

Machine Learning (ML) is a branch of Artificial Intelligence (AI) that develops algorithms and technologies that allow computers to have features of "learning", which give computers the ability to learn without being explicitly programmed [36]. The basic task of machine learning is to extract valuable information from data, which is very close to tasks of data mining, statistics, and information theory. Machine learning is closely related to (or often overlaps with) computational statistics, which also focuses on making predictions using computers. The data analytics field, which is a method used to develop complex models and algorithms suitable for the prediction process, is using machine learning commercially which is known as Predictive Analytics. These analytical models allow data scientists to produce reliable and reproducible decisions and results, and uncover hidden patterns by learning from historical relationships and trends within this data [36].

Machine learning includes a wide range of application fields, such as natural language processing, syntactic pattern recognition, search engines, medical diagnostics, bioinformatics, chemical informatics, DNA sequencing classification, speech recognition, handwriting recognition, and even object recognition, computer vision, strategic games, and robot locomotion.

Machine learning algorithms are classified into several types, the most important being:

- **Supervised Learning,** which is one of the most popular types of machine learning and is based on the presence of data and correct evidence at the time of learning.
- **Unsupervised Learning,** which is a learning that results from the presence of data without proper evidence.
- **Reinforcement learning,** which is a type of unsupervised learning in which the machine interacts with the environment and builds its expertise based on this interaction [36].

Another branch of machine learning that deals with artificial intelligence is Deep Learning (DL), which is considered a new area of research dealing with theories and algorithms that allow a machine to learn by simulating neurons in the human brain [142]. Most of the deep learning research focuses on finding methods to develop a high degree of abstraction by analyzing a huge data set using linear and nonlinear transformers. The discoveries in this field have proved significant, rapid, and effective towards progress in many areas including facial recognition, speech recognition, computer vision, natural language processing, audio recognition, social media filtering, machine translation, bioinformatics, and board game programs. The machine learns from big data using various designs of deep learning networks, including Recurrent Neural Network (RNN) frequently used with continuous texts and data, and Convolutional Neural Networks (CNN) that draw their inspiration from biological processes in the brain and other designs. Simply, most machine learning algorithms rely on two basic steps: observation and simulation (prediction). This is in the group of algorithms that rely on supervised learning or learning by observing previous events of the known results. It first monitors the input data and tries to devise distinct patterns and characteristics of the data and then simulates the behavior of functions based on the connections and relationships formed by monitoring the process of converting the input data to specific outputs [142].

2.3.1 Supervised Learning Methods

We present in this section an overview of some supervised learning methods in ML, which we will also refer to in presenting our methodology in further chapters.

Logistic Regression (LR)

Logistic Regression (LR) is a statistical model used for binary classification problems, where the output variable can take two possible values (Such as 0 and 1 or yes and no). It predicts the probability of an event occurring by fitting a logistic function to a set of input variables. The logistic function, also called the sigmoid function, transforms the output of a linear equation into a value between 0 and 1, which represents the probability of the dependent variable belonging to the positive class. The equation for logistic regression is given by:

$$p = \frac{1}{1 + e^{-z}}$$

where:

- p is the predicted probability of the positive class
- e is the base of the natural logarithm
- z is the linear combination of input variables.

The value z can be calculated as:

$$z = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \ldots + \beta_n x_n$$

where:

- β_j is the coefficient or weight assigned to each input variable.
- x_n is the value of the nth input variable

The coefficients in logistic regression are typically estimated using maximum likelihood estimation, a method that finds the values of the coefficients that makes the predictions most likely to be correct based on the available data. Logistic Regression can be used for both binary classification and multinomial classification problems, where the output variable can take more than two possible values.

K-Nearest Neighbors (KNN)

In pattern recognition, the k-Neareast Neighbours (KNN) algorithm is a non-parametric technique used for classification and regression [22]. As a part of both cases, the input comprises of the k nearest training samples in the attribute space. The result relies on whether KNN is used for classification or regression. The concept of KNN classification algorithm is that the object is classified by the majority votes of its neighbors. Given n training vectors, KNN algorithm identifies the k nearest neighbors of 'c' to estimate its class, regardless of the labels of the class. For example, if we have $k = 3$, and we have two classes 'a' and 'b', let us consider the case where the algorithm should find the class of an unclassified object 'c'. Because $k = 3$, we have to find the nearest three neighbors of object 'c'. If more 'b's are the nearest neighbors to 'c' than those in class 'a', then 'c' is classified as 'b'. Figure 2.4 explains the algorithm of KNN.

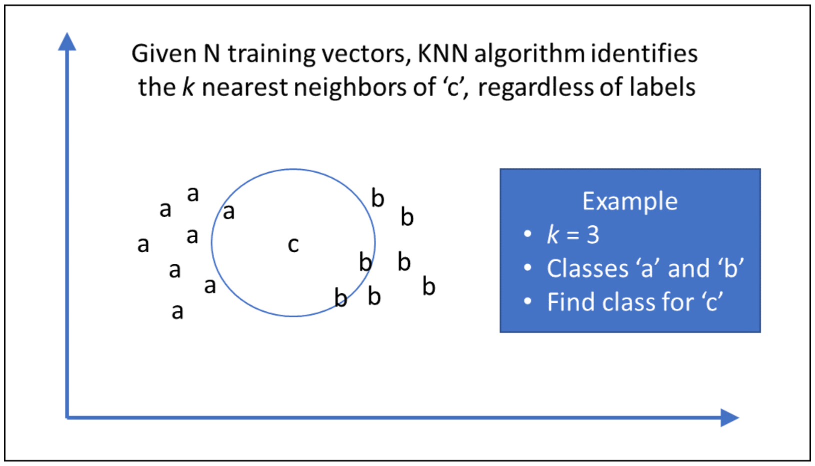

Given N training vectors, KNN algorithm identifies the *k* nearest neighbors of 'c', regardless of labels
Example
• *k* = 3
• Classes 'a' and 'b'
• Find class for 'c'

Figure 2.4: K-Nearest Neighbor methodology

For another example, if $k = 1$, each training vector defines a region in space, defining a Voronoi partition of space. Figure 2.5 explains the partitioning method in the KNN algorithm. To perform this algorithm in the best way, the k value should be an odd value for two classes' problems, and it should not be a multiple of the numbers of classes. The main disadvantage of KNN is the complexity in searching the nearest neighbors for each sample [22].

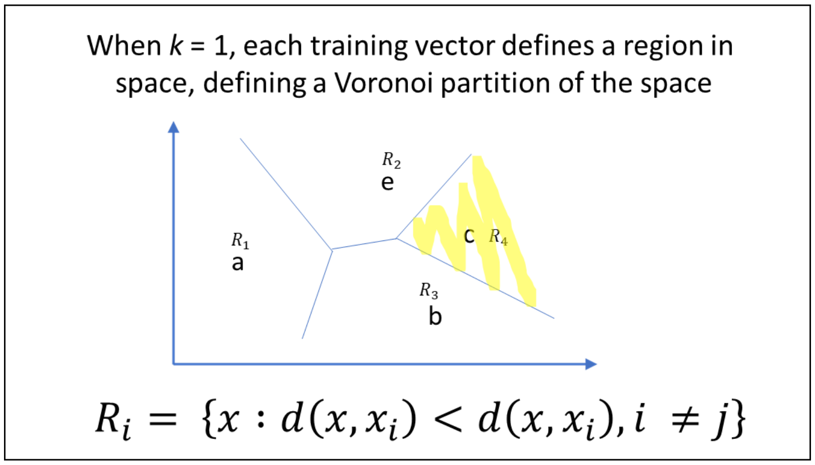

When *k* = 1, each training vector defines a region in space, defining a Voronoi partition of the space
R_1 a
R_2 e
R_3 b
R_4 c
$R_i = \{x : d(x, x_i) < d(x, x_i), i \neq j\}$

Figure 2.5: K-Nearest Neighbor area partitioning

Another negative side is that KNN is a lazy learner. It learns nothing from the training dataset and just uses the training dataset itself for classification. For new instances label prediction, the KNN algorithm will find the k closest neighbors to the new instance from the training data. The predicted class label will then be set as the most common label among the k closest neighboring points. Furthermore, it is a slow method for large data size [22].

Naïve Bayes and Bayesian Network

In machine learning, Naïve Bayes classifiers are a group of basic probabilistic classifiers used to apply Bayes' hypothesis with powerful independence assumptions between the attributes. A Naïve Bayesian model is easy to build with no complicated iterative parameter estimation, which makes it particularly useful for large datasets. Furthermore, Naive Bayesian classifier often works well and is commonly used because it often performs better than other complicated methods. Naive Bayes has been employed widely since the 1950s. It was brought in as alternate name into the text retrieval challenges in the mid-1960s [130], and remains a popular method for text categorization, judging documents as fitting in with one classification or the other, (for example, spam or legitimate, sports or political issues, etc.), and with word frequencies as the components. With proper preprocessing, it is used in this space with added strategies including support vector machines.

Bayes theorem provides a method for calculating the posterior probability, $P(c|x)$, from $P(c)$, $P(x)$, and $P(x|c)$. Naïve Bayes classifier which assumes that the result of the value of the predictor (x) on a given class (c) is independent of the values of other predictors. This assumption is called class conditional independence [70]. One of the notable limitations of Naïve Bayes is that it has strong feature independence assumptions, while, on the other hand, it is fast to train and classify, and it is not sensitive to irrelevant features. Another important feature of the Naïve Bayes algorithm is that it handles both real and discrete data, as well as handling streaming data [70].

The Naïve Bayes algorithm is based on the Bayes' theorem, which can be expressed using the following formula:

$$P(c|x) = \frac{P(x|c)P(c)}{P(x)}$$

where:

- $P(c|x)$ is the posterior probability of the class (target) c, given predictor (attributes) x,

- $P(c)$ is the prior probability of the class c,
- $P(x|c)$ is the likelihood which is the probability of the predictor x given class c, and
- $P(x)$ is the prior probability of the predictor x.

The strong assumption of the Naïve Bayes method is that the attributes $x = (x_1, x_2, \ldots, x_n)$ are independent given the class c, or in other words:

$$P(x|c) = P(x_1, x_2, \ldots, x_n|c) = P(x_1|c) \cdot P(x_2|c) \cdot \ldots \cdot P(x_n|c)$$

so the Bayes' theorem formula becomes:

$$P(c|x) = \frac{P(x_1|c) \cdot P(x_2|c) \cdot \ldots \cdot P(x_n|c) \cdot P(c)}{P(x)}$$

Decision Tree Methods (DT)

A Decision Tree (DT) is a decision support instrument that uses a tree-like chart or model of choices and their possible results, including chance occasion results, asset expenses, and utility. It is one approach to show an algorithm. Decision trees are generally used as a part of research, or examination, that is particularly in decision analysis, to recognize a technique used to achieve an objective, and they are a common tool in machine learning. This method has many algorithms, and in this research, two common algorithms were chosen to apply to the dataset and to evaluate its accuracy.

C4.5 algorithm (J48): C4.5 is an algorithm used to build a decision tree created by Ross Quinlan [140]. `C4.5` is an extension of `ID3` decision tree algorithm. The decision tree produced by C4.5 can be used for classification, and hence, `C4.5` is frequently referred to as a statistical classifier. `C4.5` assembles decision trees from an arrangement of training data similar to `ID3`, utilizing the idea of data entropy. The training dataset, $S = \{s_1, s_2, \ldots\}$, is a set of effectively grouped examples. Every specimen s_i comprises a p-dimensional vector $(x_1^i, x_2^i, \ldots, x_p^i)$, where the x_j represents feature values or feature of the sample, as well as the class in which s_i falls. At every node of the tree, `C4.5` picks the quality of the information that most successfully parts its arrangement of tests into subsets improved in one class or the other. The part basis is the standardized data pick up (distinction in entropy). The characteristic with the most noteworthy standardized data addition is settled on the choice. The `C4.5` calculation then repeats the process on the sub-lists [140].

This algorithm has a couple of base cases. The base case is all the samples in the list that fit in with the same class. At the point when this happens, it essentially makes a leaf node for the decision tree saying to pick that class. None of the features give any information gain. For this situation, C4.5 makes a decision node up the tree using the normal estimation of the class. Occasionally beforehand a concealed class is experienced. At this point, `C4.5` makes a decision node higher up the tree using the normal process. `C4.5` can build models that can be interpreted easily. Another feature of `C4.5` is that it can use categorical and continuous values, in addition to dealing with noise properly. This method cannot work properly with small training datasets, as small data can generate different decision trees [70].

Random Forest (RF): Random Forest (RF) reflects the general method of arbitrary decision forests [70] that are a group learning approach for classification, regression, and other different roles. This method works on building many choice trees at training time and resulting in the class that is the chosen method of the classes (classification) or mean prediction (regression) of the individual trees. Random decision forests are an answer for a decision trees' propensity for the overfitting issue to a training set. The method merges Breiman's "bagging" idea and the random selection of features.

Random forest runs efficiently on large databases and handles thousands of input variables without variable deletion. It has an efficient method for estimating missing data and maintains accuracy when a large proportion of the data are missing. At same time it has some limitations. Overfitting could occur in some datasets with noise, and it could be biased for some dataset features [70].

Support Vector Machine (SVM)

A Support Vector Machine (SVM) is a type of machine learning algorithm used in classification and regression analysis. It works by filtering the hyperplane (line that separates the data) that maximizes the distance between the closest data points of different classes (called the margin). For linear SVM, the goal is to find the ”maximum-margin hyperplane” that divides the group of points x_i for which $y_i = 1$ from the group of points for which $y_i = -1$, which is defined so that the distance between the hyperplane and the nearest point x_i from either group is maximized.

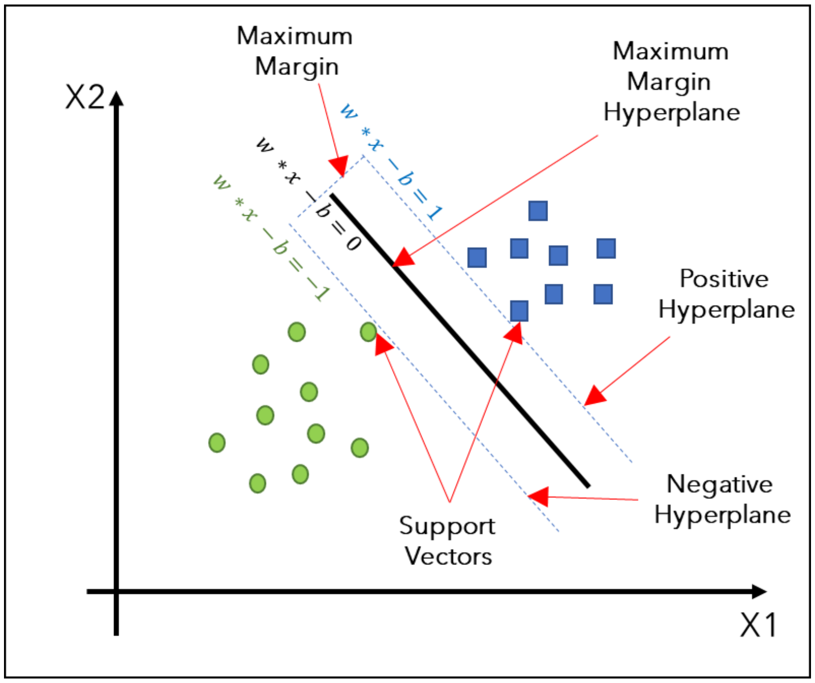


Figure 2.6: Maximum-margin hyperplane and margins for an Support Vector Machine (SVM) trained with samples from two classes.

For linear SVM, hyperplane can be written as the set of x satisfying:

$$w^T * x - b = 0$$

where w is the normal vector to the hyperplane. Figure 2.6 explains the linear SVM hyperplane.

Non-linear boundaries can be obtained using non-linear kernels that project the input features into higher-dimensional feature space. Some commonly used kernel functions in SVM include:

- **Polynomial Kernel:** This kernel can handle non-linearity by projecting the features into a higher-dimensional space using a polynomial function.

- **Radial Basis Function (RBF):** is the most commonly used kernel and it can handle non-linearity by projecting the features into infinite-dimensional space. The RBF kernel is also known as the Gaussian kernel

- **Sigmoid kernel:** This kernel can handle both linear and non-linear data and canbe used for non-linear relationship between data features.

The choice of kernel function depends on the specific problem, selecting the right kernel can drastically improve the accuracy of SVM in classification and regression tasks.

SVM is a powerful algorithm that can handle both linear and non-linear data and is widely used in a variety of applications such as image classification, text classification, and bioinformatics.

2.3.2 Deep Learning Methods

In this section, we will describe some Deep Learning (DL) methods that used in our studies.

Multilayer Perceptron (MLP)

Multilayer Perceptron (MLP) is a deep, artificial neural network that produces a series of outputs from a set of inputs. An MLP is constructed of at least three node layers: an input layer, a hidden layer, and an output layer. Except for input nodes, each node is a neuron that uses a nonlinear activation function. MLP uses a supervised learning technique called backpropagation for training [74]. If a multilayer perceptron includes a linear activation function in all neurons, that is, a linear function that maps the weighted inputs to the output of every neuron, then algebra shows that any number of layers will be reduced to a two-layer input-output model [73]. In MLPs some neurons use a nonlinear activation function that was developed to model the frequency of action potentials, or firing, of biological neurons. The two common activation functions are are both sigmoids, and described by:

$$y(v_i) = \tanh(v_i)$$

and

$$y(v_i) = (1 + e^{-v_i})^{-1}$$

The first function is a hyperbolic tangent that ranges from -1 to 1, while the second function is the logistic function, which is similar in shape but ranges from 0 to 1, where y_i is the output of the ith node (neuron) and v_i is the weighted sum of the input connections.

In recent developments of deep learning, the Rectified Linear Unit (ReLU) is more frequently used as one of the possible ways to overcome the numerical problems related to the sigmoids, defined as the positive part of its argument:

$$RELU(x) = \begin{cases} x & if x > 0, \\ 0 & \text{otherwise.} \end{cases}$$

where x is the input to a neuron.

Each neuron in the MLP can be represented by the following formula:

$$f\left(b + \sum_{i=1}^{n} x_i w_i\right) = a$$

where:

- f is an applied activation function to the neuron,
- $x_1 \ldots x_n$ are the neuron inputs,
- $w_1 \ldots w_n$ are their corresponding weights,
- b is a bias value, and
- a is the neuron output.

The MLP consists of three or more layers (an input and an output layer with one or more hidden layers) of nonlinear-activating nodes. Since MLPs are fully connected, each node in one layer connects with a certain weight w_{ij} to every node in the following layer. Dense layer, also called fully-connected layer, refers to the layer whose inside neurons connect to every neuron in the preceding layer. Training is generated in the perceptron by adjusting the link weights after processing each piece of data, based on the amount of error in the output in comparison to the expected result. The output layer outputs the classes using the MCXENT loss function, and softmax activation

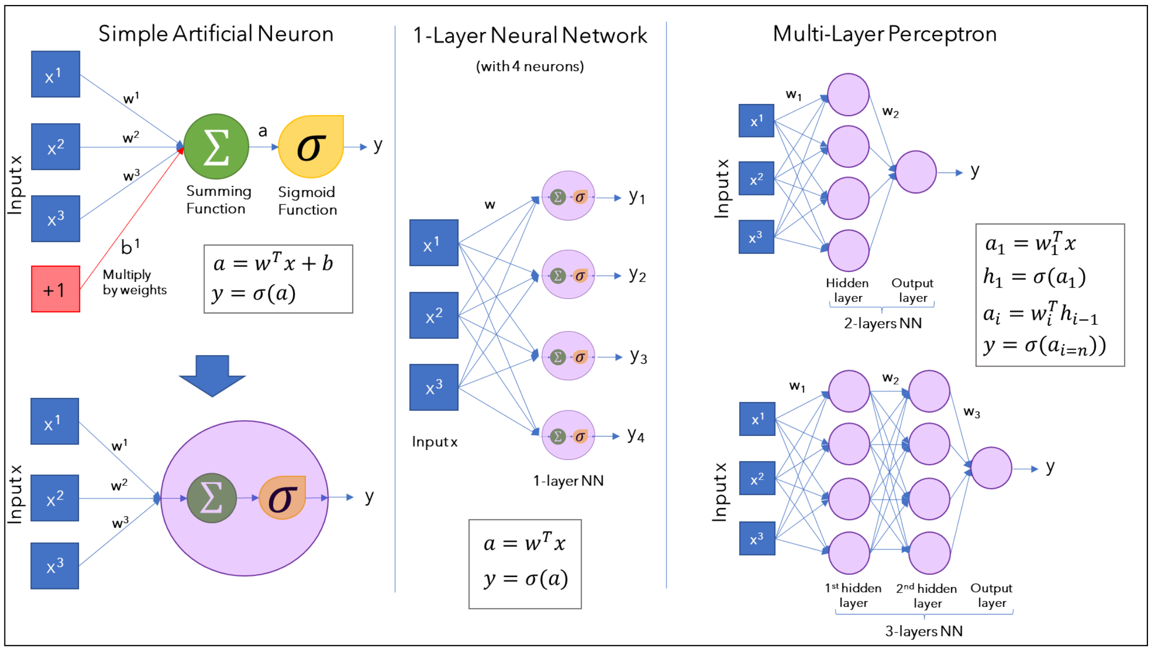


Figure 2.7: An illustration of the neural networks and the structure of the layers in neural network

function if it is a multi-class classification technique, which transforms the neurons' output exponential to class probabilities distribution, as described in the following formula:

$$softmax(a) = \frac{e^{a_i}}{\sum_{j=1}^{i} e^{a_j}}$$

where:

- a are the outputs of the neurons, and
- e is exponential of each a.

For binary classification, sigmoid activation function is used to outputs the classes, as described in the following formula:

$$\sigma(a) = \frac{1}{1 + e^{-a}}$$

Figure 2.7 illustrates the structure of the neural networks, starting from simple neural network, 1-layer neural network and the multi-layer neural network or Multi-layer Perceptron (MLP).

Weightless Neural Network (WiSARD)

Weightless Neural Network (WiSARD) is a supervised RAM-based neural network method. The RAM-based neural networks essentially use look up tables to store the function computed by each neuron, and hence are easily implemented in digital hardware and have efficient training algorithms. WiSARD was originally conceived as a pattern recognition device mainly focusing on an image processing domain. With ad-hoc data transformation, WiSARD can also be used successfully as multiclass classifier in machine learning domain. A WiSARD is formed by as many discriminators as the number of classes it must discriminate between. Each discriminator consists of a set of n RAMs that, during the training phase, learn the occurrences of n-tuples extracted from the input binary vector. The `WiSARD4WEKA` package implements a multi-class classification method based on the WiSARD weightless neural model for the Weka machine learning toolkit. This implementation adds a data-preprocessing filter method to exploit WiSARD neural model training and classification capabilities. The resulting software is a classification method based on WiSARD neural network model. Usually used for image deep learning, a few studies also used it for text classification [65].

Radial Basis Function Network (RFBN)

RBFN is an artificial neural network that uses radial basis functions as activation functions. The output of the network is a linear combination of radial basis functions of the inputs and neuron parameters. RBFN has many uses, including function approximation, time series prediction, classification, and system control. It uses the K-Means clustering algorithm to provide the basic functions and learns either a logistic regression (discrete class problems) or linear regression (numeric class problems) on top of that. Symmetric multivariate Gaussians are fit to the data from each cluster. If the class is nominal, it uses the given number of clusters per class. It standardizes all numeric attributes to zero mean and unit variance [58]. Figure 2.8 presents the Radial Basis Function Network method.

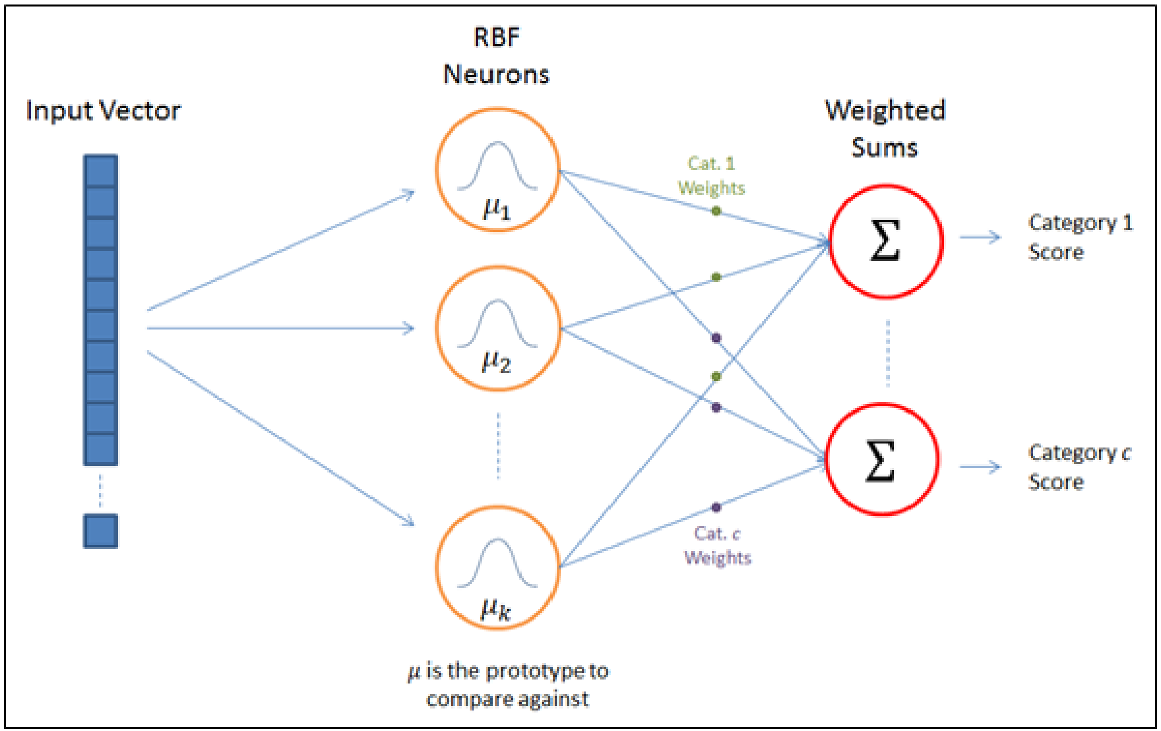


Figure 2.8: Radial Basis Function Network explanation

Long Short-Term Memory (LSTM)

LSTM is an artificial neural network architecture used in deep learning. It was designed to solve the problem of vanishing gradients in traditional Recurrent Neural Network (RNN)s [126]. LSTM networks are powerful in processing sequential data and are used in applications such as speech recognition, language modeling, and machine translation. The architecture includes memory cells, input gates, output gates, and forget gates that allow it to retain long-term memory and selectively forget or remember information.

the LSTM structure consists of a sequence of memory cells, or units, that are connected through gates. The main components of each LSTM cell are:

- **Memory cell:** which is the main storage unit of the LSTM network, which remembers information over time.
- **Input gate:** that decides which information should be stored in the memory cell, blocking irrelevant information while allowing important information to pass through.

- **Forget gate:** that decides which information to discard memory cell, removing any irrelevant or outdated information.
- **Output gate:** that decides which information from the memory cell should be used as output, allowing the LSTM to selectively output the most relevant information.

Gates in LSTM are the sigmoid activation functions i.e they output a value between 0 or 1 and in most of the cases it is either 0 or 1. The equations for the gates in LSTM are:

$$i_t = \sigma(w_i[h_{t-1}, x_t] + b_i)$$

$$f_t = \sigma(w_f[h_{t-1}, x_t] + b_f)$$

$$o_t = \sigma(w_o[h_{t-1}, x_t] + b_o)$$

where:

$i_t \rightarrow$ represents input gate.
$f_t \rightarrow$ represents forget gate.
$o_t \rightarrow$ represents output gate.
$\sigma \rightarrow$ represents sigmoid function.
$w_x \rightarrow$ weight for the respective gate(x) neurons.
$h_{t-1} \rightarrow$ output of the previous LSTM block (at timestamp $t-1$).
$x_t \rightarrow$ input at current timestamp.
$b_x \rightarrow$ biases for the respective gates(x).

The equations for the cell state, candidate cell state and the final output:

$$\tilde{c}_t = \tanh(w_c[h_{t-1}, x_t] + b_c)$$

$$c_t = f_t * c_{t-1} + i_t * \tilde{c}_t$$

$$h_t = o_t * \tanh(c^t)$$

where:

$c_t \rightarrow$ cell state (memory) at timestamp(t).
$\tilde{c}_t \rightarrow$ represents candidate for cell state at timestamp(t).

We can pass this h_t the output from current LSTM block through the softmax layer to get the predicted output y_t from the current block.

The operation of each gate is controlled by a set of weights and biases that are learned during training, allowing the LSTM network to adapt to a different tasks and datasets. The combination of these components allows the LSTM to effectively handle long-term dependencies in sequential data. Figure 2.9 shows the explanation of the architecture of the LSTM network.

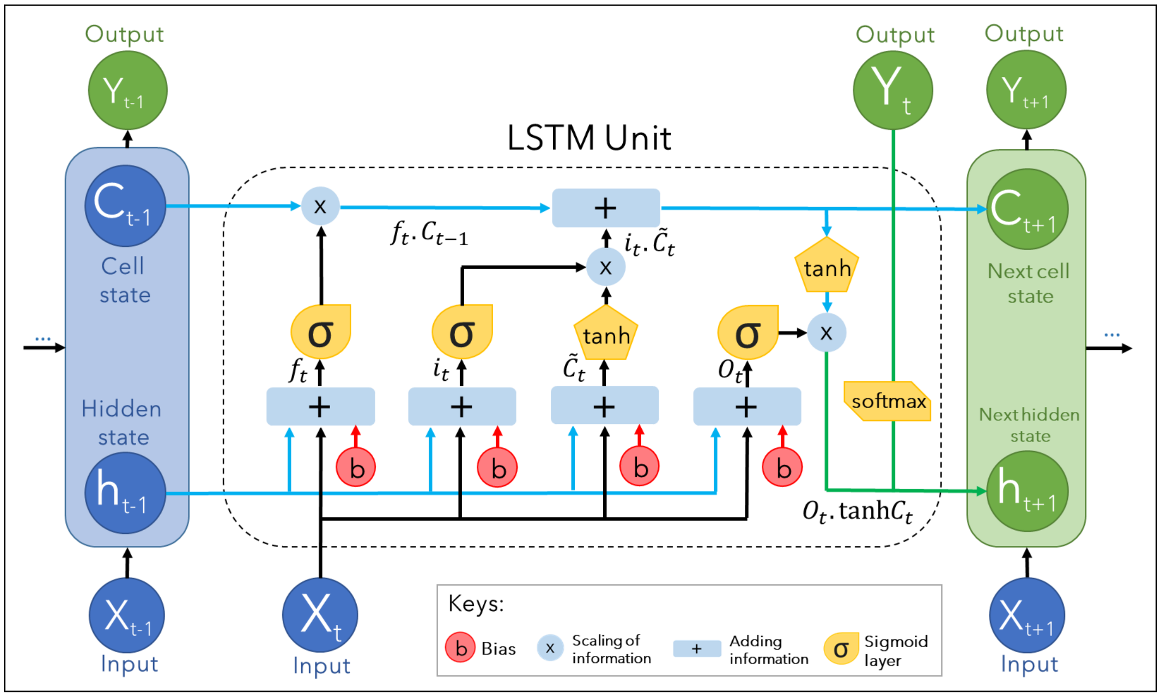


Figure 2.9: The architecture of the LSTM network

Bi-directional Long Short-Term Memory (Bi-LSTM) : is a Recurrent Neural Network (RNN) architectures used in Natural Language Processing (NLP) tasks such as named entity recognition, language modeling, and machine translation. Bi-LSTM processes the input sequence in both directions, i.e., left to right and right to left, before concatenating the outputs. In other words, bi-LSTMs have two layers of hidden units: one for the forward sequence and another for the backward sequence. Bi-LSTM captures the contextual information of both past and future states of a given sequence and thus generally produces better results than unidirectional LSTMs.

The main difference between LSTM and Bi-LSTM lies in the direction of context processing. While LSTM processes input sequences in one direction (usually left to right), Bi-LSTM processes input sequences in both directions (left to right and right to left) and concatenates the outputs. Bi-LSTM generally performs better than unidirectional LSTM due to its ability to capture contextual information from both past and future states of the given sequence.

2.4 Machine Learning Tools and Libraries

Data mining and predictive modeling are related to machine learning tools, which are algorithmic applications of artificial intelligence that enable systems to learn and advance without a lot of human input. They enable software to predict outcomes more accurately without having to be explicitly programmed. The concept is to use a model or algorithm to gather data from the outside world and then feed that data back into the model so that it can get better over time. Machine learning is used because the model "learns" as it is fed increasing amounts of data.

They can be used, for example, to create news feeds, recommender systems, search pattern predictions, spam filters, recommendation engines, and much more. Supervised, unsupervised, semi-supervised, and reinforced machine learning algorithms are the four different categories.

Weka: Waikato Environment for Knowledge Analysis (Weka) is a collection of machine learning algorithms developed by Waikato University in New Zealand, which is particularly well-organized in the framework of data mining tasks. It has tools for mining association rules, clustering, regression, classification, and visualization of data. Weka is free software distributed under the terms of the GNU General Public License, and the companion software to the book "Data Mining: Practical Machine Learning Tools and Techniques". With the help of graphical user interfaces, Weka provides a variety of visualization tools and algorithms for data analysis and predictive modeling [77].

Weka offers support for a number of common data mining operations, including feature selection, data preprocessing, clustering, classification, and regression. Weka expects input that is formatted using the Attribute-Relational File Format and has a filename ending in `.arff` extension. Each of Weka's methods are based on the premise that the data is available as a single flat file or relation, with each data point being described by a fixed number of attributes. The package manager in Weka was added to allow the easier installation of extension packages.

R: R is a programming language for statistical computing and graphics [122], developed by Ross Ihaka and Robert Gentleman, two statisticians, who made it an important tool used today by data miners, bioinformaticians, and statisticians for data analysis and creating statistical software. In order to extend the capabilities of the R language, users have created a large number of packages. R has become of the major programming languages used today in data mining [122].

The GNU package contains the official R software environment, which is open-source free software that is distributed under the terms of the GNU General Public License. It is primarily written in C, Fortran, and R (partially self-implemented). There are precompiled executables available for different operating systems. There is a command-line interface for R.

TensorFlow: `TensorFlow` is a free and open-source software library for machine learning and artificial intelligence. It can be used across a range of tasks but has a particular focus on training and inference of deep neural networks. TensorFlow was developed by the Google Brain team for internal Google use in research and production. The initial version was released under Apache License 2.0 in 2015. Google released the updated version of `TensorFlow`, named `TensorFlow 2.0`, in September 2019. TensorFlow serves as the core platform and library for machine learning. It can be used in a wide variety of programming languages, including Python, JavaScript, C++, and Java. This flexibility lends itself to a range of applications in many different sectors [6].

scikit-learn: Is a free software machine learning library for the Python programming language. It features various classification, regression and clustering algorithms including support-vector machines, random forests, gradient boosting, k-means, and DBSCAN, and is designed to interoperate with the Python numerical and scientific libraries `NumPy` and `SciPy`. `Scikit-learn` is a `NumFOCUS` fiscally sponsored project [117].

PyTorch: PyTorch is a machine learning framework based on the Torch library, used for applications such as computer vision and natural language processing, originally developed by Meta AI and is now part of the Linux Foundation umbrella. It is free and open-source software released under the modified BSD license. Although the Python interface is more polished and the primary focus of development, `PyTorch` also has a `C++` interface [115].

A number of pieces of deep learning software are built on top of `PyTorch`, including Tesla Autopilot, Uber's Pyro, Hugging Face's Transformers, PyTorch Lightning, and Catalyst.

`PyTorch` provides two high-level features:

- Tensor computing (like `NumPy` [71]) with strong acceleration via graphics processing units (GPU), and
- Deep neural networks built on a tape-based automatic differentiation system.

2.5 Feature Engineering on Twitter Data

Online social networks have emerged as a brand-new platform that offers a forum for people to discuss their opinions and points of view on various topics and issues with their friends, family, and other associates. Through text, photos, audio, and video messages and posts, we can express our ideas, feelings, moments, and positions on particular social, national, and international issues. In fact, text remains one of the most popular ways to communicate in a social network despite the availability of other forms of communication. Reda and Kashifa [137] described the goal of their study to identify and examine the sentiment and emotion people express through text in their tweets, then use that information to generate recommendations. They gathered tweets and replies on a select few topics, creating a dataset that included text, user, emotion, sentiment, and other data. They assessed user influence scores based on various user- and tweet-based parameters and used the dataset to extract sentiment and emotion from tweets and their replies. They made use of the latter data to produce both general and specific user recommendations based on their Twitter activity. The methodology they used includes some intriguing novelties, including:

- Tweet replies in the dataset and measurements,
- Incorporating agreement score, sentiment score, and emotion score of replies in influence score calculation, and
- Generating general and personalized recommendations containing a list of users who agreed on the same topic and expressed similar emotions and sentiments towards that specific topic.

In another study, Md. Rakibul Hasan et al. examine Twitter data to analyze public views toward a product. First, in order to filter tweets, they created a pre-processed data framework based on NLP. Then, in order to analyze sentiment, they use a Bag of Words (BoW) approach with the term frequency-inverse document frequency (tf-idf) weighting. Their project aims to precisely categorize positive and negative tweets using BoW and tf-idf. The accuracy of sentiment analysis can be significantly increased by using the tf-idf vectorizer, and simulation results demonstrate the effectiveness of our suggested system. Using NLP over Twitter data, they conclude that they were able to analyze sentiment with an accuracy of 85.25% [72].

Twitter sentiment analysis gives businesses the ability to track public sentiment toward their products and events in real-time. The text pre-processing of Twitter data is the first step in the sentiment analysis process. The majority of current studies on Twitter sentiment analysis concentrate on the extraction of fresh sentiment features. Zhao and Gui disregarded a pre-processing technique in their study, to examine the importance of Twitter data pre-processing. The classification performances of six pre-processing methods using two feature models and four classifiers on five Twitter datasets were summarised in this paper, which also discussed the effects of text pre-processing methods on sentiment classification performance in two types of classification tasks. The experiments show that when using the pre-processing methods of expanding acronyms and replacing negation, as well as when removing URLs, removing numbers, and removing stop words, the accuracy and F1-measure of the Twitter sentiment classification classifier are improved. When different pre-processing techniques were used, the Naive Bayes and Random Forest classifiers were more sensitive than Logistic Regression and support vector machine classifiers [84].

Recent studies discusses the use of GPT-3, a powerful language model, for sentiment analysis on Twitter. Sentiment analysis involves determining the sentiment or emotion expressed in a text, such as whether a tweet is positive, negative, or neutral. GPT-3 is trained on a massive amount of text data, which allows it to understand the nuances of language. The model is able to analyze and classify tweets based on sentiment accurately, helping to gain insights into public opinion, track brand sentiment, or monitor customer feedback. Matt Payne in his article explains the process of using GPT-3 for sentiment analysis, which involves providing the model with prompts and receiving responses that classify the sentiment of a given tweet. It also discusses the benefits of using GPT-3, such as its scalability, flexibility, and ability to adapt to different contexts and topics. The article highlights how GPT-3 can be a valuable tool in sentiment analysis on Twitter, enabling organizations to make informed decisions based on public sentiment and feedback [116].

In the next chapter, we present the first study in Twitter feature engineering in financial forecasting of the Bitcoin price.

Chapter 3

Feature Engineering for Financial Forecasting of Bitcoin Price

We discuss in this chapter feature engineering which involves selecting or generating features that are relevant to predicting the future price of Bitcoin. Social media platforms like Twitter, Reddit, and other discussion forums provide a wealth of qualitative data that could be used to gauge market sentiment. Sentiments could be analyzed across various social platforms and incorporated into a forecasting model. In this study, related tweets to Bitcoin are collected from Twitter, in different historical periods, to analyze their sentiments, and to predict Bitcoin price fluctuations using different feature generation techniques, machine learning techniques, and time series.

3.1 Background

Cryptocurrency is a representation of digital property. More precisely, it is a system using universal encryption techniques that make it impossible to penetrate and manipulate. Cryptocurrency is using encryption for security purposes to protect virtual transactions and control the establishment of new units. Therefore, it is difficult to duplicate that currency, which is an intangible exchange medium (electronically), and a branch of the alternative currency. Moreover, this exchange medium is used to denote all these applications using Blockchain, whether these applications represent a digital currency or represent other things such as smart contracts [76]. Bitcoin, alongside other cryptocurrencies, became one of the most prominent trends recently due to its redefinition of the concept of money and its price fluctuation. Especially on social media, people keep discussing Bitcoin topics, consulting, and advising about cryptocurrency trading [33]. Bitcoin was the first cryptocurrency created in 2009 and since then other currencies have been created as competitive currencies such as

Litecoin, Nemcoin, and others. Cryptocurrencies have many advantages over standard currencies and previous electronic means of payment. Firstly, they do not have a central authority, unlike other electronic currency systems, such as PayPal. Secondly, they are recorded in a public record account, which is obtained by a consensus of a large distributed and decentralized system, where the virtual transactions are recorded in an integrated manner.

Cryptocurrencies are easily used to transfer the balance between the parties in transactions. These transfers are based on the use of public and private encryption keys for security purposes. The transfer of balances ends with lower processing fees, allowing users to avoid sharp shipping fees from most banks and financial institutions. In central banks, economic systems such as the Federal Reserve, the public sector, and governments control the currency vaults by printing fiat money. However, companies and governments cannot produce encrypted currency units, or in other words, there are no bonds provided to other companies, banks, and corporations entities where the value of assets measured in a decoded currency is decentralized and the currency is created by a complete technical and fundamental system where it is created by a known or unknown group or character such as the Bitcoin inventor.

The main disadvantage of cryptocurrencies is that they are virtual. They do not have a central store. A user's budget for an encrypted digital currency can be completely erased due to a computer failure if there are no backups of their holdings. Anonymous financial transactions in encrypted currency are, of course, perfectly suited to egregious activities such as tax evasion and money laundering. Therefore, encrypted currencies are not centralized giving security, in theory, from manipulation or government intervention. While there are hundreds of cryptocurrency features, most of them are derived from a couple of protocols, such as proof of work and verification of the test. All these encrypted currencies are reserved by the mining of encrypted coins and are equipped specifically with the specific computers or currency mining devices (ASIC Miner) to participate in the interaction and procedures of transactions.

Cryptocurrencies became a popular trend these days and have gained a lot of interest, especially from those interested in investing their money or looking for an alternative solution for speculation to increase their income. Figure 3.1 shows this rapid increase in the popularity of cryptocurrencies. Bitcoin has proven that this field is fertile for investing, given that its price has doubled at least 50,000 times in seven years. Mainly, coin value relies on demand and supply, and maybe other factors related to economic news or politics.

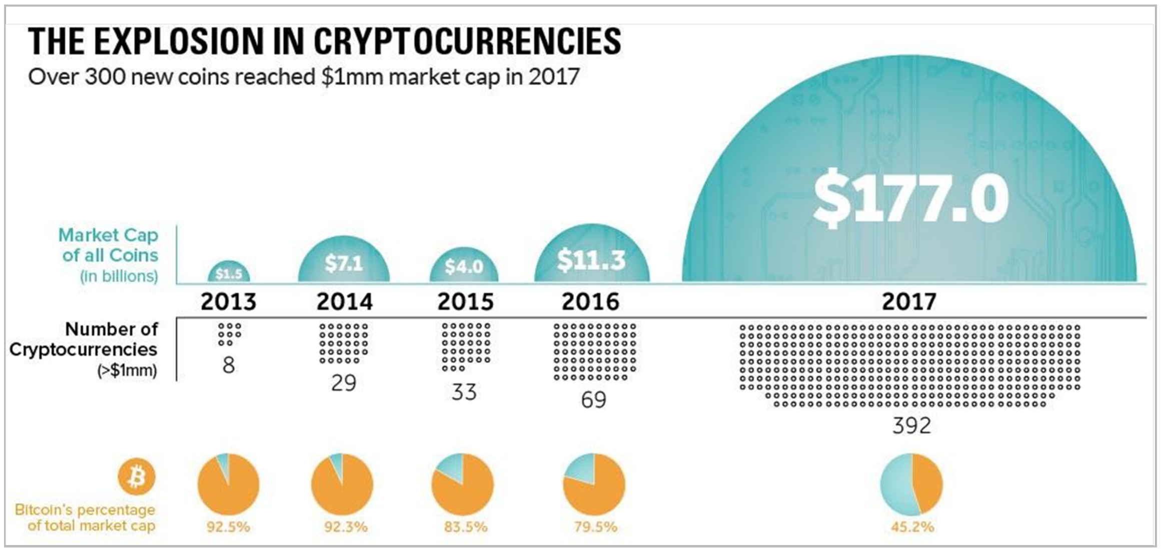


Figure 3.1: Chart that shows the rapid recent growth in popularity of cryptocurrencies

Convictions of people about cryptocurrencies have been changing, and cryptocurrencies are continuing to prove their existence as alternative money. Investing and speculation are two of the most attractive topics for people interested in increasing their income. Similarly, the cryptocurrency market has some features in common with the stock market, foreign currency market (forex), and other asset markets like crude oil, gold, and different valuable metals. Many factors may affect the change of price of different coins, related to the volume of demand and supply, and some other economic and political news and events. Investors and speculators need to have such tools that predict the increase and decrease in cryptocurrency prices and suggest to them what currency is better to invest in. It is helpful to take advantage of social media and trends about cryptocurrency and investigate if there is a strong correlation between people's posts and price changes of cryptocurrencies. Therefore, we investigate in this research these aspects:

- Is there a correlation between Twitter sentiment and Bitcoin fluctuation?
- Can a machine learning model based on polarity sentiment accuracy be used

for Bitcoin price prediction?

- Is automated sentiment classification better than manual labeling in Bitcoin price prediction?

This study has a positive perspective impact on the market of cryptocurrency that affects both the investor and the currency itself. If the investor could get rid of the fear of losing their money by having a tool that helps them predict prices and advise them as a consultant, they would be encouraged to invest more. Therefore, they will have higher profit potential. This may also be reflected in the currency, that people would be recommended to increase their demand for it—the more purchases of such a coin, the more its price and market capital increase.

Twitter as a social media platform, is rich in data that is growing rapidly, and calling e-commerce advocates to take advantage of these media services to build successful applications or extract hidden information. Among other applications, Bitcoin investors and traders require some applications that rely on social media data to help them in Bitcoin price forecasting.

Previous studies proposed a method for predicting the fluctuation of Bitcoin price considering the number of positive or negative tweets per single time period and tracing its fluctuation among the whole intervals. According to my research, there are no tools for Bitcoin price prediction that are considering machine learning for sentiment accuracy fluctuation over intervals. Traders who want to take advantage of social media search manually on social network platforms and look at the most current tweets. The main challenge is the huge number of tweets about Bitcoin per short period from one to five minutes range. This issue can be handled by focusing primarily on different selected times not exceeding 24 hours and removing irrelevant tweets. Another challenge is to have a cleaned dataset that is manually labeled to compare it with the automated sentiment approach.

The primary goal of this research is to design and build a machine-learning sentiment analysis that predicts the Bitcoin price fluctuation for future intervals. Twitter is the most suitable social medium to apply the study because of its popularity and the abundance of tweets about Bitcoin which may help in fluctuation prediction. Furthermore, Twitter may in the future develop an app or an extension in the same social network media through Twitter Developer, which helps in advising investors who invest in the cryptocurrency market. The system also could be expanded to include posts from Facebook and data from Google trends to find more accurate fluctuation. My primary research contributions are: (a) to compare different machine learning and deep learning methods on Bitcoin tweet sentiment and evaluate their performance; (b) to compare different language modeling approaches and sentiment analysis approaches; and (c) to predict the Bitcoin fluctuation using sentiment class accuracy using different machine learning approaches.

3.2 Related Work

The stock market is like the cryptocurrency market. A stock market is a platform to buy and sell stocks, and it has been driven by supply and demand, and so is the cryptocurrency market stock. The leading players in the stock market are the exchanges. Exchanges are where the sellers are matched with buyers to facilitate trading and help set the price of the shares. The stock exchange is an efficient way to create a favorable climate for active and growing new issues. Statistical algorithms and time series analysis have been used to predict the stock market. Abirami, R. and Vijaya, M.S, employed machine learning technology in computational finance since machine learning deals with techniques that allow computers to automatically learn to make accurate predictions based on past observations [121]. They used the WEKA environment [77] for training the dataset using linear regression and the LIBSVM tool for support vector regression.

In another study, Pagolu et al. used microblogging social media to predict stocks price because it is representing very well the public sentiment and opinion about current events [114]. Stock market prediction based on public sentiments expressed on Twitter has been a fascinating field of research. This study thesis observes how well the changes in stock prices of a company, the rises, and falls, are correlated with the public opinions being expressed in tweets about that company. Understanding the author's opinion from a text is the objective of sentiment analysis. The present paper has employed two different textual representations, Word2vec and n-gram, to analyze public sentiments in tweets. They applied sentiment analysis and supervised machine learning principles to the tweets extracted from Twitter and analyzed the correlation between the stock market movements of a company and sentiments in tweets. Elaborately, positive news and tweets on social media about a company would encourage people to invest in the stocks of that company. As a result, the stock price of that company would increase. The researchers concluded that the paper is shown that a strong correlation exists between the rise and fall in stock prices with the public sentiments in tweets.

Matta et al. explored if the rise of Bitcoin's cost is related to the volume of tweets or Web Search media outcomes. They compared patterns of cost with Google Trend Data, the volume of tweets, and especially with positive tweets [103]. Tweets are accessible and are effectively imported from Twitter Application Programming Interface (API). Composing the hashtag #Bitcoin or mention @Bitcoin, they assemble all tweets that specify the analyzed subject. Besides that, they imported data from Google trend and used DataStore (back-end database engine, utilizing MySQL as RDBMS), SentiStrenght tool, and Java Module. They collected more than 1,900,000 tweets, then analyzed them and identified positive and negative tweets. Using cross-correlation computation, they found similarities between Bitcoin Price with the number of tweets simultaneously, between Bitcoin price with the positive tweets, and Bitcoin price with Google trend.

Considering time series, Shah and Zhang used Bayesian Regression for predicting real-valued price of Bitcoin [143]. They used data related to Bitcoin cost obtained from the Okcoin.com cryptocurrency exchange company web site [113]. The total raw data was over 200 million. They used different t threshold—when the threshold is increased, the number of trades decreases, and the average holding time increases. At the same time, the average profit per trade increases. The strategy performs better in the middle section when the market volatility is high. In addition, the strategy is still profitable even when the price is decreasing in the last part of the testing period.

In another related research, Guo and Nino studied the capacity to make a short-term expectation of the Bitcoin trade cost fluctuation towards dollar [67]. They utilize the data of buy and sell orders collected also from Okcoin.com [113], which is one of the biggest Bitcoin exchanges. They developed a generative temporal mixture model of the instability and exchange arrange book data, which was collected through the exchange API, and it can beat the current state-of-the-art machine learning and time series statistical models. It consists of 13,730 hourly volatility observations and

701,892 order book snapshots. With the entryway weighting work of our generative temporal mixture model, they can identify administrations when the highlights of purchase and offer orders essentially influence the future tall instability periods. Besides, they also give bits of knowledge into the dynamical significance of highlights from arranged books such as showcase spread, profundity, volume, and ask/bid incline to clarify future short-term cost changes.

Krafft et al. conducted an online test to think about how vulnerable dealers in these markets are to peer impact from exchanging behavior [91]. They made bots that executed over one hundred thousand exchanges costing less than a penny each in 217 cryptocurrencies over the course of six months. They extracted data from the cryptocurrency exchange platform using API and conducted 310,222 randomly spaced trials. They discover of that people "buy" action-driven to short-term increments in ensuing buy-side activity hundreds of times the measure of our intercessions. From a planning point of view, we note that the plan choices of the trade we consider may have advanced this and other peer impacts, which highlights the potential social and financial effect of HCI in the plan of advanced teaching.

Stenqvist and Lonno in 2017 studied if sentiment analysis on Twitter data can help to indicate Bitcoin price fluctuation. They applied a naïve prediction model based on the number of sentiment fluctuations over time-series, using intervals from five minutes up to four hours and from one to four shifts [152]. By analyzing over two million tweets related to bitcoin within one month (day-to-day analyzing), they found that the most accurate aggregated time was a one-hour interval indicating a change after four hours. They mentioned that applying machine learning to the study may correlate further than the number of tweets.

Inspired by the previous studies, our study is learning the sentiment analysis of Bitcoin tweets, that collected from different periods of time, by generating tweets features, n-grams and Word2Vec embedding features, and compare the efficiency of those approaches by applying machine learning algorithms. After that, applying time-series to investigate the similarities between the Bitcoin price fluctuation and the tweet sentiment fluctuation using different time intervals.

3.3 Methodology

3.3.1 Social Media Data Selection

In this study, the dataset has been chosen from the social network Twitter based on the following reasons:

1. The availability of a huge number of training tweets, considering that Twitter users had already passed 300 million users, tweeting every second, on average, around 6,000 tweets, which corresponds to over 350,000 tweets sent per minute, 500 million per day, and around 200 billion per year. Therefore, Bitcoin's traders can find live feeds regarding Bitcoin.
2. The accessibility to collect Bitcoin tweets using a Twitter API tool.
3. Twitter is a microblogging social media platform; therefore, analyzing short blogs or 'tweets' is more manageable than analyzing articles with a big corpus.

3.3.2 System Architecture

The general form of the proposed system consists of three major parts: (1) data preprocessing part, which includes data importing from Twitter, data cleaning, data modeling process, and tweet's sentiment process; (2) data mining modeling, which includes multiple classification experiments and tests; and (3) finding events and correlations using time series. Figure 3.2 explains the phases of the proposed system architecture.

The data mining tools that have been chosen are Weka and R due to the abundance of research studies that are like this study and use the same tool. Weka is used to analyze the dataset information and to design the appropriate data mining model. R tool is used in the data preprocessing phase.

3.3.3 Data Preparation

Five lists of historical tweets related to Bitcoin have been collected to be applied in this study. Two of them are considered short datasets (less than 12 hours); `12Dec17` (Bitcoin tweets that collected on the 12th of December, 2017) and `23Mar18` (Bitcoin tweets that collected on the 23rd of March, 2018), and the other three are 24-hour tweets' datasets; `5Aug17` (Bitcoin tweets that collected on the 5th of August,

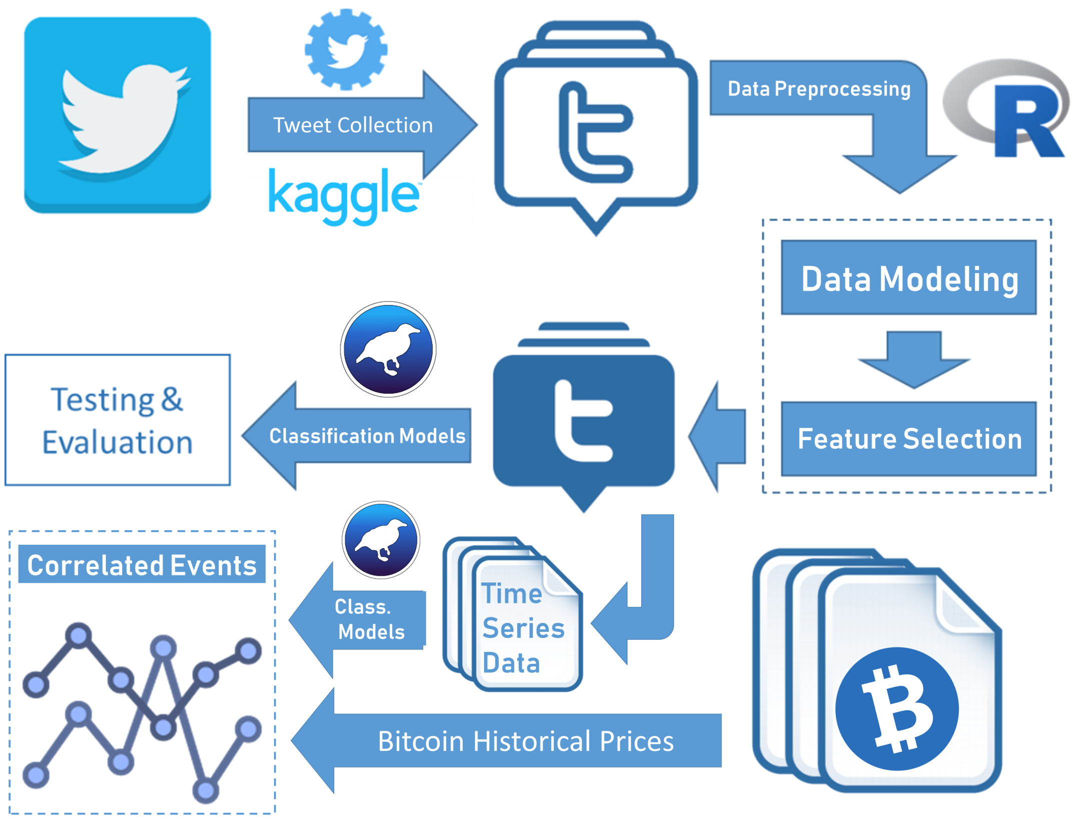
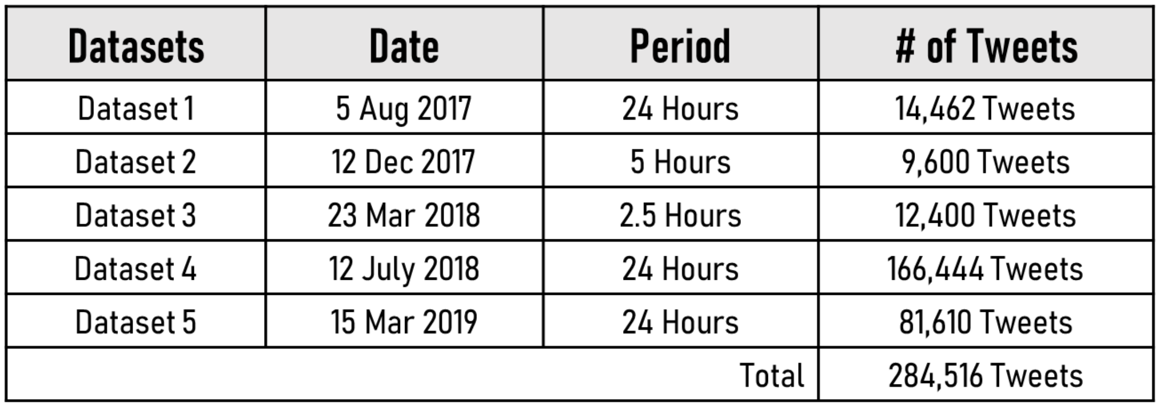


Figure 3.2: The proposed system architecture

2017), `12Jul18` (Bitcoin tweets that collected on the 12th of July, 2018), and `15Mar19` (Bitcoin tweets that collected on the 15th of March, 2019). The first short dataset (`12Dec17`) and are tweets were collected from a period that the price of Bitcoin was skyrocketing (December the 12th, 2017 — five hours period), and the second short dataset (`23Mar18`) are tweets were collected from a period that the price of Bitcoin had a significant drop (March the 23rd, 2018 — two hours and half period). The variety of datasets relying on the date and period and the number of tweets has been considered. All datasets were found in Kaggle.com and gripped using API tool from Twitter [85].

Table 3.1 shows the details of historical tweets and the number of tweets after removing irrelevant tweets. Some features such as tweet URL or Twitter user were removed because they are not relevant to the tweet sentiment analysis. Thus, we concluded that it is better to focus on the tweet text, which includes hashtags as well.



Datasets	Date	Period	# of Tweets
Dataset 1	5 Aug 2017	24 Hours	14,462 Tweets
Dataset 2	12 Dec 2017	5 Hours	9,600 Tweets
Dataset 3	23 Mar 2018	2.5 Hours	12,400 Tweets
Dataset 4	12 July 2018	24 Hours	166,444 Tweets
Dataset 5	15 Mar 2019	24 Hours	81,610 Tweets
		Total	284,516 Tweets

Table 3.1: The total of tweets are more than 282K after removing irrelevant tweets.

The data cleaning process is applied using a code implemented in the R language. All punctuations were removed, and all letters were transformed to uppercase. All unrelated tweets have been removed to become a total of 282K tweets.

3.3.4 Tweet Sentiment

Sentiment analysis is the use of NLP, computational linguistics, and textual analysis, in order to identify affective states and subjective information, whether positive, negative or neutral toward the topic of the text [160]. Fields of marketing, finance, customer service, and other areas, are considered the most common fields that are using the sentiment analysis. Sentiment analysis aims in general to determine the feelings of the speaker or writer towards a subject or identifying the dominant feeling on a document. These feelings can express the author's opinion (i.e., appraisal) or emotional state (i.e., the case of conscience of the author during the writing) or a deliberate feeling to be delivered (i.e., the influenced conscience that the author wants to deliver to the reader) [19].

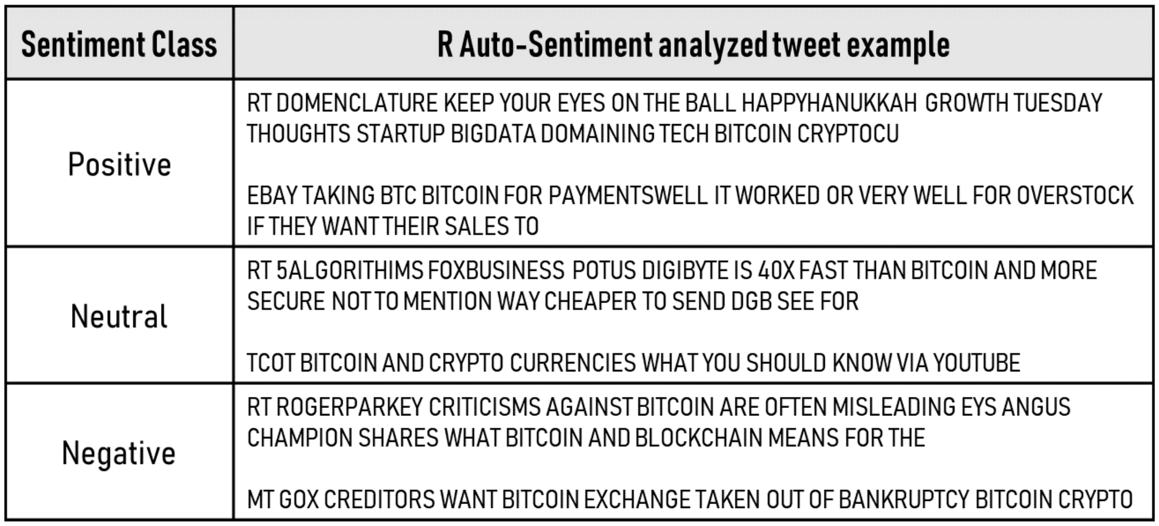
The polarity of opinion classification is considered a fundamental process in the analysis of sentiments in texts at the level of the entire document or sentence. The polarity classification refers to the determination of whether the opinion expressed in the document or sentence is positive, negative, or neutral. Other systems have evolved to look at specific emotional states such as anger, sadness, happiness, fear, etc. Sentiment analysis is widely used to evaluate the customer materials such as reviews, survey responses, social media, and materials for applications in many fields [105].

Social-Sentiment Analysis, which is also called "Opinion Mining" is a type of data extraction that measures the tendency or direction of individual opinions, which are used to extract and analyze personal information from websites. The analyzed data determine the public's feelings or reactions to certain products, people, or ideas. The term refers to the process by which consumers' feelings and opinions are analyzed on social media such as Facebook, Instagram, Twitter, and others, where individuals use these platforms to communicate with their friends, colleagues, and family and share their opinions, experiences, and feelings on a particular topic.

The rise of social media such as blogs and social networks has increased interest in sentiment analysis. Social media users share their views on a range of products, services, or news about an event or a brand. These views may enhance the event or brand's reputation or diminish it, and they may help consumers to take a position or make a decision that relies on the brand's reputation. Most sentiment analysis algorithms use simple terms to capture sentiment about a product or service. However, cultural factors, linguistic affairs, and a variety of contexts make it extremely difficult to turn a string of written text into a simple polarity sentiment. The shorter the string of text, the harder it becomes. However, sentiment analysis within microblogging such as Twitter has shown that it can be a valid online indicator of political and economic sentiment [19].

Two approaches to tweet sentiment analysis were applied: manual sentiment analysis and auto-tweet sentiment using the R language, also named as the acronym R-Auto Tweet Sentiment (RATS) [135]. In the manual sentiment approach, each tweet has been read, and it was decided to classify it into three different classes: positive, negative, and neutral. This is a time-consuming process in comparison to the RATS that can be used by the TwitteR package and get_nrc_sentiment(dataframe) function [1]. The RATS generate values ranging between 3 and -3, and the values between 1 and -1 were considered neutral. This will form two different versions of each dataset to be tested and to discover which one sentiment approach is more reliable and efficient in tweets sentiments.

[1]In the 'get_nrc_sentiment' function in R, NRC stands for "National Research Council". The function utilizes the NRC Word-Emotion Association Lexicon, which is a lexicon developed by the Computational Social Science Lab at the National Research Council in Canada. This lexicon maps words to a set of emotions and sentiments to provide sentiment analysis in text mining tasks.



Sentiment Class	R Auto-Sentiment analyzed tweet example
Positive	RT DOMENCLATURE KEEP YOUR EYES ON THE BALL HAPPYHANUKKAH GROWTH TUESDAY THOUGHTS STARTUP BIGDATA DOMAINING TECH BITCOIN CRYPTOCU EBAY TAKING BTC BITCOIN FOR PAYMENTSWELL IT WORKED OR VERY WELL FOR OVERSTOCK IF THEY WANT THEIR SALES TO
Neutral	RT 5ALGORITHIMS FOXBUSINESS POTUS DIGIBYTE IS 40X FAST THAN BITCOIN AND MORE SECURE NOT TO MENTION WAY CHEAPER TO SEND DGB SEE FOR TCOT BITCOIN AND CRYPTO CURRENCIES WHAT YOU SHOULD KNOW VIA YOUTUBE
Negative	RT ROGERPARKEY CRITICISMS AGAINST BITCOIN ARE OFTEN MISLEADING EYS ANGUS CHAMPION SHARES WHAT BITCOIN AND BLOCKCHAIN MEANS FOR THE MT GOX CREDITORS WANT BITCOIN EXCHANGE TAKEN OUT OF BANKRUPTCY BITCOIN CRYPTO

Table 3.2: Some examples of how RATS classify tweets to positive, neutral and negative sentiment.

Table 3.2 shows an example of how RATS classify tweets into positive, neutral, and negative sentiment.

3.3.5 Language Modeling Approaches

Language modeling approaches can be defined as probabilistic distribution modeling over sequence of words to provide context that helps in the natural language processing tasks. It has many modeling types, such as: Unigram and n-gram modeling, exponential modeling, neural modeling, and word embedding modeling [106].

In this study, we used three different language modeling approaches: Tweet embedding, n-gram, and a proposed method combining both generated features from embedding and n-gram.

1. Tweet Embedding: word embedding is one of the recognized approaches of modeling words into numbers to be used in text mining and deep learning. The Weka system has a tweet embedding function in a package named “Affective Tweet”. In this study, tweet embedding is applied using the “Affective Tweet” package which is mapping the tweets into vectors of real numbers. The package implements WEKA filters for calculating state-of-the-art affective analysis features [2] from tweets that

[2]The term ”state-of-the-art” in this context refers to the latest and most advanced techniques or methods available in the field of affective analysis for tweets. The Affective Tweet package in WEKA claims to implement filters (features) that are considered the cutting-edge or state-of-the-art

can be fed into machine learning algorithms [40]. Many of these features were drawn from the NRC-Canada System [3]. It also implements methods for building affective lexicons and distant supervision methods for training affective models from unlabelled tweets. [40]. The function `TweetToEmbeddingsFeatureVector` calculates a tweet-level feature representation using pre-trained word embeddings. A dummy word-embedding formed by zeroes is used for a word with no corresponding embedding. Finally, the average word-embedding is calculated and then a vector of 100 embedding values is generated for each tweet.

2. N-gram is another data modeling approach commonly used in natural language processing studies. In this process, a unique unigram/bigram vocabulary is built to use its tokens as features for each tweet [43]. Using Weka, `StringToWordVector` function is applied to generate a unigram/bigram keywords as features [161]. This function converts string attributes into numeric attributes representing word occurrence information from the text contained in the strings. The n-gram tokenizer is applied, eliminating terms that do not appear at least five times in a dataset. Moreover, the idf and tf transforms are applied in this process [141].

3. Tweet Embedding and n-gram proposed method combine the generated features in one data model. This is to observe if using both approaches together positively impacts classification accuracy.

Therefore, 30 different datasets are formed to be tested in the data mining process, 15 with manual sentiment and the rest with R-sentiment. These different feature datasets will be applied to machine learning algorithms to evaluate the best form of modeled features. Table 3.3 shows all forms of the dataset after tweet embedding and n-gram with the number of tweets for positive, negative, and neutral tweets.

---

in affective analysis, indicating that they have been developed or validated based on the most recent research advancements. And the "affective analysis feature" refers to a characteristic or attribute extracted from the text of tweets that reflects or captures the emotional or affective content. These features are designed to provide insights into the sentiment, emotion, or other affective aspects conveyed by the language used in the tweet.

[3] The NRC-Canada System refers to the National Research Council-Canada (NRC) System, which is a comprehensive suite of software tools and resources developed by the Computational Social Science Lab at the National Research Council of Canada [46].

Datasets			Positive Sentiment		Negative Sentiment		Neutral Sentiment		Total of Tweets
Name	Model	Features							
Dataset 1 5Aug17	Tweet Emb.	102	4,481	4,056	1,739	1,917	8,242	8,489	14,462
	N-Gram	471							
	Tw. Emb. + N-Gram	571							
Dataset 2 12Dec17	Tweet Emb.	102	1,523	3,139	1,503	1,427	6,621	5,081	9,647
	N-Gram	1147							
	Tw. Emb. + N-Gram	1247							
Dataset 3 23Mar18	Tweet Emb.	102	4,721	5,075	1,249	1,130	4,237	4,002	10,207
	N-Gram	1245							
	Tw. Emb. + N-Gram	1345							
Dataset 4 12Jul18	Tweet Emb.	102	46,008	36,268	25,064	10,757	95,372	119,419	166,444
	N-Gram	443							
	Tw. Emb. + N-Gram	543							
Dataset 5 15Mar19	Tweet Emb.	102	24,245	17,009	13,174	5,912	44,190	58,689	81,610
	N-Gram	441							
	Tw. Emb. + N-Gram	541							
30 forms of datasets			👤	R	👤	R	👤	R	282,370

Table 3.3: All forms of the dataset after tweet embedding and n-gram.

3.4 Machine Learning Methods

After finishing the preprocessing phase and having different remodeled datasets, it is evident that we need to apply supervised data mining techniques and Deep learning algorithms to predict sentiments.

3.4.1 Supervised Classification Methods

Our plan was to test the most common classification algorithms and compare the accuracy of predictions to each other, so that we could examine the effectiveness of different feature engineering techniques. Using Weka [162], the algorithms were chosen that rely on various theoretical classification method concepts: lazy learner method, Bayesian method, and decision tree method. Five different classification algorithms have been chosen: K-Nearest Neighbor (KNN) [22], Bayesian Network (BN) [131], Naïve Bayes (NB) [132], C4.5 Decision Tree Algorithm (J48) [140], Random Forest Decision Tree (RF) [41].

The `KNN` function in the R data mining tool from the package `{e1071}` was used to implement the experiments and to evaluate the algorithm for this study, as well as examining the algorithm IBK (Instance Based Learner) from the package(`weka.classifiers.lazy`) in the Weka data mining tool, which is the same as the KNN algorithm. We used k=3, with 100 batch size and LinearNNSearch as a nearest Neighbour Search Algorithm.

The `naiveBayes` function in R language in the package `e1071` was used to implement the experiments and tests to evaluate the algorithm for this study, as well as to examine the algorithm in the Weka data mining tool (the method `weka.classifiers.bayes.NaiveBayes`). We used 100 batch size for this experiment.

In addition, the Bayesian Network, which is a more general graphical probabilistic model that represents a set of variables and their conditional dependencies, was applied. The Bayesian Network learning is using various search algorithms and quality measures, provides data structures (network structure, conditional probability distributions, etc.) and facilities a number of common Bayes Network learning algorithms like `K2` and `B` [165]. Using Weka, the function `weka.classifiers.bayes.BayesNet` was used in the study to find the sentiment accuracy. The `K2` search algorithm, which was used, uses a hill-climbing algorithm restricted by an order on the variables. The initial network is a network with an arrow from the classifier node to each other node. A simple estimator is used for estimating the conditional probability tables of a Bayes network once the structure has been learned.

The `J48` decision tree function in Weka data mining tool was used to test and evaluate the algorithm for this study. `J48` is an open-source decision tree classifier written in Java. It represents the `C4.5` algorithm in Weka using `weka.classifiers.trees.J48` classifier class, which generates a pruned or unpruned C4.5 decision tree. For this study, an unpruned decision tree is applied. The J48 was the most time-consuming algorithm to find the sentiment accuracy for each dataset, especially with datasets that have a large number of features and instances.

The `randomForest` function in the R language in the package {`randomForest`} was used to implement the experiments and tests to evaluate the algorithm for this study, as well as to examine the algorithm in the Weka data mining tool using `weka.classifiers.trees.RandomForest` function. We used 100 batch size, 100% bag size, and 100 iterations.

All the classification methods were previously explained in Chapter 2.

3.4.2 Deep Learning Methods

Another set of methods used is a set of neural network methods, which are applied to compare their accuracy and efficiency. These methods are available in the Weka workbench using the deep learning package `WekaDeeplearning4j`, which is a Java library developed to incorporate the modern techniques of deep learning into Weka. The deep learning algorithms that applied are: Multilayer Perceptron (MLP) using `Dl4jMlpClassifier` function with two dense layers [165], from `WekaDeeplearning4j` deep learning package in Weka, Radial Basis Function Network (RBFN) [58], and Weightless Neural Network (WiSARD) [65].

`WekaDeeplearning4j` is a deep learning package for the Weka workbench. It is developed to incorporate the modern techniques of deep learning into Weka [163]. The backend is provided by the `Deeplearning4j` Java library.

Using the `Dl4jMlpClassifier` function, the MLP network was designed with two dense layers, as described in Figure 3.3. Since one epoch[4] is too big to feed to the computer at once, we used 32 epochs to perform. As the number of epochs increases, a greater number of times the weight is changed in the neural network, so the learning curve goes from underfitting to an overfitting curve [145]. The applied MLP model used a batch size[5] of 100, an instance iterator using `weka.dl4j.iterators.instance-.DefaultInstanceIterator` with a mini-batch size of one, and an iteration listener which evaluates the model while training every five epochs. The model consists of an input layer, two dense layers, and an output layer. The input layer variables are the total number of dataset attributes in addition to the total number of classes. The hidden layers are using the Rectified Linear Unit (ReLU) activation function[6], and output 100 values from the first hidden layer, and 30 values from the second hidden layer. The output layer outputs the classes using the multi-class cross-entropy (MCX-ENT) loss function, and softmax activation function, that support the multi-class classification, and transforms the neurons' output exponential to class probabilities distribution.

For the WiSARD model, The `WiSARD4WEKA` package implements a multi-class classification method based on the WiSARD weightless neural model for the Weka machine learning toolkit. We used a batch size of 100, like the previously applied neural network algorithms.

For the RBFN model, the `RBFNetwork` package is implements a multi-class classification method using Weka machine learning toolkit. Like MLP, the applied RBFN model used a batch size of 100, the number of instances to process if the batch prediction is being performed. More or fewer instances may be provided. For the K-means generation, three clusters are generated to learn. The clustering seed value, which is a random seed to pass on to K-means, was set to 1.

[4] "One Epoch" is when an entire dataset is passed forward and backward through the neural network only once.

[5] The "Batch Size" is the total number of training examples present in a single epoch.

[6] The "ReLU function" zeros any input value below zero and keep the same value for values greater than zero.

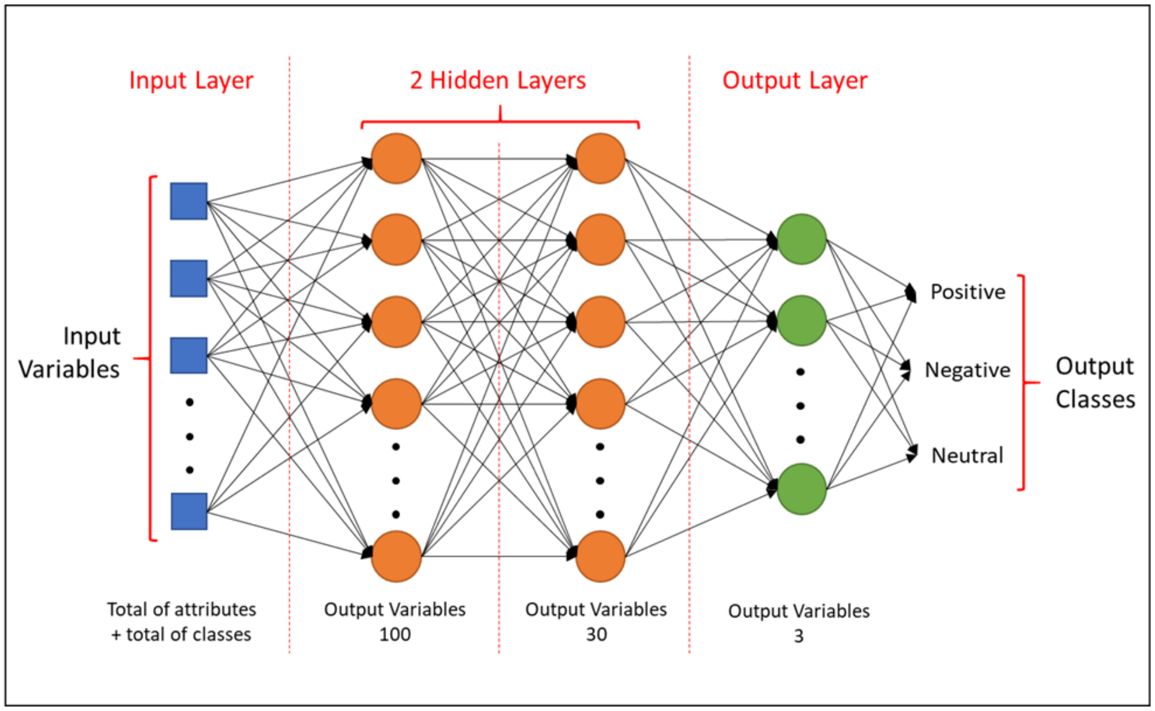


Figure 3.3: The applied Multi-layers perception model.

3.5 Price Fluctuation Prediction Methodology

Different time series frequencies are applied on the five primary datasets to find the correlation between Bitcoin price fluctuation and Twitter sentiment. The short datasets are divided into one set of intervals that is proper for the number of tweets for each dataset (15 minutes for the second dataset and five minutes for the third dataset). In contrast, the remaining datasets are divided into three sets of intervals ranging from 30 minutes up to 120 minutes. Each interval measures the fluctuation of positive sentiment accuracy and the negative sentiment accuracy (the actual positive value for each learned sentiment) and find the correlation with the Bitcoin (BTC) price change. The matching prediction events are counted depending on the learned sentiment fluctuation of the True Positive (TP), depending on which of the following four correlated criteria is matched:

- If the TP value of positive sentiment increases, and the BTC price increases in the next shift.
- If the TP value of positive sentiment decreased, and the BTC price decreased

in the next shift.

- If the TP value of negative sentiment increased, and the BTC price decreased in the next shift.
- If the TP value of positive sentiment decreased, and the BTC price increased in the next shift.

Figure 3.4 explains how the matching events are counted to predict BTC fluctuations.

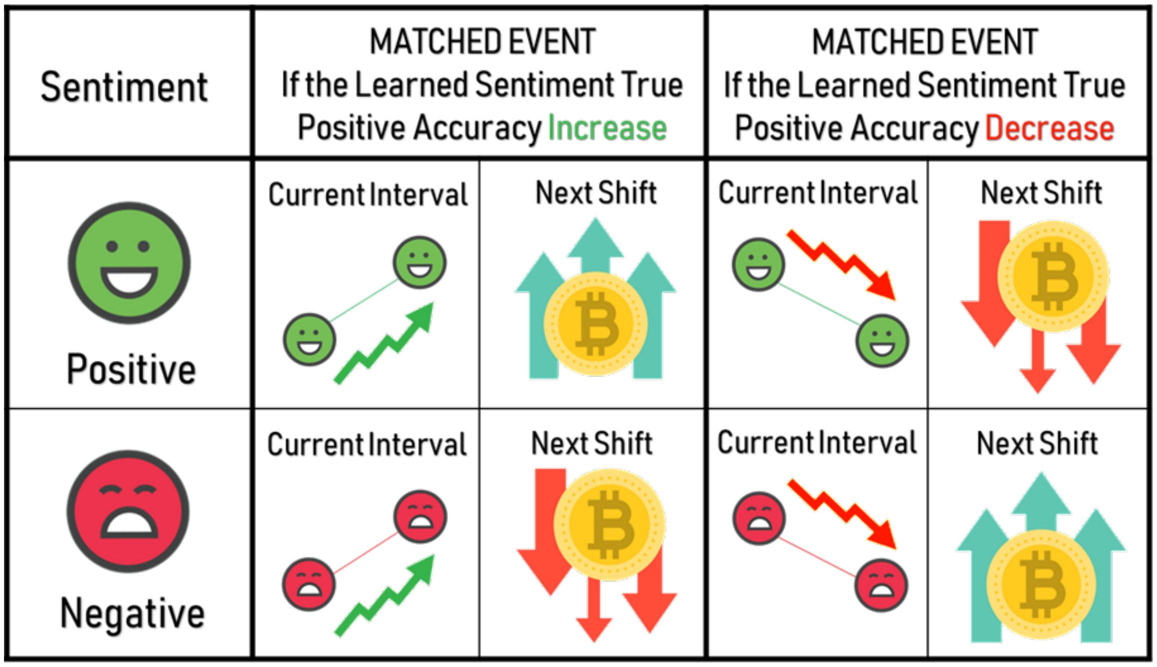


Figure 3.4: Counting matching events to predict price fluctuation depending on the sentiment classes TP value fluctuations.

3.6 Time-Series Analysis

The concept of Time-Series is a set of measurements recorded for one or more variables arranged according to their occurrence in time, and given specific phenomenon values. Time series is considered one of the most important methods of predicting the future through the facts of the past and current times. The most important time series are those of economic indicators, annual sales of companies in all aspects of their activities, education, population size, and the like. The change that occurs in the values of the time series variable is a function of time that can be represented graphically by taking the horizontal axis of time and vertical values of the variable. Time series analysis includes techniques for analyzing time series data to derive meaningful statistics and other data characteristics, while time series forecasting is the use of a model to predict future values based on values observed before [69].

To find the correlation between Bitcoin price fluctuation and Twitter sentiment, different time series frequencies are applied to the five main datasets.

The two short datasets, 12Dec17 and `23Mar18`, are divided into one set of intervals for each. That set of intervals is proper for the number of tweets that it contains (fifteen minutes for dataset #2 and five minutes for dataset #3), while the remaining datasets, `5Aug17`, `12Jul18` and `15Mar19` are divided into three sets of intervals ranging from 30 minutes up to 120 minutes. Each interval will measure the fluctuation of positive sentiment accuracy (the true positive value) and the negative class accuracy, then try to find the correlation with the Bitcoin price change. Two interval shifts will be applied to predict the matching event, single shift, and double shift.

The matching event for the positive class sentiment will be any rise or descent for the true positive value of the positive class in one interval with the same fluctuation of the Bitcoin price in the next shift. While matching event for the negative class sentiment will be any rise or descent for the true positive value of the negative in one interval with the "opposite" Bitcoin price fluctuation in the next shift. For example, if the true positive value of the negative class is raised up and the Bitcoin price drops in the next shift, this will be counted as an event for negative sentiment.

3.7 Proposed Methodology Framework

The framework diagram is divided into three phases which are the main study phases: the pre-processing phase, which includes the language modeling process; the sentiment learning phase, which includes supervised learning and deep learning; and the BTC fluctuation prediction phase, as shown in Figure 3.5.

In the diagram, several modeled datasets were generated and considered. Then after evaluating the sentiment learning, one language model with the most stable tweet sentiment dataset in performance will be selected to continue the third phase. In the third phase, the selected datasets will be segmented. Each segment will be

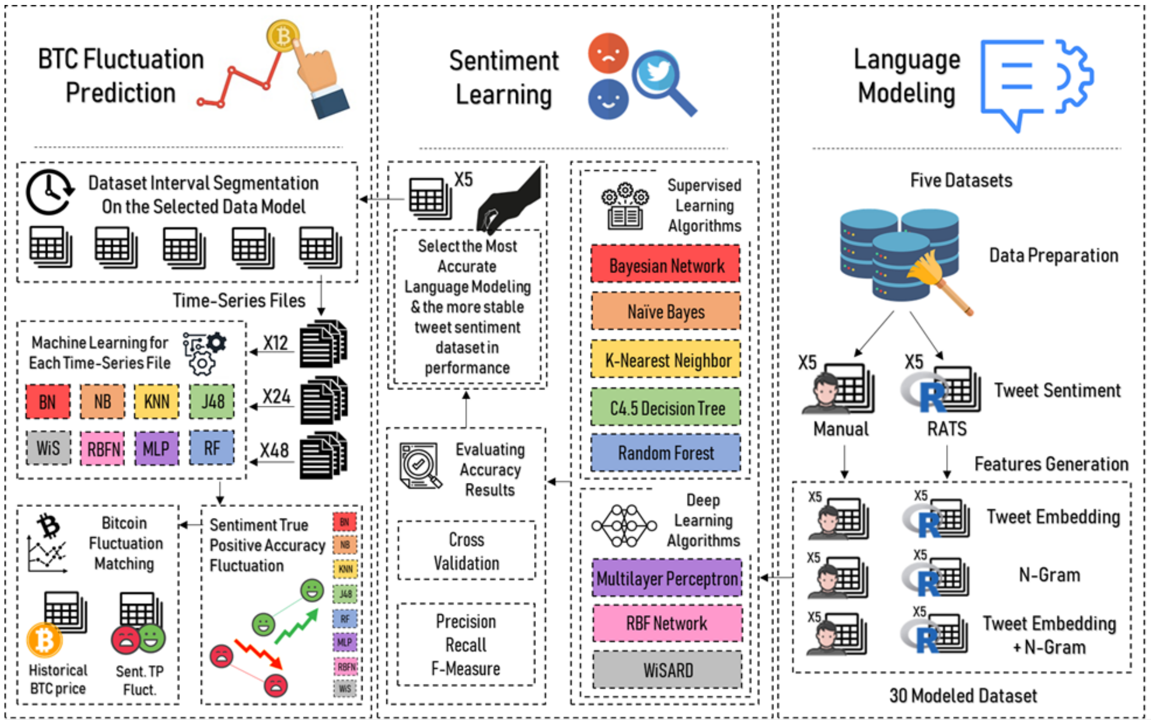


Figure 3.5: The proposed framework methodology for the main study phases

learned using the applied machine learning algorithms in this study to find the true positive score for both positive and negative sentiments. Further studies and results are conducted and discussed in the Experimental Results section to evaluate the three phases of methodologies.

3.8 Experimental Results

The research investigates three aspects: whether automated sentiment classification (RATS) is better than manual labeling in the sentiment learning process, whether a machine learning model based on polarity sentiment accuracy can be used for Bitcoin price prediction, and if there is a correlation between Twitter sentiment and BTC fluctuation. Therefore, some evaluation plans, tests, and time-series frequencies are applied to understand the interpretations of the results and to answer the previous aspects.

Classifiers / Modeled Dataset		BN		NB		KNN		J48		RF	
5Aug17	Tweet Embedding	50.98%	71.18%	50.34%	62.12%	66.43%	86.28%	62.09%	80.99%	68.52%	86.83%
	N-Gram	57.06%	77.41%	62.74%	65.57%	68.04%	85.61%	68.53%	85.90%	69.82%	88.50%
	Tweet Emb. + N-Gram	52.25%	61.96%	52.44%	71.66%	66.22%	85.85%	66.74%	84.27%	69.58%	86.62%
12Dec17	Tweet Embedding	76.13%	69.65%	54.31%	64.25%	85.92%	83.34%	80.54%	77.27%	87.56%	83.59%
	N-Gram	85.37%	79.56%	50.68%	57.99%	89.33%	83.25%	89.69%	81.64%	92.10%	86.93%
	Tweet Emb. + N-Gram	82.28%	73.54%	51.82%	58.13%	88.91%	84.08%	88.16%	81.45%	88.11%	84.08%
23Mar18	Tweet Embedding	81.92%	76.41%	49.74%	55.60%	88.79%	87.52%	83.99%	81.90%	89.67%	87.65%
	N-Gram	86.56%	75.45%	70.03%	66.50%	88.93%	86.03%	90.75%	85.22%	93.63%	88.44%
	Tweet Emb. + N-Gram	85.51%	80.01%	71.42%	67.11%	89.55%	87.04%	90.59%	85.43%	90.88%	87.68%
12Jul18	Tweet Embedding	52.45%	73.78%	48.39%	51.81%	69.07%	84.57%	72.64%	86.11%	73.96%	83.36%
	N-Gram	66.40%	81.21%	41.32%	39.92%	73.53%	85.88%	72.59%	87.48%	73.43%	88.92%
	Tweet Emb. + N-Gram	67.74%	80.02%	42.54%	41.60%	71.81%	87.51%	71.60%	87.51%	74.89 %	89.81%
15Mar19	Tweet Embedding	77.83%	67.64%	54.25%	67.72%	86.56%	86.34%	81.84%	83.44%	80.94%	85.34%
	N-Gram	82.48%	82.01%	66.92%	63.73%	88.75%	86.54%	84.95%	85.01%	83.93%	87.92%
	Tweet Emb. + N-Gram	85.12%	85.59%	58.19%	62.11%	89.12%	90.15%	88.46%	89.69%	90.01%	91.58%
		👤	R	👤	R	👤	R	👤	R	👤	R

Table 3.4: Results of accuracies of the classical classifier on the trained data

3.8.1 Sentiment Learning

Several single classifiers were trained and tested to evaluate their accuracies. Table 3.4 shows the results of the accuracies of each classifier on the trained data. The results show convergence on classifiers results, except the Naïve Bayes algorithm, which had the lowest results in all datasets. On the other hand, decision tree methods perform better in most experiments, especially in the n-gram data models.

For the deep learning classifiers, MLP and WiSARD show a significant improvement in n-gram, and tweet embedding with the n-gram data models, and it shows that manual sentiment performs better than RATS in two datasets, 12Dec17 and `23Mar18`. However, RATS gives more stable performance overall datasets (see Table 3.5).

In general, all experiments are performed a 5-folds cross-validation test. The experiments show a higher degree of accuracy of deep learning algorithms over the classical machine learning classification methods except Random Forest (RF). Manual sentiment gives better results in some datasets, while RATS is efficient in most experiments except with Naïve Bayes and Bayesian Network algorithms. Best accuracies were achieved with the manual sentiment using MLP deep learning algorithm and WiSARD, and Tweet-Embedding with n-gram data modeling. Therefore, the

Modeled Dataset \ Classifiers		MLP		RFBN		WiSARD	
5Aug17	Tweet Embedding	62.78%	74.82%	58.15%	75.38%	58.27%	65.50%
	N-Gram	64.91%	83.58%	61.95%	74.32%	61.39%	66.89%
	Tweet Emb. + N-Gram	66.49%	86.12%	62.55%	75.71%	63.06%	81.79%
12Dec17	Tweet Embedding	84.39%	81.07%	73.96%	62.34%	74.74%	67.83%
	N-Gram	91.26%	84.97%	76.25%	78.78%	90.69%	83.94 %
	Tweet Emb. + N-Gram	91.033%	85.32%	77.64%	78.30%	91.13%	84.83%
23Mar18	Tweet Embedding	88.40%	84.49%	58.76%	62.87%	53.65%	51.20%
	N-Gram	91.82%	86.12%	87.29%	77.34%	90.49%	82.18 %
	Tweet Emb. + N-Gram	92.97%	88.26%	87.45%	77.51%	91.12%	86.17%
12Jul18	Tweet Embedding	68.56%	77.83%	59.43%	76.34%	61.38%	76.49%
	N-Gram	70.97%	84.93%	62.73%	75.11%	64.58%	82.53%
	Tweet Emb. + N-Gram	71.15%	85.84%	60.96%	77.59%	59.95%	83.37%
15Mar19	Tweet Embedding	82.94%	83.49%	68.49%	67.37%	65.34%	64.88%
	N-Gram	81.37%	83.22%	82.73%	75.34%	86.44%	85.37 %
	Tweet Emb. + N-Gram	83.97%	83.38%	70.35%	74.64%	75.39%	87.49%
		👤	R	👤	R	👤	R

Table 3.5: Results of accuracies of each deep learning classifier on the trained data

Algorithm / Dataset		BN			NB			KNN			J48			RF		
		Positive	Negative	Overall	Positive	Negative	Overall	Positive	Negative	Overall	Positive	Negative	Overall	Positive	Negative	Overall
5Aug17	Precision	0.871	0.201	0.827	0.456	0.174	0.746	0.760	0.677	0.853	0.622	0.703	0.853	0.966	1.000	0.882
	Recall	0.328	0.700	0.717	0.408	0.683	0.620	0.663	0.581	0.859	0.643	0.359	0.861	0.530	0.379	0.866
	F1	0.476	0.312	0.738	0.431	0.278	0.663	0.708	0.625	0.855	0.717	0.475	0.850	0.685	0.550	0.849
12Dec17	Precision	0.649	1.000	0.836	0.515	0.271	0.745	0.819	0.879	0.887	0.748	0.798	0.866	0.994	0.995	0.898
	Recall	0.646	0.455	0.823	0.620	0.858	0.518	0.735	0.802	0.889	0.695	0.823	0.869	0.566	0.683	0.881
	F1	0.648	0.626	0.811	0.563	0.412	0.545	0.774	0.839	0.887	0.721	0.792	0.867	0.721	0.810	0.872
23Mar18	Precision	0.923	0.985	0.871	0.868	0.374	0.781	0.934	0.923	0.899	0.926	0.849	0.906	0.917	0.999	0.912
	Recall	0.818	0.681	0.855	0.760	0.826	0.714	0.878	0.826	0.896	0.925	0.809	0.906	0.923	0.744	0.909
	F1	0.867	0.805	0.855	0.811	0.515	0.731	0.906	0.872	0.896	0.926	0.828	0.906	0.920	0.853	0.908
12Jul18	Precision	0.548	0.573	0.681	0.576	0.205	0.625	0.656	0.577	0.713	0.656	0.575	0.711	0.776	0.787	0.755
	Recall	0.645	0.430	0.677	0.354	0.749	0.425	0.623	0.519	0.718	0.627	0.502	0.716	0.558	0.438	0.749
	F1	0.593	0.491	0.677	0.439	0.322	0.458	0.639	0.547	0.715	0.641	0.536	0.713	0.649	0.563	0.734
15Mar19	Precision	0.884	0.796	0.852	0.515	0.366	0.639	0.885	0.841	0.891	0.865	0.830	0.884	0.954	0.996	0.910
	Recall	0.754	0.746	0.851	0.570	0.623	0.582	0.862	0.824	0.891	0.864	0.809	0.885	0.818	0.750	0.900
	F1	0.814	0.770	0.849	0.541	0.461	0.596	0.873	0.832	0.891	0.865	0.819	0.884	0.881	0.856	0.898

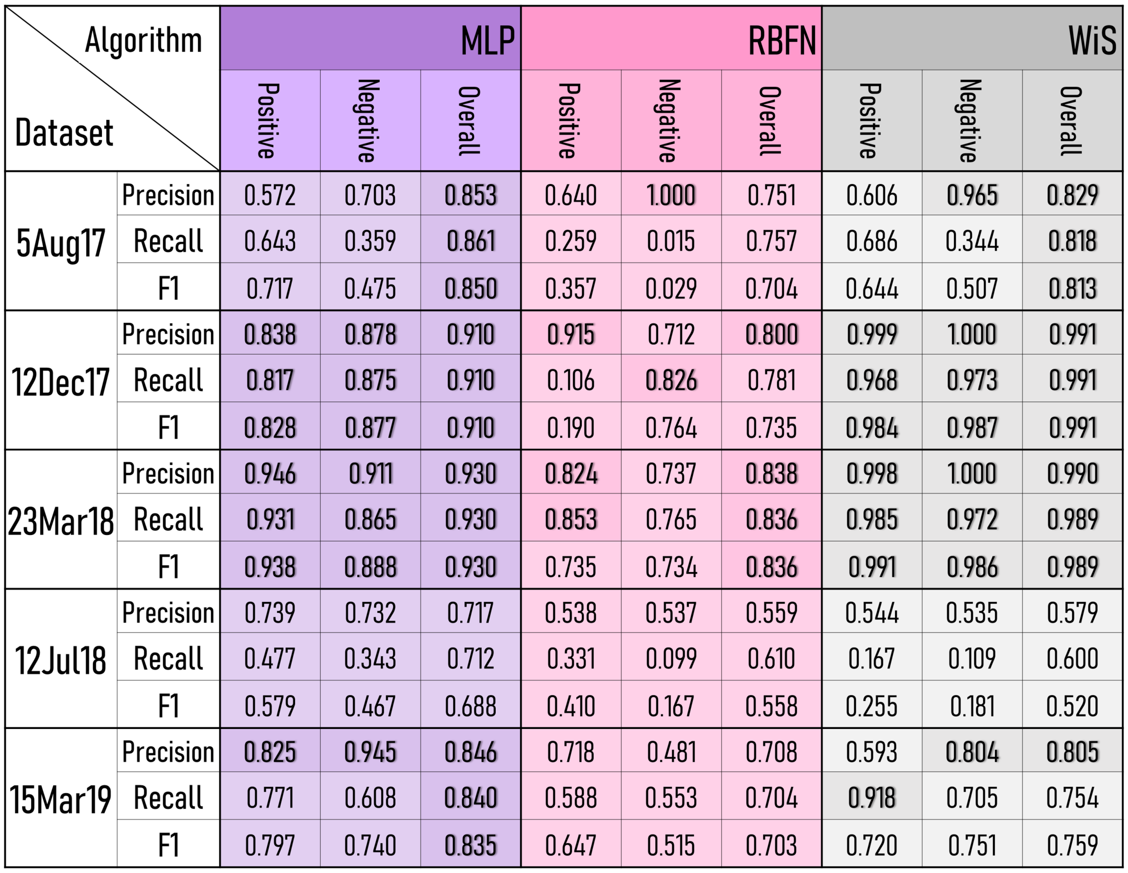
Table 3.6: Precision, recall an F1 score of the classical classifiers on the trained data

Tweet-embedding with n-gram modeling features will be applied in the time-series experiments. To evaluate the results, several measures are considered besides the algorithm accuracy: precision, recall, an F1 score. The results of these measures of all applied machine learning algorithms over all datasets are shown in Table 3.6 for classical machine learning algorithms, and Table 3.7 for deep learning algorithms, focusing on positive and negative classes. The tables have a heat-map visualization feature with the bold font for values that range from 80% to 100%.

3.8.2 Time Series Analysis

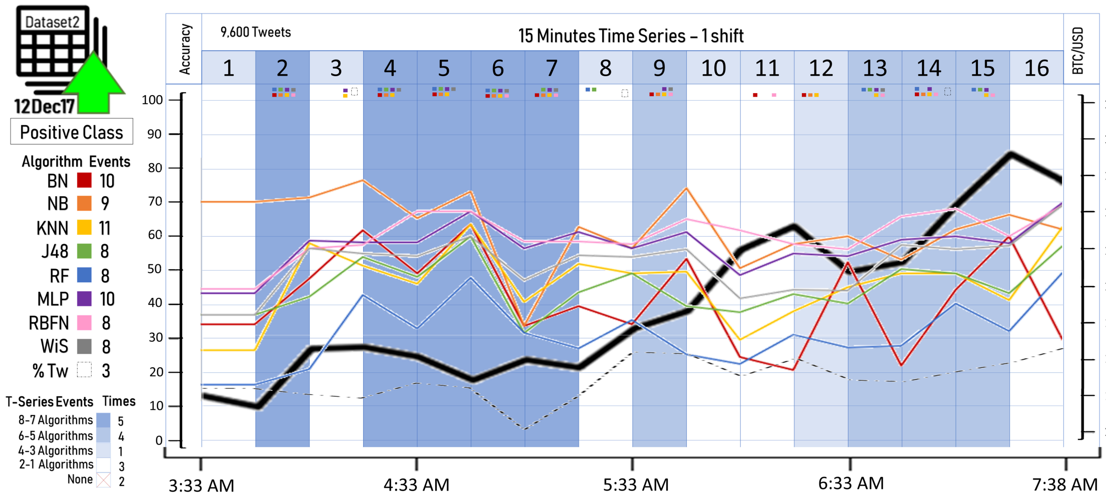
Relying on the true positive values of the positive sentiment class and the negative sentiment class, a collection of visualization charts has been built to visualize and note the events on the applied time-series. An event matches the rising and descent of the algorithm's true positive value in an interval, with the rise and descent of the Bitcoin price on the next shift.

Figure 3.6 and Figure 3.7 present a sample of what the time series charts look like. The figures show that fluctuations of the sentiment classes occur in more events than



Algorithm / Dataset		MLP			RBFN			WiS		
		Positive	Negative	Overall	Positive	Negative	Overall	Positive	Negative	Overall
5Aug17	Precision	0.572	0.703	0.853	0.640	1.000	0.751	0.606	0.965	0.829
	Recall	0.643	0.359	0.861	0.259	0.015	0.757	0.686	0.344	0.818
	F1	0.717	0.475	0.850	0.357	0.029	0.704	0.644	0.507	0.813
12Dec17	Precision	0.838	0.878	0.910	0.915	0.712	0.800	0.999	1.000	0.991
	Recall	0.817	0.875	0.910	0.106	0.826	0.781	0.968	0.973	0.991
	F1	0.828	0.877	0.910	0.190	0.764	0.735	0.984	0.987	0.991
23Mar18	Precision	0.946	0.911	0.930	0.824	0.737	0.838	0.998	1.000	0.990
	Recall	0.931	0.865	0.930	0.853	0.765	0.836	0.985	0.972	0.989
	F1	0.938	0.888	0.930	0.735	0.734	0.836	0.991	0.986	0.989
12Jul18	Precision	0.739	0.732	0.717	0.538	0.537	0.559	0.544	0.535	0.579
	Recall	0.477	0.343	0.712	0.331	0.099	0.610	0.167	0.109	0.600
	F1	0.579	0.467	0.688	0.410	0.167	0.558	0.255	0.181	0.520
15Mar19	Precision	0.825	0.945	0.846	0.718	0.481	0.708	0.593	0.804	0.805
	Recall	0.771	0.608	0.840	0.588	0.553	0.704	0.918	0.705	0.754
	F1	0.797	0.740	0.835	0.647	0.515	0.703	0.720	0.751	0.759

Table 3.7: Precision, recall an F1 score of each deep learning classifier on the trained data



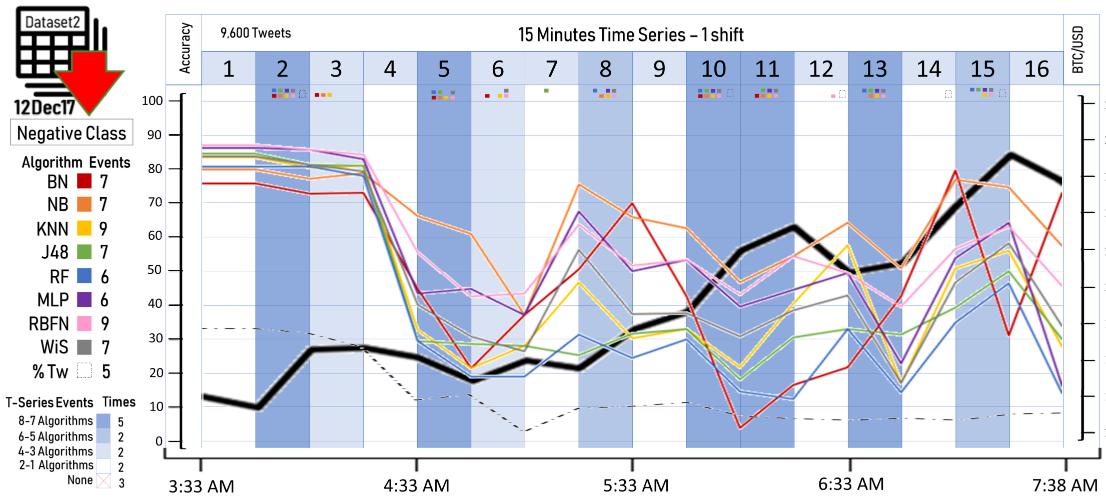
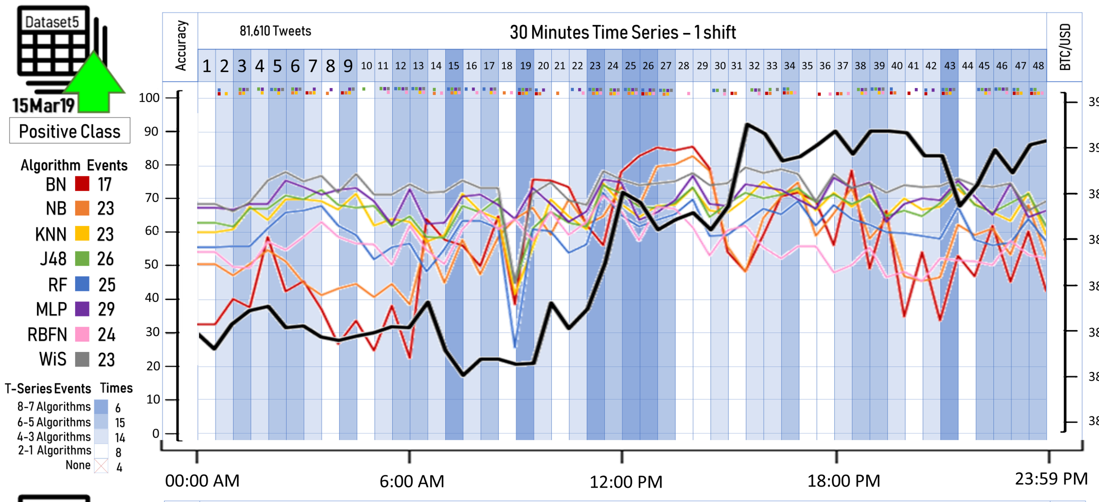


Figure 3.6: Time-series with 15 minutes interval chart of the second dataset and one shift to the future 15 minutes and compare it with the price fluctuations.

the percentage of positive and negative tweets fluctuations between the intervals and compare its fluctuation with the Bitcoin price fluctuation. Bitcoin historical price data were found on bitcoincharts.com, and it contains the BTC/USD rate for every minute. Using smoothing, the BTC price in each interval was set up to the closing price at the end of the interval. Moreover, single and double shifts are considered to find events in each dataset.

We can realize from the charts that there are some matchings in both positive and negative sentiment class accuracy and can be used to expect the price change. For example, in Figure 3.6 (that present the fluctuation of true positive accuracy over the 12Dec17 intervals) in intervals 4, 5, 6, and 7, positive sentiment accuracy had a



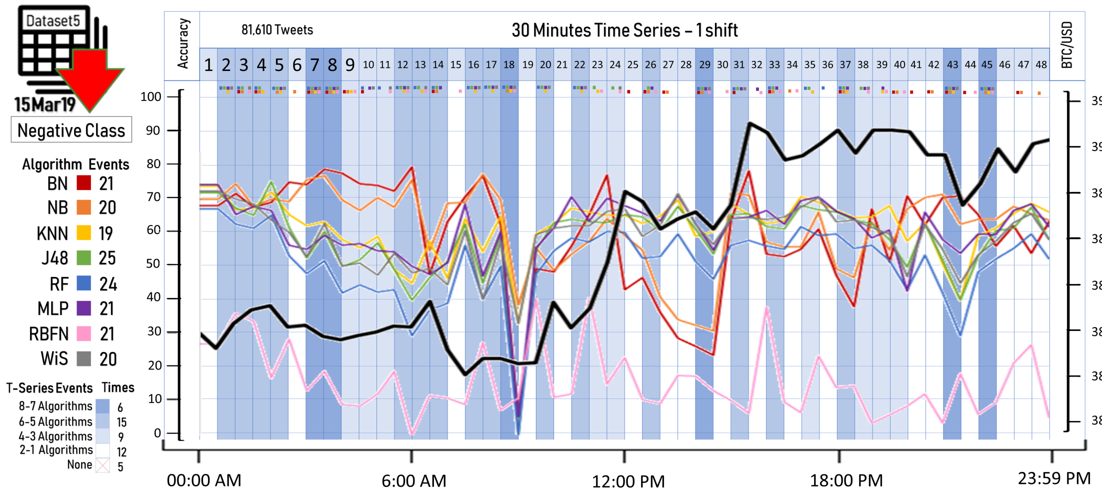


Figure 3.7: Time-series of 30 minutes interval chart of the fifth dataset one shift to the future 30 minutes, in comparison to price fluctuations.

strong value of event rate because seven to eight of the used algorithms almost predict the right fluctuation. In contrast, in the negative sentiment accuracy, intervals 4, 6, and 7 do not have the same strong value. On the other hand, intervals 10 and 11 in the negative sentiment accuracy had a very strong rate. However, it has a very weak value in positive sentiment accuracy for the same intervals. Therefore, considering both sentiments is helpful to predict the fluctuation.

We can look at the tables Table 3.8 and Table 3.9 to see overall events in all datasets with all shifts and intervals and the percentages of all the algorithms that have accuracy 50% or more of the predicted events. The results show a preponderance of the events of the positive class sentiment over the negative class sentiment in matched events. The results also showed that the Naïve Bayes algorithm had a higher percentage of predicted events on both positive and negative sentiment together.

Two evaluation tests are applied to find the correlation between the class sentiment accuracy and the Bitcoin price: the student's t-test and the Chi-square test. Table 3.10 and Table 3.11 show the associated p-values of the positive class sentiment with the Bitcoin price for the machine learning classification methods and the deep learning methods. To confirm a statistical significance, a p-value at the 0.05 probability level was determined. The results show a higher correlation in t-test using WiSARD deep learning algorithm than other algorithms. The tables also show that `5Aug17` with 30-minutes interval time-series and `23Mar18` with 5-minutes interval time-series had a signature p-value with all algorithms in both t-students test and Chi-square test. The evaluation tests showed a partial correlation between the fluctuation of the sentiment classes using different machine learning algorithms and the fluctuation of the Bitcoin price. Short intervals charts have more correlated values to the price fluctuation. The scores confirm that MLP, WiSARD, and decision tree methods correlate better among all used algorithms in the study.

3.9 Conclusion

Tweet sentiment analysis is an active field for studies in price forecasting. Using Twitter in sentiment analysis for Bitcoin is becoming an essential step for most researchers due to the many news feeds per minute regarding Bitcoin. Therefore, text

Algorithm / Dataset			BN		NB		KNN		J48		RF		MLP		RBFN		WiS		Total Events	
			P	N	P	N	P	N	P	N	P	N	P	N	P	N	P	N	P	N
5Aug17	30MinTS	1 Shift	24	25	25	23	19	22	24	20	26	19	26	21	22	19	20	20	186	169
		2 Shifts	21	22	23	22	25	23	16	23	20	24	26	24	26	26	28	27	185	191
	60MinTS	1 Shift	10	9	8	14	15	12	14	12	9	11	12	10	12	11	11	9	91	88
		2 Shifts	11	12	10	10	8	12	7	11	5	12	9	11	10	9	7	12	67	89
	120MinTS	1 Shift	5	4	5	3	7	4	7	3	7	4	8	6	10	3	7	3	56	30
		2 Shifts	7	4	5	5	6	5	4	5	6	2	6	3	4	5	5	5	43	34
12Dec17	15MinTS	1 Shift	10	7	9	7	11	9	8	7	8	6	10	6	8	9	8	7	72	58
		2 Shifts	4	5	5	7	8	6	7	6	5	7	7	7	7	6	7	6	50	50
23Mar18	5MinTS	1 Shift	7	9	11	10	12	13	8	8	8	10	10	11	10	8	10	8	76	77
		2 Shifts	9	9	9	12	9	11	9	9	8	12	10	11	9	10	11	12	74	86
12Jul18	30MinTS	1 Shift	26	24	22	18	24	25	24	26	22	21	22	23	22	22	28	26	190	185
		2 Shifts	24	16	22	24	21	27	24	26	25	28	21	22	23	23	23	25	183	191
	60MinTS	1 Shift	11	10	11	15	12	15	9	16	12	13	7	13	14	12	10	14	86	108
		2 Shifts	15	12	12	10	11	9	9	10	9	8	15	9	9	11	9	10	89	79
	120MinTS	1 Shift	7	2	9	5	7	4	7	4	7	5	5	4	6	3	4	3	52	30
		2 Shifts	4	6	2	6	5	4	3	4	5	5	7	6	4	5	8	7	38	43
15Mar19	30MinTS	1 Shift	17	21	23	20	23	19	26	25	25	24	29	21	24	21	23	20	190	171
		2 Shifts	26	23	29	27	17	26	20	26	23	23	20	28	24	25	24	25	183	203
	60MinTS	1 Shift	12	13	13	17	7	11	7	11	9	12	11	14	10	17	6	13	75	108
		2 Shifts	9	10	12	12	15	9	15	7	18	10	11	10	16	6	16	7	112	71
	120MinTS	1 Shift	6	4	5	3	5	4	6	3	6	4	7	6	7	3	6	3	48	30
		2 Shifts	7	5	6	5	3	4	4	5	4	4	4	4	3	3	6	3	37	33
		Total	272	252	276	275	270	274	258	267	267	264	283	270	280	257	277	265	2183	2124

Table 3.8: Overall events in all datasets with all shifts and intervals

Algorithm / Dataset			BN		NB		KNN		J48		RF		MLP		RBFN		WiS		# Alg. >= 50%	
			P	N	P	N	P	N	P	N	P	N	P	N	P	N	P	N	P	N
5Aug17	30MinTS	1 Shift	52%	54%	54%	50%	41%	48%	52%	43%	57%	41%	57%	46%	48%	41%	43%	43%	5	2
		2 Shifts	46%	48%	50%	48%	54%	50%	35%	50%	43%	52%	57%	52%	57%	57%	61%	59%	5	5
	60MinTS	1 Shift	45%	41%	36%	64%	68%	55%	64%	55%	41%	50%	55%	45%	55%	50%	50%	41%	5	5
		2 Shifts	50%	55%	45%	45%	36%	55%	32%	50%	23%	55%	41%	50%	45%	41%	32%	55%	1	5
	120MinTS	1 Shift	50%	40%	50%	30%	70%	40%	70%	30%	70%	40%	80%	60%	100%	30%	70%	30%	8	1
		2 Shifts	70%	40%	50%	50%	60%	50%	40%	50%	60%	20%	60%	30%	40%	50%	50%	50%	6	5
12Dec17	15MinTS	1 Shift	71%	50%	64%	50%	79%	64%	57%	50%	57%	43%	71%	43%	57%	64%	57%	50%	8	5
		2 Shifts	29%	36%	36%	50%	57%	43%	50%	43%	36%	50%	50%	50%	50%	43%	50%	43%	5	3
23Mar18	5MinTS	1 Shift	39%	50%	61%	56%	67%	72%	44%	44%	44%	56%	56%	61%	56%	44%	56%	44%	5	5
		2 Shifts	50%	50%	50%	67%	50%	61%	50%	50%	44%	67%	56%	61%	50%	56%	61%	67%	7	8
12Jul18	30MinTS	1 Shift	57%	52%	48%	39%	52%	54%	52%	57%	48%	46%	48%	50%	48%	48%	61%	57%	4	5
		2 Shifts	52%	35%	48%	52%	46%	59%	52%	57%	54%	61%	46%	48%	50%	50%	50%	54%	5	6
	60MinTS	1 Shift	50%	45%	50%	68%	55%	68%	41%	73%	55%	59%	32%	59%	64%	55%	45%	64%	5	7
		2 Shifts	68%	55%	55%	45%	50%	41%	41%	45%	41%	36%	68%	41%	41%	50%	41%	45%	4	2
	120MinTS	1 Shift	70%	20%	90%	50%	70%	40%	70%	40%	70%	50%	50%	40%	60%	30%	40%	30%	7	2
		2 Shifts	40%	60%	20%	60%	50%	40%	30%	40%	50%	50%	70%	60%	40%	50%	80%	70%	4	6
15Mar19	30MinTS	1 Shift	37%	46%	50%	43%	50%	41%	57%	54%	54%	52%	63%	46%	52%	46%	50%	43%	7	2
		2 Shifts	57%	50%	63%	59%	37%	57%	43%	57%	50%	50%	43%	61%	52%	54%	52%	54%	5	7
	60MinTS	1 Shift	55%	59%	59%	77%	32%	50%	32%	50%	41%	55%	50%	64%	45%	77%	27%	59%	3	8
		2 Shifts	41%	45%	55%	55%	68%	41%	68%	32%	82%	45%	50%	45%	73%	27%	73%	32%	7	1
	120MinTS	1 Shift	60%	40%	50%	30%	50%	40%	60%	30%	60%	40%	70%	60%	70%	30%	60%	30%	8	1
		2 Shifts	70%	50%	60%	50%	30%	40%	40%	50%	40%	40%	40%	40%	30%	30%	60%	30%	3	3
Total of Events >= 50% per Algorithm			15	10	16	15	16	12	12	13	12	13	16	12	14	10	16	11	117	94

Table 3.9: Percentages of all algorithms that have 50% or more of the predicted events.

Dataset	Algorithm	BN		NB		KNN		J48		RF	
		t.test	Chi sq.	t.test	Chi sq.	t.test	Chi sq.	t.test	Chi sq.	t.test	Chi sq.
5Aug17	30MinTS	8.62e-6	0.0002	0.0355	3.3e-5	0.0061	1.85e-8	0.0022	1.03e-8	1.74e-5	1.76e-7
	60MinTS	0.0036	0.0329	0.0649	0.0055	0.2105	0.0055	0.1342	0.0009	0.0118	0.0014
	120MinTS	0.0518	0.0947	0.2169	0.0703	0.3547	0.0777	0.3272	0.0197	0.0556	0.0169
12Dec17	15MinTS	0.3921	0.0011	0.0003	0.2315	0.1086	0.0003	0.1359	0.0003	0.1236	1e-6
23Mar18	5MinTS	0.0002	0.0001	0.0009	2e-6	0.2375	8.25e-7	0.0069	3.5e-5	2.55e-6	0.0007
12Jul18	30MinTS	0.0195	0.9499	0.1194	0.0014	0.0009	0.0069	0.0042	0.0036	0.1497	2.4e-5
	60MinTS	0.0663	0.0772	0.3732	3.9e-5	0.0274	0.0396	0.0295	0.0449	0.2086	0.0067
	120MinTS	0.0795	0.3026	0.4528	0.0151	0.0828	0.2535	0.1495	0.3156	0.2470	0.1531
15Mar19	30MinTS	0.1957	0.9522	0.0693	0.9809	0.0013	0.9999	0.0009	0.9999	0.0274	0.9953
	60MinTS	0.2308	0.2975	0.1721	0.1803	0.0037	0.8823	0.0035	0.9205	0.0229	0.7013
	120MinTS	0.2782	0.1749	0.4144	0.0586	0.0275	0.8026	0.0200	0.8384	0.0649	0.6639

Table 3.10: The associated p-values of the positive class sentiment with the Bitcoin price for the machine learning classification methods, with highlighted statistically significant values

mining and classification techniques on Twitter data that can predict the best sentiment are needed. Classifying different dataset models using tweet-embeddings and n-gram features is helpful to enhance prediction.

The main contribution of this chapter is finding a partial correlation between the Bitcoin price fluctuation and the fluctuation of the sentiment classes using different machine learning algorithms and providing a framework of an efficient tweet sentiment tool for Bitcoin tweets, whether they are positive or not. In addition, the study compares different classification methods by providing a detailed experimental evaluation. The results showed better accuracy for manual sentiment in the data preprocessing phase in some datasets. However, it also shows a more stable overall performance experiment using R-Auto Tweet Sentiment (RATS). The results also showed that tweet-embedding and n-gram data modeling features could improve sentiment prediction, especially in the MLP and WiSARD deep learning techniques and decision tree algorithms. It is notable that this could be one of the challenges, especially with large datasets, because some algorithms such as MLP deep learning or decision tree methods are time-consuming. As a future study, designing a particular lexicon for Bitcoin sentiment may improve the correlation of the sentiment analysis

Dataset	Algorithm	MLP		RBFN		WiS	
		t.test	Chi sq.	t.test	Chi sq.	t.test	Chi sq.
5Aug17	30MinTS	0.0298	2e-6	0.0154	1.74e-7	0.0094	2e-6
	60MinTS	0.4081	0.0158	0.0811	0.0080	0.2363	0.0079
	120MinTS	0.4779	0.1074	0.1884	0.0815	0.3357	0.0647
12Dec17	15MinTS	0.0018	0.0128	0.0006	0.0209	0.0141	0.0017
23Mar18	5MinTS	8.47e-6	0.0002	3.99e-5	6.2e-5	4.78e-7	0.0007
12Jul18	30MinTS	0.0125	0.0019	0.3669	2.14e-7	0.0003	0.4369
	60MinTS	0.1545	0.0269	0.2626	0.0006	0.0354	0.0682
	120MinTS	0.2644	0.1455	0.2773	0.0363	0.1856	0.2329
15Mar19	30MinTS	0.0002	0.9999	0.1047	0.9979	1.85e-5	1
	60MinTS	0.0049	0.8952	0.3124	0.4291	4.56e-4	0.9850
	120MinTS	0.0808	0.6617	0.3886	0.1804	0.0160	0.8961

Table 3.11: The associated p values of the positive class sentiment with the Bitcoin price for the deep learning classification methods, with highlighted statistically significant values

with the Bitcoin price fluctuation, considering other features like hashtags, a Twitter user, the number of tweets, and emoticons.

Chapter 4

Feature Engineering for Spam Recognition of Arabic Tweets

Feature engineering techniques can be used to identify and flag spam and bot activity in Arabic tweets, or hashtags. Since Saudi Arabia is the highest Arabic-speaking country in using Twitter and one of the top-ranked countries in using Twitter around the world, it is important to analyze and recognize spam in their tweets, because it influences public opinion and it could mislead it. By using different machine learning algorithms on different feature generation approaches, it can accurately detect and recognize spam in hashtags.

4.1 Background

Spam, also known as unsolicited commercial e-mail (UCE), is the Internet version of "junk mail." It attempts to deliver a message online to someone who would not opt-in or want to receive it. Most of the spam is advertisements for commercial products. Potential target lists are created by scanning Usenet newsgroups and Internet mailing lists or searching the Web for e-mail addresses. Automated searches are also used to collect such information to get e-mail addresses for spamming. Since e-mail services became popular, e-mail service providers aim to find solutions to filter and minimize unwanted e-mails, recognize them, and categorize them as potentially spam messages [27]. The same challenge of unwanted messages remained after the rise of social media platforms. Twitter, which is one of the microblogging social media websites, is challenged with many noisy tweets and messages from both fake and real accounts. Twitter is an excellent tool to express short ideas, giving brief feedback, and sharing essential activities such as retweets and hashtags, which are two of the most exciting tools to measure trends and the popularity of opinions. That's why companies and organizations are eager to have an account on Twitter, to know the number of followers and retweets for each tweet, and to spread the news on the high trend hashtags [57].

4.1.1 Exceptional Role of Twitter in Saudi Arabia

In the Arabian Gulf region and Saudi Arabia, Twitter occupies an advanced position on all social networking sites. It is the most used and popular site among the Saudis and the Gulf countries in general. This made Saudi Arabia of very high importance to the financially troubled company, which does not find itself — despite its large spread — a profitable business model compared to its counterparts from other social media sites. Saudi Arabia, with a population of 35 million people, is the largest market for social media in the region, as two-thirds of the Saudi population is under the age of thirty-five [66]. These age groups often tend to engage with digital media platforms, particularly since Saudi Arabia is narrowing young people's amusement and entertainment spaces. It was found that it is difficult for individuals to express their personal opinions through official channels, and therefore they resort to these networks to express themselves freely. We can see that Saudi youth go to social networking sites — on top of Twitter, Snapchat, and YouTube — to search for a space for entertainment, follow the news, and express their opinions. Communication sites are an arena for dialogue between Saudi many current topics, especially the Islamic and liberal topics [92].

In the last few years, Twitter has become a destination for all parties in Saudi society, starting with young people in the early days. Soon after, it was joined by community leaders, including thinkers, politicians, officials, media professionals, and advocates. The accounts of Saudi preachers on the Twitter platform occupy the list of the most followed accounts on the platform in Saudi and Arab countries. This gathering and widespread societal interest in the Twitter platform prompted the businessman and Saudi Prince Al-Waleed bin Talal to acquire 4.9% of its shares, making him one of the major investors in it [102]. This interest was also clearly reflected in government policies and procedures towards the platform and its celebrities, as Saudi government institutions held many conferences and workshops on the best use of the Twitter platform from the government's point of view. On the other hand, the Saudi government has given special attention to Twitter through the verified accounts of senior Saudi officials and exceptional attention to the most famous Saudi figures. The celebrity accounts on Twitter have been used in preparing for important decisions by the Saudi authorities. Thus, the Saudi authorities have invested in the Twitter platform as a media space to communicate with the Saudi public through their direct accounts or celebrities who support the Saudi government trends.

Saudi Arabia is one of the leading countries on Twitter users worldwide. It is ranked eighth globally in the number of users (around 12.45 million), and the second in the world based on the population's percentage with approximately 35% of the population using Twitter. Figure 4.1 gives more details about leading countries based on the number of Twitter users as of January 2021 [88]. The Saudi trend has become a powerful influence on leading popular public opinion. It thus can influence the decision-makers in the country to take some actions that will absorb the discontent. For example, the Saudi blogger Hamza Kashgari was subjected to a broad hashtag campaign under the hashtag #HamzaKashgari in 2012 asking the government to stop him from writing and refer him to the investigation because of an article he wrote that Twitter activists in Saudi Arabia considered insulting the Prophet of Islam [89]. Indeed the government has taken back the citizen who fled to Malaysia, in cooperation with the Malaysian authorities, after increasing the popular demands that wanted him to be punished and then investigated and imprisoned in charge of contempt of religion.

In another example, Twitter activists launched a campaign calling for dismissing an official who was recently appointed as the director of the Jeddah Downtown project in early 2020. The activists searched his account for old tweets he wrote in 2013 that carried outright hostility to the country. After a few days, the government dismissed Eng. Hisham Malaika from his position after endorsing his writing. On the other hand, the government sometimes ignores some hashtags in the Saudi trend, such as the "The salary is not enough" hashtag [23], for example, or some human rights claims, because the government believes that the claims are not based on reasonable justifications for increasing salaries, or making some legal amendments. It is claimed that these hashtags are administered from abroad to fuel criticism of the citizens' living conditions and incite their discontent [35]. Some hashtags have been proven to rise in the trend due to the tweeting of many fake accounts with tweets that may not have any relationship to the topic of the hashtag only to raise it in the trend to gain more popular interaction, which was known under the term of "electronic flies phenomena" [112].

These accounts are often anonymous, and some are managed from outside Saudi Arabia, which was evident specifically after June 2017 and the decision of Arab countries, namely Saudi Arabia, Egypt, the United Arab Emirates, and Bahrain, to cut diplomatic relations with the State of Qatar, claiming its interference in the affairs of these countries and its support for radical Islamist groups for the sake of creating chaos in these countries. This caused the creation of a lot of fake accounts from both sides to develop trends and lead direct public opinion, creating electronic warfare-like situations, with the participation of hundreds of anonymous accounts [164]. It has also been observed that many hashtags were also rising due to the involvement of unwanted tweets from unknown accounts carrying commercial advertisements and the promotion of fraud or sexual products, as well as some tweets concerning the promotion of terrorism and violence, such as the tweets of electronic cells of the terrorist organization of ISIS. These tweets generally have nothing to do with the name of hashtags, which affects the credibility of the hashtag and its position in the trend.

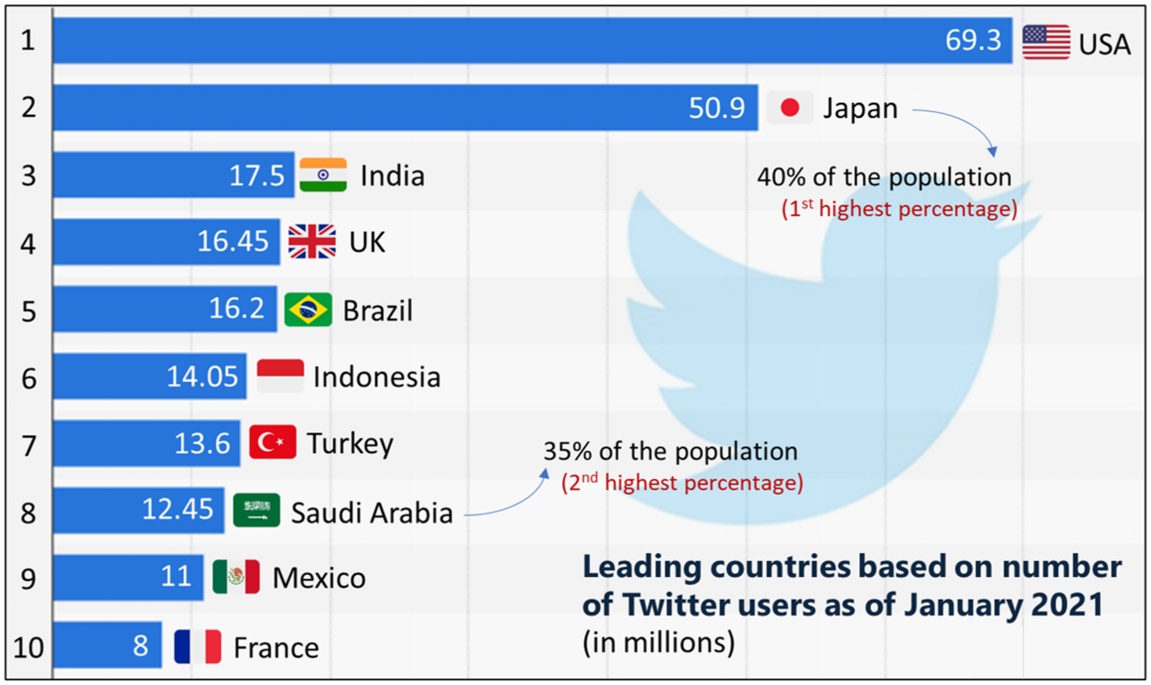


Figure 4.1: Leading countries based on number of Twitter users.

Spam Detection on Arabic Language

Recognizing spam over tweets is an active research area in the field of Natural Language Processing. There are many advanced studies on spam recognition on tweets in the English language, but not as much in the case of the Arabic language for many reasons. Some of the most important reasons are the lack of Arabic content on the Internet compared to English content and the late application of artificial intelligence techniques on Arabic texts. The activity in spam detection on Arabic content has begun with the increase of Arabic-speaking researchers and those interested in it. In this study, we apply machine learning experiments on tweets in the Arabic language, gathered from the Saudi Arabian trends, to recognize the possibly unwanted messages that are irrelevant to the hashtag topic by generating features of the tweet text, then evaluate different approaches and the spam class quality. Some of the main research questions that we aim to study are the effectiveness of n-gram and embedding-based generating features, different machine learning algorithms, their relations to different topic domains of tweets, and the balance of datasets in terms of spam vs. non-spam classes.

4.2 Related Work

Spam and non-related tweets mostly go against the community standard of using Twitter. It can be misleading and affect the credibility of the hashtag and the trend. It can be bots that spread hate or promote terrorism. Therefore generating features can help in the process of learning Spam tweets and recognizing them. Albadi and Kurdi made a study on spreading religious hatred on Arabic Twitter, and they found that bots were responsible for 11% of hateful tweets in the hate speech dataset [15]. They apply that to different hashtags topics from political events to the sport-related discussion by developing a bot detection model trained on various features extracted from 86,346 tweets disseminated by 450 manually-labeled accounts. They also extract 35 features categorized based on content, tweet, sentiment, and account features. Another study made by Almerekhi and Elsayed, extract features of Arabic tweets based on their formality, structure, temporal, and other specific features to detect the Automatically-Generated tweets [20]. In a similar approach, Alharbi and Aljaedi analyzed the 47 features generated and then selected the 16 most significant ones [17]. They achieved high performance by implementing one of the well-known classification algorithms, random forest, with accuracy rates greater than 90%.

A further study made by Boreggah et al. extracted the top 10 features based on tweet features, tweet content, account behavior, and account profile [37]. They applied three classification algorithms: Random Forest (RF), Naïve Bayes (NB), and Support Vector Machine (SVM). Random Forest accomplished the highest accuracy result, which was 98.68%. On the other hand, Alorini and Rawat extracted only three features to find automatic spam on tweets [21]. They rely on the number of hashtags in tweets, the number of shortened URLs, and the existence of profanity words, but they only focused on spam in Gulf Dialectical Arabic tweets. Results show that NB produces more accurate outcomes by 86%. Similarly, AlTwairesh et al. extracted these words and constructed an Arabic Spam Detecting Lexicon (ASDL) [156]. The lexicon contains 108 words. After this empirical analysis, four features were identified to be used in the classification model (URL, phone number, number of hashtags, spam lexicon). They mentioned that the features used were identified through empirical analysis and then applied in the classification approaches developed. Both approaches showed comparable results in terms of performance measures reported reaching an average F-measure of 85% for the rule-based approach and 91.6% for the supervised learning approach.

In a recent study, AlZoubi et al. made a study on affect detection from Arabic tweets using ensemble and deep learning techniques [24]. They used pre-trained embedding AraVec [148], and our results showed that our proposed approach outperformed the baseline results as well as other related work models with an enhancement of 0.7% over the best performing model effect. Finally, AlAzani and ElAlfy, used Word2vec embedding the extract features based on two approaches: the Continuous Bag-of-words (CBOW) and the skip-grams (SG) [13].

Our primary research contribution is to compare different supervised classification methods and evaluate their performance on different hashtags based on two different feature generation approaches, the Word2Vec embedding, and the n-gram approach. A further goal is to investigate if grouping the tweets based on hashtag topics and using the same methodology has a notable impact on the spam recognition results.

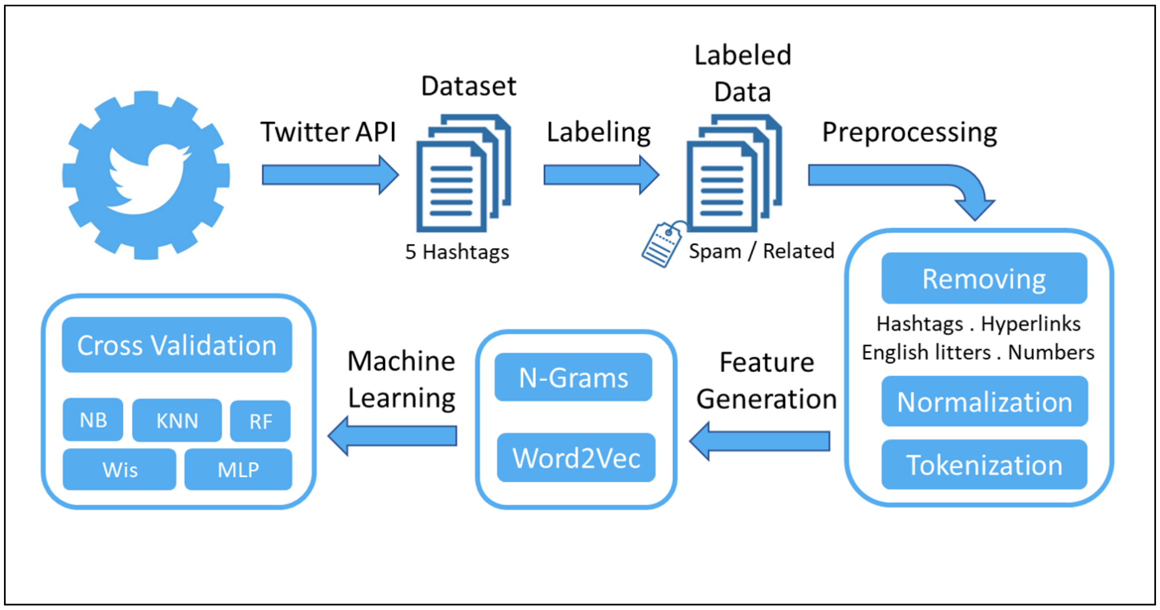


Figure 4.2: The proposed system architecture

4.3 Methodology and Data

The general form of the proposed system consists of three major parts: (1) the data pre-processing part, which includes data collection from Twitter, data cleaning, data modeling process, and tweet labeling; (2) the feature generation process that includes n-gram approach and Word2Vec; and, (3) the machine learning part which includes multiple classification experiments and tests. Figure 4.2 shows the system architecture of this study.

Eight Saudi Arabian trend hashtags from Twitter have been collected using Twitter API to be applied in this study. Tweets have been gathered during October 2020. Two of them are about health topics related to Covid-19, SA_COV and SA_2Wv. The other two cover political topics about boycotting Turkish products in Saudi Arabia (SA_TUR) and a TV interview with the former Saudi ambassador in the United States (SA_BAN). Three more datasets are covering some internal national affairs in Saudi Arabia, like breaking news regarding royal orders (SA_ROY), the occasions of the Saudi National Day (SA_SND), and the news about increasing the percentage of value-added tax (VAT) in Saudi Arabia (SA_VAT), in addition to one small dataset about the "Derby" football match between Al-Hilal and Al-Nasser (SA_DER). After removing only English tweets, the total number of collected tweets is over 40,000

Datasets	Description	Topic	#Tweets	Relevant	Spam	Spam%
SA_SND	The Saudi National Day	National Affairs	5,760	2,911	2,849	49%
SA_VAT	Value Added Tax	National Affairs	1,652	994	658	40%
SA_Cov	Covid-19	Health	7,617	4,861	2,756	39%
SA_Roy	Royal orders from the king	National Affairs	4,696	3,013	1,683	35%
SA_Tur	Boycotting Turkish products	Politics	9,603	7,101	2,502	26%
SA_2Wv	Second Wave of Covid-19	Health	2,293	1,760	533	23%
SA_Ban	Prince Bandar Bin Sultan interview	Politics	6,040	5,152	888	14%
SA_Der	Football match between Al-Nasser and Al-Hilal	Sport	2,365	2,206	169	7%
		Total	40,026	27,998	12,038	

Table 4.1: The total of tweets are more than 40,026 after removing irrelevant tweets

tweets. All tweets have been labeled manually into two classes; Relevant and Spam. Spam tweets are unrelated tweets to the hashtag topic, like ads, prayers, and funny jokes.

Table 4.1 shows the details of the collected tweets after labeling, sorted from the dataset that has the most Spam percentage to the least.

Because the study is focusing on generating features from the text, some tweet features such as tweet URL or Twitter user were removed, and the data cleaning process was applied using a code implemented in Python [120] language. All punctuations were removed, and Tweets' words were counted and tokenized using a regular expression. The n-gram code generates unigram, bigram, and trigram features using the `StringToWordVector` function in Weka [77], then limited to the 500 most frequent n-grams [162]. Tf-idf technique were applied to vectoraized the tokened n-grams. Figure 4.3 shows the process of generating n-gram features for this study.

The Word2Vec features approach generates the 200-dimension embedding from the same hashtag tweets and applies the skip-gram model by one, using the `gensim.-models.Word2Vec()` function from `Gensim` library [171] in Python [120]. The model is trained on skip-grams (SG), which are n-grams that allow tokens to be skipped. The context of a word can be represented through a set of skip-gram pairs of target word and context word, where context word appears in the neighboring context of target word. We used windows size of 5, which represents the maximum distance between the current and predicted word within a tweet. The window size value is

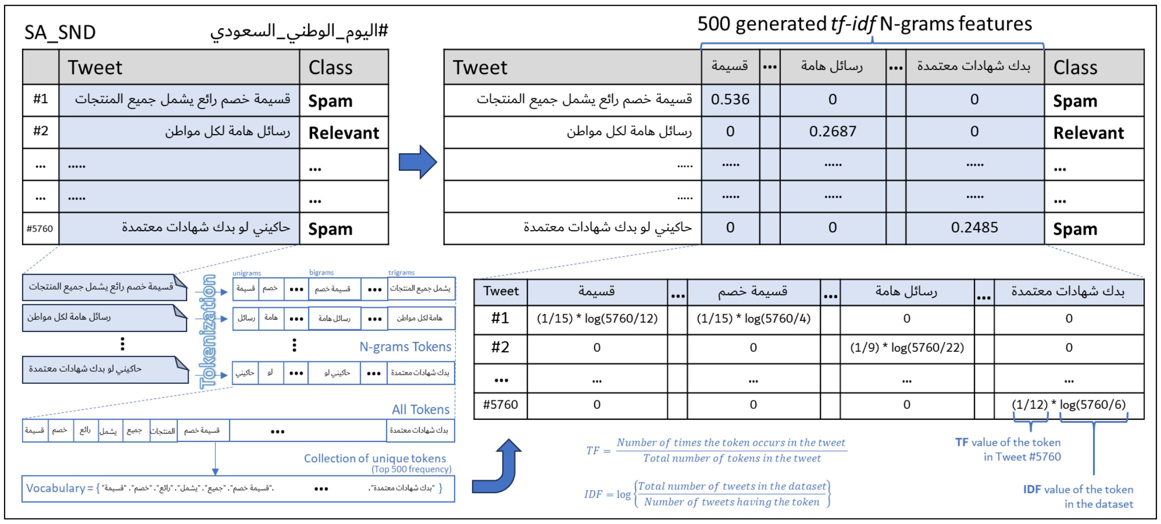

SA_SND
#اليوم_الوطني_السعودي
500 generated tf-idf N-grams features
Tweet
Class
Spam
Relevant
Tokenization
unigrams
bigrams
trigrams
N-grams Tokens
All Tokens
Collection of unique tokens (Top 500 frequency)
Vocabulary
TF = Number of times the token occurs in the tweet / Total number of tokens in the tweet
IDF = log (Total number of tweets in the dataset / Number of tweets having the token)
TF value of the token in Tweet #5760
IDF value of the token in the dataset

Figure 4.3: Illustrating the steps of generating tf-idf n-grams features for spam tweets, starting from tokenizing words from the all tweets, building a collection of unique tokens, selecting the top frequent 500 tokens, Then applying tf-idf technique to have weighted n-gram features

preferred to be small as it used in this study when we are focusing on capturing local syntax and specific tasks, while larger window size are better suited to capturing semantic relationships at a larger scale of data. The model calculates a tweet-level feature representation using pre-trained word embeddings. A dummy word-embedding formed by zeroes is used for a word with no corresponding embedding. Then, the average word-embedding is calculated and then a vector of 200 embedding values is generated for each tweet. We set negative sampling value to 10, which means that for each training example , the algorithm will randomly select 10 negative samples (words that are not the actual context word or target word) to update their weights. That is, for each training example, the algorithm updates the weights for 11 words in total, one actual context word and 10 negative samples. Figure 4.4 shows the process of generating Word2Vec embeddings for each tweet in the dataset. For more understandig to Word2Vec model, see details on figure 4.4

Using 10-fold cross-validation in Weka, five machine learning algorithms are applied. The algorithms were chosen relying on various classification methods' concepts: lazy learner method, the Bayesian method, and the decision tree method. Therefore, three of them are classical supervised learning algorithms: Naïve Bayes (NB), k-Neareast Neighbours (KNN), and Random Forest (RF), and the other two are deep

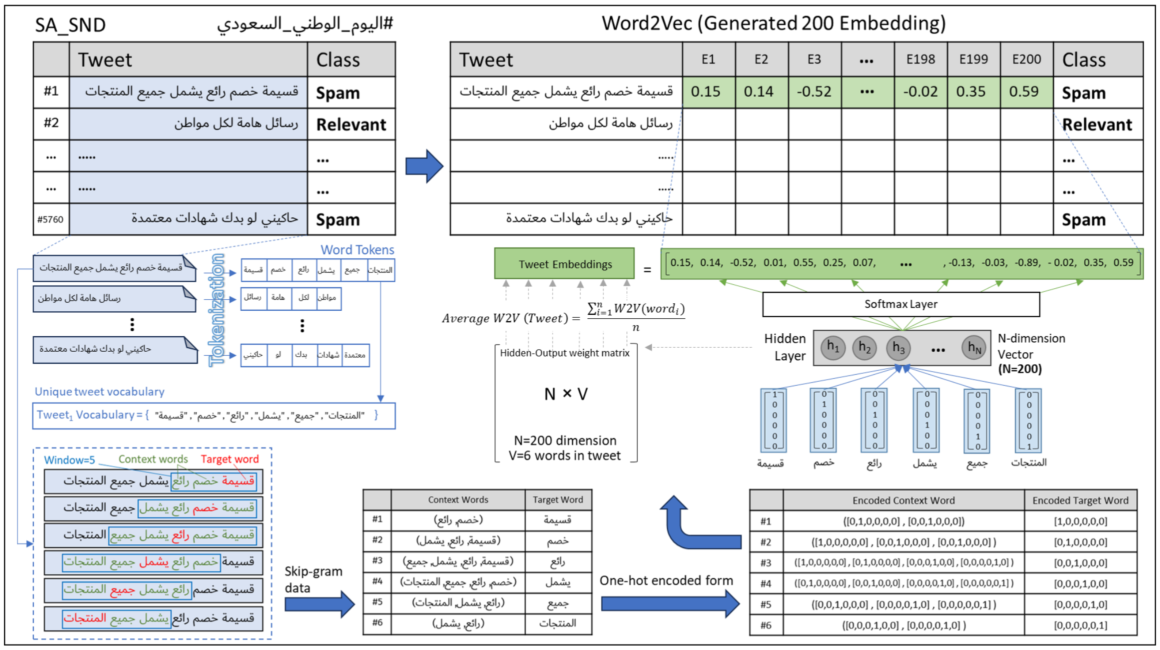


Figure 4.4: Illustrating the steps of generating word2Vec embedding features for spam tweets, starting from tokenizing words from the tweet, building a unique tweet vocabulary, building the skip-gram data for the tweet, then one-hot encoded form, after that the applying 200-dimension vectors of Wored2Vec model, and finally taking the average word2Vec for each tweet

learning algorithms: Multilayer Perceptron (MLP) and Weightless Neural Network (WiSARD). These were Weka-based implementations of the deep learning algorithms.

4.4 Machine Learning Methods

After finishing the preprocessing phase and having different remodeled datasets, it is evident that we need to apply supervised data mining techniques and Deep learning algorithms to predict spam. All experiments applied 10-folds cross validation testing technique.

4.4.1 Supervised Classification Methods

Our plan was to test the most common classification algorithms and compare the accuracy of predictions to each other, so that we could examine the effectiveness of different feature engineering techniques. Using Weka [162], the algorithms were chosen that rely on various theoretical classification method concepts: lazy learner method,

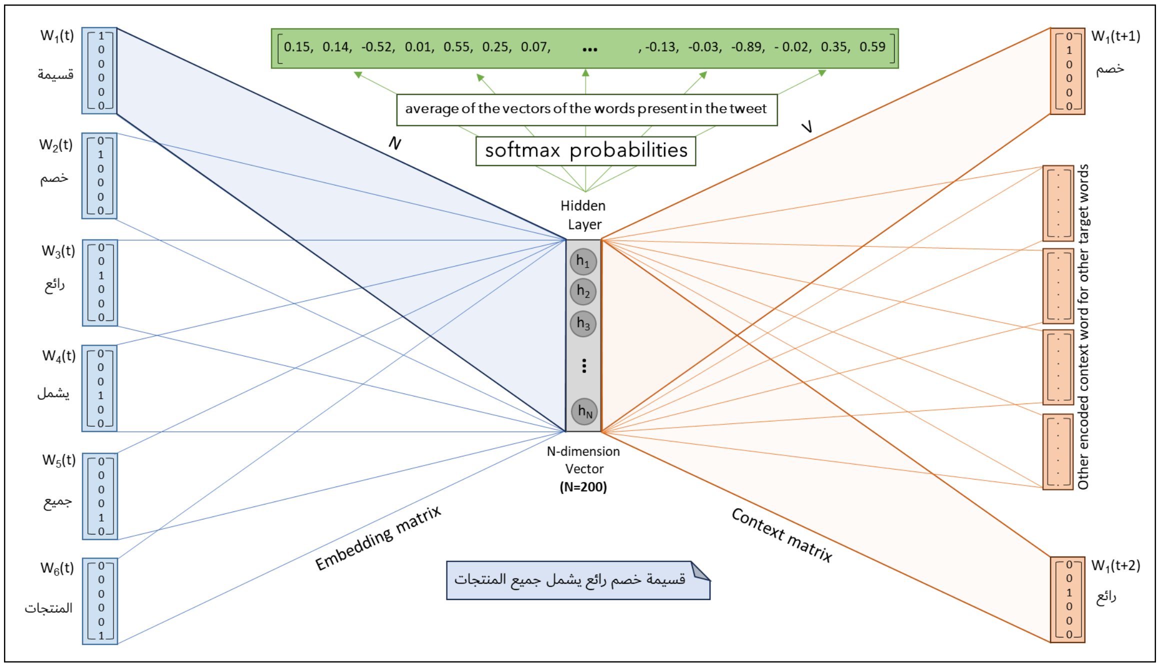


Figure 4.5: More understanding to Word2Vec model

Bayesian method, and decision tree method. Three different classification algorithms have been chosen: K-Nearest Neighbor (KNN) [22], Naïve Bayes (NB) [132], Random Forest Decision Tree (RF) [41].

The `IBK` function in the Weka data mining tool from the package {`lazy`} was used to implement the experiments and to evaluate the KNN algorithm. IBK is the same as the KNN algorithm. We used k=3, with 100 batch size and LinearNNSearch as a nearest Neighbour Search Algorithm.

The `NaiveBayes` function in Weka data mining tool in the package {`bayes`} was used to implement the experiments and tests to evaluate the algorithm for this study. We used 100 batch size for this experiment.

The `RandomForest` function in the Weka data mining tool in the package {`trees`} was used to implement the experiments and tests to evaluate the algorithm for this study. We used 100 batch size, 100% bag size, and 100 iterations.

All the classification methods were previously explained in Chapter 2.

4.4.2 Deep Learning Methods

Another set of methods used is a set of neural network methods, which are applied to compare their accuracy and efficiency. These methods are available in the Weka workbench using the deep learning package `WekaDeeplearning4j`, which is a Java library developed to incorporate the modern techniques of deep learning into Weka. The two deep learning algorithms that applied are: Multilayer Perceptron (MLP) using `Dl4jMlpClassifier` function with two dense layers [165], from `WekaDeeplearning4j` deep learning package in Weka, and Weightless Neural Network (WiSARD) [65].

For the first algorithm, training of a perceptronn is performed by adjusting the link weights after processing each piece of data, based on the amount of error in the output in comparison to the expected result. Using the `Dl4jMlpClassifier` function, the MLP network was designed with two dense layers. Since one epoch is too big to feed to the computer at once, we used 32 epochs to perform. As the number of epochs increases, a greater number of times the weight is changed in the neural network, so the learning curve goes from underfitting to an overfitting curve [145]. The applied MLP model used a batch size of 100, an instance iterator using `weka.dl4j.iterators.instance.DefaultInstanceIterator` with a mini-batch size of one, and an iteration listener which evaluates the model while training every five epochs. The model consists of an input layer, two dense layers, and an output layer. The input layer variables are the total number of dataset attributes in addition to the total number of classes. The hidden layers are using the Rectified Linear Unit (ReLU) activation function, and output 100 values from the first hidden layer, and 30 values from the second hidden layer. The output layer outputs the classes using the MCXENT loss function, and softmax activation function, which transforms the neurons' output exponentials to class probabilities distribution.

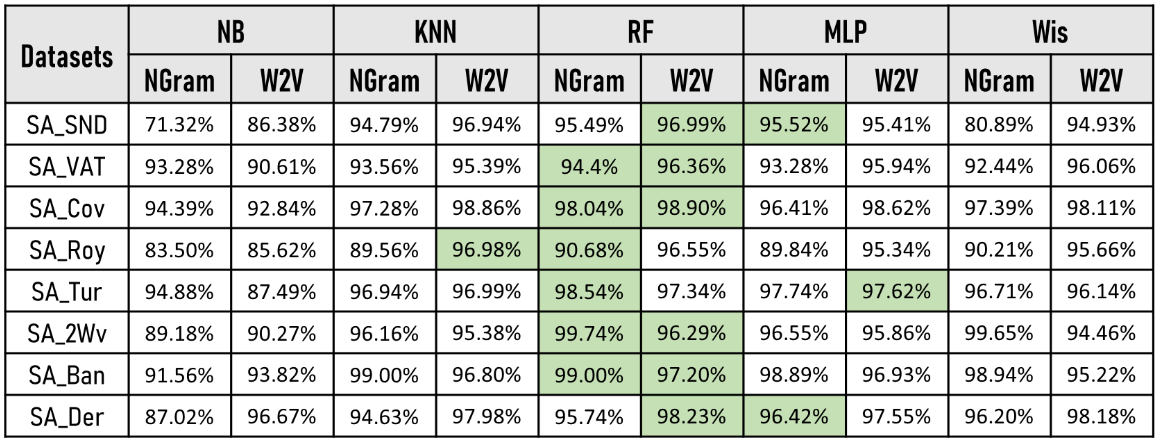
For the WiSARD model, The `WiSARD4WEKA` package implements a multi-class classification method based on the WiSARD weightless neural model for the Weka machine learning toolkit. We used a batch size of 100, like the previously applied neural network algorithms.

All used Deep Learning Methods were briefly explained in Chapter 2.

4.5 Experimental Results

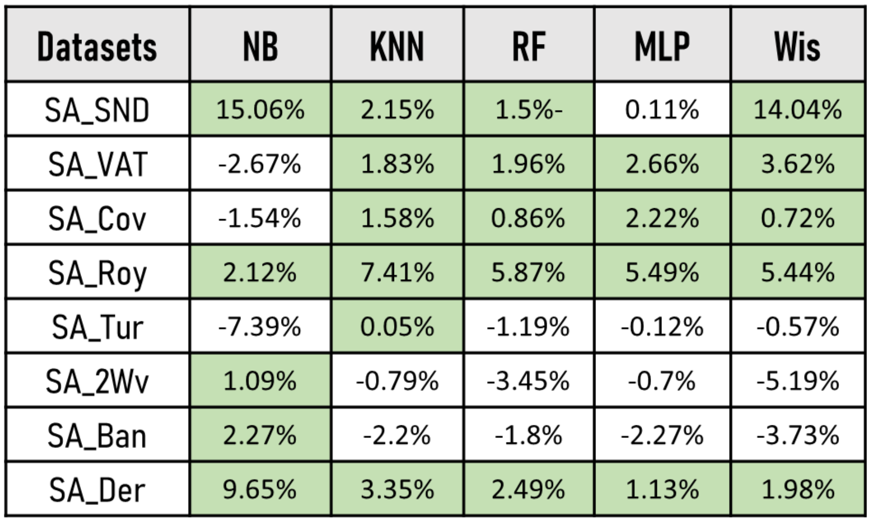
The results of the applied machine learning algorithms show high accuracy for spam recognition for Arabic tweets in general. Random Forest algorithm is showing the mostly better performance overall applied algorithms, especially with n-gram generated features. While K-nearest neighbor is showing the best accuracy on `SA_ROY` dataset that has features generated by Word2Vec embedding, and the MLP deep learning algorithm showing the best result on
texttttSA_TUR dataset using Word2Vec. MLP is also showed the best accuracy on `SA_SND` and `SA_DER` datasets using the n-gram approach. The NB algorithm is showing an improvement on embedding over the n-gram in three datasets. See Table 4.2 for the accuracy details using n-gram and Word2Vec; the highest accuracy results for each approach are shown in green color. The main observed outcome from these experimental results is that n-gram performs better than Word2Vec on three out of the eight used datasets. These three datasets have spam percentages ranging from 26% to 14% successively (least balanced). On the other hand, Word2Vec performs better on the more balancing datasets with spam percentages ranging from almost half to almost the third (most balanced), and one more dataset with only 7% of tweets labeled as spam (unbalanced). Another interesting finding is that the Word2Vec embedding method with 100 generated features has more accuracy than n-gram features with 500 features for each tweet. This can show the superiority of Word2Vec over the n-gram approach. This calls to try 300 embeddings of Word2Vec and see if there is any advance of it over 100 embeddings. Table 4.3 shows the percentage difference between n-gram generated features and Word2Vec on all applied machine learning algorithms.

In further support for this finding, we applied the same experiments on the grouped datasets by topic (see Table 4.4 and Table 4.5). We found that grouped hashtags by topic with more balanced classes have better results using the Word2Vec embedding approach. The three grouped datasets by topics are: national affairs hashtags, COVID-19-related hashtags, and political hashtags. The datasets `SA_National` and `SA_Health` that have from 42% and 33% of spam tweets have more accurate results with 10-folds cross-validation than the n-gram approach. At the same time, the `SA_Politics` that has 21% of spam tweets has better results in most of the algorithms



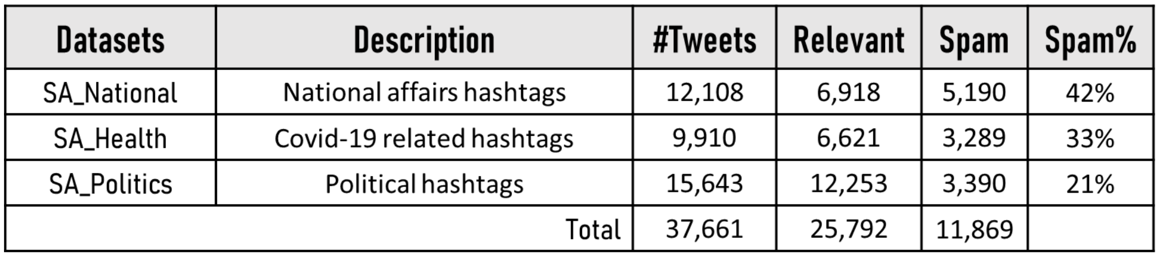
Datasets	NB		KNN		RF		MLP		Wis	
	NGram	W2V	NGram	W2V	NGram	W2V	NGram	W2V	NGram	W2V
SA_SND	71.32%	86.38%	94.79%	96.94%	95.49%	96.99%	95.52%	95.41%	80.89%	94.93%
SA_VAT	93.28%	90.61%	93.56%	95.39%	94.4%	96.36%	93.28%	95.94%	92.44%	96.06%
SA_Cov	94.39%	92.84%	97.28%	98.86%	98.04%	98.90%	96.41%	98.62%	97.39%	98.11%
SA_Roy	83.50%	85.62%	89.56%	96.98%	90.68%	96.55%	89.84%	95.34%	90.21%	95.66%
SA_Tur	94.88%	87.49%	96.94%	96.99%	98.54%	97.34%	97.74%	97.62%	96.71%	96.14%
SA_2Wv	89.18%	90.27%	96.16%	95.38%	99.74%	96.29%	96.55%	95.86%	99.65%	94.46%
SA_Ban	91.56%	93.82%	99.00%	96.80%	99.00%	97.20%	98.89%	96.93%	98.94%	95.22%
SA_Der	87.02%	96.67%	94.63%	97.98%	95.74%	98.23%	96.42%	97.55%	96.20%	98.18%

Table 4.2: 10-folds cross validation accuracy for n-gram and Word2Vec approaches



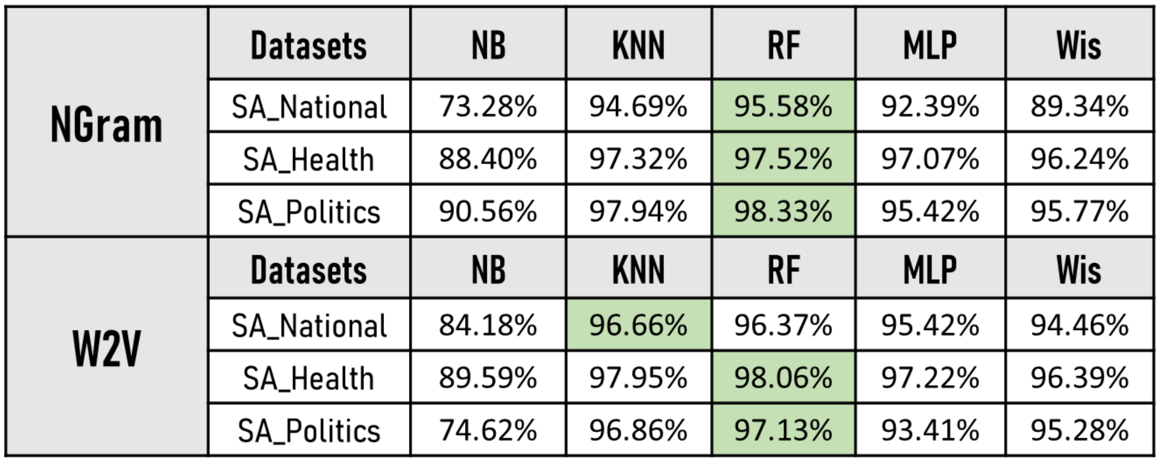
Datasets	NB	KNN	RF	MLP	Wis
SA_SND	15.06%	2.15%	1.5%-	0.11%	14.04%
SA_VAT	-2.67%	1.83%	1.96%	2.66%	3.62%
SA_Cov	-1.54%	1.58%	0.86%	2.22%	0.72%
SA_Roy	2.12%	7.41%	5.87%	5.49%	5.44%
SA_Tur	-7.39%	0.05%	-1.19%	-0.12%	-0.57%
SA_2Wv	1.09%	-0.79%	-3.45%	-0.7%	-5.19%
SA_Ban	2.27%	-2.2%	-1.8%	-2.27%	-3.73%
SA_Der	9.65%	3.35%	2.49%	1.13%	1.98%

Table 4.3: Word2Vec accuracy improvement over the n-gram approach



Datasets	Description	#Tweets	Relevant	Spam	Spam%
SA_National	National affairs hashtags	12,108	6,918	5,190	42%
SA_Health	Covid-19 related hashtags	9,910	6,621	3,289	33%
SA_Politics	Political hashtags	15,643	12,253	3,390	21%
	Total	37,661	25,792	11,869	

Table 4.4: The grouped datasets by the hashtag's topic



	Datasets	NB	KNN	RF	MLP	Wis
NGram	SA_National	73.28%	94.69%	95.58%	92.39%	89.34%
	SA_Health	88.40%	97.32%	97.52%	97.07%	96.24%
	SA_Politics	90.56%	97.94%	98.33%	95.42%	95.77%
W2V	Datasets	NB	KNN	RF	MLP	Wis
	SA_National	84.18%	96.66%	96.37%	95.42%	94.46%
	SA_Health	89.59%	97.95%	98.06%	97.22%	96.39%
	SA_Politics	74.62%	96.86%	97.13%	93.41%	95.28%

Table 4.5: Comparison between Word2Vec and n-gram 10-folds cross validation accuracy results on grouped datasets by topic

in the n-gram feature generation approach.

Moreover, the precision results of the spam class are showing higher scores using Word2Vec embedding approaches on most machine learning experiments in the applied algorithms, especially for the more balanced dataset and last unbalanced dataset `SA_DER`. F-measure results also show better results of the n-gram approach in all experiments of the least balanced dataset. Figures 4.6 and 4.7: the scores of the precision, recall, and F-measures for spam class for all datasets in n-gram and Word2Vec. It also shows higher scores for spam class precision and F-measure, in general, using the Random Forest algorithm in comparison with other algorithms, which gives an advantage of Random Forest as a better algorithm option to be applied for this study. Figure 4.8 shows what approach has better results in precision, recall, and F-measures in all experiments. Random Forest performance was also in processing time compared to the deep learning algorithms, while MLP was the most

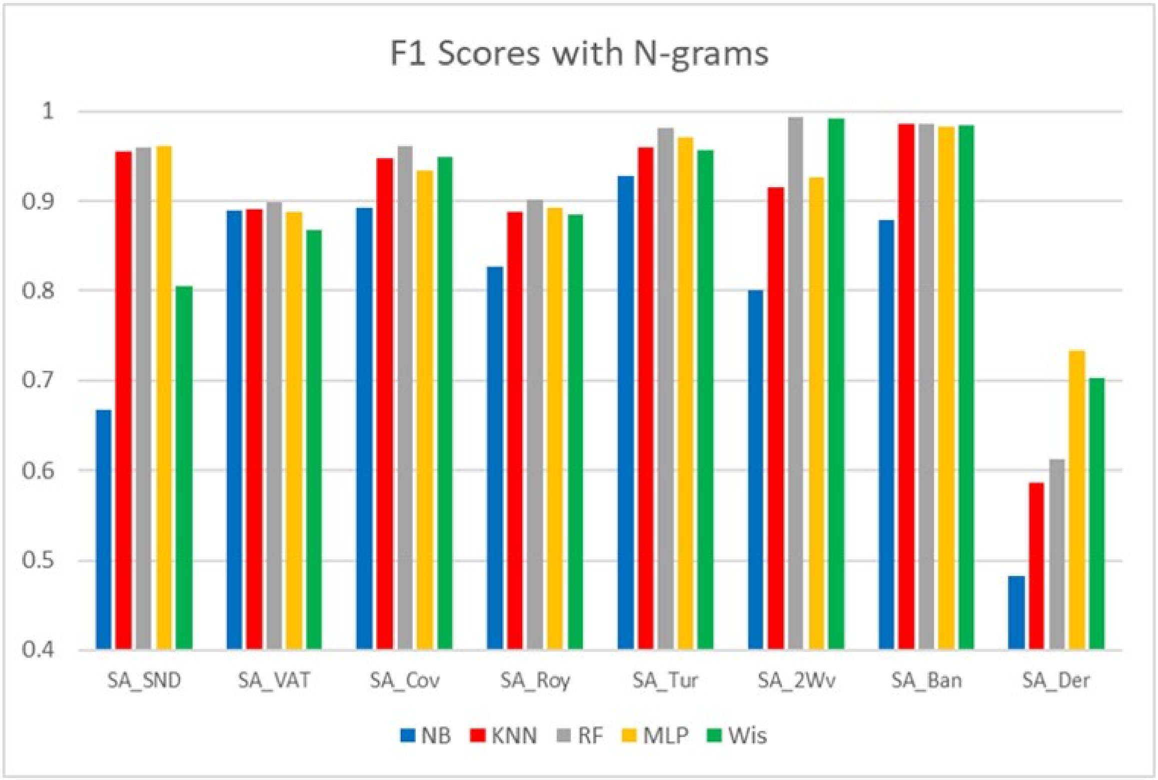


Figure 4.6: F-Measures of the Spam class for n-gram Approach

time consumable approach for most experiments. The lower accuracy results and other spam class evaluation measures were in Naïve Bayes algorithm in both applied feature generation approaches.

In general, we can conclude that the Word2Vec embedding feature generation approach excels the n-gram approach on the most balanced datasets in learning and recognizing the spam tweets. Moreover, Random Forest performs better in 10-folds cross-validation accuracy results, spam class precision, recall, F-measure scores, and processing time for most applied datasets.

4.6 Conclusion and Future Work

Tweet feature generation is an active field for studies and research. Arabic tweets are more challenging due to fewer resources than English tweets. It interests many people following news and trends on Twitter, especially on Saudi Arabian trends that may be affected by spam and unrelated tweets. In this study, we applied two different

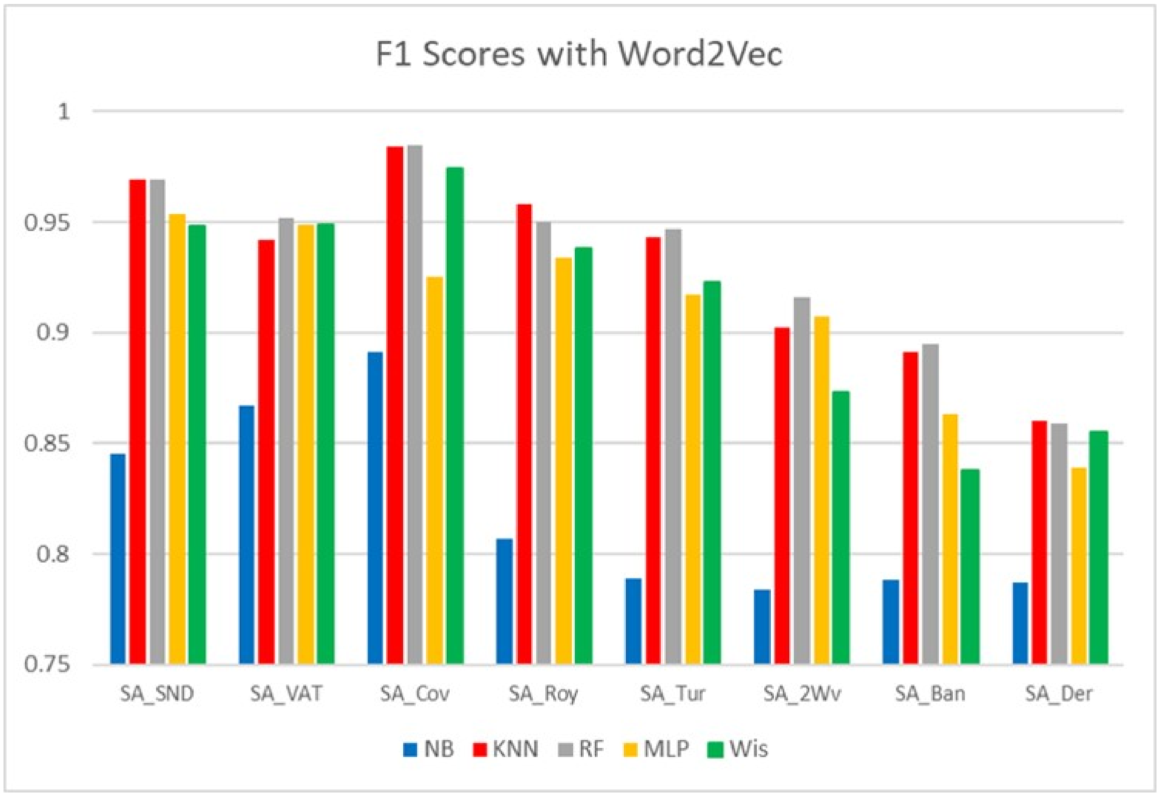


Figure 4.7: F-Measures of the Spam class for Word2Vec Approach

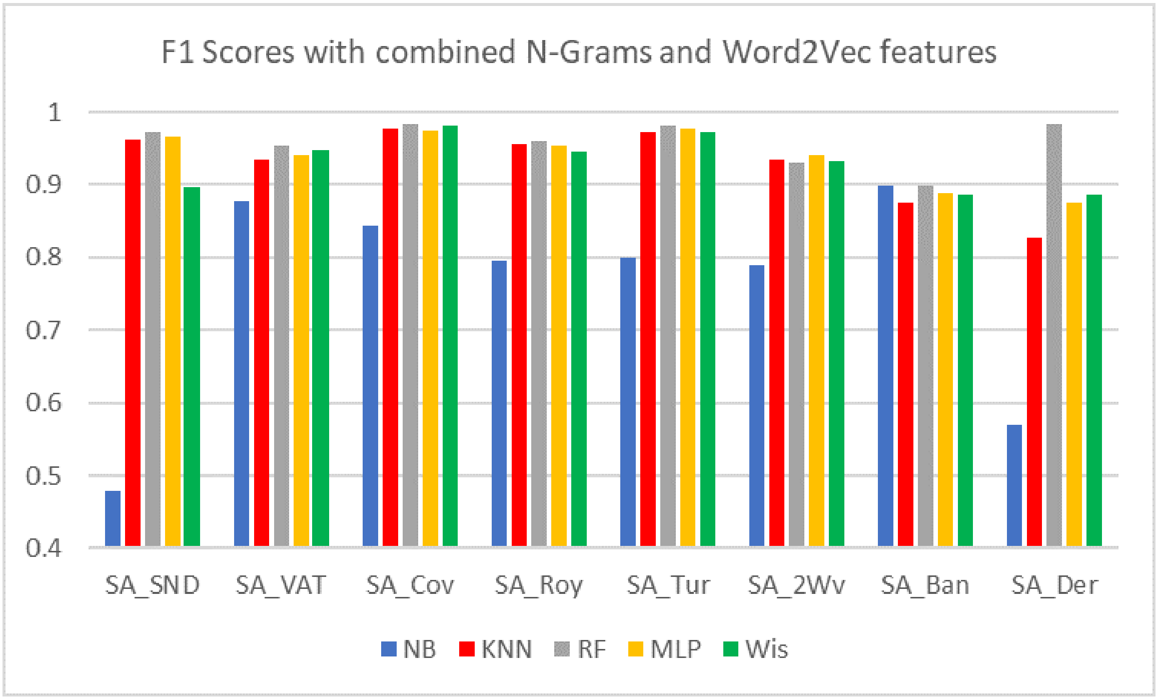


Figure 4.8: F-Measures of the Spam class for combined features of n-gram and Word2Vec Approaches

feature generations on Arabic tweets to see a better method for learning the spam on Arabic tweets. The main contribution of this paper is finding the advantage of Word2Vec approach over the n-gram feature generation approach, and the advantage is on the more balanced datasets, while the datasets that are least balancing are showing better results for n-gram. We also found the same conclusion in applying the same methodology on the grouped hashtags by topic type. Another finding is that Random Forest performs better than other algorithms in most experiments, in cross-validation accuracy results, spam class evaluation scores, and processing time. Considering that Word2Vec generated 100 embeddings in this study, in comparison to 500 top n-gram features, it is encouraging to try generating 300 embeddings and apply the same machine learning algorithms in the future to see if there is a correlation between the generated feature number results. Another future work idea is doing the same study on datasets in English to compare the results between different languages.

Chapter 5

Feature Engineering for Arabic Dialect Recognition

Arabic dialect recognition is the process of identifying the dialect of a given Arabic text or speech. The Arabic language has many dialects that are different from Arabic-speaking country to another and among the sub-areas of these countries. Generating features of Arabic tweets can be used to train a machine-learning model for Arabic dialect recognition. In this chapter, we used pre-trained models such as BERT fine-tuned generated features and demonstrated that these features are helpful in the process of recognizing the Arabic country or region with that the Arabic tweet is associated.

5.1 Background

The Arabic language is the most spoken Semitic language and one of the most widely spoken languages in the world. It is spoken by more than 467 million people [25], and its speakers are distributed in the Arab world, as well as to many other neighboring regions such as the Ahvaz region in Iran, Turkey, Chad, Mali, Senegal, Eritrea, Ethiopia, South Sudan. It can be ranked the fifth widely spoken language in terms of the most widespread languages globally, And the fourth language in terms of the number of users on the Internet. Moreover, the Arabic language is important to Muslims because it is a sacred language as it is the language of the Quran. It is the language of prayer and is essential in carrying out Islamic worship. Arabic is an official language in all countries of the Arab world, which are the countries that politically belong to the Arab League, in addition to being an official language in Chad, Eritrea, and Israel.

5.1.1 Arabic Dialects

There are many factors that contributed to the formation of different Arabic dialects, the most important is the Arab immigration after the spread of Islam and the expansion of the Rashidun Empire outside the Arabian Peninsula, the mainland of Arabs, and the influence of other nations' languages. The multiplicity of dialects in Arabic existed among the Arabs from the days of pre-Islamic times, as there was a dialect for each of the tribes. When the Quran was revealed, it was in the dialect of "Quraish" tribe, the tribe of the Prophet of Muslims. Therefore, classical Arabic, or the standard Arabic dialect, is originally the dialect of "Quraish" that became the standard dialect of other Arab tribes [80].

After the coming of Islam, the modern vernacular dialects likely started at the time of the Islamic conquests, as the new Muslims in the new lands (many of which are now Arab countries) began to learn Arabic, but — naturally — they could not speak it exactly as the Arabs in the Arabian Peninsula speak it; thus it was slightly changed. At that time, the difference was not very clear. However, gradually Arabic was transformed, and its phonemic features, sentences, etc., were changed until it turned into modern colloquial dialects [14]. Arabic-speaking lands were invaded, ruled, and colonized in the latest centuries by other non-Arabic kingdoms and empires, such as Turkey, France, the United Kingdom, Italy, and Spain. This is another factor that affects the current Arabic dialects. In Fig. 5.1 we tried to categorize the Arabic dialects regarding the geographic location, the historical understanding, words, and sound similarities, and the strong influence of other nations, whether they were the native nation of the land or the foreign country that colonized these lands.

Language is the primary vessel that contains science, technology, culture, history, civilization, identity, and feelings. Although the Arabic language is one of the most widely spoken languages worldwide, it is also clear that there is a problem in teaching and disseminating science in the current era in the Arabic language, and the adoption of the English language often in education in Arab universities and sometimes also in schools. This issue exists due to the weakness of the Arabic content in the new sciences and the lack of the Arabic scientific side of the literature written in the Arabic language. Lack of Arabic content on the web is another challenge besides the variety of dialects in Arabic and having no unified standard of language, making the

Arabic language very difficult for Natural Language Processing and machine learning tasks. Moreover, we should consider that the Arabic alphabet differs from the English or Latin alphabet. Also, it is written from right to left, making it complicated and not compatible with many applied rules to the processing of Latin-based alphabet languages.

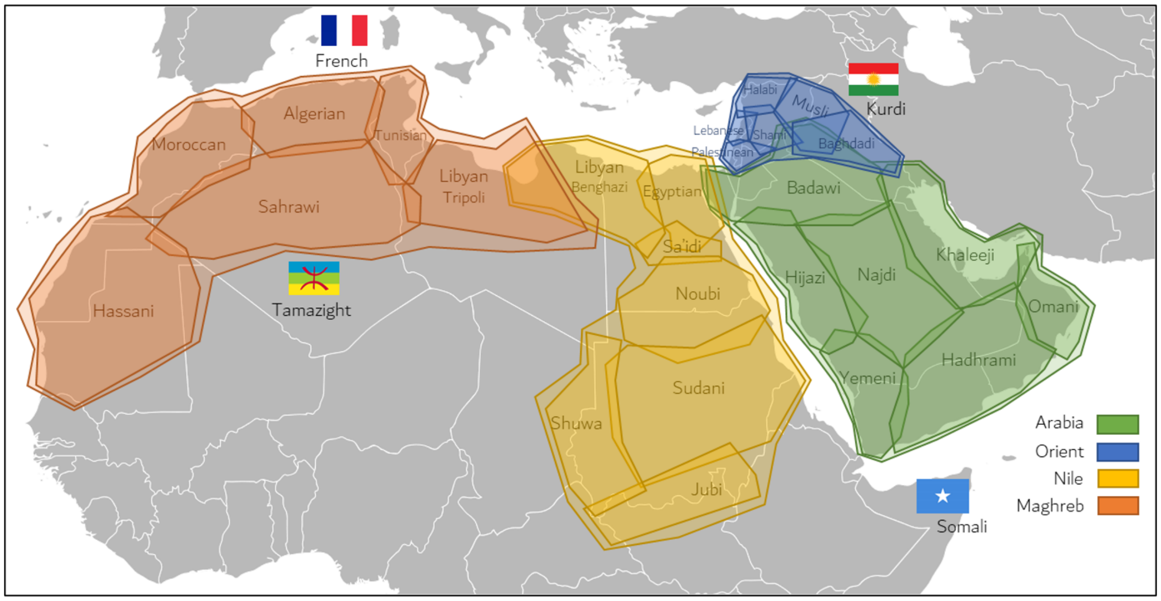


Figure 5.1: Arabic dialect regions and sub-regions

5.2 Related Work

Arabic dialect identification is a difficult task. The lack of corpora specifically devoted to the subject, the absence of a common orthography between and among the various dialects, and the nature of the language itself are a few causes for this (e.g., its morphological richness among other peculiarities). The Arabic NLP community has developed several solutions to address these issues. One of the solutions was the creation of annotated corpora with a primary focus on dialectical information, such as the Arabic On-line Commentary dataset [169], the Multi-Arabic Dialect Applications and Resources (MADAR) Arabic dialect corpus and lexicon [38], the Arap-Tweet corpus [168], as well as a city-level dataset of Arabic dialects that was curated by Abdul-Mageed et al. [7]. Another solution is organizing NLP workshops and shared tasks, which are only focused on creating methods and models that can identify and categorize the use of Arabic dialects in written text, which is another well-liked form of response. The MADAR shared task, which focuses on dialect detection at the level of Arab countries and cities, is one example [39].

Another example was the Nuanced Arabic Dialect Identification (NADI) shared task 2020 [9], and NADI shared task 2021 [10]. In NADI 2021, they received a total of 53 unique team registrations, and after the evaluation phase, they received a total of 68 submissions. Based on the official metric, the macro F1 score, the best performance obtained was with 22.38% F1 score for the country level. For the province level, the best test results were achieved with the best F1 score which is 32.26%. These results were for Modern Standard Arabic (MSA) tweets. For the dialectal Arabic tweets, the results were weaker, and the tasks were more challenging. For the country level, the best test result for all teams who submitted the task was with 6.43% F1 score, while the best results of all teams who submitted the province-level task for dialectal Arabic achieved an F1 score of 8.60%.

With the varying emphasis on feature engineering, ensemble methods, and the level of supervision required, the Arabic NLP community's efforts have led to several publications that examine the application of various Machine Learning (ML) tools to the problem of dialect identification [139, 59, 78, 154]. A number of papers have also been published recently that investigate the potential of Deep Learning (DL) models for dialect detection. These papers range from the work by Abdul-Mageed et al. and Ali [7, 18], which demonstrate the improved performance that can be attained using Long Short-Term Memorys (LSTMs) and Convolutional Neural Networkss (CNNs), to Zhang and Abdul-Mageed [170], which emphasizes the potential of pre-trained language models to achieve state-of-the-art performance on the task of dialect detection. In our study, we are focusing more on the role of feature engineering by comparing different feature generation methodologies with considering the number of classes, then applying an ensemble model to notice any improvement in the dialect prediction accuracy.

5.3 Data

As the participants in NADI shared task 2020 [110] and NADI shared task 2021 [111], we were provided a labeled dataset that consisted of 42,000 tweets in total, collected from 21 Arab world countries and 100 provinces from them, to learn and identify different Arabic dialects using Natural Language Processing techniques. We also provided 10,000 tweets from the shared task, split in two parts: 5000 tweets are used to validate models during the training phase, and 5000 tweets to test the models at the end of training. NADI shared a task to focus on the sub-country level of identification for dialects. Therefore, they announced two tasks: the first is a country-level task to learn and classify tweets belonging to a specific country, and the second is a province-level to learn and identify tweets belonging to a specific province. They also provided an additional 10 million unlabeled tweets that can be used in developing our proposed system for both tasks. The labeled dataset was unbalanced, and it has more tweets from Egypt, Iraq, and Saudi Arabia than other countries such as Sudan or Mauritania. For example, there were 21 provinces from Egypt, 12 from Iraq, and 10 from Saudi Arabia, while there was just one province from Sudan and Mauritania each. We also found that some of the non-Arabic tweets in the dataset from certain countries and provinces belong to non-Arabic languages that use the Arabic alphabet. For instance, Kurdish in at least 4 Iraqi's provinces, Urdu in some tweets from the United Arab Emirates and Oman, and Farsi in Some tweets from Iraq and Yemen. We also realized many considerable numbers of tweets in languages that use the Latin alphabet, such as French from Tunisia, Algeria, and Morocco, and tweets in Somali from Somalia, as well as tweets in English from the rest of the regions. Figure 5.2 shows the distribution of tweets over country. Both labeled and unlabeled datasets were cleaned by removing the non-Arabic words, diacritics "tashkeel", punctuation, numbers. Figure 5.3 shows a histogram of the number of words in the different tweets. We can observe that the longest tweet is less than 60 words.

We re-modelled the data to have different versions from the original version provided by NADI, the first re-modelled version is having 19 classes instead of 21 classes of countries. The deleted countries were Somalia and Djibouti due to two reasons: too few tweets in the dataset from these countries, and these are not basically Arabic speaking countries as the rest of countries in the study. Even though Somalia and Djibouti are members in the Arab League and they do declare the Arabic language as official, it is not commonly and publicly used there at the same level as the indigenous languages, which are the Somalian langauge in Somalia, and the Somalian

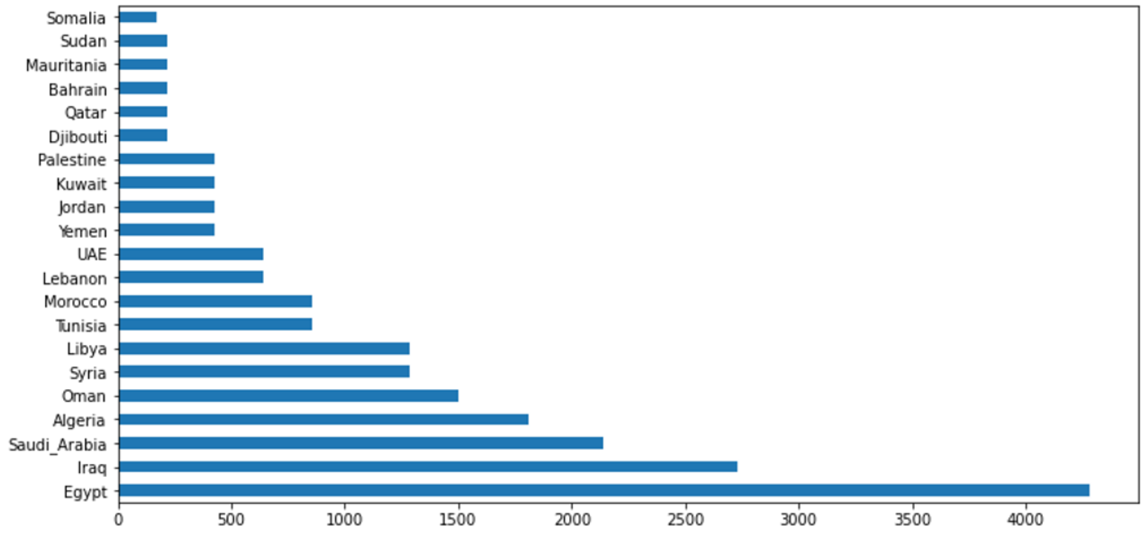


Figure 5.2: The distribution of tweets over countries

and the Afar langauges in Djibouti. There are other countries that also have a small number of tweets and they do not provide much traning data, but we did not eliminate them because they are still actual Arabic speaking countries, and we want to use there tweets in the next re-modelled version. The second re-modelled version is having region-based classes instead of countries. We grouped country classes considering the geographical location and neighbouring to form four new classes. Gulf class that contains seven Arabian Peninsula countries which are: Saudi Arabia, Kuwait, Bahrain, Qatar, United Arab Emirates, Oman, and Yemen. The next region class is Orient class that contains five countries from Mesopotamia and Levant area countries which are: Iraq, Syria, Lebanon, Jordan, and Palestine. Another region class is Nile class that contains two countries from Nile River basin countries which are: Egypt and Sudan. The last region class is Maghreb class, and Maghreb means the west in Arabic language, that contains the five countries from the rest of north Africa and far west of Arab world countries which are: Libya, Tunisia, Algeria, Morocco, and Mauritania.

The last re-modelling approach version is called the plus version, which is motivated by an idea to notice the impact of increasing the data of the countries that have fewer tweets. We increased the main dataset (NADI 2020), with more tweets from (NADI 2021) dataset from all countries, except Egypt, top country in the number of tweets, because of its high percentage. We added those tweets to the new re-modelled

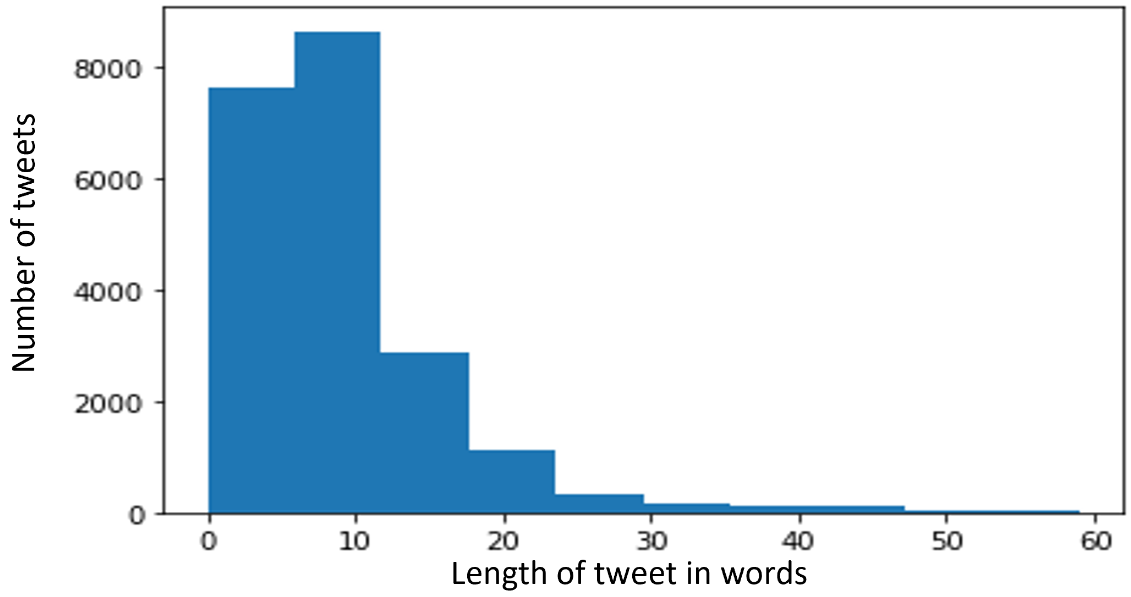


Figure 5.3: Histogram of the number of words in the different tweets

datasets as well. Therefore, we had six different versions of tweets. From Figure 5.4 we can see the train tweets count before and after increasing the dataset, and we can see in the tables Table 5.1 and Table 5.2 the descriptions of the datasets.

5.4 Proposed System

Our proposed classification pipeline, after collecting the data, consists of three steps: the pre-processing step, feature engineering step, and the learner. These steps and overall system architecture is shown in Figure 5.5.

5.4.1 Data Preprocessing Step

The pre-processing step involves data cleaning, such as removal of punctuation, stop-words, user mentions, onomatopoeias, repeated character contraction, and hashtag word splitting. We re-modelled the datasets to have different versions to compare the learning accuracies and notice the impact of changing the class percentages, as discussed in the previous section.

Dataset version	Original Dataset C21					C21Plus			C19		C19Plus	
Country	Train	Dev	Test	Total	%	Train	Total	%	Total	%	Total	%
Egypt	4220	1032	989	6241	20%	4220	6241	14%	6241	21%	6241	15%
Iraq	2719	671	652	4042	13%	4598	5921	14%	4042	13%	5921	14%
Saudi Arabia	2110	510	510	3130	10%	4011	5031	12%	3130	10%	5031	12%
Algeria	1899	427	439	2765	9%	2793	3659	8%	2765	9%	3659	9%
Oman	1477	341	357	2175	7%	2149	2847	7%	2175	7%	2847	7%
Syria	1266	309	306	1881	6%	2007	2622	6%	1881	6%	2622	6%
Libya	1266	310	307	1883	6%	2004	2621	6%	1883	6%	2621	6%
Morocco	844	207	205	1256	4%	1765	2177	5%	1256	4%	2177	5%
Tunisia	844	170	176	1190	4%	1427	1773	4%	1190	4%	1773	4%
UAE	633	154	153	940	3%	1670	1977	5%	940	3%	1977	5%
Lebanon	633	155	141	929	3%	1097	1393	3%	929	3%	1393	3%
Jordan	422	103	102	627	2%	754	959	2%	627	2%	959	2%
Yemen	422	88	102	612	2%	1296	1486	3%	612	2%	1486	3%
Kuwait	422	103	102	627	2%	714	919	2%	627	2%	919	2%
Palestine	422	102	102	626	2%	746	950	2%	626	2%	950	2%
Qatar	211	52	51	314	1%	450	553	1%	314	1%	553	1%
Mauritania	211	52	51	314	1%	362	465	1%	314	1%	465	1%
Bahrain	211	52	51	314	1%	331	434	1%	314	1%	434	1%
Sudan	211	48	51	310	1%	377	476	1%	310	1%	476	1%
Djibouti	211	52	51	314	1%	322	425	1%	-	-	-	-
Somalia	346	62	102	510	2%	625	789	2%	-	-	-	-
Total	21000	5000	5000	31000	-	33718	43718	-	30176	-	42504	-

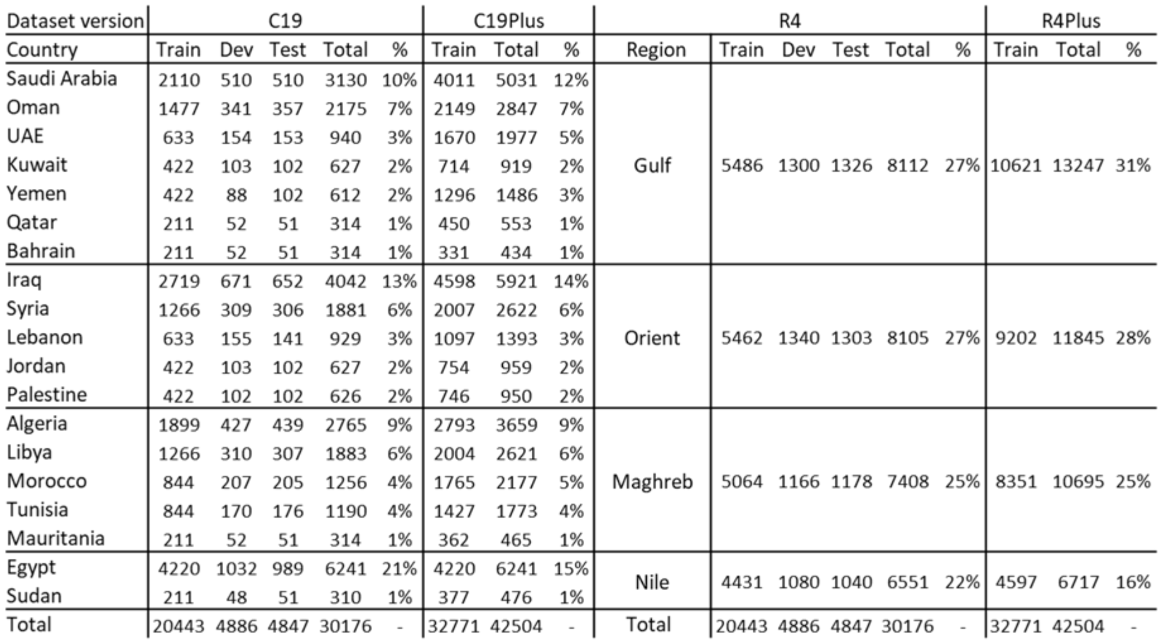
Table 5.1: The details of the original dataset `C21`, and the re-modelled datasets `C21P`, `C19`, and `C19P`

5.4.2 Feature Engineering Step

In the feature engineering step, we chose to generate features using 3 different techniques: tf-idf n-grams, AraVec, and BERT pre-trained models.

TF-IDF N-grams

The first technique is generating n-grams ($n \in \{1, 2, 3\}$) features using the tf-idf technique. We used `TfidfVectorizer` [118] from `scikit-learn` library [117] to gain the maximum of 10,000 features to feed later into the traditional machine learning algorithms to train the Support Vector Machine (SVM), Random Forest (RF), Naïve Bayes (NB), and Logistic Regression (LR) models.



Dataset version	C19					C19Plus				R4					R4Plus		
Country	Train	Dev	Test	Total	%	Train	Total	%	Region	Train	Dev	Test	Total	%	Train	Total	%
Saudi Arabia	2110	510	510	3130	10%	4011	5031	12%									
Oman	1477	341	357	2175	7%	2149	2847	7%									
UAE	633	154	153	940	3%	1670	1977	5%									
Kuwait	422	103	102	627	2%	714	919	2%	Gulf	5486	1300	1326	8112	27%	10621	13247	31%
Yemen	422	88	102	612	2%	1296	1486	3%									
Qatar	211	52	51	314	1%	450	553	1%									
Bahrain	211	52	51	314	1%	331	434	1%									
Iraq	2719	671	652	4042	13%	4598	5921	14%									
Syria	1266	309	306	1881	6%	2007	2622	6%									
Lebanon	633	155	141	929	3%	1097	1393	3%	Orient	5462	1340	1303	8105	27%	9202	11845	28%
Jordan	422	103	102	627	2%	754	959	2%									
Palestine	422	102	102	626	2%	746	950	2%									
Algeria	1899	427	439	2765	9%	2793	3659	9%									
Libya	1266	310	307	1883	6%	2004	2621	6%									
Morocco	844	207	205	1256	4%	1765	2177	5%	Maghreb	5064	1166	1178	7408	25%	8351	10695	25%
Tunisia	844	170	176	1190	4%	1427	1773	4%									
Mauritania	211	52	51	314	1%	362	465	1%									
Egypt	4220	1032	989	6241	21%	4220	6241	15%	Nile	4431	1080	1040	6551	22%	4597	6717	16%
Sudan	211	48	51	310	1%	377	476	1%									
Total	20443	4886	4847	30176	-	32771	42504	-	Total	20443	4886	4847	30176	-	32771	42504	-

Table 5.2: The mapping of the re-modelled datasets `C19`, `C19P` in comparison to the region based re-modelled datasets `R4` and `R4P`

AraVec

The second technique is generating Arabic word2vec embeddings using AraVec [150], which is a pre-trained distributed word representation (word embedding) open-source project. Aravec created using deep learning algorithms, specifically CBOW and SG models, and it uses a large corpus of Arabic text as training data. It is particularly useful for various NLP tasks such as, text classification, text clustering, information retrieval, and sentiment analysis for Arabic language. The pre-trained word embedding can also be used as an input to downstream machine learning models to further improve performance on various tasks. We used the Genism library to generate word vectors using Word2Vec by loading the `full_grams_cbow_100_twitter` pre-trained Twitter-based word embedding model, which contains 66,900,000 documents, 1,476,715 vocabularies, and applied CBOW technique with 100 vectors [149]. An embedding matrix was built using these vectors and then used as weighted parameters in the embedding layer within an Bi-LSTM-based model which consists of an embedding layer, an Bi-LSTM layer, and a fully connected layer [79].

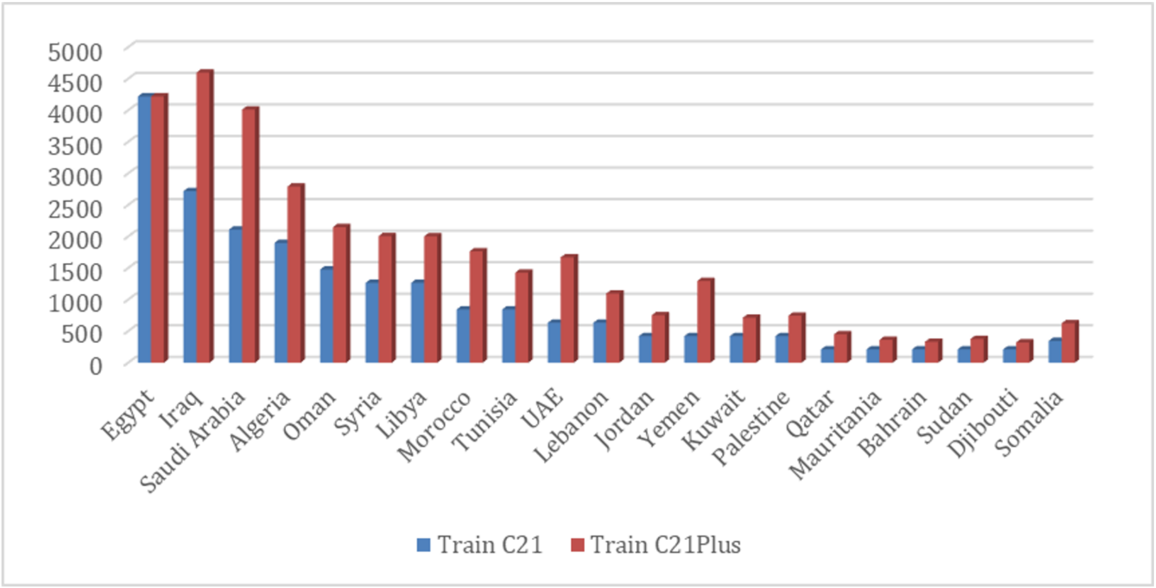


Figure 5.4: Train tweets counts before and after increasing the dataset

BERT

The third technique is based in finding the fine-tuned generated features using the Bidirectional Encoder Representations from Transformers (BERT) model, which is a transformer-based machine learning technique for natural language processing pre-training developed by Google.

In this study, we pretrained six different BERT-based pre-trained models, using the HuggingFace [166] Pytorch library [115], in order to classify the tweets into the wanted number of classes. The six pre-trained models are:

- asafaya/bert-base-arabic (ASAFAYA) [134], a retrained BERT base language model for Arabic which was pretrained on 8.2 Billion words,
- aubmindlab/bert-base-arabert (ARABERT) [26], an Arabic pretrained lanaguage model based on Google's BERT architechture, which was pretrained on 2.7 Billion words,
- aubmindlab/bert-base-arabertv02-twitter (ARABERTV2T) [26], a new model for Arabic dialects and tweets, trained by continuing the pre-training using the MLM task on 60M Arabic tweets (filtered from a collection on 100M), and 8.6 Billion word,

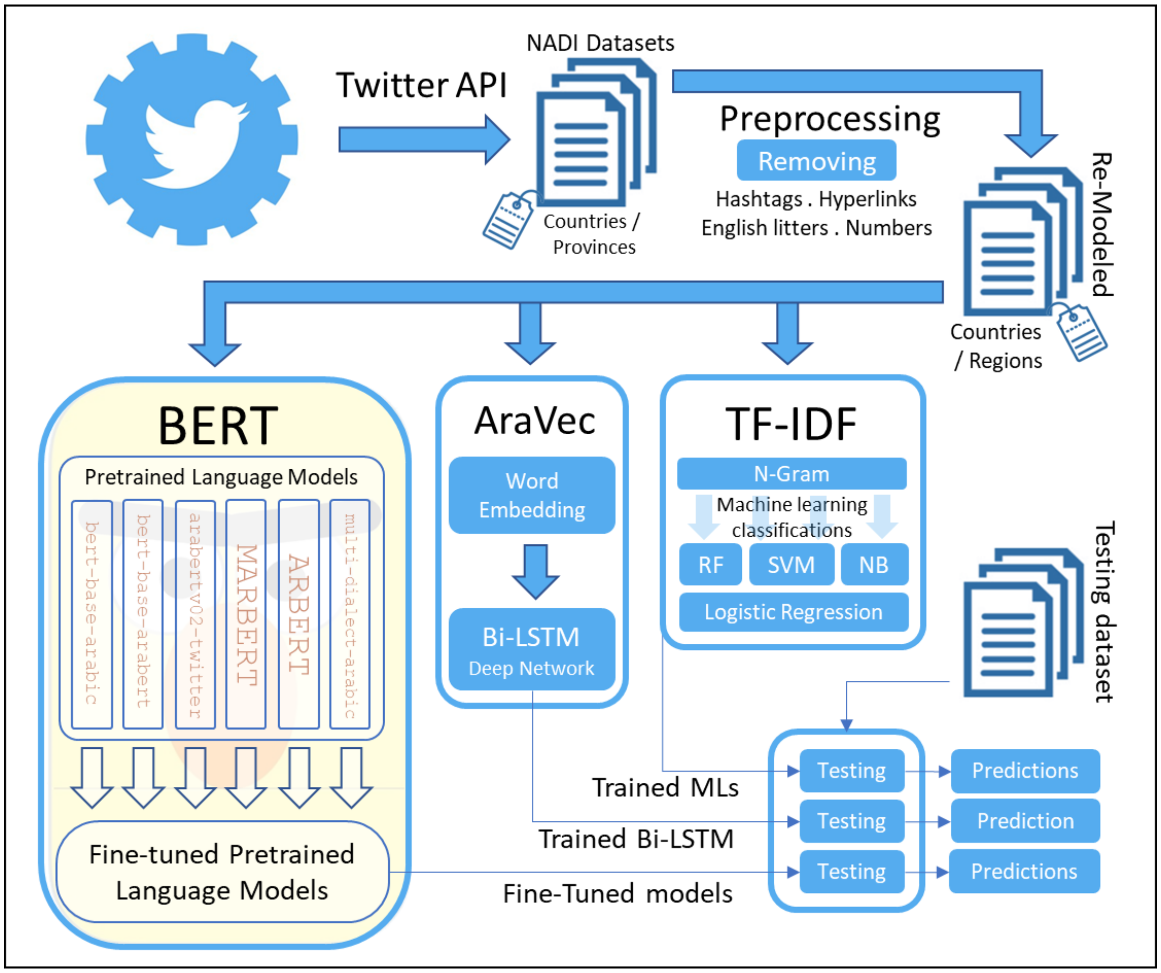


Figure 5.5: The proposed system architecture

- UBC-NLP/marbert (MARBERT) [8], a large scale pre-training masked language model focused on both DA and MSA. Arabic has multiple varieties. To train MARBERT, randomly sample 1 Billion Arabic tweets from a large dataset of about 6 Billion tweets,

- UBC-NLP/arbert (ARBERT) [8], a large scale pre-training masked language model focused on MSA. To train ARBERT, the same architecture as BERT-base is used: 12 attention layers, each has 12 attention heads and 768 hidden dimensions, a vocabulary of 100K WordPieces, making 163M parameters. To train ARBERT on a collection of Arabic datasets comprising 61 GB of text (6.2 Billion tokens),

- bashar-talafha/multi-dialect-bert-base-arabic (MULTIDB) [153], a model using

Arabic-BERT and trained it on 10M arabic tweets from the unlabled data of The Nuanced Arabic Dialect Identification (NADI) shared task.

We extracted the maximum number of words in the applied tweets in the study, which is 59 words. Bert models use the maximum number of words in the document to generated a tensor feature named `InputFeature` object, which is a single set of features of data, this tensor include the following important argument:

- `input_ids`: sequence of token ids, the rest of the 59 words filled with zeros.
- `attention_mask`: sequence of 1s (for "real" tokens) and 0s (for padding tokens),
- `token_type_ids`: aka `segment_ids`. Indicators for which sentence (or sequence each token belongs to). Classical BERT supports only 0s and 1s (for first and second sentence, respectively),
- `label`: Represents training example classification labels,

Figure 5.6 shows an example of the design of the generated BERT input features for an Arabic tweet. After finding the fine-tuned model we apply the argmax function [44] for the learning step.

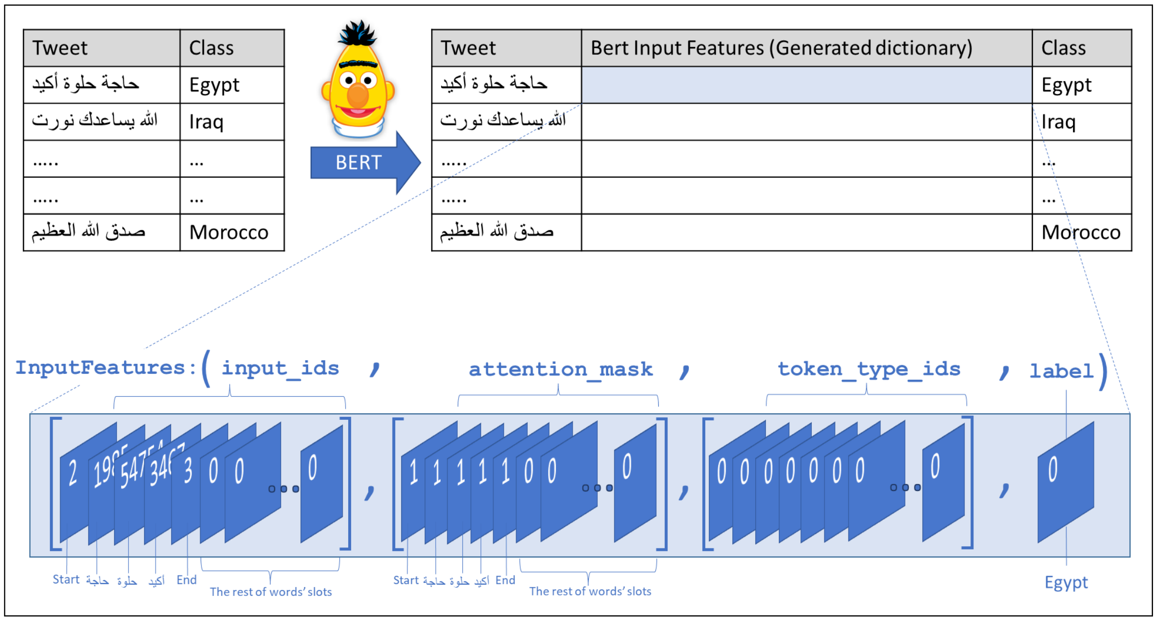


Figure 5.6: The design of the generated BERT input features for an Arabic tweet

5.4.3 Learning Step

The last step is the learning step, we applied each of the three machine learning approaches, trained machine learning algorithms from tf-idf , the trained Bi-directional Long Short-Term Memory (Bi-LSTM) Vectors from AraVec, and the fine-tuned BERT model, to the same testing dataset, to find the prediction and the accuracy results. The performance of all the experiments was then compared regarding to the accuracy of the learning and the F-score results. Given that we have used less than half of the unlabelled data, there is still room for improvement.

To learn dialects from the generated n-grams features we used traditional machine learning; SVM, RF, MultinominalNB, and LR models.

Support Vector Machine (SVM) is a machine learning algorithm used for classification and regression analysis. Using Python, from `sklearn.svm` we import `SVC` function. We experimented SVM with RBF kernel.

For the Random Forest (RF), Using Python, from `sklearn.ensemble` we import `RandomForestClassifier` function. we set `n_estimatorsint` with 200, which is he number of trees in the forest.

For Multinominal Naive Bayes (MultinominalNB), is a probabilistic algorithm used for text classification. it is a variant of the Naive Bayes, which is based on Bayes theorem and assumption of Independence between features. Using Python, from `sklearn.ensemble` we import `MultinomialNB` function.

Logistic Regression (LR), is a statistical model used to predict the probability of binary outcomes, the model fits a logistic function, also known as sigmoid function, to the observed data, which transforms the output of the linear regression model into a probability value between 0 and 1. It can be also extended to multiclass classification by using softmax function. relationship between a dependent variable and one or more independent variables. Using Python, from `sklearn.linear_model` we import `LogisticRegression` function.

For the AraVec experiment, Bi-LSTM model was applied to learn the Arabic dialect. In word embedding it is a NLP technique that combine the Bi-directional Long Short-Term Memory archetecture and word embedding. in this approach, a Bi-LSTM is trained to model sequential relationships between words in a text cotpus, while word embedding s are used to represent each word as a dense vector. Using Bi-LSTM with Aravec embedding is a powerful approach for natural language processing tasks in the Arabic language. After providing pretrained word embedding model using Aravec, it can be used as input to a Bi-LSTM network to obtain context-aware representations of Arabic word. Using Python, from `lstm_classifier` we import `SequenceDataset` function with `lstm_classifier`, we set epoch by 30, learning rate by 0.001, dropout by 0.4, and dimension of the embedding by 100 vectors.

For the BERT experiments, the last step is a simple linear classifier (the final layer in the neural network) which was finally trained on the training subset provided by the organizers. The library for the language model experiments used is `huggingface` [166] in Python. In training the BERT classifiers, we applied batch size of 16, with three epochs and learning rate of $\epsilon = 3.5 \cdot 10^{-6}$. In training the BERT classifiers, we applied 30 epochs and a learning rate of 0.001.

It is important to mention that we started our experiments with the simpler architectures using probabilistic classifiers with no embedding layers. However, none of the models could converge, and rare classes such as Bahrain and Qatar were not detected. We experimented with linear kernel SVM, RBF kernel SVM, Random Forest, and Logistic Regression. We used generated lists of characteristic n-grams per country described in the previous section in this experimental setup.

5.5 Results

As previously mentioned, several approaches were investigated in the experiments we conducted, starting with the classical Machine Learning techniques, moving to Deep Learning technique, and finally settling on our successful BERT-based model. For our traditional Machine Learning experiments, we tried various models such as Support Vector Machine (SVM), Radial Basis Function (RBF) kernel SVM, Logistic Regression (LR), Multinominal Naive Bayes (MultinominalNB), and Random Forest (RF) Random Forest (RF), along with tf-idf generated n-gram features. We apply these techniques on all dataset models that we have. Table 5.3 shows the results of the learning accuracy in terms of the F1-score.

The results show a superiority of the RBF algorithm over the other machine learning algorithm in most of the experiments, with rivalry of MultinominalNB in the

Regions datasets, and rivalry of LR in `C21P`, `C19`, `C19P` datasets. MultinominalNB and RBF methods score the highest accuracy and F1-score with 53% for each of them. In the original data C21, we found the same result of F1-score with 17% when compared to the work of Abdul-Mageed et al. ARBERT [8] , but we had a higher accuracy with 32% using RBF in comparison to their work which gained 30%.

TF-IDF N-Grams		Datasets					
ML Classifier		C21	C21Plus	C19	C19Plus	R4	R4Plus
Linear SVM	Acc:	0.3	0.33	0.3	0.33	0.47	0.5
	F1:	0.17	0.23	0.28	0.32	0.47	0.5
RBF Kernel SVM	Acc:	0.32	0.34	0.32	0.35	0.49	0.53
	F1:	0.17	0.19	0.28	0.32	0.5	0.53
Multinominal NB	Acc:	0.29	0.33	0.3	0.33	0.48	0.53
	F1:	0.08	0.15	0.21	0.26	0.49	0.53
Random Forest	Acc:	0.29	0.32	0.3	0.32	0.43	0.47
	F1:	0.15	0.2	0.25	0.28	0.44	0.47
Logistic Regression	Acc:	0.31	0.35	0.32	0.35	0.48	0.52
	F1:	0.13	0.2	0.26	0.31	0.49	0.53

Table 5.3: The results of accuracy and F-scores in the tf-idf classification experiments

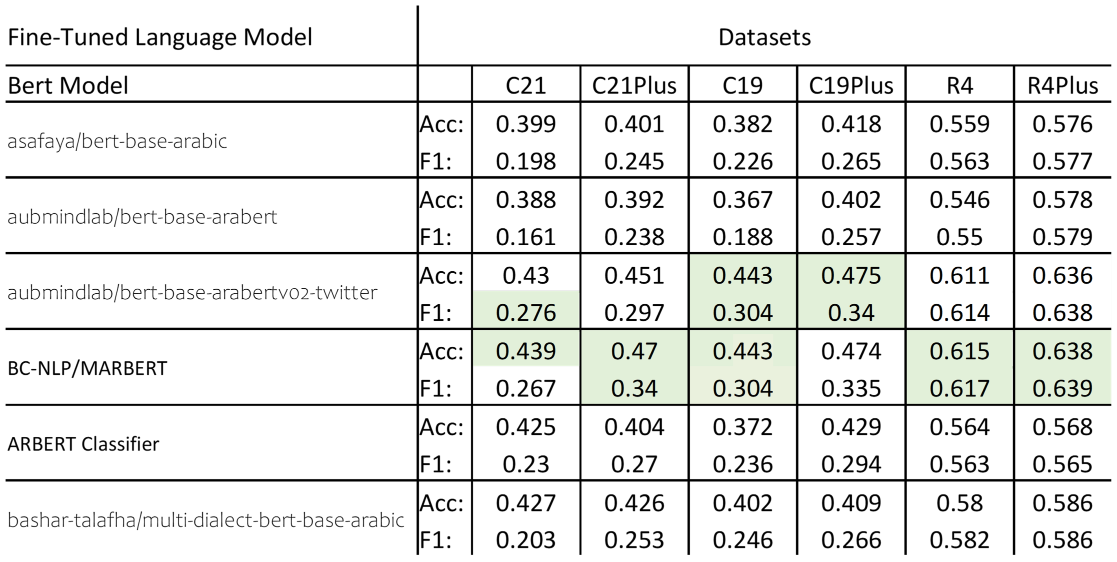
For word embedding deep learning technique, we used AraVec pre-trained distributed word embedding vectors, And use the embedding layer within an Bi-directional Long Short-Term Memory (Bi-LSTM) model to predict the dialect.

The results shows a slight improvement on accuracy results in all experiments in comparison to RBF kernel SVM algorithm in the tf-idf method. This improvement was missing in the F-scores. In `C21`, `C19`, and `R4` experiments, we found over 56,000 tokenized unique words with embedding, but AraVec could not find vector for 19528 words (almost 25% of the words). In the "Plus" datasets, we found over 57,000 tokenized unique words with embedding, but AraVec could not find vector for 31,436 words (almost 29% of the words). That shows the importance of having vectors for all the possible words, but AraVec vectors not covering all Arabic words in all dialects. Table 5.4 shows the results of the AraVec experiments using the Bi-LSTM neural network.

The architecture with AraBERT embeddings is the only one able to detect all the

AraVec Embeddings		Datasets					
		C21	C21Plus	C19	C19Plus	R4	R4Plus
Bi-LSTM	Acc:	0.33	0.36	0.33	0.36	0.5	0.54
	F1:	0.14	0.2	0.14	0.18	0.5	0.54

Table 5.4: The results of accuracy and f-scores in the AraVec experiments



Fine-Tuned Language Model		Datasets					
Bert Model		C21	C21Plus	C19	C19Plus	R4	R4Plus
asafaya/bert-base-arabic	Acc:	0.399	0.401	0.382	0.418	0.559	0.576
	F1:	0.198	0.245	0.226	0.265	0.563	0.577
aubmindlab/bert-base-arabert	Acc:	0.388	0.392	0.367	0.402	0.546	0.578
	F1:	0.161	0.238	0.188	0.257	0.55	0.579
aubmindlab/bert-base-arabertv02-twitter	Acc:	0.43	0.451	0.443	0.475	0.611	0.636
	F1:	0.276	0.297	0.304	0.34	0.614	0.638
BC-NLP/MARBERT	Acc:	0.439	0.47	0.443	0.474	0.615	0.638
	F1:	0.267	0.34	0.304	0.335	0.617	0.639
ARBERT Classifier	Acc:	0.425	0.404	0.372	0.429	0.564	0.568
	F1:	0.23	0.27	0.236	0.294	0.563	0.565
bashar-talafha/multi-dialect-bert-base-arabic	Acc:	0.427	0.426	0.402	0.409	0.58	0.586
	F1:	0.203	0.253	0.246	0.266	0.582	0.586

Figure 5.7: The results of accuracy and f-scores in fine-tuned BERT models

dialects to some level. The other models, such as SVM, Random Forest, and Random Forest, could not deal with under-resourced dialects.

5.6 Discussion

We build a confusion matrix for all countries to visualize the percentage of correctly predicted tweets and the similarities between countries. In Figure 5.8 we picked up two examples of grouped neighboring countries. The first group has Saudi Arabia, Kuwait, Iraq, Bahrain, and Syria. It shows that most tweets of small countries such as Kuwait and Bahrain went to bigger countries, mostly Saudi Arabia, which confirms the influence of the Saudi dialect on other countries in the Gulf area. The second subset, which includes Egypt, Sudan, Libya, Tunisia, and Palestine, shows the big influence of Egypt on the closer countries to it, first Sudan and Palestine, and then

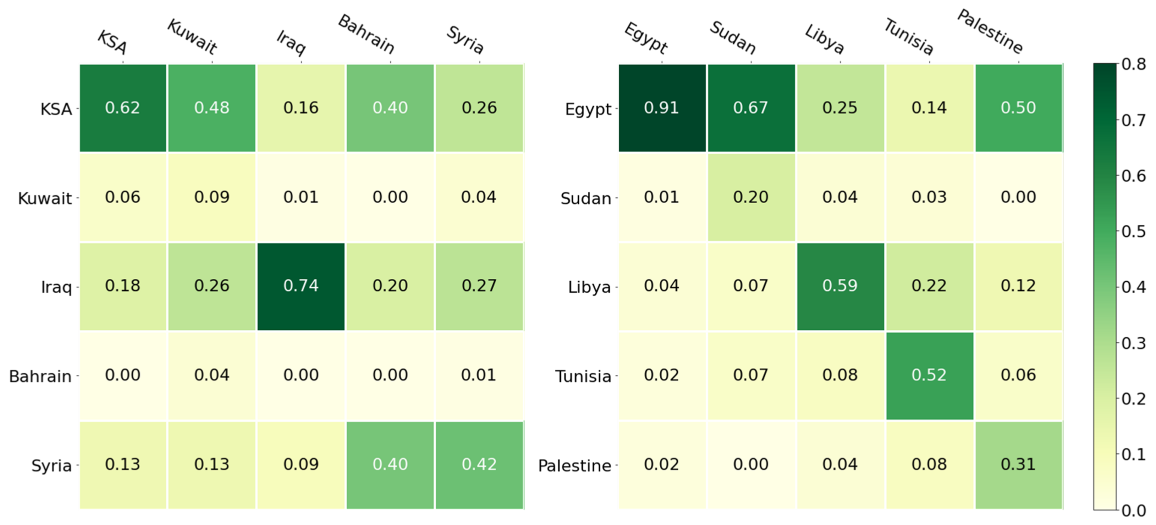


Figure 5.8: Confusion matrix for two subsets of Arab countries

Libya. We also observed the confusion matrix of each region that was previously shown in Figure 5.1. We found the big influence of Algerian dialect on Maghreb region, Iraq dialect over orient region and Saudi Arabia on Arabia region, and there is a bigger influence of Egyptian dialect over all countries. That may interpret the effect of having an unbalanced dataset due to the higher number of tweets from these four countries than from other countries. Among other interesting observations, we found a strong percentage of predicted tweets of Yemen in Arabia region countries, even with having a lesser number of tweets than Saudi Arabia. In future work, we would like to try more balanced tweets from all countries to see if it impacts results. Furthermore, we would like to consider the proposed categorization in Figure 5.1 as the main classification task, then get deeper with more classes by countries and sub-regions. As another future direction, would could try other tweet embedding techniques such as ArWordVec [61] and enhance the generated n-gram dictionaries with more words for each region and country to discover its impact on the results.

5.7 Conclusion and Future Work

In this study, we presented an architecture for Arabic dialect detection. We experimented feature engineering techniques, the n-gram feature using tf-idf, the word embedding feature using AraVec, and the fine-tuned language model using BERT models, on the Twitter dataset from the NADI shared task 2020 and 2021. On six different versions of re-modelled datasets, we used more traditional algorithms involving probabilistic models such as SVM, Random Forest, Naïve Bayes, RBF, and Logistic Regression, as well as a model with embedding representation which was trained on a large external Arabic corpus, then using the Bi-LSTM learning on the trained model. Moreover, we build six fine-tuned language models based on BERT to apply them in the learning process. We also applied an ensemble model that combines the features of tf-idf using RBF, AraVec, and BERT. The results showed that BERT is more powerful for this study than other methods, specifically `bert-base-arabertv02-twitter` and MARBERT. Our main contribution is comparing different feature engineering methods for Arabic tweets and stating that BERT is the best technique for language modeling and Arabic dialect recognition. The training set is very unbalanced, and that was reflected in the results. After increasing the data, we found that having more balanced data will help to find better accuracy and F1-score results. Another finding is having lesser classes by grouping the classes based on geographical location contributes to increasing the accuracy. That call to apply the study in the future on different region groupings or consider developing a hierarchical model consisting of many specialized models for each sub-region. Another future consideration is to expand the training sub-sets for under-resourced dialects and making the datasets balanced.

Chapter 6

Conclusion

Feature engineering is one of the essential steps in machine learning. When it comes to NLP, it helps to create better data which helps the model to understand it well, and provide reasonable results. Therefore, generating and extracting features in Twitter is one of the challenges that we need to apply on many experiments to assist the machine learning field. In our study, we propose applying three different feature generation tasks on three fields: financial forecasting, spams filtering, and linguistic dialect identification. We applied n-grams and word embedding approaches to generate features of Bitcoin tweets for sentiment analysis. The main contributions of this research are finding a partial correlation between the Bitcoin price fluctuation and the fluctuation of the sentiment classes using different machine learning algorithms and providing a framework of an efficient tweet sentiment tool for Bitcoin tweets, whether they are positive or not. In the second study, we applied Word2Vec and n-grams on Arabic tweets for spam filtering in the current work. The main contribution of this research is finding the advantage of Word2Vec approach the over the n-gram feature generation approach. The advantage is on the more balanced datasets, while the least balancing datasets show better n-gram results. We also found the same conclusion in applying the same methodology on the grouped hashtags by topic type. In the third study, we applied six BERT pretrained fine-tuned model on Arabic tweets to recognize the Arabic dialect, aside with the tf-idf n-gram feature generation, and AraVec word embedding model. The results showed that BERT is more powerful for this study than other methods, specifically `bert-base-arabertv02-twitter` and MARBERT. A common finding that we discovered is that generating features for Tweets is improving the learning process, weather the Tweet were in Arabic or in English. Another expected finding is that having balanced class data help in enhancing learning accuracy, and our finding is that n-gram-based features better cope with class imbalance. For the imbalanced data (`C21`, `C19`, `R4`) we noticed a better

F-score results for tf-idf n-grams in comparison with AraVec embeddings, but still the `bert-base-arabertv02-twitter` and MARBERT has better F-scores results overall experiments. These are some lessons that should be considered in future studies in the effect of balanced and unbalanced datasets on the choice of feature engineering techniques in Twitter datasets and similar short-text datasets.

6.1 Achieved Objectives

Based on the thesis problem statement, we applied several methods to increase the number of limited features on Twitter text data. We generated features in the financial forecasting study using n-grams and tweet embeddings using Weka. We generated features in the spam recognition study by applying n-grams and Word2Vec approaches, and we generated features on Arabic dialects identification study, by applying tf-idf n-grams, Ara-Vec, and six different BERT based fine-tuned models on the tweets. We also achieved other goals:

- Comparing the quality of multiple feature engineering techniques on Tweets.
- Applying those techniques to tweets from different fields of interest, in different languages.

These results were published in six peer-referred publications, four In-proceeding papers, and two journals, as well as two published video presentations.

6.2 Study Limitations

When conducting feature engineering on Twitter data, there are several limitations that should be considered, and these affected research presentend here. These limitations were:

- Limited access to the Twitter API to get more related tweets in a short time window.
- Lack of research and resources on generating features for tweets in the Arabic language.

- Time-consuming experiments to get the results due to the limitation in the computer memory.

6.3 Future Work

As this thesis consists of three different studies, there are some future work directions that can be generally considered in all studies, like applying advanced text processing or making further analysis on tweet sentiment, but we can be more specific in ideas based on the current findings:

- We can involve more than one language in each study, for example, applying Arabic tweets in the financial forecasting study, and English tweets in the spam recognition study, and noticing the difference between the two languages when we apply the same methodology.
- Considering the balancing of classes in the dataset. We noticed an impact of having more tweets from one class over the other class, especially in the spam recognition study, and the dialect identification study. The results encourage us to apply the fine-tuned model to equal classes in the number of tweets for the Arabic dialect identification study.
- Applying the fine-tuned models in the financial forecasting study, and in the spam recognition study, since we noticed its efficiency on the Arabic dialect identification study.

Bibliography

[1] fse/word2vec-google-news-300. *Hugging Face*, https://huggingface.co/fse/word2vec-google-news-300, last accessed 2023-03-31.

[2] Most visited website - 1996/2022 - statistics and data. Statistics and Data, Mar 2022, https://statisticsanddata.org/data/most-popular-website/, last accessed 2023-03-31.

[3] Twitter API. Twitter Developer platform , https://developer.twitter.com/en/docs/twitter-api, last accessed 2023-03-31.

[4] Twitter, inc. net promoter score 2023 benchmarks. Customer.guru , https://customer.guru/net-promoter-score/twitter-inc, last accessed 2023-03-31.

[5] Twitter to expand 280-character tweets. BBC News, Nov 2017, https://www.bbc.com/news/technology-41900880, last accessed 2023-03-31.

[6] Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S. Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Ian Goodfellow, Andrew Harp, Geoffrey Irving, Michael Isard, Yangqing Jia, Rafal Jozefowicz, Lukasz Kaiser, Manjunath Kudlur, Josh Levenberg, Dandelion Mané, ..., and Xiaoqiang Zheng. TensorFlow: Large-scale machine learning on heterogeneous systems, 2015. Software available from tensorflow.org.

[7] Muhammad Abdul-Mageed, Hassan Alhuzali, and Mohamed Elaraby. You tweet what you speak: A city-level dataset of Arabic dialects. In *Proceedings of the Eleventh International Conference on Language Resources and Evaluation (LREC 2018)*, Miyazaki, Japan, May 2018. European Language Resources Association (ELRA).

[8] Muhammad Abdul-Mageed, AbdelRahim Elmadany, and El Moatez Billah Nagoudi. ARBERT & MARBERT: Deep bidirectional transformers for Arabic. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 7088–7105, Online, Aug 2021. Association for Computational Linguistics.

[9] Muhammad Abdul-Mageed, Chiyu Zhang, Houda Bouamor, and Nizar Habash. NADI 2020: The first nuanced Arabic dialect identification shared task. In *Proceedings of the Fifth Arabic Natural Language Processing Workshop*, pages 97–110, Barcelona, Spain (Online), December 2020. Association for Computational Linguistics.

[10] Muhammad Abdul-Mageed, Chiyu Zhang, AbdelRahim Elmadany, Houda Bouamor, and Nizar Habash. NADI 2021: The second nuanced Arabic dialect identification shared task. In *Proceedings of the Sixth Arabic Natural Language Processing Workshop*, pages 244–259, Kyiv, Ukraine (Virtual), April 2021. Association for Computational Linguistics.

[11] Sarah Aboulhosn. Twitter trending topics: How they work and how to use them, Mar 2021.

[12] Abien Fred Agarap. Deep learning using rectified linear units (relu). *CoRR*, abs/1803.08375, 2018.

[13] Sadam Al-Azani and El-Sayed M. El-Alfy. Detection of arabic spam tweets using word embedding and machine learning. *2018 International Conference on Innovation and Intelligence for Informatics*, 2018.

[14] Ahmad Al-Jallad. The polygenesis of the Neo-Arabic dialects. *Journal of Semitic Studies*, Volume 54:Pages 515–536, Oct 2009.

[15] Nuha Albadi. Hateful people or hateful bots? *Proceedings of the ACM on Human-Computer Interaction*, 2019.

[16] Alexa. Alexa web site, 2023. https://www.alexa.com, last accessed 2023-03-31.

[17] Adel R. Alharbi and Amer Aljaedi. Predicting rogue content and Arabic Spammers on Twitter. *Future Internet*, 2019.

[18] Mohamed Ali. Character level convolutional neural network for Arabic dialect identification. In *Proceedings of the Fifth Workshop on NLP for Similar Languages, Varieties and Dialects (VarDial 2018)*, pages 122–127, Santa Fe, New Mexico, USA, Aug 2018. Association for Computational Linguistics.

[19] Nada Allouch. Sentiment and emotional analysis: The absolute difference - emojics blog. *Emojics*, 2019.

[20] Hind Almerekhi and Tamer Elsayed. Detecting automatically-generated arabic tweets. *Springer*, 2015.

[21] Dema Alorini and Danda B. Rawat. Automatic spam detection on gulf dialectical arabic tweets. *IEEE*, 2020.

[22] Naomi S. Altman. An introduction to kernel and nearest-neighbor nonparametric regression. *The American Statistician*, 46(3):175–185, 1992.

[23] Nada Altuwaijri. “Salaries not enough”: Saudis shout on social media. https://english.alarabiya.net/variety/2013/08/13/Millions-of-Saudis-complain-on-Twitter-about-salaries, last accessed 2023-03-31.

[24] Omar AlZoubi, Saja Khaled Tawalbeh, and Mohammad AL-Smadi. Affect detection from Arabic tweets using ensemble and deep learning techniques. *Journal of King Saud University – Computer and Information Sciences*, 2020.

[25] Ulrich Ammon. *World Languages: Trends and Futures*, pages 101–122. 10 2010.

[26] Wissam Antoun, Fady Baly, and Hazem Hajj. AraBERT: Transformer-based model for Arabic language understanding. In *LREC 2020 Workshop Language Resources and Evaluation Conference 11–16 May 2020*, page 9.

[27] Rahul Awati and Taina Teravainen. What is email spam and how to fight it? *TechTarget*. https://www.techtarget.com/searchsecurity/definition/spam, last accessed 2023-03-31.

[28] V. Kishore Ayyadevara. *Word2vec*, pages 167–178. Apress, Berkeley, CA, 2018.

[29] Ahmed Balfagih and Vlado Keselj. Twitter arabic dialect identification using arabert. In *39th IBIMA Computer Science Virtual Conference 30-31 May 2022*. IBIMA, 2022.

[30] Ahmed Balfagih and Vlado Keselj. Twitter arabic dialect identification using arabert. In *40th IBIMA Conference: 23-24 November 2022, Seville, Spain*. IBIMA, 2022.

[31] Ahmed Balfagih, Vlado Keselj, and Stacey Taylor. N-gram and word2vec feature engineering approaches for spam recognition on some influential twitter topics in saudi arabia. In *Proceedings of the 6th International Conference on Information System and Data Mining*, ICISDM '22, page 101–107, New York, NY, USA, 2022. Association for Computing Machinery.

[32] Ahmed Balfagih, Vlado Keselj, and Stacey Taylor. N-gram and word2vec feature engineering approaches for spam recognition on some influential twitter topics in saudi arabia. *Journal of Advances in Information Technology (JAIT)*, 13 No. 6:562–568, 2022.

[33] Ahmed M. Balfagih and Vlado Keselj. Evaluating sentiment c1assifiers for Bitcoin tweets in price prediction task. In *2019 IEEE International Conference on Big Data (Big Data)*, page 5499–5506, Dec 2019.

[34] Trisha Dowerah Baruah, Krishna Kanta, and Handiqui State. Effectiveness of social media as a tool of communication and its potential for technology enabled connections: A micro-level study. *International Journal of Scientific and Research Publications*, 2, 2012.

[35] BBC. Hisham Malaikah: Who is he? why did he have to delete his account from Twitter?, 2021. Original from Arabic: https://www.bbc.com/arabic/trending-50973909, last accessed 2023-03-31.

[36] Christopher M. Bishop. *Pattern Recognition and Machine Learning*. Springer New York, New York, NY, 2016.

[37] Bayan Boreggah, Arwa Alrazooq, Muna Al-Razgan, and Hana AlShabib. Analysis of Arabic bot behaviors. *IEEE*, 2018.

[38] Houda Bouamor, Nizar Habash, Mohammad Salameh, Wajdi Zaghouani, Owen Rambow, Dana Abdulrahim, Ossama Obeid, Salam Khalifa, Fadhl Eryani, Alexander Erdmann, and Kemal Oflazer. The madar arabic dialect corpus and lexicon. In *International Conference on Language Resources and Evaluation*, 2018.

[39] Houda Bouamor, Sabit Hassan, and Nizar Habash. The MADAR shared task on Arabic fine-grained dialect identification. In *Proceedings of the Fourth Arabic Natural Language Processing Workshop*, pages 199–207, Florence, Italy, August 2019. Association for Computational Linguistics.

[40] Felipe Bravo-Marquez, Bernhard Pfahringer, Saif Mohammad, and Eibe Frank. AffectiveTweets: a Weka package for analyzing affect in tweets. *Journal of Machine Learning Research*, 20:1–6, 06 2019.

[41] Leo Breiman. Random forests. In *Machine Learning*, page 5–32, 2001.

[42] James Briggs. Masked-language modelling with bert, Sep 2021. https://towardsdatascience.com/masked-language-modelling-with-bert-7d49793e5d2c.

[43] Andrei Z. Broder, Steven C. Glassman, Mark S. Manasse, and Geoffrey Zweig. Syntactic clustering of the web. *Computer Networks and ISDN Systems*, 29(8):1157–1166, 1997. Papers from the Sixth International World Wide Web Conference.

[44] Jason Brownlee. What is argmax in machine learning?, March 2017. https://machinelearningmastery.com/argmax-in-machine-learning/, last accessed 2023-03-31.

[45] Bala Priya C. Softmax activation function: Everything you need to know. https://www.pinecone.io/learn/softmax-activation/, last accessed 2023-03-31.

[46] National Research Council Canada. Government of canada. https://nrc.canada.ca/en, last accessed 2023-03-31.

[47] Hugo Caselles-Dupré, Florian Lesaint, and Jimena Royo-Letelier. Word2vec applied to recommendation: Hyperparameters matter. *CoRR*, abs/1804.04212, 2018.

[48] Sudhanshu Chauhan and Nutan Kumar Panda. Chapter 2 - open source intelligence and advanced social media search. In Sudhanshu Chauhan and Nutan Kumar Panda, editors, *Hacking Web Intelligence*, pages 15–32. Syngress, Boston, 2015.

[49] Michelle Chen. A guide: Text analysis, text analytics amp; text mining. *Medium*, Oct 2020.

[50] Mitchell Clark. Now twitter blue subscribers can write 4,000-character tweets, Febraury 2023. https://www.theverge.com/2023/2/8/23591472/twitter-blue-subscribers-longer-tweets-4000-characters.

[51] Idit Cohen. Optimizing feature generation. *Medium, Towards Data Science*, Oct 2019. https://towardsdatascience.com/optimizing-feature-generation-dab98a049f2e, last accessed 2022-11-17.

[52] Stefania Cristina. The Transformer model. *MachineLearningMastery.com*, Jan 2023. https://machinelearningmastery.com/the-transformer-model/.

[53] Marc Damashek. Gauging similarity with n-grams: Language-independent categorization of text. *Science*, 267(5199):843–848, 1995.

[54] Suzanne Van den Bosch. Automatic feature generation and selection in predictive analytics solutions. 2017.

[55] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. BERT: pre-training of deep bidirectional transformers for language understanding. *CoRR*, abs/1810.04805, 2018.

[56] Guozhu Dong. A quick guide to feature engineering. *KDnuggets*. https://www.kdnuggets.com/a-quick-guide-to-feature-engineering.html/, last accessed 2021-11-17.

[57] Saige Driver. Twitter for Business: Everything You Need to Know. *Business News Daily*. 2021. https://www.businessnewsdaily.com/7488-twitter-for-business.html, last accessed 2023-03-31.

[58] Frank Eibe. Fully supervised training of gaussian radial basis function networks in weka. In *Department of Computer, Science University of Waikato*, 2014.

[59] Heba Elfardy and Mona Diab. Sentence level dialect identification in Arabic. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 456–461, Sofia, Bulgaria, August 2013. Association for Computational Linguistics.

[60] Facebook. Facebook web site, 2023. https://www.facebook.com, last accessed 2023-03-31.

[61] Mohammed Fouad, Ahmed Mahany, Naif Aljohani, Rabeeh Abbasi, and Saeed-Ul Hassan. ArWordVec: Efficient word embedding models for Arabic tweets. *Soft Computing*, 06 2019.

[62] Evgeniy Gabrilovich and Shaul Markovitch. Feature generation for text categorization using world knowledge. IJCAI'05, page 1048–1053, San Francisco, CA, USA, 2005. Morgan Kaufmann Publishers Inc.

[63] Shreyal Gajare. Tf – idf for bigrams trigrams, September 2019. https://www.geeksforgeeks.org/tf-idf-for-bigrams-trigrams/.

[64] Shreya Ghelani. Breaking bert down. *Medium, Towards Data Science*, Jul 2019. https://towardsdatascience.com/breaking-bert-down-430461f60efb.

[65] Di Gregorio Giordano. The wisard classifier. In *ESANN*, 2016.

[66] Varun Godinho. Two-thirds of saudi arabia's population is under the age of 35. *Gulf Business*, 2020. https://gulfbusiness.com/two-thirds-of-saudi-arabias-population-is-under-the-age-of-35/, last accessed 2023-03-31.

[67] Tian Guo, Albert Bifet, and Nino Antulov-Fantulin. Bitcoin volatility forecasting with a glimpse into buy and sell orders. In *2018 IEEE International Conference on Data Mining (ICDM)*. IEEE, nov 2018.

[68] Felix Hamborg and Karsten Donnay. NewsMTSC: A dataset for (multi-)target-dependent sentiment classification in political news articles. In *Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics: Main Volume*, pages 1663–1675, Online, April 2021. Association for Computational Linguistics.

[69] James Hamilton. *Time Series Analysis*. Princeton University Press, 1994.

[70] Jiawei Han, Micheline Kamber, and Jian Pei. 8 - classification: Basic concepts. In Jiawei Han, Micheline Kamber, and Jian Pei, editors, *Data Mining (Third Edition)*, The Morgan Kaufmann Series in Data Management Systems, pages 327–391. Morgan Kaufmann, Boston, third edition edition, 2012.

[71] Charles R. Harris, K. Jarrod Millman, Stéfan J. van der Walt, Ralf Gommers, Pauli Virtanen, David Cournapeau, Eric Wieser, Julian Taylor, Sebastian Berg, Nathaniel J. Smith, Robert Kern, Matti Picus, Stephan Hoyer, Marten H. van Kerkwijk, Matthew Brett, Allan Haldane, Jaime Fernández del Río, Mark Wiebe, Pearu Peterson, Pierre Gérard-Marchant, Kevin Sheppard, ..., and Travis E. Oliphant. Array programming with NumPy. *Nature*, 585(7825):357–362, September 2020.

[72] Md. Rakibul Hasan, Maisha Maliha, and M. Arifuzzaman. Sentiment analysis with nlp on twitter data. In *2019 International Conference on Computer, Communication, Chemical, Materials and Electronic Engineering (IC4ME2)*, pages 1–4, 2019.

[73] Trevor Hastie, Robert Tibshirani, and Jerome Friedman. *The Elements of Statistical Learning*. Springer New York, New York, NY, 2009.

[74] Trevor Hastie, Robert Tibshirani, and Jerome Friedman. *Neural Networks*, pages 389–416. Springer New York, New York, NY, 2009.

[75] heavy.ai. What is feature engineering? *OmniSci*. https://www.omnisci.com/technical-glossary/feature-engineering last accessed last accessed 2022-11-12.

[76] Garrick Hileman and Michel Rauchs. 2017 global cryptocurrency benchmarking study. *SSRN Electronic Journal*, 2017.

[77] Geoffrey Holmes, Andrew Donkin, and Ian Witten. Weka: A machine learning workbench. In *Proc Second Australia and New Zealand Conference on Intelligent Information Systems*, 1994.

[78] Fei Huang. Improved Arabic dialect classification with social media data. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 2118–2126, Lisbon, Portugal, September 2015. Association for Computational Linguistics.

[79] Zhiheng Huang, Wei Xu, and Kai Yu. Bidirectional LSTM-CRF models for sequence tagging. *CoRR*, abs/1508.01991, 2015.

[80] Taha Hussain. *On Pre-Islamic poetry*. 1926.

[81] IBM. What is natural language processing? https://www.ibm.com/topics/natural-language-processing.

[82] Instagram. Instagram web site, 2023. https://www.instagram.com, last accessed 2023-03-31.

[83] Rajneesh Jha. Automated feature engineering tools. *Analytics Vidhya*, Feb 2020.

[84] Zhao Jianqiang and Gui Xiaolin. Comparison research on text pre-processing methods on twitter sentiment analysis. *IEEE Access*, 5:2870–2879, 2017.

[85] Kaggle. Kaggle datasets, 2019. https://www.kaggle.com/datasets, last accessed 2023-03-31.

[86] James Max Kanter and Kalyan Veeramachaneni. Deep feature synthesis: Towards automating data science endeavors. In *2015 IEEE International Conference on Data Science and Advanced Analytics (DSAA)*, pages 1–10, 2015.

[87] Andreas Kaplan and Michael Haenlein. Users of the world, unite! the challenges and opportunities of social media. *Business Horizons*, 53(1):59–68, 2010.

[88] Simon Kemp. Digital 2020: Global Digital Overview. *DataReportal – Global Digital Insights*, 2021. https://datareportal.com/reports/digital-2020-global-digital-overview, last accessed 2023-03-31.

[89] David Keyes. Saudi writer Hamza Kashgari faces charge of blasphemy after tweets about Muhammad. *Washington Post*, 2012. Original from Arabic: https://www.washingtonpost.com/opinions/saudi-writer-detained-after-tweets-about-muhammad/2012/02/09/gIQApsgW2Q_story.html, last accessed 2023-03-31.

[90] Olga Kolchyna, Tharsis T. P. Souza, Philip Treleaven, and Tomaso Aste. Twitter sentiment analysis: Lexicon method, machine learning method and their combination, 2015.

[91] Peter M. Krafft, Nicolas Della Penna, and Alex Sandy Pentland. An experimental study of cryptocurrency market dynamics. In *Proceedings of the 2018 CHI Conference on Human Factors in Computing Systems*, CHI '18, page 1–13, New York, NY, USA, 2018. Association for Computing Machinery.

[92] Stéphane Lacroix. Between islamists and liberals: Saudi arabia's new "islamo-liberal" reformists. The Middle East Journal. 58(3), pp.345-365. 2004.

[93] Quanzhi Li, Sameena Shah, Rui Fang, Armineh Nourbakhsh, and Xiaomo Liu. Tweet sentiment analysis by incorporating sentiment-specific word embedding and weighted text features. In *2016 IEEE/WIC/ACM International Conference on Web Intelligence (WI)*, pages 568–571, 2016.

[94] Jimmy Lin and Alek Kolcz. Large-scale machine learning at twitter. In *Proceedings of the 2012 ACM SIGMOD International Conference on Management of Data*, SIGMOD '12, page 793–804, New York, NY, USA, 2012. Association for Computing Machinery.

[95] Bin Liu and Bang Wang. Bayesian negative sampling for recommendation. *arXiv*, 2022.

[96] Huan Liu and Hiroshi Motoda. A data mining perspective. In *Feature Extraction, Construction and Selection*, pages 67–72, 2002.

[97] Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. Roberta: A robustly optimized BERT pretraining approach. *CoRR*, abs/1907.11692, 2019.

[98] Christopher G. Lynch. Twitter tips: Url shorteners do more than save space, Jun 2009. https://www.computerworld.com/article/2524807/twitter-tips--url-shorteners-do-more-than-save-space.html, last accessed 2023-03-31.

[99] Edward Ma. 3 silver bullets of word embeddings in nlp. *Towards Data Science*, 2018. https://towardsdatascience.com/3-silver-bullets-of-word-embedding-in-nlp-10fa8f50cc5a, last accessed 2023-03-31.

[100] Amanda MacArthur. The history of twitter you didn't know. *Lifewire*, Nov 2020.

[101] Shaul Markovitch and Dan Rosenstein. Feature generation using general constructor functions. *Machine Learning*, 49(1):59–98, 2002.

[102] Matthew Martin, Archana Narayanan, and Sarah Algethami. Why apple, citigroup and twitter will be tracking this arrest in saudi arabia. *Bloomberg*. The Economic Times. 2017.

[103] Martina Matta, Maria Ilaria Lunesu, and Michele Marchesi. Bitcoin spread prediction using social and web search media. 06 2015.

[104] Chris McCormick. Word2vec tutorial - The skip-gram model. *Chris McCormick Blog*, Apr 2016. https://www.fer.unizg.hr/_download/repository/TAR-2020-reading-05.pdf, last accessed 2023-03-31.

[105] Rada Mihalcea, Carmen Banea, and Janyce Wiebe. Learning multilingual subjective language via cross-lingual projections. *Association for Computational Linguistics (ACL)*, page 976–983, 2007.

[106] Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. Efficient estimation of word representations in vector space. *Association for Computational Linguistics (ACL)*, 2013.

[107] Piero Molino. Word embedding, past, present, and future. https://w4nderlu.st/media/pages/teaching/word-embeddings/2739226721-1665242033/word-embeddings.pdf, last accessed 2023-03-31.

[108] Lampros Mouselimis. *fastText: Efficient Learning of Word Representations and Sentence Classification using R*, 2022. R package version 1.0.3.

[109] Karsten Müller, Carlo Schwarz, and Thomas Fujiwara. How twitter affected the 2016 presidential election, Oct 2020. https://cepr.org/voxeu/columns/how-twitter-affected-2016-presidential-election, last accessed 2023-03-31.

[110] Nadi. Nadi shared task 2020, 2020. https://sites.google.com/view/nadi-shared-task, last accessed 2023-03-31.

[111] Nadi. Nadi shared task 2021, 2021. https://sites.google.com/view/nadi-shared-task, last accessed 2023-03-31.

[112] Bensoula Noureddine. Electronic flies and public opinion. *Al-Naciriya: Journal of Sociological and Historical Studies*, June 2020. https://www.asjp.cerist.dz/en/article/115793, last accessed 2023-03-31.

[113] OkCoin. Okcoin cryptocurrency exchange. https://www.okcoin.com/.

[114] Sasank Pagolu, Kamal Reddy, Ganapati Panda, and Babita Majhi. Sentiment analysis of twitter data for predicting stock market movements. pages 1345–1350, 10 2016.

[115] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. Pytorch: An imperative style, high-performance deep learning library. In *Advances in Neural Information Processing Systems 32*, pages 8024–8035. Curran Associates, Inc., 2019.

[116] Matt Payne. Building a gpt-3 twitter sentiment analysis product, Oct 2022. https://www.width.ai/post/twitter-sentiment-analysis-using-gpt3.

[117] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

[118] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Tfidfvectorizer, scikit-learn, 2011. https://scikit-learn.org/stable/modules/generated/sklearn.feature_extraction.text.TfidfVectorizer.html.

[119] Jeffrey Pennington, Richard Socher, and Christopher Manning. GloVe: Global vectors for word representation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532–1543, Doha, Qatar, Oct 2014. Association for Computational Linguistics.

[120] Python. Python programming language. https://www.python.org/, last accessed 2023-03-31.

[121] Abirami R. and Vijaya M.S. *Stock Price Prediction Using Support Vector Regression*, volume 269, pages 588–597. 01 2012.

[122] R Core Team. *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing, Vienna, Austria, 2022.

[123] Alec Radford, Karthik Narasimhan, Tim Salimans, and Ilya Sutskever. Improving language understanding by generative pre-training. OpenAI, 2018.

[124] Leena Rao. Six-year-old twitter now has 140m active users sending 340m tweets per day, Mar 2012. https://techcrunch.com/2012/03/21/six-year-old-twitter-now-has-140m-active-users-sending-340m-_tweets-per-day/, last accessed 2023-03-31.

[125] Ash Read. Longer tweets are here: All you need to know about twitter's 140 character update, Jun 2020. https://buffer.com/resources/longer-tweets-coming-twitter/, last accessed 2023-03-31.

[126] Antônio H. Ribeiro, Koen Tiels, Luis Antonio Aguirre, and Thomas B. Schön. The trade-off between long-term memory and smoothness for recurrent networks. *CoRR*, abs/1906.08482, 2019.

[127] Jair Ribeiro. What is natural language processing (nlp), and why does it matter to you? *Towards Data Science*, 2021.

[128] Martin Riva. Word embeddings: Cbow vs skip-gram. https://www.baeldung.com/cs/word-embeddings-cbow-vs-skip-gram.

[129] Adam Roberts and Colin Raffel. Exploring transfer learning with t5: the text-to-text transfer transformer, September 2019. https://www.geeksforgeeks.org/tf-idf-for-bigrams-trigrams/.

[130] Fabrizio Ruggeri, Ron Kenett, and Frederick Faltin. *Encyclopedia of Statistics in Quality and Reliability*, volume 2. 10 2007.

[131] Fabrizio Ruggeri, Ron Kenett, and Frederick Faltin. Bayesian networks. In *Encyclopedia of Statistics in Quality and Reliability*, 2008.

[132] Stuart Russell and Peter Norvig. A modern, agent-oriented approach to introductory artificial intelligence. In *ACM SIGART Bulletin*, pages 24–26, 1995.

[133] Mehreen Saeed. A gentle introduction to sigmoid function. *Machine Learning Mastery*, 2021. https://machinelearningmastery.com/a-gentle-introduction-to-sigmoid-function/.

[134] Ali Safaya, Moutasem Abdullatif, and Deniz Yuret. KUISAIL at SemEval-2020 task 12: BERT-CNN for offensive speech identification in social media. In *Proceedings of the Fourteenth Workshop on Semantic Evaluation*, pages 2054–2059, Barcelona (online), Dec 2020. International Committee for Computational Linguistics.

[135] Chaitanya Sagar. Twitter sentiment analysis using R. *Dataaspirant*, 2018. http://dataaspirant.com/2018/03/22/twitter-sentiment-analysis-using-r/, last accessed 2023-03-31.

[136] Punyajoy Saha, Binny Mathew, Pawan Goyal, and Animesh Mukherjee. Hatemonitors: Language agnostic abuse detection in social media. *CoRR*, abs/1909.12642, 2019.

[137] Kashfia Sailunaz and Reda Alhajj. Emotion and sentiment analysis from twitter text. *J. Comput. Sci.*, 36, 2019.

[138] Sheel Saket. Count vectorizer vs tfidf vectorizer: Natural language processing. *LinkedIn*, Jan 2020.

[139] Mohammad Salameh, Houda Bouamor, and Nizar Habash. Fine-grained Arabic dialect identification. In *Proceedings of the 27th International Conference on Computational Linguistics*, pages 1332–1344, Santa Fe, New Mexico, USA, August 2018. Association for Computational Linguistics.

[140] Steven Salzberg. C4.5: Programs for machine learning by j. ross quinlan. In *Morgan Kaufmann Publishers*, pages 235–240, 1993.

[141] Claude Sammut and Geoffrey I. Webb. Tf–idf. *Springer US*, pages 986–987, 2010. https://doi.org/10.1007/978-0-387-30164-8_832, last accessed March 2, 2023.

[142] Jürgen Schmidhuber. Deep learning in neural networks: An overview. *Neural Networks*, 61:85–117, jan 2015. https://doi.org/10.1016%2Fj.neunet.2014.09.003.

[143] Devavrat Shah and Kang Zhang. Bayesian regression and bitcoin. *arXiv*, 2014.

[144] Claude Shannon. Prediction and entropy of printed english. 1950.

[145] Sagar Sharma. Epoch vs batch size vs iterations. *Towards Data Science*, Mar 2017. https://towardsdatascience.com/epoch-vs-iterations-vs-batch-size-4dfb9c7ce9c9, last accessed March 2, 2022.

[146] Prajwal Shreyas. Tweet analytics using nlp. *Analytics Vidhya*, Jun 2020. https://www.sciencedirect.com/science/article/pii/B9780128018675000021.

[147] Jack Slater. A timeline of twitter as elon musk completes buyout. *Metro*, Oct 2022. https://metro.co.uk/2022/10/28/who-owned-twitter-before-elon-musk-timeline-of-ownership-17655892/, last accessed 2023-03-31.

[148] Abu Bakr Soliman and Ali Abdelaal. Aravec, 2018. https://github.com/bakrianoo/aravec, last accessed 2023-03-31.

[149] Abu Bakr Soliman, Kareem Eisa, and Samhaa R. El-Beltagy. Aravec 2.0. https://github.com/bakrianoo/aravec/blob/master/AraVec%202.0/README.md.

[150] Abu Bakr Soliman, Kareem Eissa, and Samhaa R. El-Beltagy. Aravec: A set of arabic word embedding models for use in arabic nlp. *Procedia Computer Science*, 117:256–265, 2017. the 3rd International Conference on Arabic Computational Linguistics (ACLing 2017).

[151] Robyn Speer. Conceptnet Numberbatch: A new name for the best word embeddings you ca, May 2016. http://blog.conceptnet.io/posts/2016/conceptnet-numberbatch-a-new-name-for-the-best-word-embeddings-you-can-download/.

[152] Evita Stenqvist and Jacob Lönnö. Predicting bitcoin price fluctuation with twitter sentiment analysis, 2017.

[153] Bashar Talafha, Mohammad Ali, Muhy Eddin Za'ter, Haitham Seelawi, Ibraheem Tuffaha, Mostafa Samir, Wael Farhan, and Hussein T. Al-Natsheh. Multi-dialect arabic bert for country-level dialect identification, 2020.

[154] Bashar Talafha, Wael Farhan, Ahmed Altakrouri, and Hussein Al-Natsheh. Mawdoo3 AI at MADAR shared task: Arabic tweet dialect identification. In *Proceedings of the Fourth Arabic Natural Language Processing Workshop*, pages 239–243, Florence, Italy, Aug 2019. Association for Computational Linguistics.

[155] TensorFlow. Fine-tuning a bert model. https://www.tensorflow.org/tfmodels/nlp/fine_tune_bert.

[156] Nora Al Twairesh, Mawaheb Al Tuwaijri, Afnan Al Moammar, and Sarah Al Humoud. Arabic spam detection in twitter. *The 2nd Workshop on Arabic Corpora and Processing Tools 2016 Theme: Social Media*, 2016.

[157] Twitter. Twitter web site, 2023. https://www.twitter.com, last accessed 2023-03-31.

[158] Khushali Verma. Using countvectorizer to extracting features from text, July 2022. https://www.geeksforgeeks.org/using-countvectorizer-to-extracting-features-from-text/.

[159] Jesse Vig. Deconstructing Bert, part 2: Visualizing the inner workings of attention. *Medium, Towards Data Science*, Apr 2022. https://towardsdatascience.com/deconstructing-bert-part-2-visualizing-the-inner-workings-of-attention-60a16d86b5c1.

[160] Chuan-Ju Wang, Ming-Feng Tsai, Tse Liu, and Chin-Ting Chang. Financial sentiment analysis for risk prediction. 10 2013.

[161] Weka. Class stringToWordVector. https://weka.sourceforge.io/doc.dev/weka/filters/unsupervised/attribute/StringToWordVector.html, last accessed 2023-03-31.

[162] Weka. Weka 3: Machine learning software in java. https://www.cs.waikato.ac.nz/ml/weka/, last accessed 2023-03-31.

[163] Weka. WekaDeeplearning4j: Deep learning using weka. https://deeplearning.cms.waikato.ac.nz/, last accessed 2023-03-31.

[164] Patrick Wintour. Qatar given 10 days to meet 13 sweeping demands by saudi arabian. *The Guardian*. https://www.theguardian.com/world/2017/jun/23/close-al-jazeera-saudi-arabia-issues-qatar-with-13-demands-to-end-blockad, last accessed 2023-03-31.

[165] Ian Witten. More data mining with weka - simple neural networks. In *New Zealand: Department of Computer Science University of Waikato.*, 2019.

[166] Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Rémi Louf, Morgan Funtowicz, and Jamie Brew. Huggingface's Transformers: State-of-the-art Natural Language Processing. *CoRR*, 2019. http://arxiv.org/abs/1910.03771, last accessed 2023-03-31.

[167] YouTube. Youtube web site, 2023. https://www.youtube.com, last accessed 2023-03-31.

[168] Wajdi Zaghouani and Anis Charfi. Arap-tweet: A large multi-dialect Twitter corpus for gender, age and language variety identification. In *Proceedings of the Eleventh International Conference on Language Resources and Evaluation (LREC 2018)*, Miyazaki, Japan, May 2018. European Language Resources Association (ELRA).

[169] Omar F. Zaidan and Chris Callison-Burch. Arabic Dialect Identification. *Computational Linguistics*, 40(1):171–202, 03 2014.

[170] Chiyu Zhang and Muhammad Abdul-Mageed. No army, no navy: BERT semi-supervised learning of Arabic dialects. In *Proceedings of the Fourth Arabic Natural Language Processing Workshop*, pages 279–284, Florence, Italy, Aug 2019. Association for Computational Linguistics.

[171] Radim Řehůřek. Gensim: Topic modelling for humans in Python. https://radimrehurek.com/gensim/models/word2vec.html, last accessed 2023-03-31.

Appendix A

Time-Series Intervals for Bitcoin price fluctuation

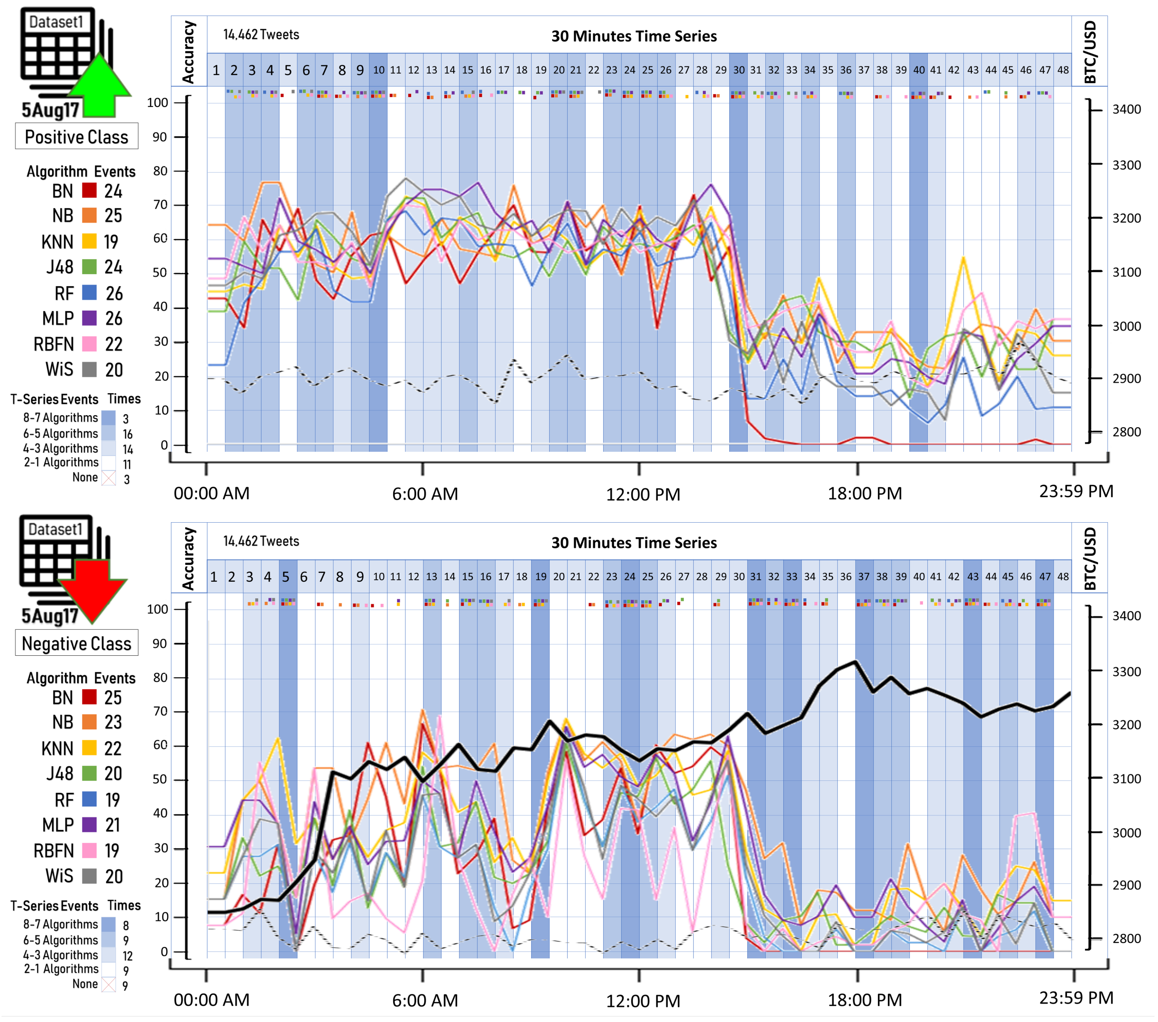


Figure A.1: Time-series with 30 minutes interval chart of the `5Aug17` dataset and one shift to the future 30 minutes and compare it with the price fluctuations.

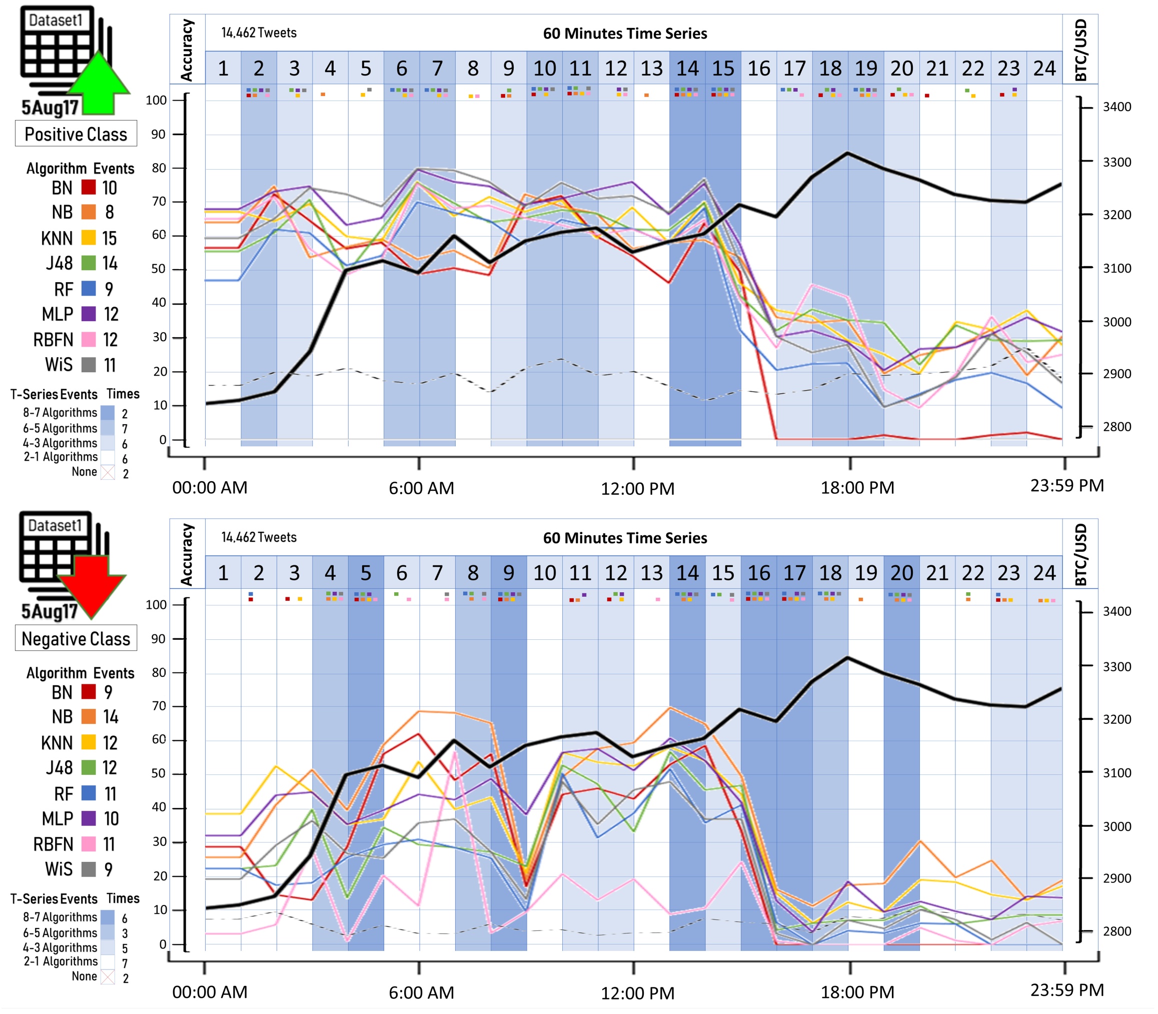


Figure A.2: Time-series with 60 minutes interval chart of the `5Aug17` dataset and one shift to the future 60 minutes and compare it with the price fluctuations.

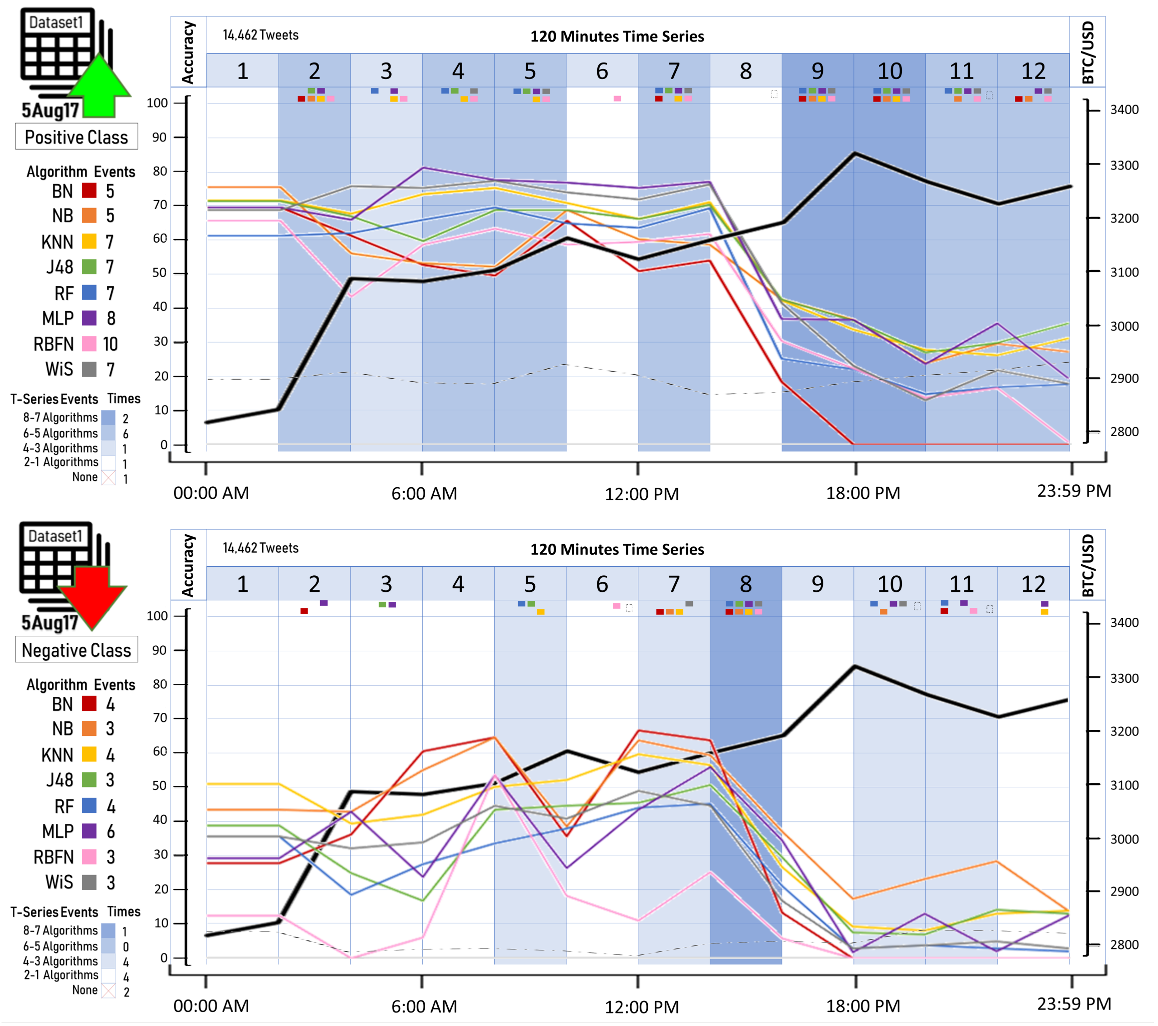
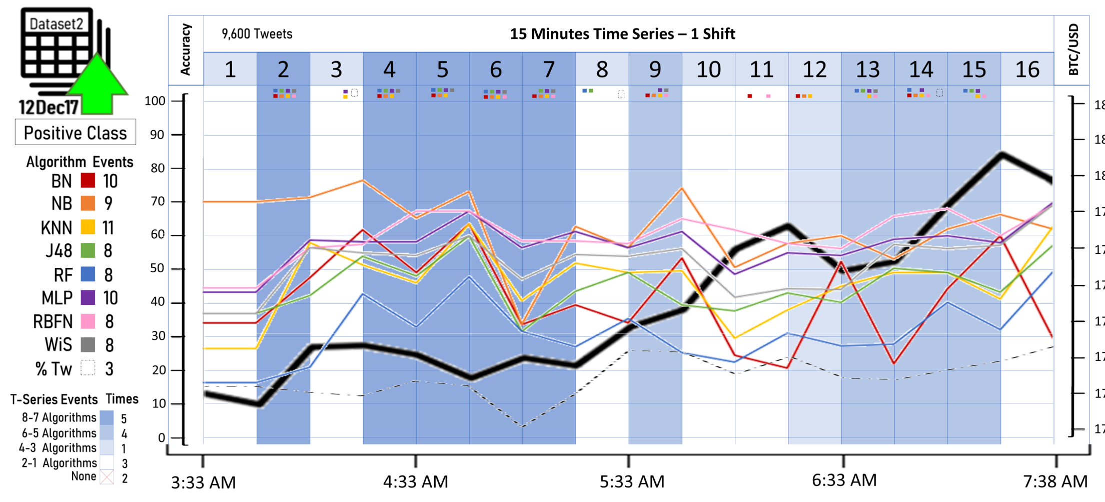


Figure A.3: Time-series with 120 minutes interval chart of the `5Aug17` dataset and one shift to the future 120 minutes and compare it with the price fluctuations.



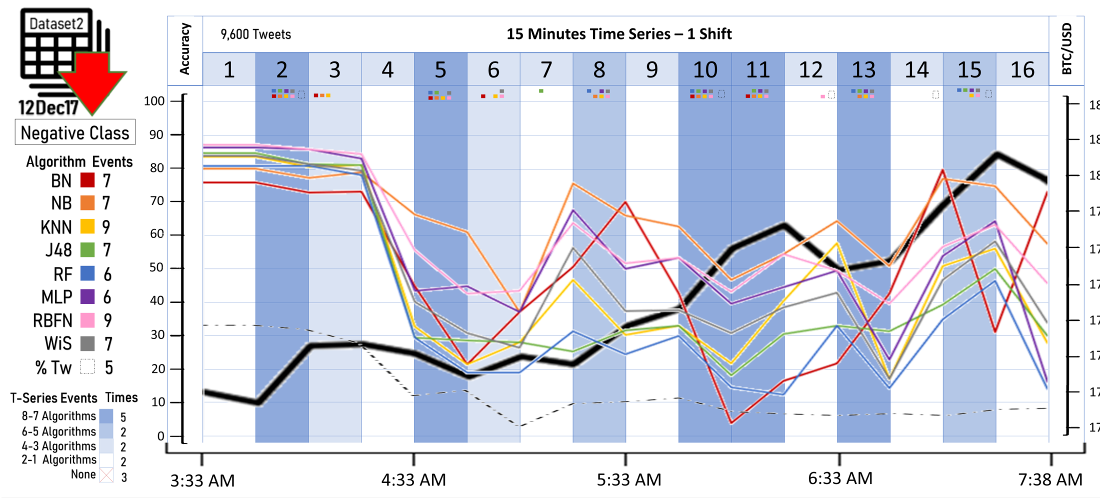
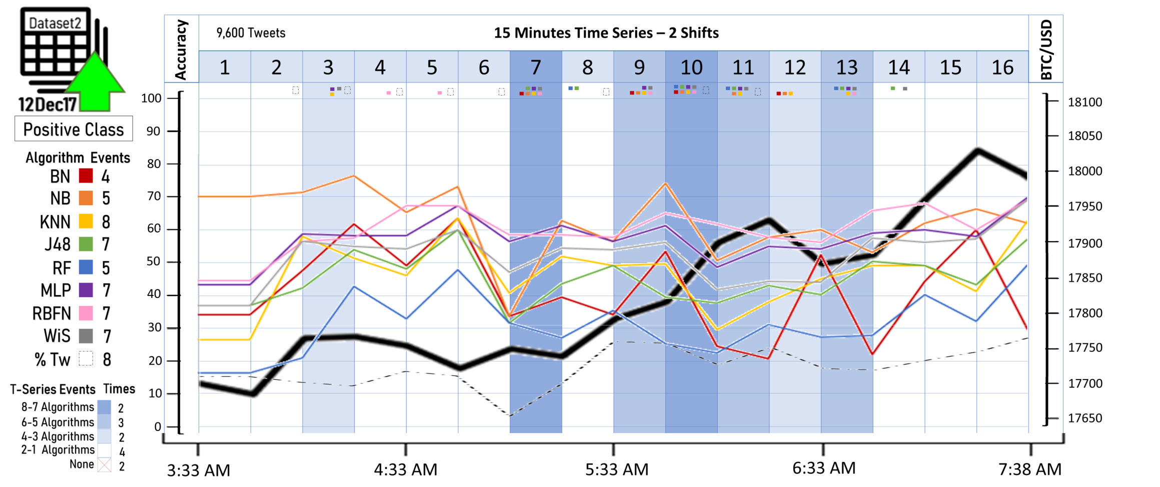


Figure A.4: Time-series with 15 minutes interval chart of the 12Dec17 dataset and one shift to the future 15 minutes with 1 shift, and compare it with the price fluctuations.



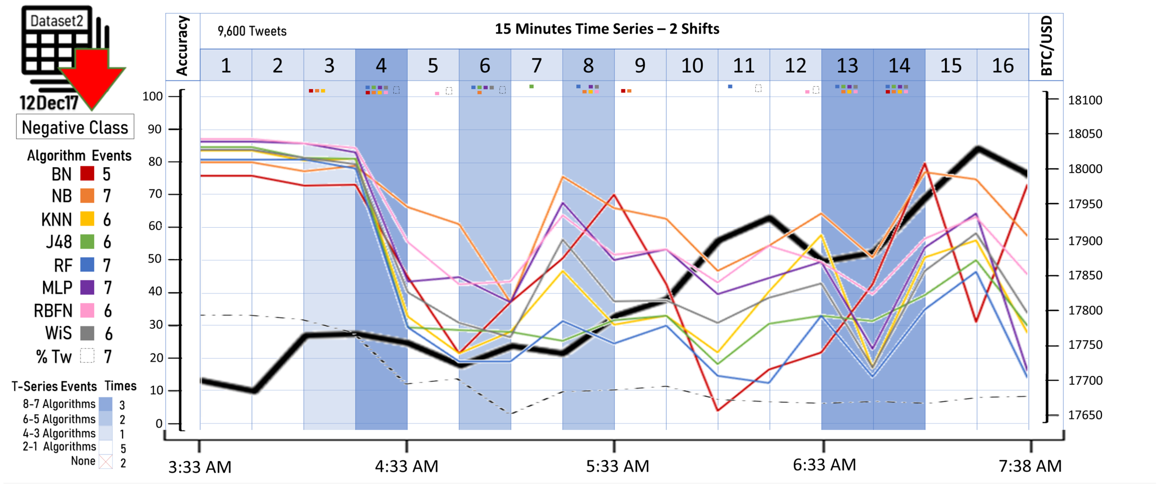
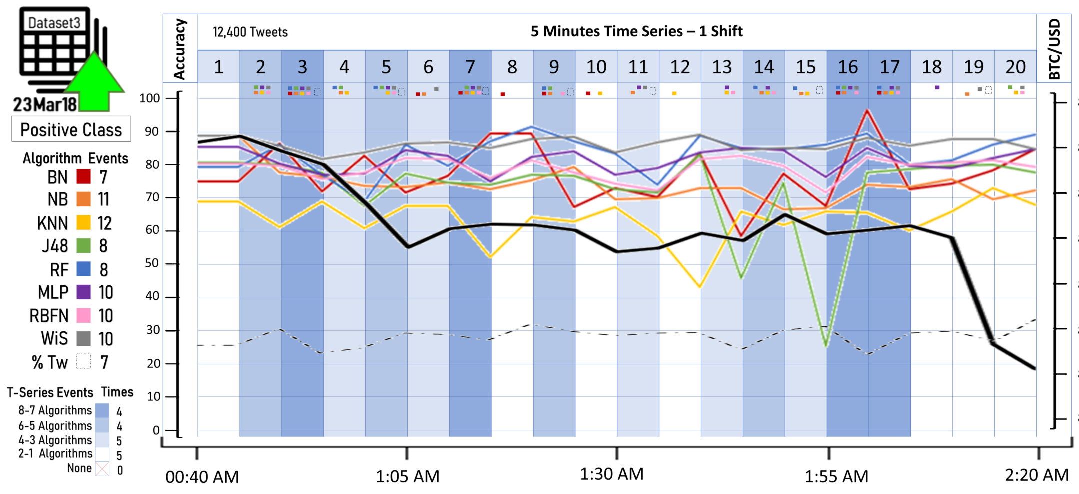


Figure A.5: Time-series with 15 minutes interval chart of the 12Dec17 dataset and one shift to the future 15 minutes with 2 shifts, and compare it with the price fluctuations.



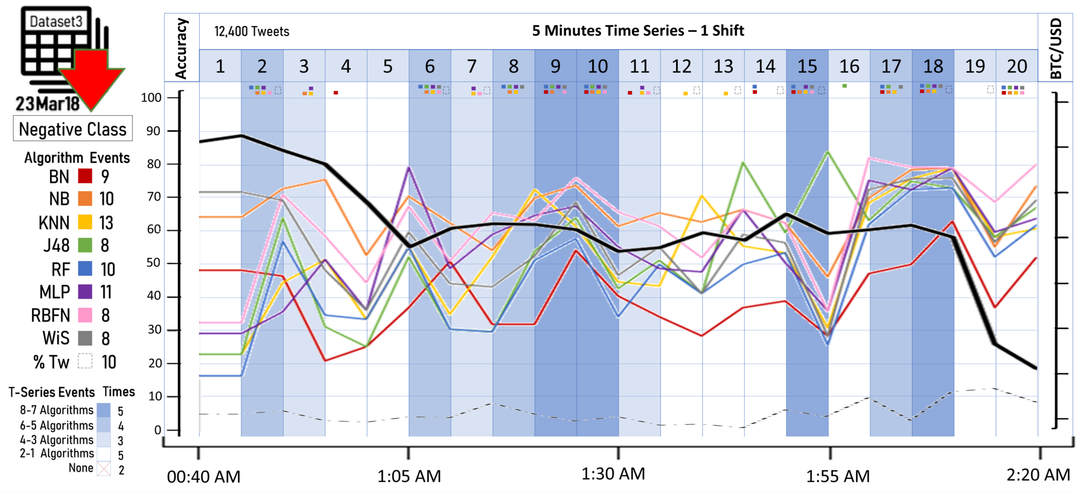


Figure A.6: Time-series with 5 minutes interval chart of the 23Mar18 dataset and one shift to the future 5 minutes with 1 shift, and compare it with the price fluctuations.

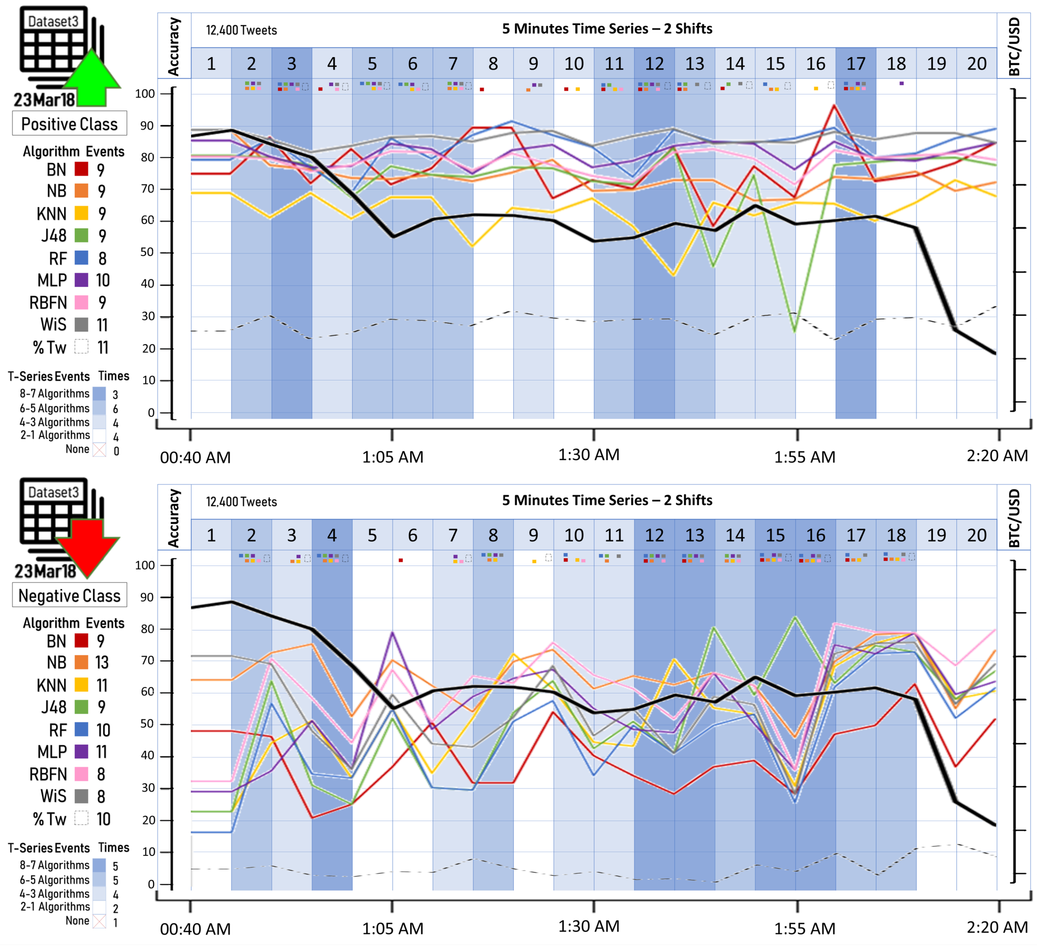


Figure A.7: Time-series with 5 minutes interval chart of the `23Mar18` dataset and one shift to the future 5 minutes with 2 shifts, and compare it with the price fluctuations.

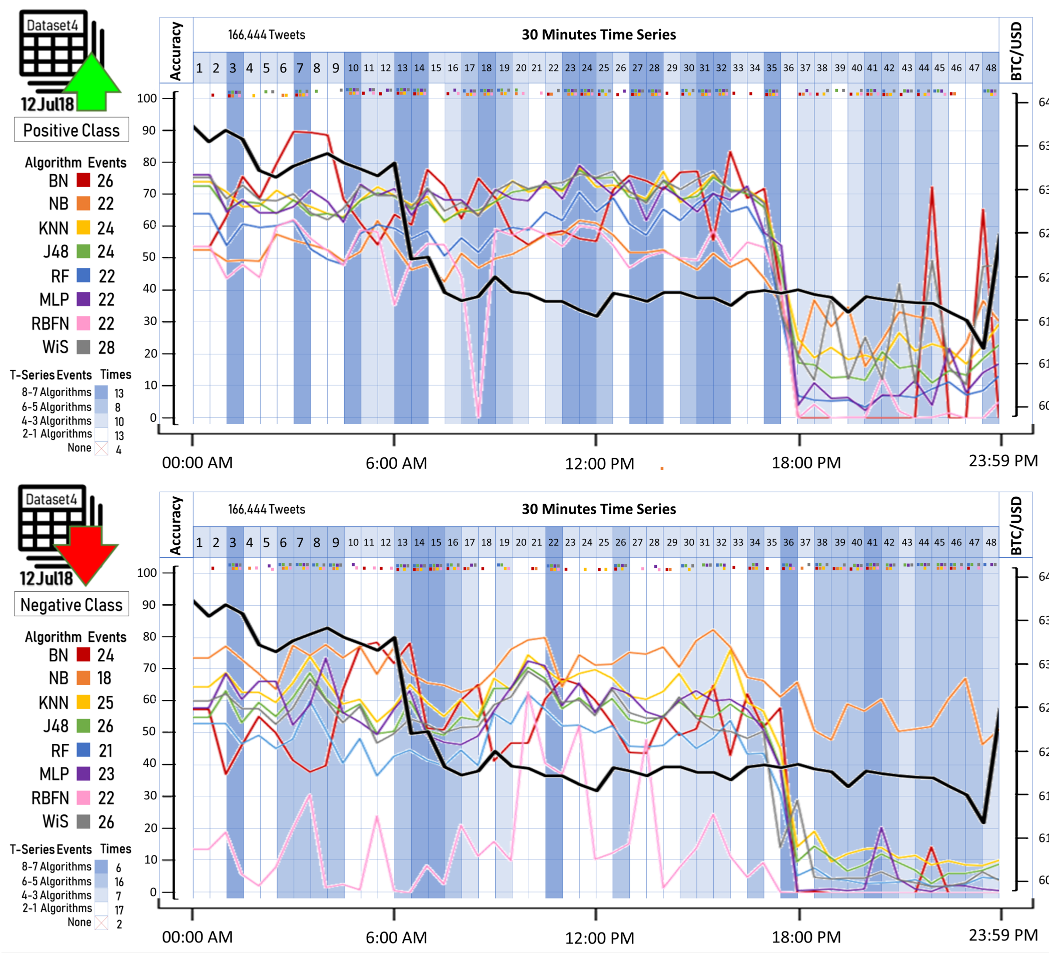
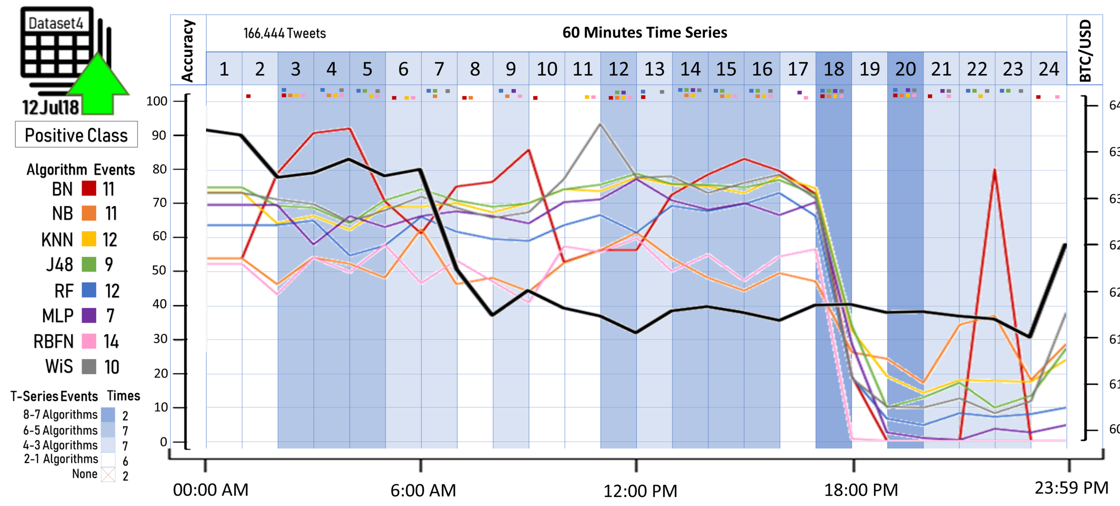


Figure A.8: Time-series with 30 minutes interval chart of the `12Jul18` dataset and one shift to the future 30 minutes and compare it with the price fluctuations.



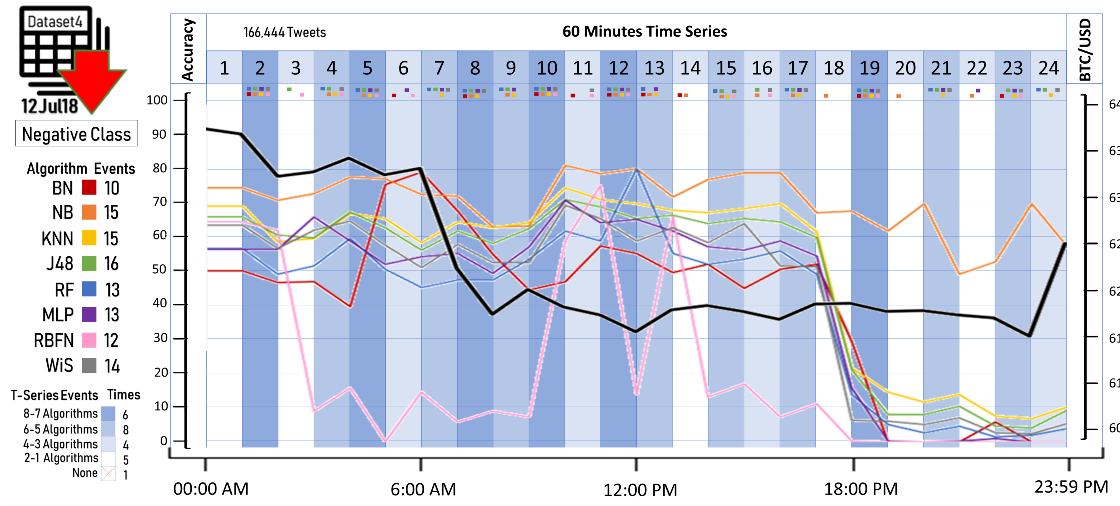


Figure A.9: Time-series with 60 minutes interval chart of the `12Jul18` dataset and one shift to the future 60 minutes and compare it with the price fluctuations.

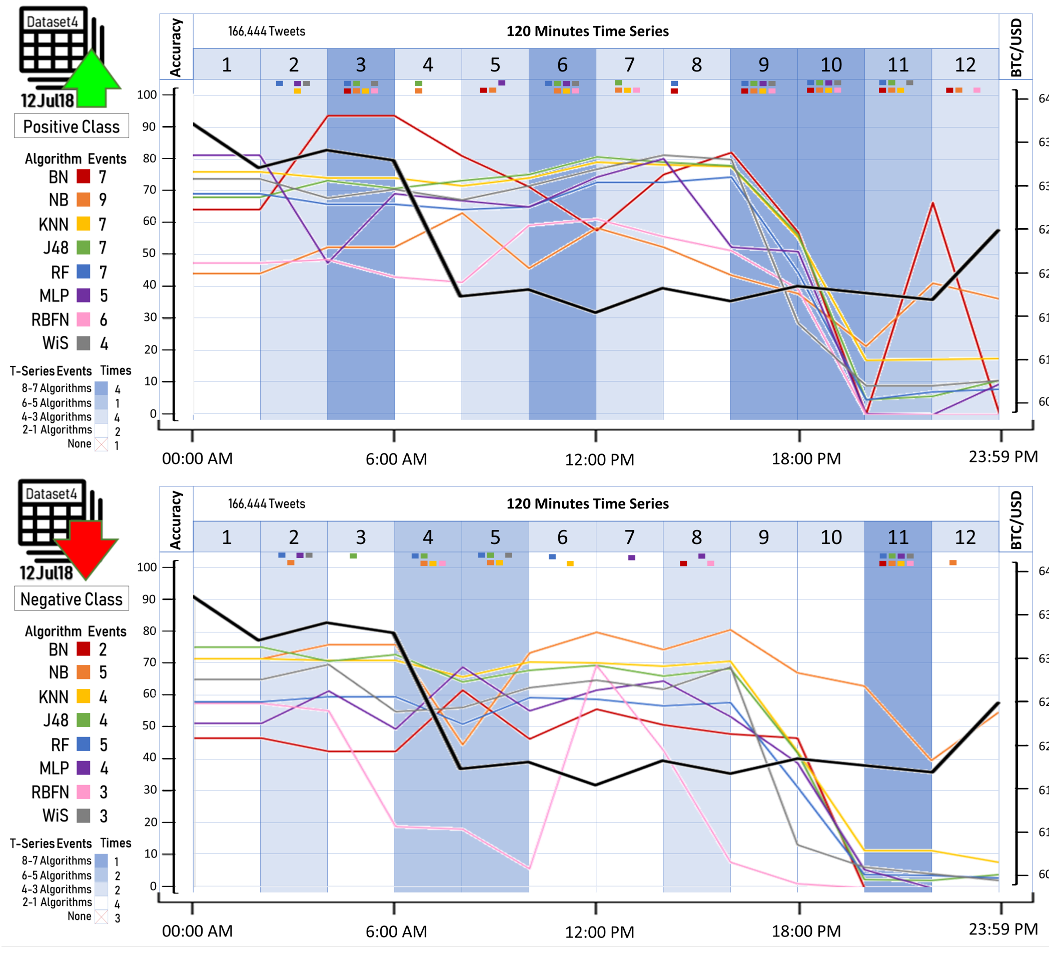


Figure A.10: Time-series with 120 minutes interval chart of the 12Jul18 dataset and one shift to the future 120 minutes and compare it with the price fluctuations.

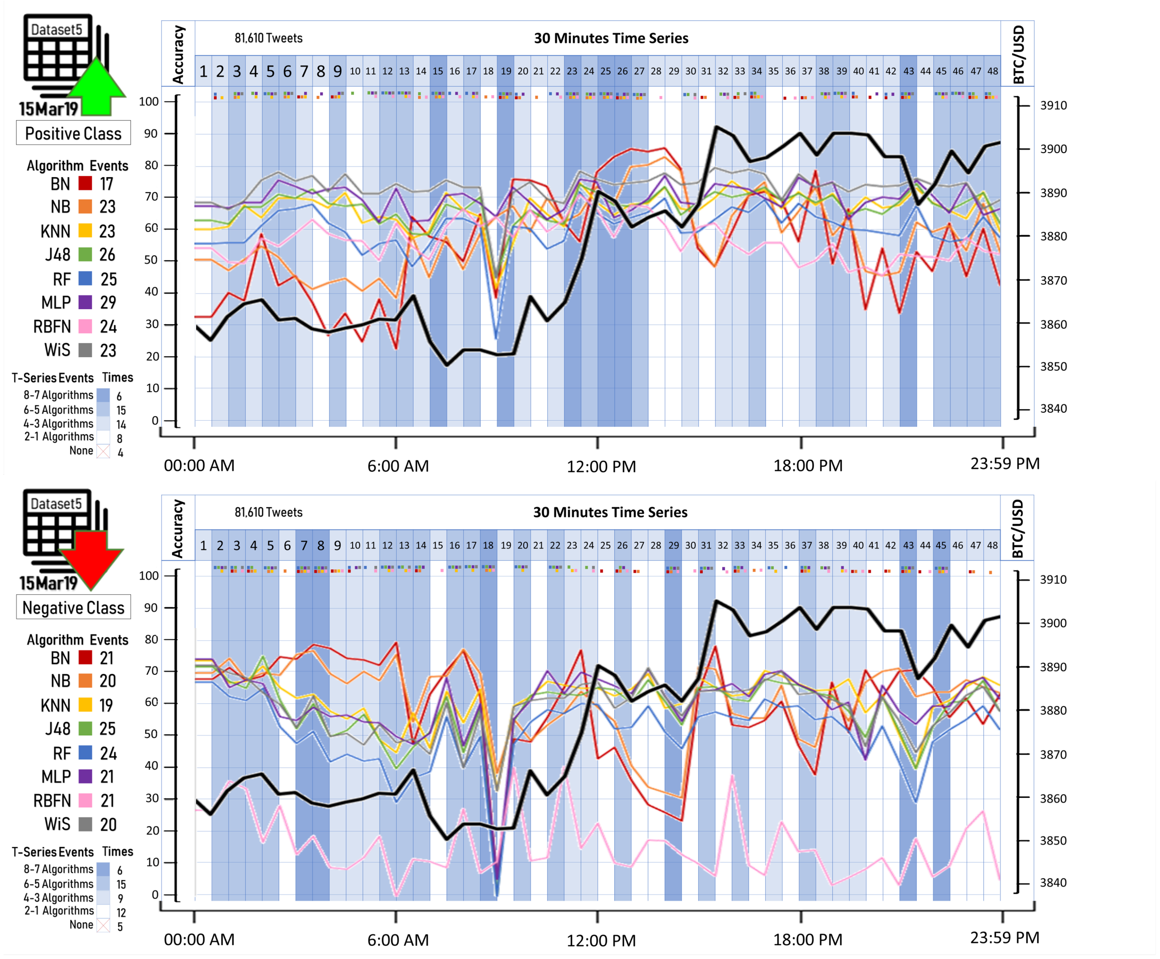


Figure A.11: Time-series with 30 minutes interval chart of the `15Mar19` dataset and one shift to the future 30 minutes and compare it with the price fluctuations.

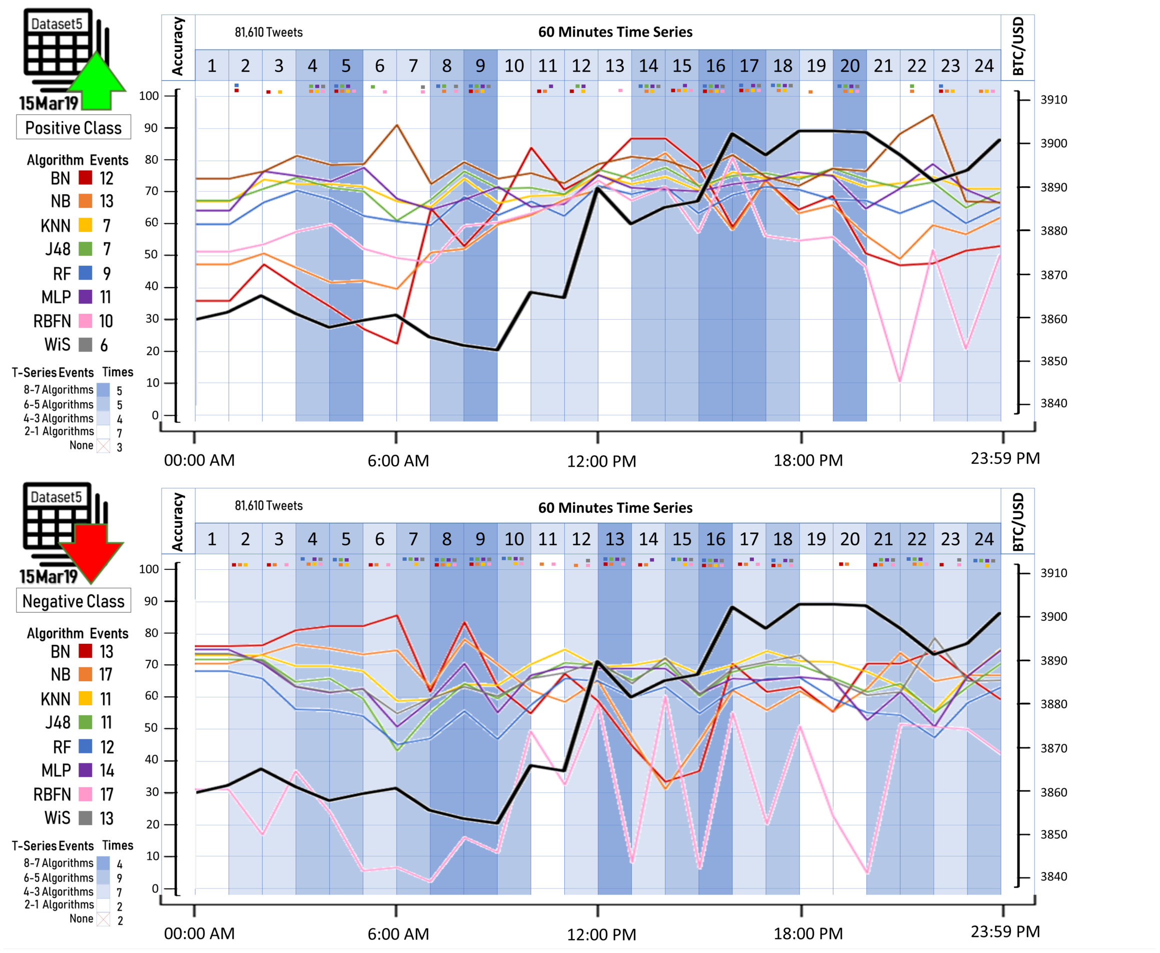


Figure A.12: Time-series with 60 minutes interval chart of the `15Mar19` dataset and one shift to the future 60 minutes and compare it with the price fluctuations.

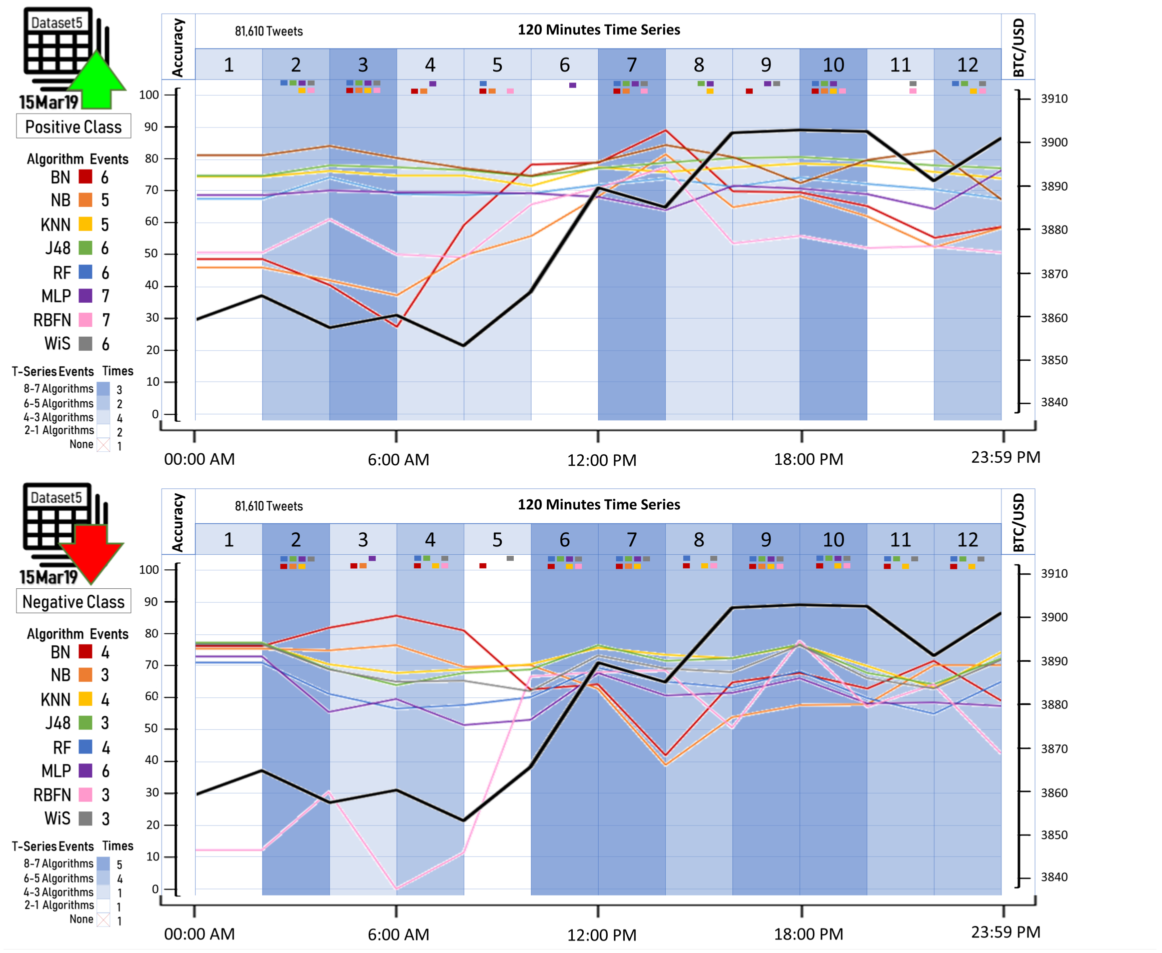
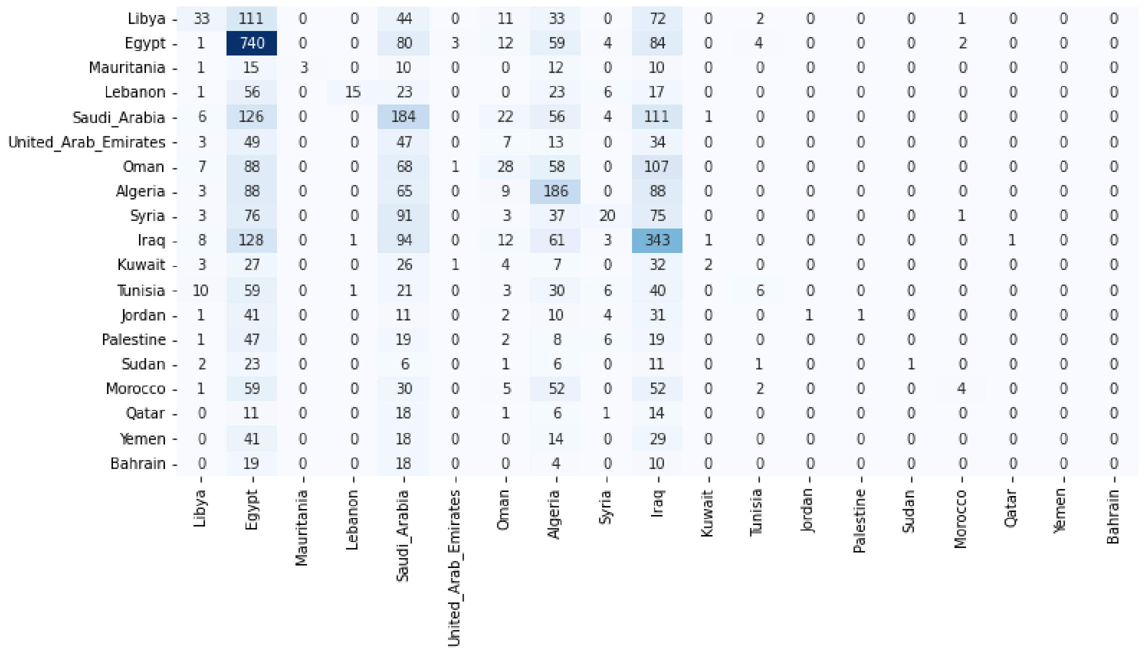


Figure A.13: Time-series with 120 minutes interval chart of the `15Mar19` dataset and one shift to the future 120 minutes and compare it with the price fluctuations.

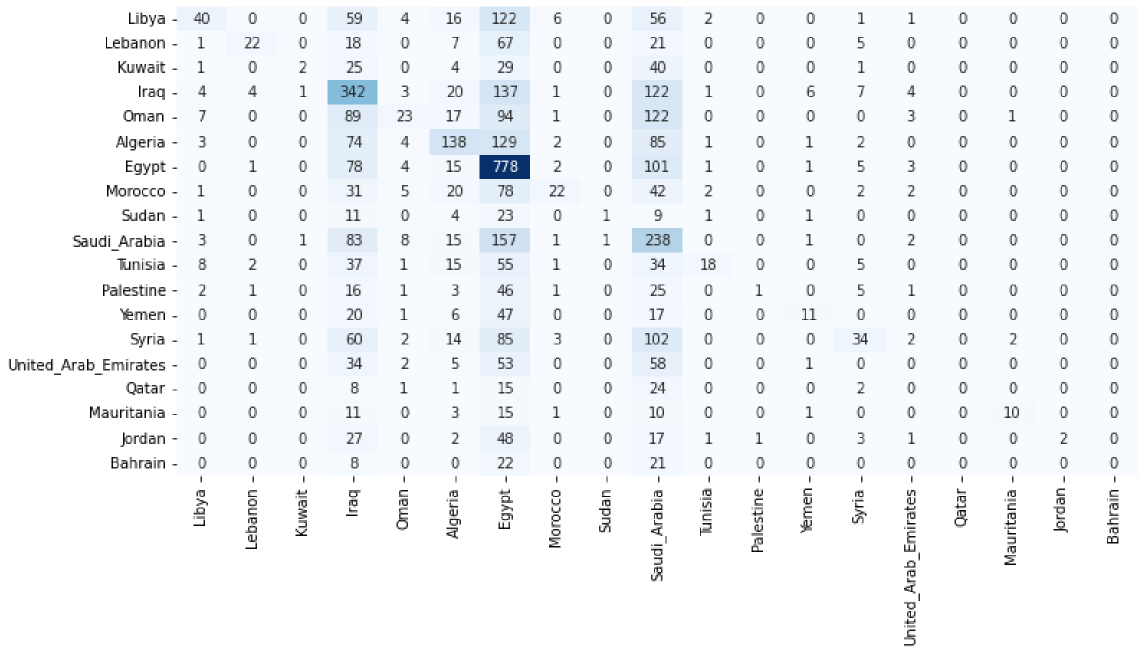
Appendix B

Additional Arabic Dialect Confusion Matrices



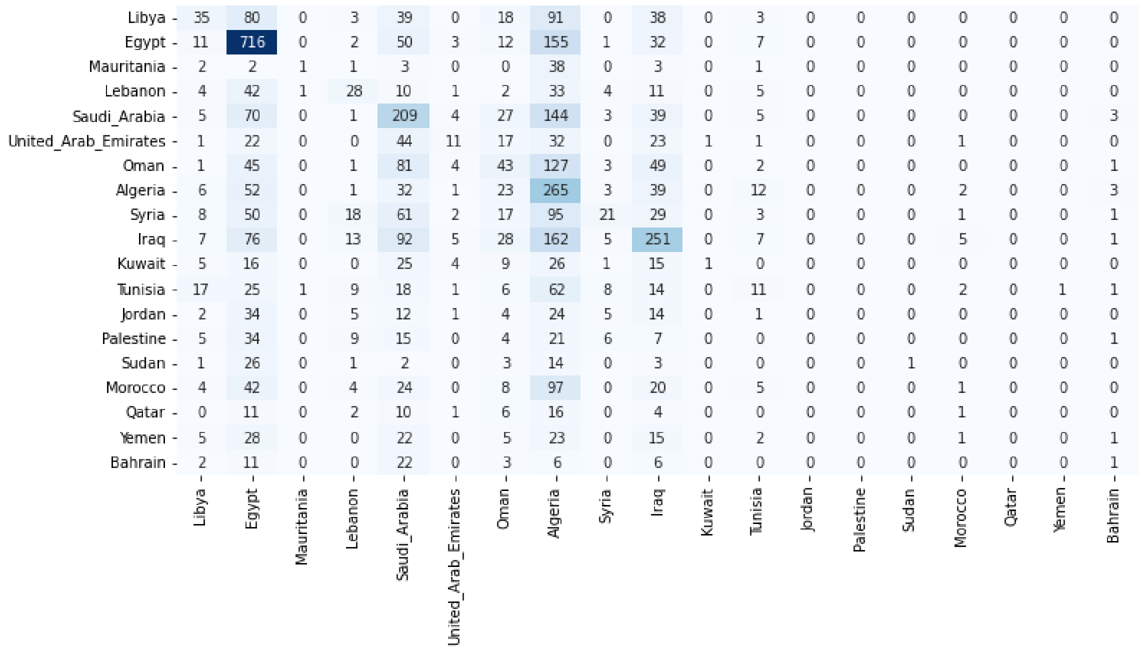
	Libya	Egypt	Mauritania	Lebanon	Saudi_Arabia	United_Arab_Emirates	Oman	Algeria	Syria	Iraq	Kuwait	Tunisia	Jordan	Palestine	Sudan	Morocco	Qatar	Yemen	Bahrain
Libya	33	111	0	0	44	0	11	33	0	72	0	2	0	0	0	1	0	0	0
Egypt	1	740	0	0	80	3	12	59	4	84	0	4	0	0	0	2	0	0	0
Mauritania	1	15	3	0	10	0	0	12	0	10	0	0	0	0	0	0	0	0	0
Lebanon	1	56	0	15	23	0	0	23	6	17	0	0	0	0	0	0	0	0	0
Saudi_Arabia	6	126	0	0	184	0	22	56	4	111	1	0	0	0	0	0	0	0	0
United_Arab_Emirates	3	49	0	0	47	0	7	13	0	34	0	0	0	0	0	0	0	0	0
Oman	7	88	0	0	68	1	28	58	0	107	0	0	0	0	0	0	0	0	0
Algeria	3	88	0	0	65	0	9	186	0	88	0	0	0	0	0	0	0	0	0
Syria	3	76	0	0	91	0	3	37	20	75	0	0	0	0	0	1	0	0	0
Iraq	8	128	0	1	94	0	12	61	3	343	1	0	0	0	0	0	1	0	0
Kuwait	3	27	0	0	26	1	4	7	0	32	2	0	0	0	0	0	0	0	0
Tunisia	10	59	0	1	21	0	3	30	6	40	0	6	0	0	0	0	0	0	0
Jordan	1	41	0	0	11	0	2	10	4	31	0	0	1	1	0	0	0	0	0
Palestine	1	47	0	0	19	0	2	8	6	19	0	0	0	0	0	0	0	0	0
Sudan	2	23	0	0	6	0	1	6	0	11	0	1	0	0	1	0	0	0	0
Morocco	1	59	0	0	30	0	5	52	0	52	0	2	0	0	0	4	0	0	0
Qatar	0	11	0	0	18	0	1	6	1	14	0	0	0	0	0	0	0	0	0
Yemen	0	41	0	0	18	0	0	14	0	29	0	0	0	0	0	0	0	0	0
Bahrain	0	19	0	0	18	0	0	4	0	10	0	0	0	0	0	0	0	0	0

Figure B.1: SVM tf-idf N-gram confusion matrix for dataset `C19`



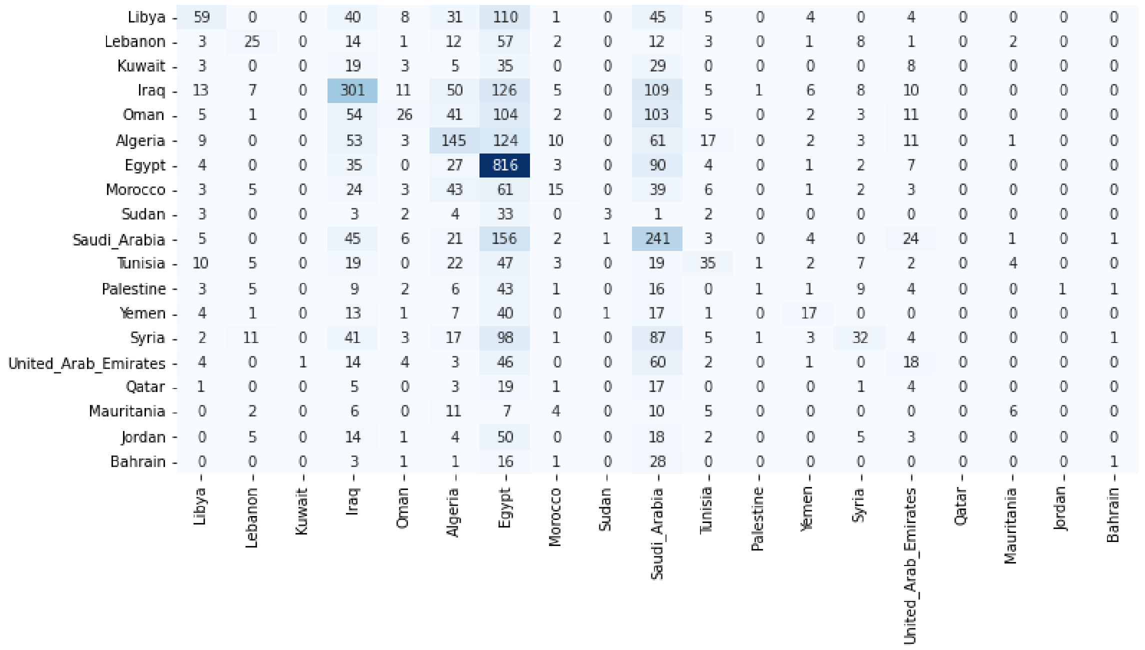
	Libya	Lebanon	Kuwait	Iraq	Oman	Algeria	Egypt	Morocco	Sudan	Saudi_Arabia	Tunisia	Palestine	Yemen	Syria	United_Arab_Emirates	Qatar	Mauritania	Jordan	Bahrain
Libya	40	0	0	59	4	16	122	6	0	56	2	0	0	1	1	0	0	0	0
Lebanon	1	22	0	18	0	7	67	0	0	21	0	0	0	5	0	0	0	0	0
Kuwait	1	0	2	25	0	4	29	0	0	40	0	0	0	1	0	0	0	0	0
Iraq	4	4	1	342	3	20	137	1	0	122	1	0	6	7	4	0	0	0	0
Oman	7	0	0	89	23	17	94	1	0	122	0	0	0	0	3	0	1	0	0
Algeria	3	0	0	74	4	138	129	2	0	85	1	0	1	2	0	0	0	0	0
Egypt	0	1	0	78	4	15	778	2	0	101	1	0	1	5	3	0	0	0	0
Morocco	1	0	0	31	5	20	78	22	0	42	2	0	0	2	2	0	0	0	0
Sudan	1	0	0	11	0	4	23	0	1	9	1	0	1	0	0	0	0	0	0
Saudi_Arabia	3	0	1	83	8	15	157	1	1	238	0	0	1	0	2	0	0	0	0
Tunisia	8	2	0	37	1	15	55	1	0	34	18	0	0	5	0	0	0	0	0
Palestine	2	1	0	16	1	3	46	1	0	25	0	1	0	5	1	0	0	0	0
Yemen	0	0	0	20	1	6	47	0	0	17	0	0	11	0	0	0	0	0	0
Syria	1	1	0	60	2	14	85	3	0	102	0	0	0	34	2	0	2	0	0
United_Arab_Emirates	0	0	0	34	2	5	53	0	0	58	0	0	1	0	0	0	0	0	0
Qatar	0	0	0	8	1	1	15	0	0	24	0	0	0	2	0	0	0	0	0
Mauritania	0	0	0	11	0	3	15	1	0	10	0	0	1	0	0	0	10	0	0
Jordan	0	0	0	27	0	2	48	0	0	17	1	1	0	3	1	0	0	2	0
Bahrain	0	0	0	8	0	0	22	0	0	21	0	0	0	0	0	0	0	0	0

Figure B.2: SVM tf-idf N-gram confusion matrix for dataset `C19P`



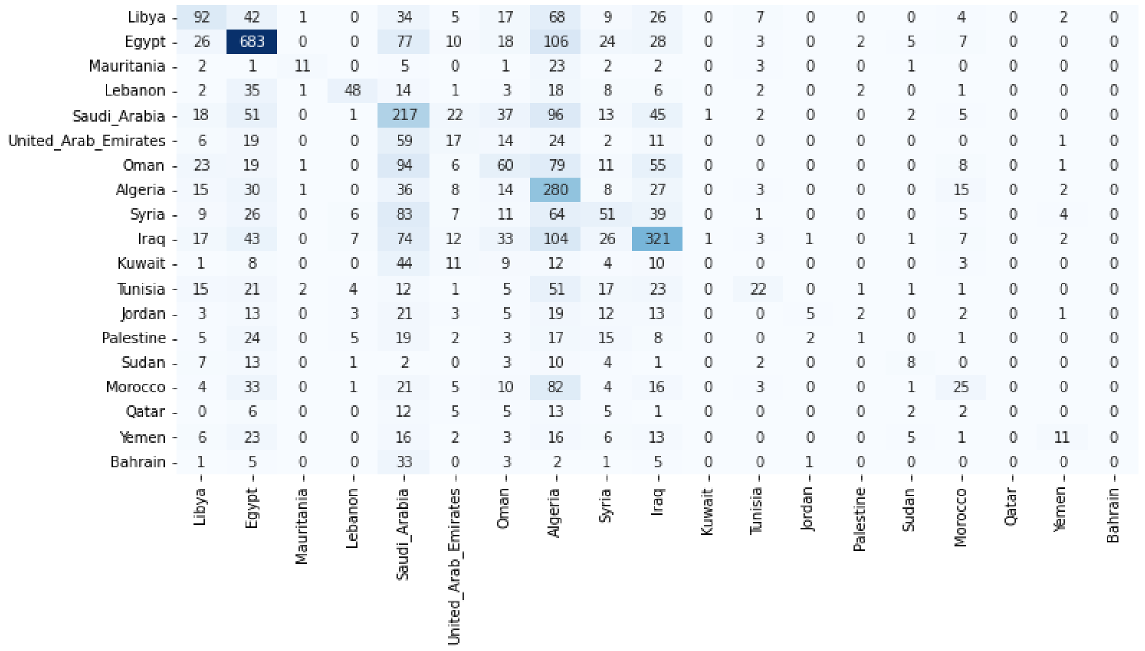
	Libya	Egypt	Mauritania	Lebanon	Saudi_Arabia	United_Arab_Emirates	Oman	Algeria	Syria	Iraq	Kuwait	Tunisia	Jordan	Palestine	Sudan	Morocco	Qatar	Yemen	Bahrain
Libya	35	80	0	3	39	0	18	91	0	38	0	3	0	0	0	0	0	0	0
Egypt	11	716	0	2	50	3	12	155	1	32	0	7	0	0	0	0	0	0	0
Mauritania	2	2	1	1	3	0	0	38	0	3	0	1	0	0	0	0	0	0	0
Lebanon	4	42	1	28	10	1	2	33	4	11	0	5	0	0	0	0	0	0	0
Saudi_Arabia	5	70	0	1	209	4	27	144	3	39	0	5	0	0	0	0	0	0	3
United_Arab_Emirates	1	22	0	0	44	11	17	32	0	23	1	1	0	0	0	1	0	0	0
Oman	1	45	0	1	81	4	43	127	3	49	0	2	0	0	0	0	0	0	1
Algeria	6	52	0	1	32	1	23	265	3	39	0	12	0	0	0	2	0	0	3
Syria	8	50	0	18	61	2	17	95	21	29	0	3	0	0	0	1	0	0	1
Iraq	7	76	0	13	92	5	28	162	5	251	0	7	0	0	0	5	0	0	1
Kuwait	5	16	0	0	25	4	9	26	1	15	1	0	0	0	0	0	0	0	0
Tunisia	17	25	1	9	18	1	6	62	8	14	0	11	0	0	0	2	0	1	1
Jordan	2	34	0	5	12	1	4	24	5	14	0	1	0	0	0	0	0	0	0
Palestine	5	34	0	9	15	0	4	21	6	7	0	0	0	0	0	0	0	0	1
Sudan	1	26	0	1	2	0	3	14	0	3	0	0	0	0	1	0	0	0	0
Morocco	4	42	0	4	24	0	8	97	0	20	0	5	0	0	0	1	0	0	0
Qatar	0	11	0	2	10	1	6	16	0	4	0	0	0	0	0	1	0	0	0
Yemen	5	28	0	0	22	0	5	23	0	15	0	2	0	0	0	1	0	0	1
Bahrain	2	11	0	0	22	0	3	6	0	6	0	0	0	0	0	0	0	0	1

Figure B.3: AraVec confusion matrix for dataset `C19`



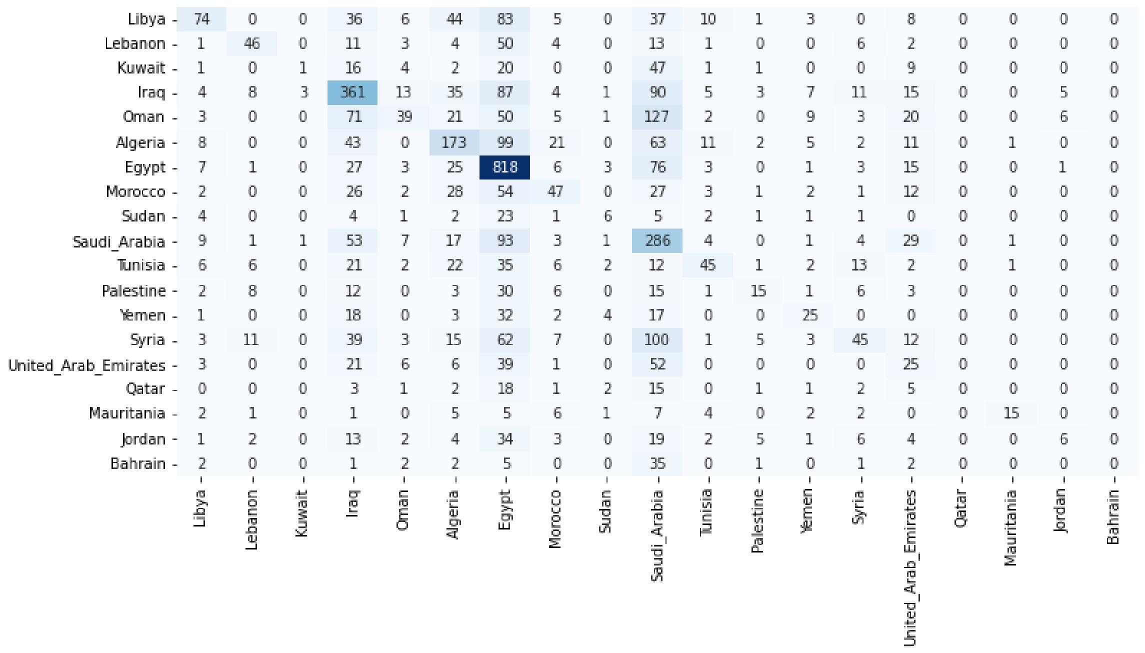
	Libya	Lebanon	Kuwait	Iraq	Oman	Algeria	Egypt	Morocco	Sudan	Saudi_Arabia	Tunisia	Palestine	Yemen	Syria	United_Arab_Emirates	Qatar	Mauritania	Jordan	Bahrain
Libya	59	0	0	40	8	31	110	1	0	45	5	0	4	0	4	0	0	0	0
Lebanon	3	25	0	14	1	12	57	2	0	12	3	0	1	8	1	0	2	0	0
Kuwait	3	0	0	19	3	5	35	0	0	29	0	0	0	0	8	0	0	0	0
Iraq	13	7	0	301	11	50	126	5	0	109	5	1	6	8	10	0	0	0	0
Oman	5	1	0	54	26	41	104	2	0	103	5	0	2	3	11	0	0	0	0
Algeria	9	0	0	53	3	145	124	10	0	61	17	0	2	3	11	0	1	0	0
Egypt	4	0	0	35	0	27	816	3	0	90	4	0	1	2	7	0	0	0	0
Morocco	3	5	0	24	3	43	61	15	0	39	6	0	1	2	3	0	0	0	0
Sudan	3	0	0	3	2	4	33	0	3	1	2	0	0	0	0	0	0	0	0
Saudi_Arabia	5	0	0	45	6	21	156	2	1	241	3	0	4	0	24	0	1	0	1
Tunisia	10	5	0	19	0	22	47	3	0	19	35	1	2	7	2	0	4	0	0
Palestine	3	5	0	9	2	6	43	1	0	16	0	1	1	9	4	0	0	1	1
Yemen	4	1	0	13	1	7	40	0	1	17	1	0	17	0	0	0	0	0	0
Syria	2	11	0	41	3	17	98	1	0	87	5	1	3	32	4	0	0	0	1
United_Arab_Emirates	4	0	1	14	4	3	46	0	0	60	2	0	1	0	18	0	0	0	0
Qatar	1	0	0	5	0	3	19	1	0	17	0	0	0	1	4	0	0	0	0
Mauritania	0	2	0	6	0	11	7	4	0	10	5	0	0	0	0	0	6	0	0
Jordan	0	5	0	14	1	4	50	0	0	18	2	0	0	5	3	0	0	0	0
Bahrain	0	0	0	3	1	1	16	1	0	28	0	0	0	0	0	0	0	0	1

Figure B.4: AraVec confusion matrix for dataset `C19P`



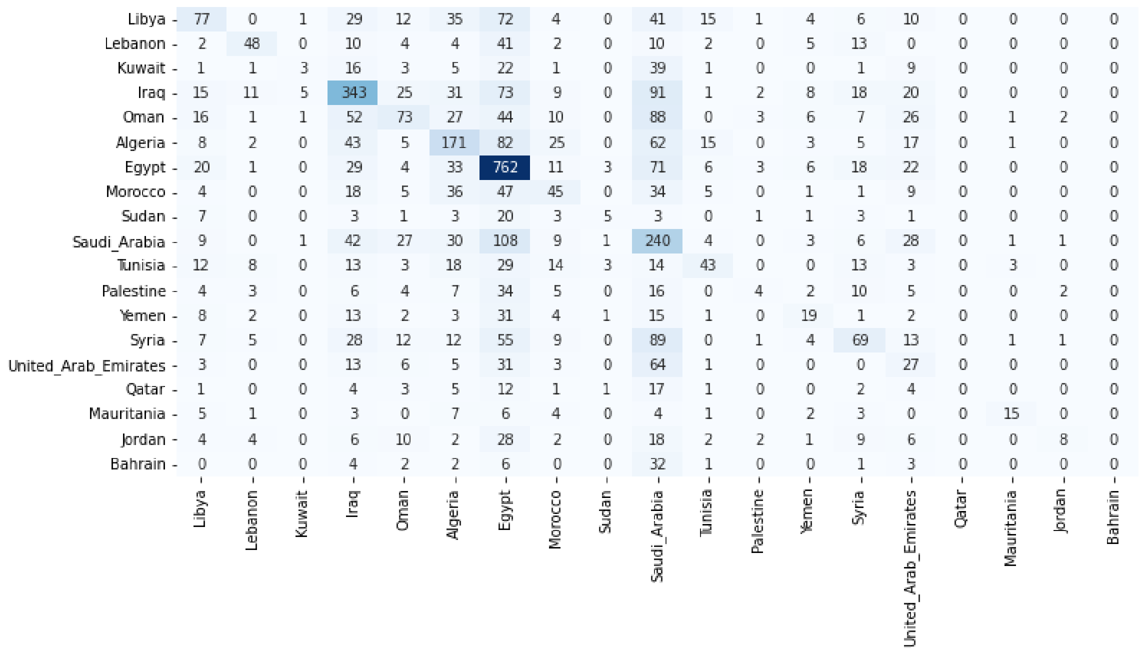
	Libya	Egypt	Mauritania	Lebanon	Saudi_Arabia	United_Arab_Emirates	Oman	Algeria	Syria	Iraq	Kuwait	Tunisia	Jordan	Palestine	Sudan	Morocco	Qatar	Yemen	Bahrain
Libya	92	42	1	0	34	5	17	68	9	26	0	7	0	0	0	4	0	2	0
Egypt	26	683	0	0	77	10	18	106	24	28	0	3	0	2	5	7	0	0	0
Mauritania	2	1	11	0	5	0	1	23	2	2	0	3	0	0	1	0	0	0	0
Lebanon	2	35	1	48	14	1	3	18	8	6	0	2	0	2	0	1	0	0	0
Saudi_Arabia	18	51	0	1	217	22	37	96	13	45	1	2	0	0	2	5	0	0	0
United_Arab_Emirates	6	19	0	0	59	17	14	24	2	11	0	0	0	0	0	0	0	1	0
Oman	23	19	1	0	94	6	60	79	11	55	0	0	0	0	0	8	0	1	0
Algeria	15	30	1	0	36	8	14	280	8	27	0	3	0	0	0	15	0	2	0
Syria	9	26	0	6	83	7	11	64	51	39	0	1	0	0	0	5	0	4	0
Iraq	17	43	0	7	74	12	33	104	26	321	1	3	1	0	1	7	0	2	0
Kuwait	1	8	0	0	44	11	9	12	4	10	0	0	0	0	0	3	0	0	0
Tunisia	15	21	2	4	12	1	5	51	17	23	0	22	0	1	1	1	0	0	0
Jordan	3	13	0	3	21	3	5	19	12	13	0	0	5	2	0	2	0	1	0
Palestine	5	24	0	5	19	2	3	17	15	8	0	0	2	1	0	1	0	0	0
Sudan	7	13	0	1	2	0	3	10	4	1	0	2	0	0	8	0	0	0	0
Morocco	4	33	0	1	21	5	10	82	4	16	0	3	0	0	1	25	0	0	0
Qatar	0	6	0	0	12	5	5	13	5	1	0	0	0	0	2	2	0	0	0
Yemen	6	23	0	0	16	2	3	16	6	13	0	0	0	0	5	1	0	11	0
Bahrain	1	5	0	0	33	0	3	2	1	5	0	0	1	0	0	0	0	0	0

Figure B.5: The pre-trained model asafaya/bert-base-arabic (ASAFAYA) confusion matrix for dataset `C19`



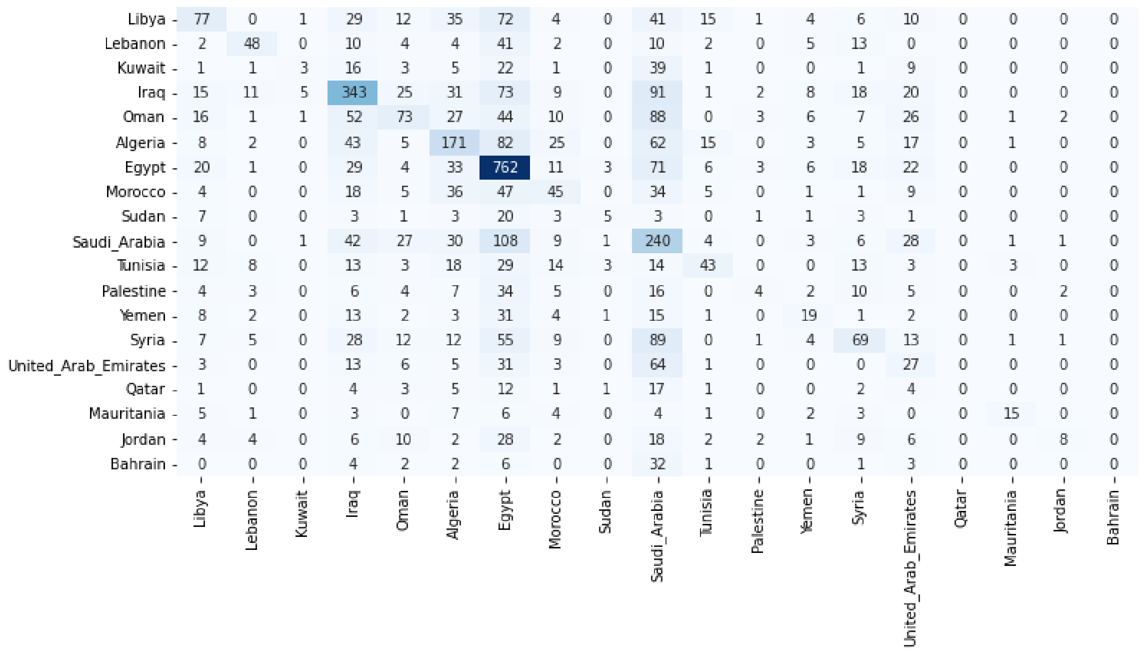
	Libya	Lebanon	Kuwait	Iraq	Oman	Algeria	Egypt	Morocco	Sudan	Saudi_Arabia	Tunisia	Palestine	Yemen	Syria	United_Arab_Emirates	Qatar	Mauritania	Jordan	Bahrain
Libya	74	0	0	36	6	44	83	5	0	37	10	1	3	0	8	0	0	0	0
Lebanon	1	46	0	11	3	4	50	4	0	13	1	0	0	6	2	0	0	0	0
Kuwait	1	0	1	16	4	2	20	0	0	47	1	1	0	0	9	0	0	0	0
Iraq	4	8	3	361	13	35	87	4	1	90	5	3	7	11	15	0	0	5	0
Oman	3	0	0	71	39	21	50	5	1	127	2	0	9	3	20	0	0	6	0
Algeria	8	0	0	43	0	173	99	21	0	63	11	2	5	2	11	0	1	0	0
Egypt	7	1	0	27	3	25	818	6	3	76	3	0	1	3	15	0	0	1	0
Morocco	2	0	0	26	2	28	54	47	0	27	3	1	2	1	12	0	0	0	0
Sudan	4	0	0	4	1	2	23	1	6	5	2	1	1	1	0	0	0	0	0
Saudi_Arabia	9	1	1	53	7	17	93	3	1	286	4	0	1	4	29	0	1	0	0
Tunisia	6	6	0	21	2	22	35	6	2	12	45	1	2	13	2	0	1	0	0
Palestine	2	8	0	12	0	3	30	6	0	15	1	15	1	6	3	0	0	0	0
Yemen	1	0	0	18	0	3	32	2	4	17	0	0	25	0	0	0	0	0	0
Syria	3	11	0	39	3	15	62	7	0	100	1	5	3	45	12	0	0	0	0
United_Arab_Emirates	3	0	0	21	6	6	39	1	0	52	0	0	0	0	25	0	0	0	0
Qatar	0	0	0	3	1	2	18	1	2	15	0	1	1	2	5	0	0	0	0
Mauritania	2	1	0	1	0	5	5	6	1	7	4	0	2	2	0	0	15	0	0
Jordan	1	2	0	13	2	4	34	3	0	19	2	5	1	6	4	0	0	6	0
Bahrain	2	0	0	1	2	2	5	0	0	35	0	1	0	1	2	0	0	0	0

Figure B.6: The pre-trained model asafaya/bert-base-arabic (ASAFAYA) confusion matrix for dataset `C19P`



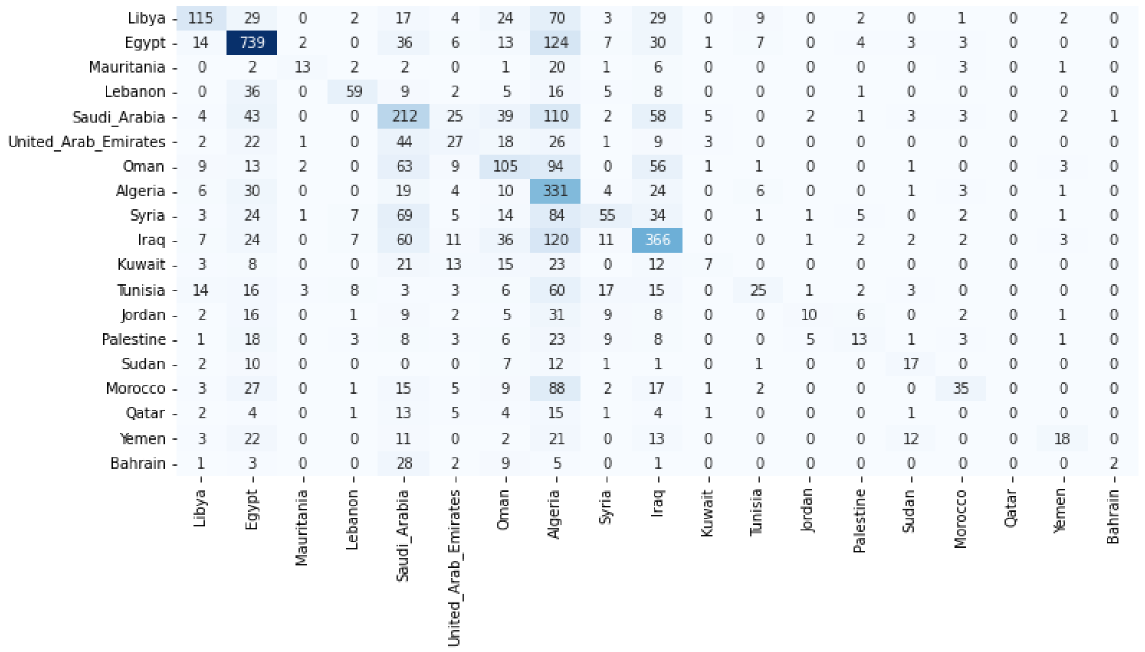
	Libya	Lebanon	Kuwait	Iraq	Oman	Algeria	Egypt	Morocco	Sudan	Saudi_Arabia	Tunisia	Palestine	Yemen	Syria	United_Arab_Emirates	Qatar	Mauritania	Jordan	Bahrain
Libya	77	0	1	29	12	35	72	4	0	41	15	1	4	6	10	0	0	0	0
Lebanon	2	48	0	10	4	4	41	2	0	10	2	0	5	13	0	0	0	0	0
Kuwait	1	1	3	16	3	5	22	1	0	39	1	0	0	1	9	0	0	0	0
Iraq	15	11	5	343	25	31	73	9	0	91	1	2	8	18	20	0	0	0	0
Oman	16	1	1	52	73	27	44	10	0	88	0	3	6	7	26	0	1	2	0
Algeria	8	2	0	43	5	171	82	25	0	62	15	0	3	5	17	0	1	0	0
Egypt	20	1	0	29	4	33	762	11	3	71	6	3	6	18	22	0	0	0	0
Morocco	4	0	0	18	5	36	47	45	0	34	5	0	1	1	9	0	0	0	0
Sudan	7	0	0	3	1	3	20	3	5	3	0	1	1	3	1	0	0	0	0
Saudi_Arabia	9	0	1	42	27	30	108	9	1	240	4	0	3	6	28	0	1	1	0
Tunisia	12	8	0	13	3	18	29	14	3	14	43	0	0	13	3	0	3	0	0
Palestine	4	3	0	6	4	7	34	5	0	16	0	4	2	10	5	0	0	2	0
Yemen	8	2	0	13	2	3	31	4	1	15	1	0	19	1	2	0	0	0	0
Syria	7	5	0	28	12	12	55	9	0	89	0	1	4	69	13	0	1	1	0
United_Arab_Emirates	3	0	0	13	6	5	31	3	0	64	1	0	0	0	27	0	0	0	0
Qatar	1	0	0	4	3	5	12	1	1	17	1	0	0	2	4	0	0	0	0
Mauritania	5	1	0	3	0	7	6	4	0	4	1	0	2	3	0	0	15	0	0
Jordan	4	4	0	6	10	2	28	2	0	18	2	2	1	9	6	0	0	8	0
Bahrain	0	0	0	4	2	2	6	0	0	32	1	0	0	1	3	0	0	0	0

Figure B.7: The pre-trained model aubmindlab/bert-base-arabert (ARABERT) confusion matrix for dataset `C19P`



	Libya	Lebanon	Kuwait	Iraq	Oman	Algeria	Egypt	Morocco	Sudan	Saudi_Arabia	Tunisia	Palestine	Yemen	Syria	United_Arab_Emirates	Qatar	Mauritania	Jordan	Bahrain
Libya	77	0	1	29	12	35	72	4	0	41	15	1	4	6	10	0	0	0	0
Lebanon	2	48	0	10	4	4	41	2	0	10	2	0	5	13	0	0	0	0	0
Kuwait	1	1	3	16	3	5	22	1	0	39	1	0	0	1	9	0	0	0	0
Iraq	15	11	5	343	25	31	73	9	0	91	1	2	8	18	20	0	0	0	0
Oman	16	1	1	52	73	27	44	10	0	88	0	3	6	7	26	0	1	2	0
Algeria	8	2	0	43	5	171	82	25	0	62	15	0	3	5	17	0	1	0	0
Egypt	20	1	0	29	4	33	762	11	3	71	6	3	6	18	22	0	0	0	0
Morocco	4	0	0	18	5	36	47	45	0	34	5	0	1	1	9	0	0	0	0
Sudan	7	0	0	3	1	3	20	3	5	3	0	1	1	3	1	0	0	0	0
Saudi_Arabia	9	0	1	42	27	30	108	9	1	240	4	0	3	6	28	0	1	1	0
Tunisia	12	8	0	13	3	18	29	14	3	14	43	0	0	13	3	0	3	0	0
Palestine	4	3	0	6	4	7	34	5	0	16	0	4	2	10	5	0	0	2	0
Yemen	8	2	0	13	2	3	31	4	1	15	1	0	19	1	2	0	0	0	0
Syria	7	5	0	28	12	12	55	9	0	89	0	1	4	69	13	0	1	1	0
United_Arab_Emirates	3	0	0	13	6	5	31	3	0	64	1	0	0	0	27	0	0	0	0
Qatar	1	0	0	4	3	5	12	1	1	17	1	0	0	2	4	0	0	0	0
Mauritania	5	1	0	3	0	7	6	4	0	4	1	0	2	3	0	0	15	0	0
Jordan	4	4	0	6	10	2	28	2	0	18	2	2	1	9	6	0	0	8	0
Bahrain	0	0	0	4	2	2	6	0	0	32	1	0	0	1	3	0	0	0	0

Figure B.8: The pre-trained model aubmindlab/bert-base-arabert (ARABERT) confusion matrix for dataset `C19P`



	Libya	Egypt	Mauritania	Lebanon	Saudi_Arabia	United_Arab_Emirates	Oman	Algeria	Syria	Iraq	Kuwait	Tunisia	Jordan	Palestine	Sudan	Morocco	Qatar	Yemen	Bahrain
Libya	115	29	0	2	17	4	24	70	3	29	0	9	0	2	0	1	0	2	0
Egypt	14	739	2	0	36	6	13	124	7	30	1	7	0	4	3	3	0	0	0
Mauritania	0	2	13	2	2	0	1	20	1	6	0	0	0	0	0	3	0	1	0
Lebanon	0	36	0	59	9	2	5	16	5	8	0	0	0	1	0	0	0	0	0
Saudi_Arabia	4	43	0	0	212	25	39	110	2	58	5	0	2	1	3	3	0	2	1
United_Arab_Emirates	2	22	1	0	44	27	18	26	1	9	3	0	0	0	0	0	0	0	0
Oman	9	13	2	0	63	9	105	94	0	56	1	1	0	0	1	0	0	3	0
Algeria	6	30	0	0	19	4	10	331	4	24	0	6	0	0	1	3	0	1	0
Syria	3	24	1	7	69	5	14	84	55	34	0	1	1	5	0	2	0	1	0
Iraq	7	24	0	7	60	11	36	120	11	366	0	0	1	2	2	2	0	3	0
Kuwait	3	8	0	0	21	13	15	23	0	12	7	0	0	0	0	0	0	0	0
Tunisia	14	16	3	8	3	3	6	60	17	15	0	25	1	2	3	0	0	0	0
Jordan	2	16	0	1	9	2	5	31	9	8	0	0	10	6	0	2	0	1	0
Palestine	1	18	0	3	8	3	6	23	9	8	0	0	5	13	1	3	0	1	0
Sudan	2	10	0	0	0	0	7	12	1	1	0	1	0	0	17	0	0	0	0
Morocco	3	27	0	1	15	5	9	88	2	17	1	2	0	0	0	35	0	0	0
Qatar	2	4	0	1	13	5	4	15	1	4	1	0	0	0	1	0	0	0	0
Yemen	3	22	0	0	11	0	2	21	0	13	0	0	0	0	12	0	0	18	0
Bahrain	1	3	0	0	28	2	9	5	0	1	0	0	0	0	0	0	0	0	2

Figure B.9: The pre-trained model aubmindlab/bert-base-arabertv02-twitter (ARABERTV2T) confusion matrix for dataset `C19`

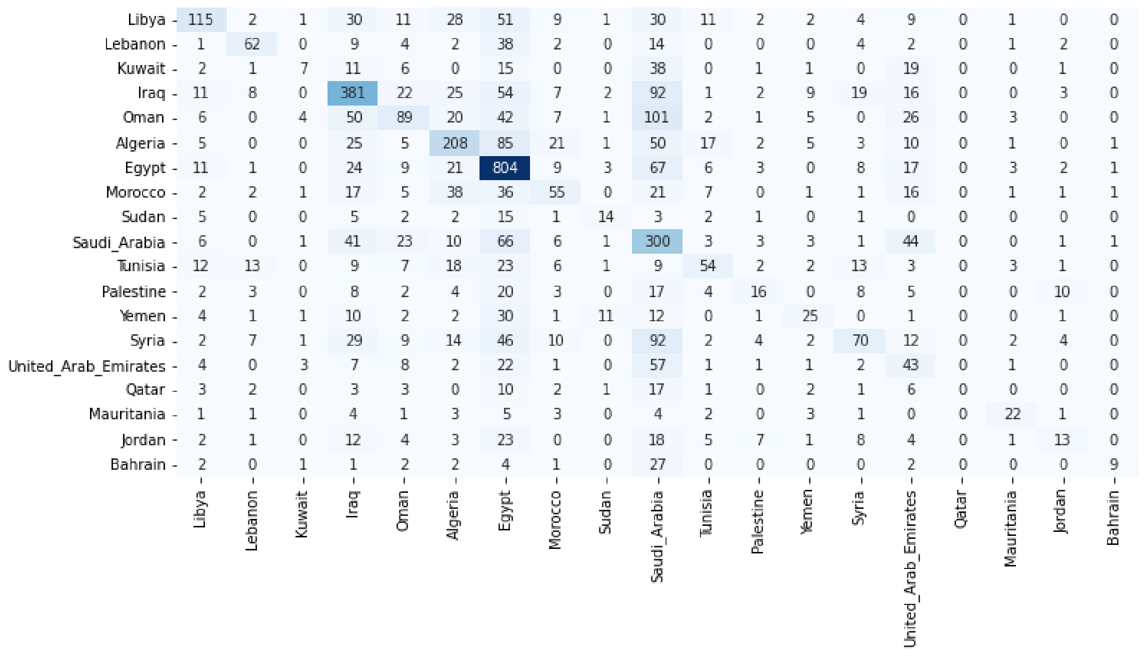


Figure B.10: The pre-trained model aubmindlab/bert-base-arabertv02-twitter (ARABERTV2T) confusion matrix for dataset `C19P`

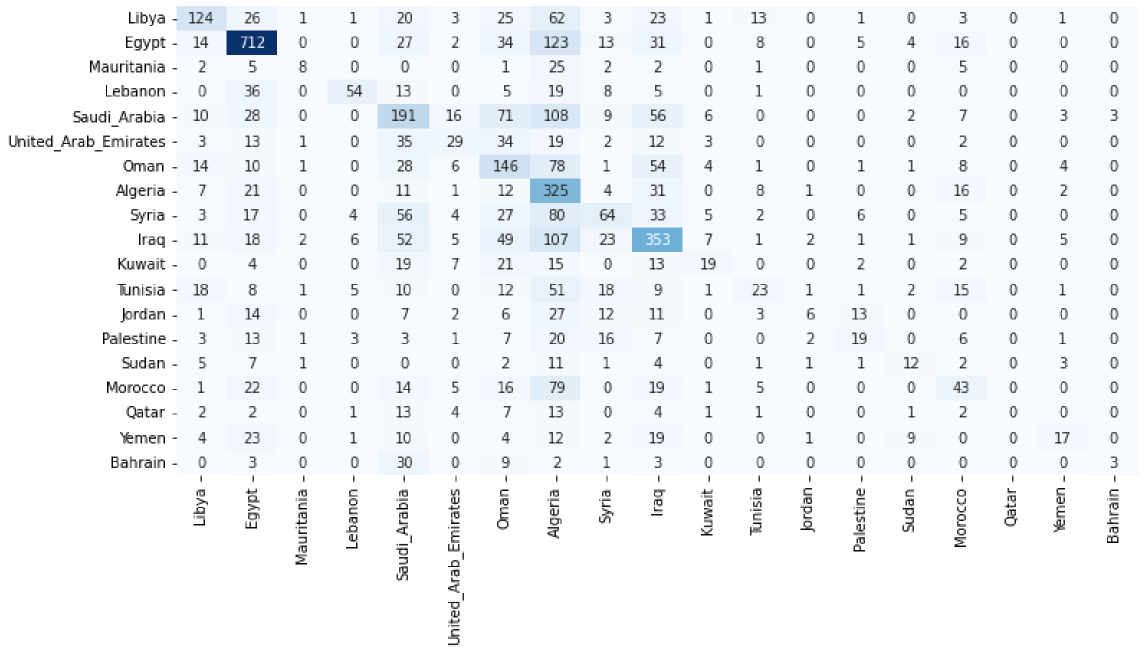
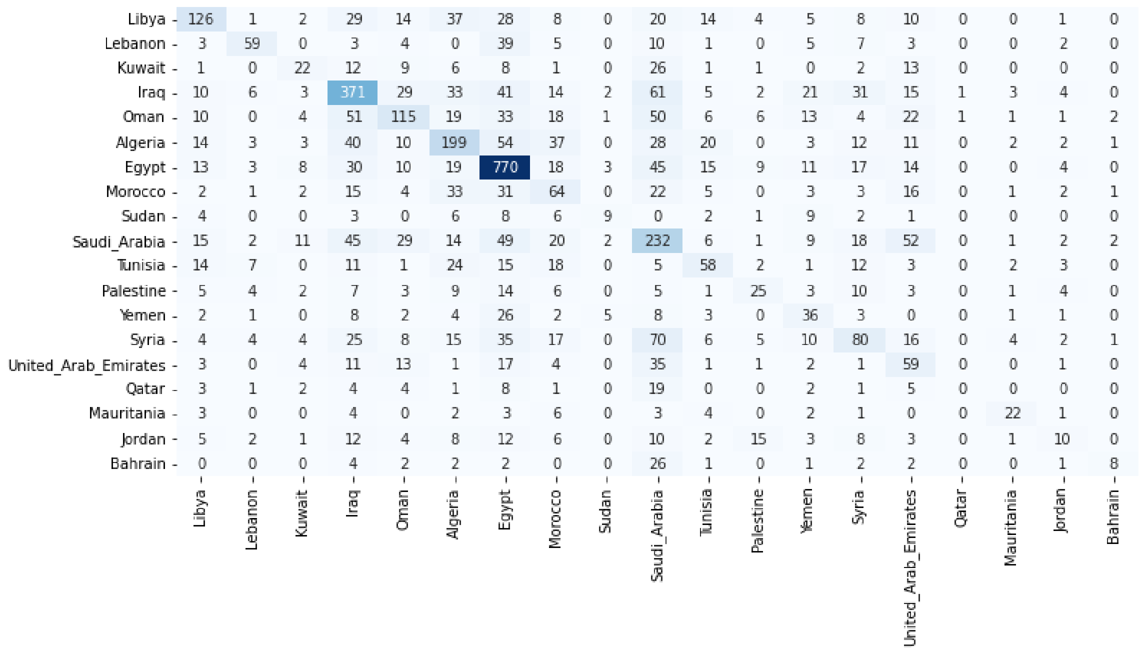
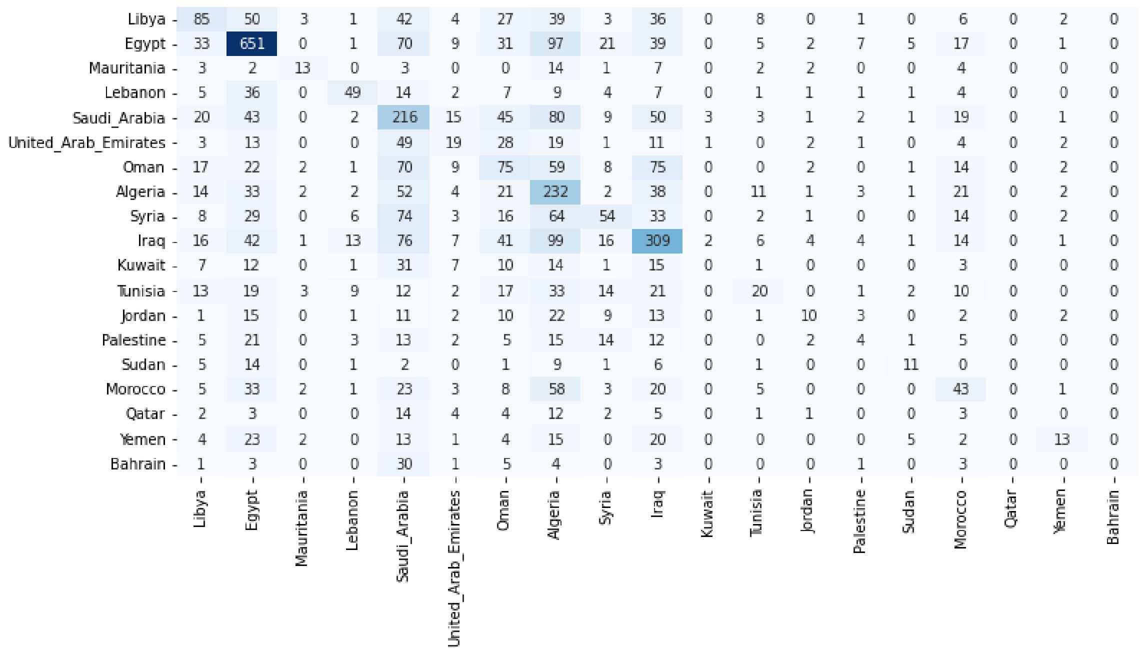


Figure B.11: The pre-trained model UBC-NLP/marbert (MARBERT) BERT confusion matrix for dataset `C19`



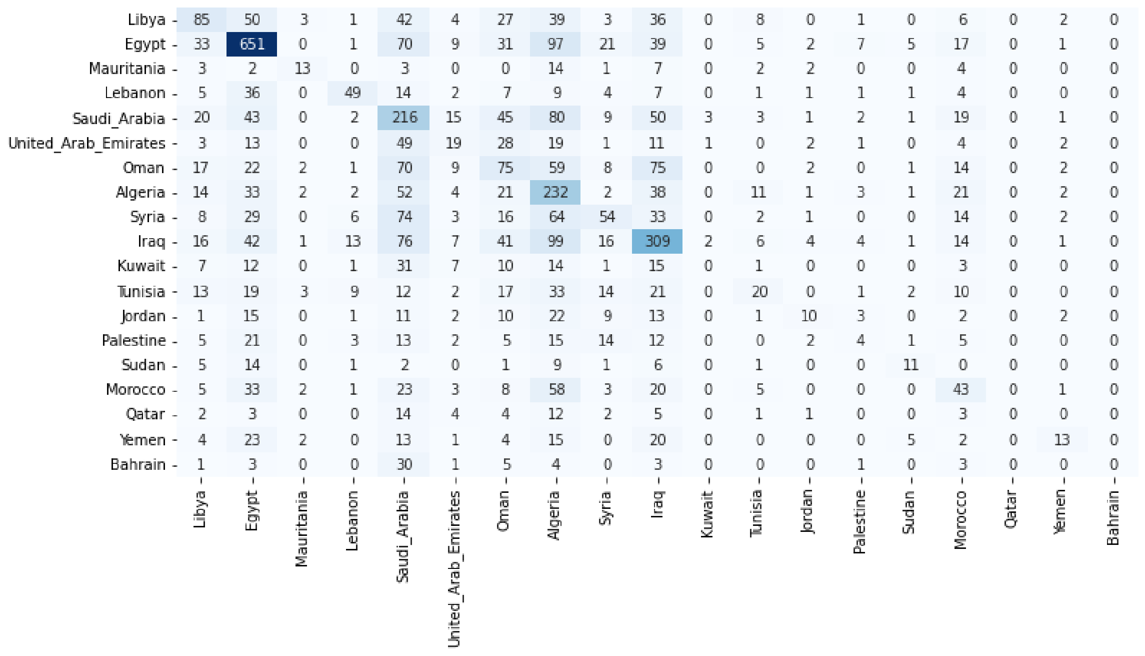
	Libya	Lebanon	Kuwait	Iraq	Oman	Algeria	Egypt	Morocco	Sudan	Saudi_Arabia	Tunisia	Palestine	Yemen	Syria	United_Arab_Emirates	Qatar	Mauritania	Jordan	Bahrain
Libya	126	1	2	29	14	37	28	8	0	20	14	4	5	8	10	0	0	1	0
Lebanon	3	59	0	3	4	0	39	5	0	10	1	0	5	7	3	0	0	2	0
Kuwait	1	0	22	12	9	6	8	1	0	26	1	1	0	2	13	0	0	0	0
Iraq	10	6	3	371	29	33	41	14	2	61	5	2	21	31	15	1	3	4	0
Oman	10	0	4	51	115	19	33	18	1	50	6	6	13	4	22	1	1	1	2
Algeria	14	3	3	40	10	199	54	37	0	28	20	0	3	12	11	0	2	2	1
Egypt	13	3	8	30	10	19	770	18	3	45	15	9	11	17	14	0	0	4	0
Morocco	2	1	2	15	4	33	31	64	0	22	5	0	3	3	16	0	1	2	1
Sudan	4	0	0	3	0	6	8	6	9	0	2	1	9	2	1	0	0	0	0
Saudi_Arabia	15	2	11	45	29	14	49	20	2	232	6	1	9	18	52	0	1	2	2
Tunisia	14	7	0	11	1	24	15	18	0	5	58	2	1	12	3	0	2	3	0
Palestine	5	4	2	7	3	9	14	6	0	5	1	25	3	10	3	0	1	4	0
Yemen	2	1	0	8	2	4	26	2	5	8	3	0	36	3	0	0	1	1	0
Syria	4	4	4	25	8	15	35	17	0	70	6	5	10	80	16	0	4	2	1
United_Arab_Emirates	3	0	4	11	13	1	17	4	0	35	1	1	2	1	59	0	0	1	0
Qatar	3	1	2	4	4	1	8	1	0	19	0	0	2	1	5	0	0	0	0
Mauritania	3	0	0	4	0	2	3	6	0	3	4	0	2	1	0	0	22	1	0
Jordan	5	2	1	12	4	8	12	6	0	10	2	15	3	8	3	0	1	10	0
Bahrain	0	0	0	4	2	2	2	0	0	26	1	0	1	2	2	0	0	1	8

Figure B.12: The pre-trained model UBC-NLP/marbert (MARBERT) BERT confusion matrix for dataset C19P



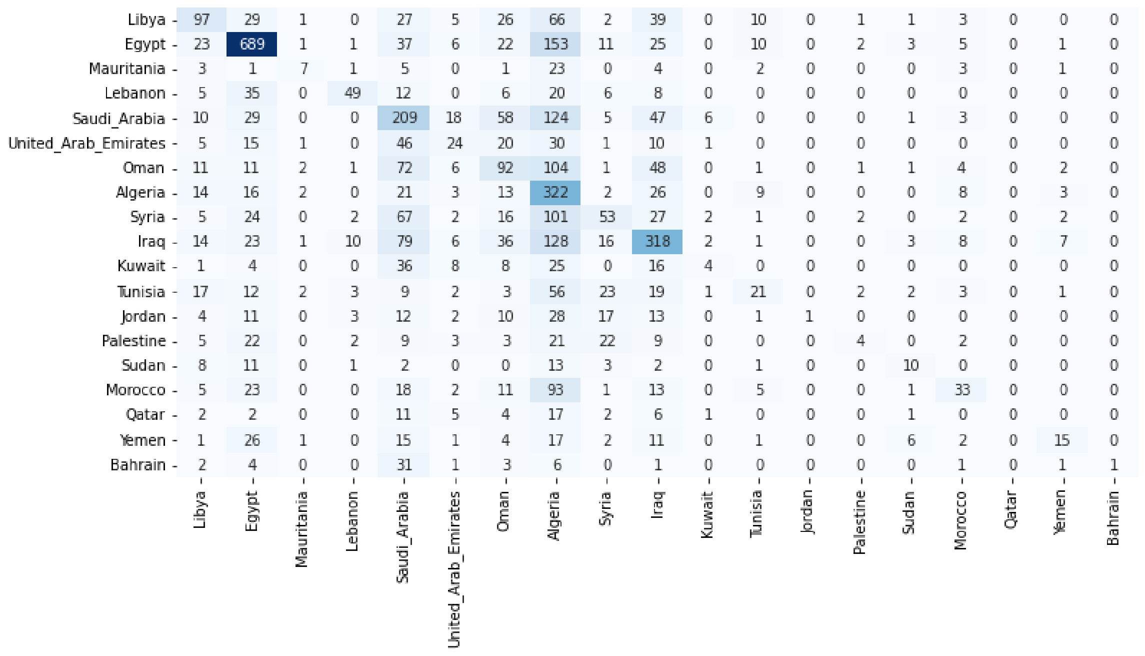
	Libya	Egypt	Mauritania	Lebanon	Saudi_Arabia	United_Arab_Emirates	Oman	Algeria	Syria	Iraq	Kuwait	Tunisia	Jordan	Palestine	Sudan	Morocco	Qatar	Yemen	Bahrain
Libya	85	50	3	1	42	4	27	39	3	36	0	8	0	1	0	6	0	2	0
Egypt	33	651	0	1	70	9	31	97	21	39	0	5	2	7	5	17	0	1	0
Mauritania	3	2	13	0	3	0	0	14	1	7	0	2	2	0	0	4	0	0	0
Lebanon	5	36	0	49	14	2	7	9	4	7	0	1	1	1	1	4	0	0	0
Saudi_Arabia	20	43	0	2	216	15	45	80	9	50	3	3	1	2	1	19	0	1	0
United_Arab_Emirates	3	13	0	0	49	19	28	19	1	11	1	0	2	1	0	4	0	2	0
Oman	17	22	2	1	70	9	75	59	8	75	0	0	2	0	1	14	0	2	0
Algeria	14	33	2	2	52	4	21	232	2	38	0	11	1	3	1	21	0	2	0
Syria	8	29	0	6	74	3	16	64	54	33	0	2	1	0	0	14	0	2	0
Iraq	16	42	1	13	76	7	41	99	16	309	2	6	4	4	1	14	0	1	0
Kuwait	7	12	0	1	31	7	10	14	1	15	0	1	0	0	0	3	0	0	0
Tunisia	13	19	3	9	12	2	17	33	14	21	0	20	0	1	2	10	0	0	0
Jordan	1	15	0	1	11	2	10	22	9	13	0	1	10	3	0	2	0	2	0
Palestine	5	21	0	3	13	2	5	15	14	12	0	0	2	4	1	5	0	0	0
Sudan	5	14	0	1	2	0	1	9	1	6	0	1	0	0	11	0	0	0	0
Morocco	5	33	2	1	23	3	8	58	3	20	0	5	0	0	0	43	0	1	0
Qatar	2	3	0	0	14	4	4	12	2	5	0	1	1	0	0	3	0	0	0
Yemen	4	23	2	0	13	1	4	15	0	20	0	0	0	0	5	2	0	13	0
Bahrain	1	3	0	0	30	1	5	4	0	3	0	0	0	1	0	3	0	0	0

Figure B.13: The pre-trained model UBC-NLP/arbert (ARBERT) confusion matrix for dataset C19



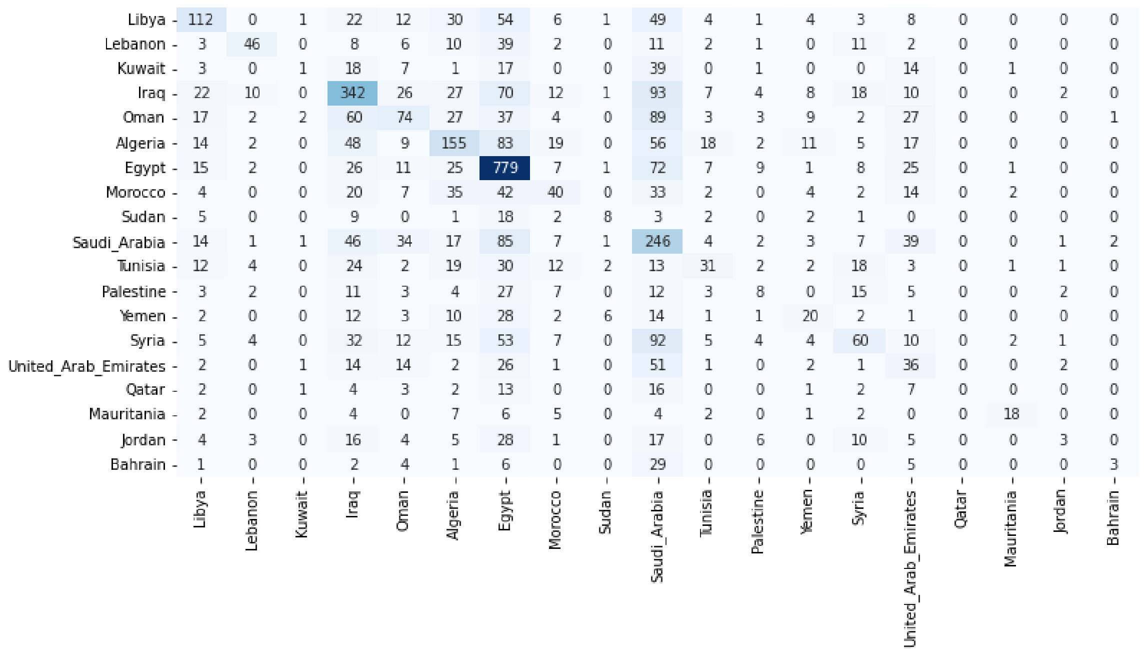
	Libya	Egypt	Mauritania	Lebanon	Saudi_Arabia	United_Arab_Emirates	Oman	Algeria	Syria	Iraq	Kuwait	Tunisia	Jordan	Palestine	Sudan	Morocco	Qatar	Yemen	Bahrain
Libya	85	50	3	1	42	4	27	39	3	36	0	8	0	1	0	6	0	2	0
Egypt	33	651	0	1	70	9	31	97	21	39	0	5	2	7	5	17	0	1	0
Mauritania	3	2	13	0	3	0	0	14	1	7	0	2	2	0	0	4	0	0	0
Lebanon	5	36	0	49	14	2	7	9	4	7	0	1	1	1	1	4	0	0	0
Saudi_Arabia	20	43	0	2	216	15	45	80	9	50	3	3	1	2	1	19	0	1	0
United_Arab_Emirates	3	13	0	0	49	19	28	19	1	11	1	0	2	1	0	4	0	2	0
Oman	17	22	2	1	70	9	75	59	8	75	0	0	2	0	1	14	0	2	0
Algeria	14	33	2	2	52	4	21	232	2	38	0	11	1	3	1	21	0	2	0
Syria	8	29	0	6	74	3	16	64	54	33	0	2	1	0	0	14	0	2	0
Iraq	16	42	1	13	76	7	41	99	16	309	2	6	4	4	1	14	0	1	0
Kuwait	7	12	0	1	31	7	10	14	1	15	0	1	0	0	0	3	0	0	0
Tunisia	13	19	3	9	12	2	17	33	14	21	0	20	0	1	2	10	0	0	0
Jordan	1	15	0	1	11	2	10	22	9	13	0	1	10	3	0	2	0	2	0
Palestine	5	21	0	3	13	2	5	15	14	12	0	0	2	4	1	5	0	0	0
Sudan	5	14	0	1	2	0	1	9	1	6	0	1	0	0	11	0	0	0	0
Morocco	5	33	2	1	23	3	8	58	3	20	0	5	0	0	0	43	0	1	0
Qatar	2	3	0	0	14	4	4	12	2	5	0	1	1	0	0	3	0	0	0
Yemen	4	23	2	0	13	1	4	15	0	20	0	0	0	0	5	2	0	13	0
Bahrain	1	3	0	0	30	1	5	4	0	3	0	0	0	1	0	3	0	0	0

Figure B.14: The pre-trained model UBC-NLP/arbert (ARBERT) confusion matrix for dataset `C19P`



	Libya	Egypt	Mauritania	Lebanon	Saudi_Arabia	United_Arab_Emirates	Oman	Algeria	Syria	Iraq	Kuwait	Tunisia	Jordan	Palestine	Sudan	Morocco	Qatar	Yemen	Bahrain
Libya	97	29	1	0	27	5	26	66	2	39	0	10	0	1	1	3	0	0	0
Egypt	23	689	1	1	37	6	22	153	11	25	0	10	0	2	3	5	0	1	0
Mauritania	3	1	7	1	5	0	1	23	0	4	0	2	0	0	0	3	0	1	0
Lebanon	5	35	0	49	12	0	6	20	6	8	0	0	0	0	0	0	0	0	0
Saudi_Arabia	10	29	0	0	209	18	58	124	5	47	6	0	0	0	1	3	0	0	0
United_Arab_Emirates	5	15	1	0	46	24	20	30	1	10	1	0	0	0	0	0	0	0	0
Oman	11	11	2	1	72	6	92	104	1	48	0	1	0	1	1	4	0	2	0
Algeria	14	16	2	0	21	3	13	322	2	26	0	9	0	0	0	8	0	3	0
Syria	5	24	0	2	67	2	16	101	53	27	2	1	0	2	0	2	0	2	0
Iraq	14	23	1	10	79	6	36	128	16	318	2	1	0	0	3	8	0	7	0
Kuwait	1	4	0	0	36	8	8	25	0	16	4	0	0	0	0	0	0	0	0
Tunisia	17	12	2	3	9	2	3	56	23	19	1	21	0	2	2	3	0	1	0
Jordan	4	11	0	3	12	2	10	28	17	13	0	1	1	0	0	0	0	0	0
Palestine	5	22	0	2	9	3	3	21	22	9	0	0	0	4	0	2	0	0	0
Sudan	8	11	0	1	2	0	0	13	3	2	0	1	0	0	10	0	0	0	0
Morocco	5	23	0	0	18	2	11	93	1	13	0	5	0	0	1	33	0	0	0
Qatar	2	2	0	0	11	5	4	17	2	6	1	0	0	0	1	0	0	0	0
Yemen	1	26	1	0	15	1	4	17	2	11	0	1	0	0	6	2	0	15	0
Bahrain	2	4	0	0	31	1	3	6	0	1	0	0	0	0	0	1	0	1	1

Figure B.15: The pre-trained model bashar-talafha/multi-dialect-bert-base-arabic (MULTIDB) confusion matrix for dataset `C19`



	Libya	Lebanon	Kuwait	Iraq	Oman	Algeria	Egypt	Morocco	Sudan	Saudi_Arabia	Tunisia	Palestine	Yemen	Syria	United_Arab_Emirates	Qatar	Mauritania	Jordan	Bahrain
Libya	112	0	1	22	12	30	54	6	1	49	4	1	4	3	8	0	0	0	0
Lebanon	3	46	0	8	6	10	39	2	0	11	2	1	0	11	2	0	0	0	0
Kuwait	3	0	1	18	7	1	17	0	0	39	0	1	0	0	14	0	1	0	0
Iraq	22	10	0	342	26	27	70	12	1	93	7	4	8	18	10	0	0	2	0
Oman	17	2	2	60	74	27	37	4	0	89	3	3	9	2	27	0	0	0	1
Algeria	14	2	0	48	9	155	83	19	0	56	18	2	11	5	17	0	0	0	0
Egypt	15	2	0	26	11	25	779	7	1	72	7	9	1	8	25	0	1	0	0
Morocco	4	0	0	20	7	35	42	40	0	33	2	0	4	2	14	0	2	0	0
Sudan	5	0	0	9	0	1	18	2	8	3	2	0	2	1	0	0	0	0	0
Saudi_Arabia	14	1	1	46	34	17	85	7	1	246	4	2	3	7	39	0	0	1	2
Tunisia	12	4	0	24	2	19	30	12	2	13	31	2	2	18	3	0	1	1	0
Palestine	3	2	0	11	3	4	27	7	0	12	3	8	0	15	5	0	0	2	0
Yemen	2	0	0	12	3	10	28	2	6	14	1	1	20	2	1	0	0	0	0
Syria	5	4	0	32	12	15	53	7	0	92	5	4	4	60	10	0	2	1	0
United_Arab_Emirates	2	0	1	14	14	2	26	1	0	51	1	0	2	1	36	0	0	2	0
Qatar	2	0	1	4	3	2	13	0	0	16	0	0	1	2	7	0	0	0	0
Mauritania	2	0	0	4	0	7	6	5	0	4	2	0	1	2	0	0	18	0	0
Jordan	4	3	0	16	4	5	28	1	0	17	0	6	0	10	5	0	0	3	0
Bahrain	1	0	0	2	4	1	6	0	0	29	0	0	0	0	5	0	0	0	3

Figure B.16: The pre-trained model bashar-talafha/multi-dialect-bert-base-arabic (MULTIDB) confusion matrix for dataset C19P

Appendix C

List of publications

- **Evaluating Sentiment Classifiers for Bitcoin Tweets in Price Prediction Task** paper that is published in 2019 IEEE International Conference on Big Data (Big Data) [33].

- **N-gram and Word2Vec Feature Engineering Approaches for Spam Recognition on Some Influential Twitter Topics in Saudi Arabia** paper is published at the ICISDM 2022: the 6th International Conference on Information System and Data Mining and published by Association for Computing Machinery (ACM) [31].

- Journal article **N-gram and Word2Vec Feature Engineering Approaches for Spam Recognition on Some Influential Twitter Topics in Saudi Arabia paper** is published in JAIT: Journal of Advances in Information Technology [32].

- **Twitter Arabic Dialect Identification using AraBERT** is a short paper and presentation that is published at the 39th IBIMA conference as an in-proceeding short paper, and the presentation is also published as a YouTube video on IBIMA's official channel as presented at their virtual conference [29].

- **Twitter Arabic Dialect Identification using Different Feature Engineering Models** is published at the 40th IBIMA conference as a full paper, with an invitation for journal submission of the paper, and published YouTube video on IBIMA's official channel as presented at their virtual conference [30].

- Journal article **Twitter Arabic Dialect Identification using Different Feature Engineering Models** was accepted to be published at "The Communications of the IBIMA", by IBIMA Publishing.