ENHANCING MULTI-CLASS TEXT CLASSIFICATION IN ONLINE DRUG REVIEWS THROUGH MACHINE LEARNING APPROACHES

BAVAHRNI A/P SUBRAMANIAM

PROJECT SUBMITTED IN PARTIAL FULFILMENT FOR THE DEGREE OF MASTER OF DATA SCIENCE

FACULTY OF INFORMATION SCIENCE AND TECHNOLOGY
UNIVERSITI KEBANGSAAN MALAYSIA
BANGI
2024

ENHANCING MULTI-CLASS TEXT CLASSIFICATION IN ONLINE DRUG REVIEWS THROUGH MACHINE LEARNING APPROACHES

BAVAHRNI A/P SUBRAMANIAM

PROJEK YANG DIKEMUKAKAN UNTUK MEMENUHI SEBAHAGIAN
DARIPADA SYARAT UNTUK MEMPEROLEH IJAZAH
SARJANA SAINS DATA

FAKULTI TEKNOLOGI DAN SAINS MAKLUMAT
UNIVERSITI KEBANGSAAN MALAYSIA
BANGI

2024

DECLARATION

I hereby declare that the work in this project is my own except for quotations and summaries which have been duly acknowledged.

26 June 2024

BAVAHRNI SUBRAMANIAM
P122569

ACKNOWLEDGEMENT

I would like to express my sincere gratitude to all those who have contributed to the completion of this project. First and foremost, I am deeply thankful to my project supervisor, Associate Professor Dr.Suhaila Zainudin for her guidance, support, and valuable insights throughout this project journey. Her expertise and encouragement have been instrumental in shaping this project.

I owe hugely to my dear parents, Mr. Subramaniam Allagan and Mdm. Leelawathy N. Nalliah for their permanent love and confidence in encouraging me to go ahead in my study and career. My thanks equally go to my parents-in–law, sister, Kasthuri, brother, Rooben, brothers-in-law, sisters-in-law, nephews and nieces for the love and support shown throughout my studies.

This project would have remained a dream had it not been for my dearest husband, Ir.Ts.Dr. Narendren Rengasamy. His unwavering love, support, and understanding have been my anchor, providing me with the strength and motivation to persevere during challenging times.

Finally, I would like to express my profound gratitude to the Almighty for His blessings, guidance, and strength that have sustained me throughout this endeavour.

ABSTRAK

Fasa pandemik yang lalu telah menekankan kepentingan dan keperluan tindak balas yang segera dalam pelaksanaan polisi dan amalan bidang kesihatan, terutamanya apabila melibatkan pengenalan ubat baru. Salah satu langkah kritikal dalam pengenalan ubat baru adalah, analisis ulasan ubat, di mana maklum balas klinikal tentang penggunaan ubat dinilai, mengenal pasti risiko berpotensi, dan kesan-kesan sampingan. Walaubagaimanpun, saiz data yang besar, bentuk data yang tidak berstruktur dan penggunaan bahasa tabii menyebabkan analisis ulasan ubat yang segera secara tepat dan berkesan adalah suatu cabaran. Pembangunan terkini dalam pembelajaran mesin, khususnya pemprosesan bahasa tabii, memberikan peluang untuk menangani cabaran-cabaran ini. Dalam kajian ini, data yang diperolehi daripada pangkalan data *drugs.com* dan *drug.lib* dianalisis dengan menggunakan lapan algoritma pembelajaran mesin iaitu Multinomial Naïve Bayes(MNB), Random Forest, Decision Tree, Extra Tree, Extreme Gradient Boost, Linear Support Vector Classifier(SVC), Logistic Regression, dan K-Nearest Neighbour. Bagi mengkaji kesan fitur, algoritma dan pecahan data latihan/ujian pada skor-F1 dan masa, empat jenis fitur (bag-of-words (BOW), term frequency-inverse document frequency (tf-idf) unigram, bigram, dan trigram) dan 5 jenis pecahan latihan/ujian (50/50, 60/40, 70/30, 80/20, 90/10) telah digunakan. Keputusan kajian mendapati Linear SVC memiliki skor-F1 yang tertinggi, manakala MNB pula mengambil masa yang paling singkat. Selain itu, kajian juga mendapati bahawa pemilihan fitur memberikan impak kepada skor-F1 dan keseluruhan masa yang diambil oleh sesuatu algoritma bagi menyelesaikan turutan kod dimana pengunaan BOW memberikan skor-F1 yang tertinggi dan masa yang paling pendek. Secara keseluruhannya, kombinasi model linear SVC dengan pecahan 90/10 bersama tf-idf bigram memberikan keputusan yang paling optimum. Kajian ini memberikan pengetahuan yang berharga tentang impak fitur, pembahagian set data latihan/ujian, dan algoritma pada data ulasan ubat yang boleh digunapakai di dalam bidang kesihatan.

ABSTRACT

The recent pandemic highlighted the importance and need for a quick response in the implementation of healthcare policies, and practices, especially when it comes to introduction of new drugs. Drug review analysis, where the clinical feedback on the drug usage is evaluated, identifying potential risks, and adverse effects is a critical step in this. However, as the data obtained is typically large in number, unstructured in form, and often expressed using natural language, processing it accurately, effectively and in time was challenging. The recent development in machine learning, specifically Natural Language processing, provided an opportunity for these challenges to be addressed. In this study, data obtained from drugs.com and drug.lib was analysed under eight different machine learning algorithms, Multinomial Naïve Bayes (MNB), Random Forest(RF), Decision Tree (DT), Extra Tree (ET), Extreme Gradient Boost (XGB), Linear Support Vector Classifier(SVC), Logistic Regression, and K-Nearest Neighbour(KNN). Four different feature extractors bag-of-words (BOW), term frequency-inverse document frequency (tf-idf) unigram, bigram and trigram, and 5 different train/test splits (50/50, 60/40, 70/30, 80/20, 90/10) were also employed to study the effect of the different feature extractors, algorithms, and splits on the F1 -score and total time. Linear SVC had the highest F1 score, while MNB had the shortest time. It was observed that the choice of feature extractor had an impact on the F1 score and total time, with BOW having the highest F1 score and lowest time. Overall, the linear SVC with 90/10 split and tf-idf bigram combination was the most optimum approach. This study provided valuable insight on the impact of feature extractor, split and algorithm on drug review data which is particularly useful in the healthcare field.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF ILLUSTRATION

LIST OF ABBREVIATIONS

ADHD	Attention Deficit Hyperactivity Disorder
ADR	Adverse Drug Reaction
AI	Artificial Intelligence
AUC	Area Under the Curve
BI	Business Intelligence
BOW	Bag-of-Words
CART	Classification and Regression Tree
COVID	coronavirus disease
CPU	Central Processing Unit
DCT	Discrete Cosine Transform
DNH	Distributed Nearest Hash
DT	Decision Tree
ET	Extra Tree
ETC	Extra Tree Classifier
FN	False Negative
FP	False Positive
FPGA	Field-Programmable Gate Arrays
GAN	Generative Adversarial Networks
GCN	graph convolutional networks
GPU	Graphics Processing Unit
IQR	Inter Quartile Range
KNN	K-Nearest Neighbour
LASSO	Least Absolute Shrinkage and Selection Operator
LDA	Latent Dirichlet Allocation
LR	Logistic Regression
ML	Machine Learning
MLP	Multi-Layer Perceptron

MNB	Multinomial Naïve Bayes
NB	Naïve Bayes
NLP	Natural Language Processing
RF	Random Forest
SGD	Stochastic Gradient Descent
SVC	Support Vector Classifier
SVM	Support Vector Machine
TF-IDF	Term Frequency-Inverse Document Frequency
TN	True Negative
TP	True Positive
UCI	University of California, Irvine
USA	United States of America
VC	Vapnik-Chervonenkis theory
XAI	Explainable Artificial Intelligence
XGB	Extreme Gradient Boost

CHAPTER I

INTRODUCTION

1.1 BACKGROUND

The recent pandemic that ravaged globally not only disrupted the livelihood of the masses, but it also took a hefty toll on the healthcare system, highlighted the shortcomings and limitation of the current practises, specifically when it came to the introduction of new medication and treatments. The sudden demand for new vaccines and drugs, and the time pressure exposed the challenges associated with processing of clinical trials data. When large scale trials were conducted, there was a need for an accurate and fast way to process and analyse the data collected (Kashte et al. 2021). The multiple parallel trials conducted across the globe generated huge amounts of data that was not only limited to the adverse drug effects and other medical parameters, but also contained additional information such as demographics and geographical data. This large data sets, if used well, would be of utmost beneficial in the introduction of new drugs (Hiscott et al. 2020).

Another lesson learned from the recent pandemic was on the public perception and the need for greater transparency. In Malaysia, the Ministry of Health (MoH) lead the way by making Covid-19 related raw data publicly available to the masses. The demand and expectation for information from the general public was not limited to merely gaining access to clinical data but, it also involved a feedback system where they can also provide real time valuable input back to the healthcare policymakers. It was also observed that the public expected that their feedback be taken seriously and in most cases, a personalised reply is provided (Lee et al. 2022). This was a crucial aspect when it came to introduction of new treatments and drugs as it not only resulted in an increase in consumer confidence and acceptance of the drugs, but it also lead to an effective

implementation of various healthcare policies, such as the national Covid -19 vaccination policy introduced by MoH (Ain Umaira Md Shah et al. 2020). Artificial Intelligence and machine learning (ML) have woven themselves into the fabric of our daily lives, subtly and sometimes overtly shaping our experiences. From movies and music to news and products, AI-powered algorithms personalize content, catering to our individual preferences (Shuai Zhang et al. 2019). Virtual assistants like Siri and Alexa handle tasks like scheduling appointments, controlling smart home devices, and providing information - enhancing convenience and automating daily routines (Rajakumaran et al. 2023). Adaptive learning platforms powered by ML tailor educational content and pace to individual student needs, improving learning outcomes (Daugherty et al. 2022). Ride-sharing apps and smart traffic management systems powered by AI are optimizing routes and reducing congestion, impacting our daily commutes (Ang et al. 2022). Chatbots and automated systems handle basic customer inquiries, freeing up human agents for complex issues and improving overall efficiency(Ngai et al. 2021). From manufacturing robots to automated data analysis, AI is driving increased efficiency and productivity across various sectors (Jan et al. 2023). Artificial Intelligence algorithms are assisting doctors in analysing medical images and making diagnoses, potentially leading to earlier detection and improved outcomes (Huang et al. 2021). ML models are accelerating the drug discovery process, bringing us closer to faster development of life-saving medications (Choudhury et al. 2022). Tailoring treatment plans to individual patients based on vast data sets, AI holds promise for a future of precise and effective healthcare (Alowais et al. 2023)

The digital age has witnessed an explosion of user-generated content on health and wellness, with online drug reviews taking centre stage. This readily available data, constantly growing with every shared experience, holds immense potential for understanding patient perspectives on various medications (Anjali & GK 2022). While clinical data relies on formal medical reports and records, online reviews provide insights through unfiltered, everyday language (Joshi & Abdelfattah 2021) which includes slang, emojis, or personal anecdotes. Deriving valuable knowledge about patient treatment responses from these large dataset reviews often hinges on time-consuming, laborious manual reviews of textual data, underscoring the need for automated analysis methods (Ling 2023). The rise of machine learning and NLP has

unlocked exciting possibilities for extracting crucial insights from online drug reviews, enhancing pharmacovigilance monitoring and proactive identification of potential drug safety concerns (Anjali & GK 2022).

1.2 PROBLEM STATEMENT

With the increased use of information technology, social media and other digital documentation and data gathering tools especially within the healthcare industry, a huge amount of data is amassed. These large datasets, often running into hundreds of thousands if not million data points contains important information that when put to use would greatly benefit and accelerate response time and enable a comprehensive drug review to be conducted prior to its release. However, these large data cannot be processed using conventional means, either manually or via low level spreadsheets. The full magnitude of information obtained in the data would not be realised in this way. Hence, a new problem is found, where although huge amounts of data is available, a new method and tool needs to be used to effectively make sense of it.

The use of and access to information technology tools such as web-based data collection and feedback system is continuously growing. Locally, the collection of adverse drug reaction ADR reports on the Covid-19 vaccine via mysejahtera was an example where the consumer gets to directly input their feedback and reaction on certain drugs and vaccines to the relevant authorities. Besides resulting in huge amounts of data, the data obtained is also often unstructured in form. The use of natural language, conversational slang, non-numerical feedback and non-standard terminology add to the complexity in processing these information. This leads to a time-consuming process where each review and response needs to be evaluated manually, once again leading to not only delay in obtaining information but also a possible lower accuracy.

Joshi and Abdelfattah (2021) had used similar method and drug review datasets for multi-label classification, however the study did not explore on the use of different features, sampling methods and machine learning models such as XGB and KNN.

1.3 RESEARCH OBJECTIVES

1. To analyse the effect of different machine learning models and feature extraction techniques on drug review dataset
2. To identify the best machine learning model and feature extraction technique in terms of F1-score and runtime

1.4 RESEARCH SCOPE

This study was focused on multi-label text classification of patient conditions based on online drug review dataset obtained from publicly available database (UCI Machine Learning Repository) which was last updated in 2018. Only top 10 conditions were selected for the study based on the reviews submitted by consumers in USA. Only eight machine learning classifiers (Multinomial Naïve Bayes, Linear Support Vector classifier, Logistic Regression, Random Forest, Extra Tree, Decision Tree, Extreme Gradient Boost and K-Nearest Neighbour) and four feature extraction techniques were utilized for the study. To ensure balanced class representation in both training and test data, we employed stratified sampling techniques, examining five split ratios: 90/10, 80/20, 70/30, 60/40, and 50/50. Due to the limitations of accuracy in imbalanced datasets, F1-score, which considers both precision and recall, was chosen as the primary performance metric for this study. Besides that, to assess the computational speed time, the time taken to train and test each algorithm in Colab environment were recorded.

1.5 PROJECT STRUCTURE

This project is structured in a conventional format, split into 6 different chapters.

Chapter 1 : Introduction, the overall background of the project, together with the research objectives and research scope and limitations were discussed.

Chapter 2 : Literature Review presents a theoretical overview on key terms and aspects of machine learning and text classification approaches. Past research that utilised similar machine learning algorithm were also discussed.

Chapter 3 : Methodology. In this chapter, the overall methods used for this study was explained in detail.

Chapter 4 : Results provides a graphical representation and a brief explanation of the results obtained from the different approaches applied.

Chapter 5 : Discussion is where in depth discussions from an holistic perspective was done. Looking at the various aspects and comparing the results to past studies.

Chapter 6 : Conclusion. This chapter highlights the general conclusion of the study and how it ties to the research objectives. Suggestions for future work and key contributions of this study was also discussed in this chapter.

CHAPTER II

LITERATURE REVIEW

2.1 INTRODUCTION TO DRUG REVIEW

Pharmacovigilance, as defined by the World Health Organization (WHO), encompasses the activities of identifying, evaluating, understanding, and preventing Adverse Drug Reactions (ADRs) (Gawich & Alfonse 2022). While drugs undergo rigorous testing before approval, many adverse effects are only discovered after widespread use in the real world (Saad et al. 2021). User-generated drug reviews, often containing information on patient conditions, experiences, and side effects, offer a valuable source of data for pharmacovigilance efforts. The rapid growth of electronically accessible health data, coupled with the ability of Natural Language Processing (NLP) and machine learning to handle large volumes of text, has opened new avenues for pharmacological monitoring (Anjali & GK 2022). One application lies in drug selection accuracy. By analyzing user reviews and patient conditions, NLP and machine learning can help healthcare professionals identify the most effective drugs for specific situations. Text mining techniques can further aid this process by extracting data patterns, trends, and potential knowledge from user-generated reviews (Haryadi et al. 2022). Classifying patient conditions based on drug reviews can not only reveal previously unknown ADRs but also alert healthcare professionals to potential risks associated with medications. Ultimately, selecting the right drug for a patient's condition not only improves treatment outcomes but also increases patient satisfaction and medication adherence (Haque et al. 2023) Therefore, careful selection of appropriate machine learning methods for drug review classification tasks is crucial for advancing pharmacovigilance efforts.

2.2 INTRODUCTION TO MACHINE LEARNING

The promising field of machine learning (ML) has rapidly ascended to a prominent position within the burgeoning domain of artificial intelligence (AI). It is crucial to dispel the misconception that ML and AI are synonymous. As delineated by Russell and Norvig (2010), AI encompasses a broader spectrum of capabilities, including reasoning, problem-solving, and adaptation. Within this framework, ML emerges as a potent engine, enabling AI systems to acquire and refine knowledge from data in an autonomous manner, independent of explicit programming (Alpaydin 2021 ; Kufel et al. 2023). This paradigm shift empowers AI with the ability to learn from experience, akin to a dynamic and adaptable actor in a complex performance. ML boasts a diverse repertoire of techniques, each meticulously designed for specific roles. Supervised learning, akin to a seasoned mentor, guides algorithms by providing labelled data (paired inputs and desired outputs), enabling them to map future inputs to the corresponding outputs with accuracy (Russell & Norvig 2010). This empowers predictive feats across diverse domains, such as market trend forecasting or image categorization (James et al. 2021). Conversely, unsupervised learning, the intrepid explorer, thrives in the absence of labels, uncovering hidden patterns and relationships within data (Alloghani et al. 2020).

However, the development of ML within AI is not without its significant challenges. Data biases, which can arise from stereotypical representations within training data, necessitate the adoption of responsible data sourcing practices and the implementation of bias mitigation techniques (Verma et al. 2021) The lack of explanation ability and interpretability in complex ML models raises concerns regarding accountability and potential misuse, highlighting the need for further research into model transparency techniques (Miller 2019). Furthermore, ethical considerations surrounding the societal implications of ML necessitate careful reflection and the development of robust ethical frameworks to guide responsible development and deployment (Floridi 2023). Despite these challenges, the future of ML within AI remains promising. Recent advancements, such as the Transformer architecture, have revolutionized natural language processing with breakthroughs in translation, summarization, and question answering tasks (Choi & Lee 2023).. Federated learning

addresses privacy concerns by enabling collaborative training on decentralized data, thereby enhancing model generalizability (Yu et al. 2023) Additionally, the development of Explainable AI (XAI) techniques is shedding light on the inner workings of complex models, fostering trust and transparency in AI systems (Arrieta et al. 2020).

2.3 HISTORY OF MACHINE LEARNING

The seeds of machine learning (ML) sprouted in the 1940s and 50s, nurtured by the burgeoning fields of cybernetics, control theory, and computational learning. Alan Turing's visionary 1950 paper, "Computing Machinery and Intelligence," and Frank Rosenblatt's groundbreaking creation of the Perceptron in 1958, the first artificial neural network, laid the groundwork for the explosive growth of ML research in the following decades (Rosenblatt 1962 ; Turing 2009). The 1960s and 1970s witnessed a golden age, with Arthur Samuel officially coining the term "machine learning" in 1959, the rise of powerful algorithms like decision trees and support vector machines, and even a temporary fascination with knowledge-based systems that relied on symbolic reasoning (Alpaydin 2020 ; Samuel 2000). While these early forays may not resemble the data-driven giants of today's ML landscape, they offer a fascinating glimpse into the evolution of this transformative technology.

The 1980s and 90s saw a decline in AI research funding and interest, often referred to as the "AI winter" (Toosi et al. 2021). However, this period also laid the groundwork for future breakthroughs. Theoretical advances like Vapnik-Chervonenkis theory (VC dimension) provided a rigorous framework for understanding generalization and model selection in ML (Liu 2023). Renewed interest in neural networks emerged, spurred by successes in backpropagation algorithms and advancements in computational power. The proliferation of data in the 21st century has fuelled an unprecedented renaissance in ML. Key drivers include the rise of deep learning, deep neural networks with multiple hidden layers have achieved remarkable performance in diverse tasks, from image recognition to natural language processing (Alom et al. 2019) Availability of powerful computing resources is another factor affecting the uptake of ML. GPUs and specialized hardware have significantly accelerated training times for

complex models(Khan et al. 2022). Similarly, tools like TensorFlow and PyTorch have made ML accessible to a broader audience, fuelling innovation and collaboration (Mayer & Jacobsen 2020) Types of Machine Learning

Machine learning (ML) has permeated diverse fields, prompting a need for a structured understanding of its various approaches. ML techniques can generally be grouped in to 3 main groups namely Supervised Learning, Unsupervised Learning, and Reinforced learning (Alsharif et al. 2020).

2.4 SUPERVISED LEARNING

The foundation of supervised learning lies in its labelled data, where each input observation is paired with its corresponding desired output. This allows algorithms to learn a mapping between inputs and outputs, enabling them to predict future outputs with remarkable accuracy. Supervised learning algorithms thrive on labelled training data, where each data point consists of features (inputs) paired with corresponding target values (labels). During training, the algorithm analyses this data, seeking patterns and relationships between features and labels. It essentially learns a mapping function that associates certain feature combinations with specific outputs. Once trained, the algorithm can leverage this mapping function for prediction. When given new input data, it analyses the features and identifies the output (label in classification, value in regression) most likely associated with those features based on the learned patterns (Tufail et al. 2023).

The ultimate goal of a supervised learning model is to perform well on unseen data. Whether classifying data points into distinct categories or estimating continuous values, the model strives to accurately predict the appropriate label or output for any new input it encounters (Tufail et al. 2023). This prowess manifests in various tasks, including regression, where algorithms predict continuous values like market trends or housing prices (Hastie et al. 2009 ; James et al. 2021), and classification, where algorithms categorize data points into distinct classes, as seen in image recognition (Sarker 2021) and sentiment analysis (Wankhade et al. 2022).

Recent advancements in supervised learning are pushing the boundaries of what's possible. Transformers, for instance, have revolutionized natural language processing with their groundbreaking performances in translation, summarization, and question answering (Patwardhan et al. 2023). Another noteworthy leap forward is Bayesian Deep Learning, which integrates probabilistic approaches with deep neural networks, enhancing the quantification of uncertainty and interpretability of complex models (Deodato 2019). These advancements highlight the dynamic and ever-evolving nature of supervised learning, promising further breakthroughs in diverse domains (Burkart & Huber 2021).

2.5 UNSUPERVISED LEARNING

In contrast to its supervised counterpart, unsupervised learning operates in the realm of unlabelled data. This category seeks to unearth hidden patterns and relationships within data, akin to an explorer deciphering an unknown map. Common tasks within this realm include clustering where data points are grouped together based on shared characteristics, enabling tasks like customer segmentation for targeted marketing (James et al. 2021) . Dimensionality Reduction where complex data is simplified for efficient analysis, facilitating visualization of high-dimensional datasets, is another task under the unsupervised learning approach. (Usama et al. 2019).

Recent advancements within unsupervised learning have brought forth the Generative Adversarial Networks (GANs). These models generate realistic data, enabling tasks like creating high-resolution images or synthetic speech(Dash et al. 2023). Autoencoders are another recent development in unsupervised machine learning. This technique learns compressed representations of data, proving valuable for anomaly detection and data dimensionality reduction (Boquet et al. 2020).

2.6 SEMI-SUPERVISED LEARNING

Partially labelled data, where most labels are missing, poses a challenge for learning accurate models. Semi-supervised methods can leverage these few labelled examples to infer labels for the vast unlabelled data, enriching the training set. Similarly, classifying movie genres from synopses can benefit from utilizing labelled examples alongside

unlabelled synopses to refine genre detection. While semi-supervised learning can boost performance compared to purely supervised methods with limited labelled data, it's important to acknowledge that the accuracy may not always surpass supervised learning with a full set of labels (Alpaydin 2021 ; Tufail et al. 2023). The effectiveness depends on data characteristics, chosen algorithms, and the specific task at hand. Nonetheless, for scenarios with scarce labelled data, semi-supervised learning offers a powerful way to unlock the potential of partially-labelled datasets (Tufail et al. 2023).

There have been recent developments in the semi supervised machine learning approach. The graph-based methods where the inherent structure of data through graphs is exploited has led to powerful semi-supervised algorithms. Techniques like graph convolutional networks (GCNs) can effectively propagate label information across data points connected through edges, leading to improved performance on tasks like image segmentation and protein folding (Song et al. 2022).. Generative adversarial networks (GANs) and their variants are also increasingly being used to generate synthetic labelled data, augmenting the existing labelled dataset and enriching the training process. This can be particularly beneficial in scenarios with limited labelled data (Tu & Yang 2019). Active learning is another advancement in semi-supervised ML techniques. Choosing which unlabelled data points to label strategically can significantly improve the efficiency of semi-supervised learning. Recent advances in active learning algorithms focus on identifying the most informative and impactful data points for labelling, leading to better model performance with less manual effort (Flores & Verschae 2022). Uncertainty quantification that involves understanding the uncertainty associated with predictions from semi-supervised models is another crucial improvement as it facilitates building of trust and transparency. Recent research focuses on developing techniques to quantify uncertainty, allowing users to assess the confidence of the model's predictions {Zhao, 2020 #191.

2.7 REINFORCED LEARNING

Reinforcement learning embodies the self-taught student, continuously interacting with its environment to refine its behaviour. This category operates through trial and error, receiving rewards for desirable actions and penalties for undesirable ones, ultimately

maximizing its reward. Common tasks tackled by reinforcement learning include controlling of robots and automation, enabling robots to learn optimal navigation and manipulation strategies (Mnih et al. 2015 ; Szepesvári 2022), and Game Playing, where complex games like Go and StarCraft 2 are mastered by employing an interactive approach (Moerland et al. 2023 ; Silver et al. 2016).

Recent advancements within reinforcement learning include Deep Q-learning. This technique combines deep neural networks with reinforcement learning for improved performance and scalability (Kufel et al. 2023 ; Mnih et al. 2015). Multi-agent Reinforcement Learning is another recent development in reinforced learning. This framework enables multiple agents to learn and cooperate within complex environments (Lowe et al. 2017 ; Zhang et al. 2021).

The frontiers of ML are constantly expanding, witnessing the emergence of new categories and hybrid approaches. Notably, Explainable AI (XAI) is gaining traction, aiming to demystify complex models and foster trust and transparency (Belle & Papantonis 2021). Additionally, Federated learning addresses privacy concerns by enabling model training on decentralized data, enhancing generalizability and security (Li et al. 2020) Natural Language Processing

Natural Language Processing (NLP) is a rapidly evolving field within Artificial Intelligence (AI) focused on empowering computers to understand and manipulate human language in its diverse forms, including text and speech (Brown et al. 2020). This pursuit delves into the intricate workings of human language comprehension and usage, seeking to unlock the secrets of how we communicate effectively. By gleaning these insights, NLP researchers develop sophisticated tools and techniques that equip computers with the ability to handle natural language tasks like humans do. These tasks encompass a wide range, from sentiment analysis and text classification to speech recognition/synthesis and semantic analysis (Chowdhary & Chowdhary 2020).

The foundation of NLP lies in a rich confluence of disciplines, drawing upon expertise from computer science, information science, linguistics, artificial intelligence, robotics, and even psychology. This interdisciplinary approach fosters a holistic

understanding of language, enabling the development of robust and adaptable NLP models. The impact of NLP is readily apparent in our daily lives, with applications permeating diverse sectors. Companies leverage NLP capabilities to automate services like machine translation, facilitating global communication and knowledge exchange (Chowdhary & Chowdhary 2020). Additionally, NLP powers tasks like parts-of-speech tagging and resume parsing, streamlining processes and enhancing data analysis (Baviskar et al. 2021).

Looking beyond current applications, NLP holds immense potential for shaping the future of human-computer interaction. Advancements in conversational AI promise seamless and nuanced dialogue between humans and machines, blurring the lines between natural and artificial communication (Brown et al. 2020 ; Chowdhary & Chowdhary 2020). Moreover, NLP's influence is extending into fields like healthcare and education, offering innovative solutions for personalized medical diagnosis and interactive learning experiences. The future of NLP is brimming with exciting possibilities, driven by continuous research and advancements in areas like deep learning and neural networks. As NLP continues to evolve, we can expect even more profound breakthroughs that bridge the gap between machines and human communication, empowering us to interact with technology in a more natural and intuitive way.

2.8 TEXT CLASSIFICATION

Text classification, a cornerstone of NLP, revolves around automatically analysing textual data and assigning relevant categories based on its content. Also known as document classification, it leverages machine learning techniques to map predefined categories onto text documents (Rosquist 2021). This supervised learning approach finds applications in diverse areas such as spam filtering, search engine optimization, sentiment analysis, and product review mining (Anvar Shathik & Krishna Prasad 2020).

Since the early 1990s, the combination of machine learning and natural language processing (NLP) has attracted a lot of interest in the understanding of linguistic structures (McShane & Nirenburg 2021). Text classification tasks, a subfield within

NLP, can be categorized based on their reliance on machine learning or not, as well as their labelling strategy. In multi-class classification, each data point belongs exclusively to one predefined category, while multi-label classification treats each label as a separate binary classification problem, allowing data points to belong to multiple categories or none (Brask & Gellerman 2021 ; Read et al. 2011).

One early approach, popular in the 1980s, was the rule-based method. This involved crafting extensive sets of logical rules to classify texts based on keyword occurrences. For instance, a news article classifier might categorize texts containing "football," "team," or "score" as "sports." Notably, the Construe system developed by Hayes and Weinstein (1990) for Reuters news categorization exemplified this approach. While effective with well-crafted rules, this method suffers from limitations. As Sebastiani (2002) argues, rule-based systems require both a knowledge engineer and domain expert to build and maintain the rules, making them susceptible to changes in categorization needs. Recent research explored combining rule-based and probabilistic approaches using associative classification rules and a Naive Bayes classifier, achieving improved accuracy compared to standalone methods (Hadi et al. 2018). However, further comparisons with more advanced supervised techniques are needed.

Another prevalent approach, topic modelling, utilizes unsupervised learning to discover latent topics within unlabelled document collections. By clustering documents based on word occurrences, topic modelling identifies thematic clusters, each defined by a set of keywords. One such widely used technique is Latent Dirichlet Allocation (LDA) developed by Blei et al. (2003). LDA finds applications in various areas, such as uncovering key themes in unlabelled research papers (Jelodar et al. 2019) or analysing privacy policies (Sarne et al. 2019). While effective for latent topic discovery, topic modelling becomes less relevant when labelled data is readily available (Chauhan & Shah 2021).

Finally, the supervised approach leverages labelled data and supervised learning algorithms to train classifiers for assigning one or more category labels to a document. Supervised text classification typically involves two steps: data preprocessing and feature vector extraction, followed by classifier training using the extracted features

(Flores & Verschae 2022). This approach offers a robust and flexible solution for various text classification tasks.

2.9 TEXT PRE-PROCESSING

Text pre-processing is a critical initial stage in numerous NLP algorithms, significantly impacting the downstream performance of classification models (Muaad et al. 2022). It acts as a preparatory phase, transforming raw textual data into a format suitable for machine learning algorithms. Social media or used generated data contains slangs or pictograms (emojis/emoticon) as a mean for graphical expression of emotions which are particularly useful in reviewing effectiveness of product or service (Li et al. 2019). Therefore, the multi-step text-preprocessing phase lays the foundation for tasks like text classification, topic modelling, sentiment analysis and text summarization, ultimately influences model accuracy. Omitting any of these steps can compromise the effectiveness of these tasks.

The pre-processing pipeline typically involves a series of text cleansing operations, including:

1. Tokenization: Splitting sentences into individual words or meaningful units (e.g., n-grams) (Haryadi et al. 2022).

2. Stop-word removal: Eliminating common words that lack semantic content (e.g., articles, conjunctions) (Piter et al. 2021).

3. Lowercase conversion: Standardizing all letters to lowercase for consistent representation (Hickman et al. 2022).

4. Lemmatization: Reducing words to their base form (e.g., converting "running" to "run") (Hickman et al. 2022).

These operations aim to remove noise from the text, such as unnecessary words and characters (punctuation, special symbols). This not only reduces the dimensionality of the data, making it more manageable for machine learning algorithms, but also facilitates effective feature selection (Kowsari et al. 2019) and ultimately enhances classification accuracy (HaCohen-Kerner et al. 2020)

2.10 CLASSIFICATION ALGORITHMS

While there are many different ML algorithms, in this study, the focus was on 8 main models as follows.

2.10.1 Multinomial Naive Bayes

Multinomial Naive Bayes (MNB) is a powerful and effective tool for text classification. It excels in handling data represented as counts or rates, making it ideal for tasks like sentiment analysis and, movie review classification (Mtetwa et al. 2018). MNB is based on multinomial distribution, which calculated the probability of observing counts among multiple categories. This characteristic makes it particularly suited for features like word counts or frequencies in text classification setup (Mtetwa et al. 2018 ; Rana & Kolhe 2015).

MNB differs from its parent, Naive Bayes, in its handling of feature dependencies. While Naive Bayes assumes conditional independence between features, MNB treats them as independent despite potential underlying relationships facilitating efficient learning and prediction, especially for large datasets with bag-of-words features (Rahman & Akter 2019). However, MNB's assumption of conditional independence has potential impact on model accuracy, particularly when semantic relationships between words are crucial. Delving deeper, MNB utilizes the multinomial distribution to estimate the likelihood of observing specific word counts in each class (Riego & Villarba 2023) This estimation assumes independence between words within documents and disregards their context or position. Additionally, MNB assumes equal prior probabilities for all classes, which can be adjusted based on prior knowledge or data characteristics (Gawich & Alfonse 2022). To avoid zero probabilities during calculations, MNB often employs Laplace smoothing, adding a small pseudo-count to each word occurrence across all classes (Gawich & Alfonse 2022) This technique ensures smooth probability estimations and avoids numerical instability.

Recent MNB research aims to improve its performance by incorporating linguistic features like n-grams or TF-IDF to capture semantic relationships and improve accuracy (Sharifani et al. 2022). Additionally, research on semi-supervised

learning approaches utilizing both labelled and unlabelled data shows promising results in further enhancing MNB's effectiveness, particularly for large datasets with limited labelled examples (Abbas et al. 2019 ; Jiang et al. 2016).

2.10.2 Logistic Regression

Logistic regression is a widely used text classification tool, achieving superior results in diverse applications and fields (Harsha Kadam & Paniskaki 2020). Its effectiveness is evident in studies like Kotzé et al. (2020), where it achieved an impressive 0.899 accuracy in classifying violent events within WhatsApp messages, surpassing even SVM classifiers. While linear regression thrives with continuous response variables, logistic regression shines in the realm of categorical outputs, where standard methods falter (Cheng & Hüllermeier 2009 ; Wu 2018). Its ability to transform the mean response and model probabilities via a sigmoid function makes it ideal for tasks like sentiment analysis, where reviews are categorized as "positive" or "negative" (Gupta 2020). logistic regression also offers optimization and adaptation, such as identifying informative data points for labelling, reducing labelling costs and improving generalization performance (Thangaraj & Sivakami 2018). Additionally, kernel methods can transform data into higher dimensions, addressing imbalanced rare events data (Hajibabaee et al. 2021). (Yen et al. 2011). For large datasets with high dimensionality, logistic regression demonstrates scalability through N-gram smoothing techniques (Kanish Shah et al. 2020)

However, challenges exist in scenarios with numerous features and limited observations. Overfitting becomes a concern, as models memorize the training data instead of generalizing effectively (Zabor et al. 2022). Logistic regression exhibits an advantage over SVMs due to its ability to control model complexity through techniques like model selection (Occhipinti et al. 2022) Despite its popularity, ridge logistic regression faces limitations in large-scale settings, necessitating further exploration. Combining sparse solutions with ridge regression could address this issue by removing irrelevant features (Qin & Lou 2019). In the realm of natural language processing, logistic regression’s close relationship with neural networks further elevates its significance. Compared to Naive Bayes, a generative classifier, logistic regression

adopts a discriminative approach, providing advantages in interpretability and training simplicity (Solovyeva & Abdullah 2022). It performs well with linearly separable datasets but can lead to overfitting in certain scenarios, particularly with high number of features (Han 2020). This limitation confines its applicability to problems with discrete functions, as complex relationships are often beyond its reach.

2.10.3 Linear Support Vector Classification

Linear Support Vector Classification (Linear SVC) is a powerful algorithm for large-scale, multi-class classification tasks. Derived from the classic Support Vector Machine (SVM) algorithm, it uses linearity to maximize class margins through hyperplanes. (Gawich, 2022 #138}. This simplification unlocks a multitude of advantages. Firstly, Linear SVC offers superior scalability compared to its kernel-based SVM counterpart, particularly for large datasets. Linear computations significantly speed up training and prediction times (Joshi & Abdelfattah 2021) while also enhancing interpretability by allowing easier understanding of decision boundaries and feature importance, providing valuable insights into model behavior.

Beyond these core strengths, Linear SVC unlike standard SVMs offers flexibility allowing users to customize penalties like "squared_hinge" to enhance convergence speed and improve robustness to outliers (Gawich & Alfonse 2022). Furthermore, the ability to customize penalties, such as L2 regularization, empowers tailoring the model for specific tasks and controlling model complexity, preventing overfitting (Scott Zhang et al. 2019). For multi-class scenarios, Linear SVC employs the efficient "one-vs-the-rest" approach, building separate binary classifiers for each class against all others (Géron 2022). This strategy makes it a versatile tool across diverse domains, from text classification and image recognition to bioinformatics and financial forecasting.

However, its effectiveness depends on linear separability between classes. For complex non-linear relationships, kernel SVMs or other non-linear models might be more suitable. Additionally, interpretability can still be challenging with high-dimensional data, but feature importance analysis techniques can provide insight

(Géron 2022). Despite these limitations, Linear SVC stands as a potent and flexible workhorse for large-scale, multi-class classification problems.

2.10.4 Random Forest

Random Forest (RF), a powerful ensemble method, in machine learning algorithms, that uses multiple decision trees, to overcome the limitations in individual trees. Unlike Bagging, Random Forest delves deeper. Both utilize multiple decision trees, but their approaches diverge when choosing split points within each tree. Bagging simply samples data with replacement and builds trees using the entire feature set, while Random Forest randomly selects a smaller subset of features for each split, injecting diversity into the tree-building process (Géron 2022 ; Joshi & Abdelfattah 2021). This diversification reduces overfitting, a notorious weakness of decision trees, and strengthens the overall ensemble. By averaging predictions from diverse trees, Random Forest achieves exceptional accuracy on various datasets, often exceeding other popular algorithms like Support Vector Machines (Brask & Gellerman 2021). The ensemble nature makes Random Forest inherently robust to noise and outliers in the data, further bolstering its performance (Mesa-Jiménez et al. 2022). Unlike "black-box" models, Random Forest offers valuable insights into feature importance, allowing data scientists to understand which features drive the model's predictions (Harsha Kadam & Paniskaki 2020).

Random Forest's versatility shines across numerous domains including sentiment analysis, spam filtering, classifying textual data based on its content (Gupta 2020) and image categorization in healthcare and autonomous driving applications (Binkhonain 2021). Identifying anomalous patterns in financial transactions is crucial for fraud prevention, makes it a valuable tool in this domain (Tufail et al. 2023). However, ongoing research strives on addressing specific challenges associated with RF such as selecting optimal hyperparameters (Saad et al. 2021) understanding high-dimensional data,(Parmar et al. 2023) and exploring alternative ensemble methods that leverage different principles like boosting. , These challenges require careful consideration and domain-specific knowledge, as well as novel techniques to

understand the model's decision-making process potentially leading to further performance improvements in specific tasks (Shawkat et al. 2022).

2.10.5 Decision Tree

Decision trees are widely used supervised learning algorithm in machine learning (Brask & Gellerman 2021 ; Safavian & Landgrebe 1991)that breaks down complex decision-making processes into simpler, rule-based steps. They use tree-like structure to classify data points by tracing a path from the root node down to a leaf, with each internal node representing a decision based on a specific feature. The feature with the highest information gain, , is chosen as the parent node, while subsequent features are assigned to child nodes until reaching the final leaf nodes containing the predicted class labels (Aghaei et al. 2019) . Decision trees model offer several advantages, including training and prediction, , for large datasets (Brask & Gellerman 2021), a clear visual representation of decision rules used for classification, making them easier to understand and interpret compared to black-box models (Apallius de Vos 2023). As decision trees are robust to missing data and require minimal data preprocessing, it makes them a good choice for real-world applications (Apallius de Vos 2023).

However, certain limitations need to be considered. Decision trees can easily overfit, meaning they memorize the training data but struggle to generalize to unseen examples (Apallius de Vos 2023 ; Gupta 2020). This can be mitigated through techniques like pruning or ensemble methods. Simple decision trees may not be able to capture complex relationships within the data, leading to suboptimal performance for intricate tasks (Apallius de Vos 2023). Decision trees can be sensitive to noise and outliers, potentially leading to inaccurate predictions (Feofanov 2021). Various architectures and splitting criteria exist, with its Gini impurity measure, can be used to tailor the algorithm to specific problems (Feofanov 2021). Additionally, combining multiple decision trees through ensemble methods like boosting or bagging can significantly improve accuracy and reduce overfitting (González et al. 2020).

2.10.6 Extra Tree

Extra Tree Classifier (ETC) is a tree-based ensemble learning algorithm gaining traction for its effectiveness and simplicity (Barhoom et al. 2022). Similar to its cousin, Random Forest (RF), ETC combines multiple decision trees for improved accuracy and robustness. However, unlike RF, ETC boasts distinct characteristics that set it apart. One key difference lies in the training data used for each tree. While RF utilizes random bootstrapping, drawing samples with replacement from the original dataset, ETC employs the entire training set for each individual tree (Darbanian et al. 2020) (Geurts et al. 2006). This leads to a higher variance in the individual trees compared to RF, but also potentially reduces bias, contributing to better generalization (Bhati & Rai 2020). Another distinguishing feature of ETC is its approach to split point selection within the decision trees. Instead of relying on a specific optimality criterion like the Gini index used in RF (El Bouchefry & de Souza 2020), ETC randomly selects the best split point for each feature at each node (Darbanian et al. 2020)This element of randomness further contributes to the diversity of the trees and helps avoid overfitting (Binkhonain 2021).

Despite these differences, ETC shares some key strengths with RF. Its ensemble nature makes it robust to noise and outliers in the data (Rustam et al. 2021). Furthermore, the individual trees are relatively simple and interpretable, allowing for easier understanding of the model's decision-making process (Bhati & Rai 2020). Additionally, feature importance can be derived from ETC forests, providing insights into the most influential features for the classification task (Saad et al. 2021).

2.10.7 Extreme Gradient Boosting

XGBoost, short for Extreme Gradient Boosting, has emerged as a dominant force in the machine learning landscape (Li et al. 2022). Built upon the foundation of gradient boosting, it combines multiple weak decision trees into a powerful ensemble, achieving unparalleled accuracy and efficiency. Its prowess extends to various domains, including medicine, finance, and even load forecasting. Several key features contribute to XGBoost's advantage over other models. It prevents overfitting through LASSO (Least Absolute Shrinkage and Selection Operator) and Ridge penalties, ensuring models generalize well to unseen data. Missing values are handled intelligently, and different

sparsity patterns are tackled efficiently. The algorithm eliminates the need for manual cross-validation, simplifying the hyperparameter tuning. Training is significantly accelerated by leveraging parallel processing, making XGBoost accessible even on modest hardware (Mesa-Jiménez et al. 2022).

In the realm of medicine, XGBoost shines in diverse tasks like disease diagnosis, prognosis prediction, and treatment selection (Kufel et al. 2023). Its ability to handle both structured and unstructured medical data, such as clinical notes and images, makes it an invaluable tool for healthcare professionals (Inoue et al. 2020 ; Ramakrishnan & Ganapathy 2022 ; Wang et al. 2022). Under the hood, XGBoost iteratively builds decision trees, focusing on minimizing the loss function with each step (Occhipinti et al. 2022). Splitting criteria are based on CART principles, while the least square loss and logarithmic function are commonly used (Qi 2020). This continuous refinement leads to a progressively more accurate model. While sharing core principles with gradient boosting, XGBoost excels through its implementation details. Regularization techniques effectively control tree complexity, leading to improved performance (Piter et al. 2021). Hyperparameter tuning plays a crucial role in optimizing the learning process, and XGBoost offers ample flexibility in this regard (Piter et al. 2021). Its ability to handle real-world problems, both small and large-scale, with minimal resources is a testament to its elegance and efficiency (Afifah et al. 2021).

2.10.8 K-Nearest Neighbours

K-Nearest Neighbours (KNN) is a fundamental algorithm in machine learning known for its simplicity, versatility, and interpretability (Gasparetto et al. 2022). Its core principle is that similar data points reside in close proximity within a feature space. KNN meticulously searches the training set for the K nearest neighbours which are the data points that bear the closest resemblance to the newcomer. This kinship is quantified using distance metrics like Euclidean distance, which measures the "straight-line" separation between points. However, the choice of metric isn't a one-size-fits-all affair. Depending on the data's characteristics, Manhattan, Minkowski, or even Hamming distances might prove more suitable travel companions for KNN's exploration (Chen et al. 2020). Once the K nearest neighbours have been identified, KNN then determines

the fate of the newcomer by taking a majority vote to assign the most frequent class label among the neighbours to the new instance. This democratic approach leverages the inherent structure of the data, where similar points tend to cluster together, making it particularly effective for tasks like text classification, where KNN excels in handling feature-extracted text data like TF-IDF vectors (Brask & Gellerman 2021). KNN can also tackle regression tasks, where the goal is to predict continuous values. In this scenario, KNN consults its nearest neighbours once again, but instead of taking a majority vote, it averages their target variables to arrive at a predicted value for the new instance (Tufail et al. 2023). This makes KNN a versatile tool, capable of tackling diverse tasks across various domains.

However, one hurdle in KNN lies in finding the optimal value for K, the number of neighbours considered. Choosing the right K is akin to striking a delicate balance – too few neighbours can lead to overfitting, while too many can result in underfitting. Balancing factors like data size, noise levels, and desired model complexity requires careful consideration (Rahman & Akter 2019). Another challenge KNN faces is its computational cost. As the number of data points in the training set grows, so does the computational burden of calculating distances to all of them during classification. This "curse of dimensionality" can make KNN less efficient for large datasets, prompting researchers to explore techniques like dimensionality reduction to mitigate this issue (Haryadi & Mandala 2019). However, KNN's strengths includes its ease of use, interpretability, and non-parametric nature, meaning it doesn't require strong assumptions about the underlying data distribution, making it a valuable tool for practitioners across various fields. From text classification and sentiment analysis to recommender systems and anomaly detection, KNN continues to be a relevant and powerful algorithm in the ever-evolving landscape of machine learning.

2.11 FEATURE EXTRACTORS

Analysing textual data for text classification requires transforming it into a format suitable for machine learning algorithms. This crucial step, known as feature extraction, involves identifying and representing the most informative words within the text. Two main approaches dominate this process: word embedding and term weighting. Methods

like Word2Vec capture the semantic relationships between words by assigning them N-dimensional vectors. Words with similar meanings have closer vector representations in this "semantic space"(Pilehvar & Camacho-Collados 2020). This allows the model to learn and utilize complex relationships between words, improving its ability to understand the overall sentiment of a text. Techniques like TF-IDF assign weights to each word in a document, reflecting their importance in distinguishing that document from other Words that appear frequently within a document but rarely elsewhere are given higher weights, as they are more likely to be indicative of the document's specific content. The effectiveness of these weighting schemes is evaluated based on recall (the proportion of relevant terms extracted) and precision (the proportion of irrelevant terms excluded (Ahuja et al. 2019).

It's important to distinguish between feature extraction and feature selection. While feature extraction transforms existing features into a reduced set, feature selection involves choosing a subset of these features that best represent the data for analysis (Tiruneh & Fayek 2019). This process helps to reduce redundancy and improve the efficiency of the machine learning model. The choice of feature extraction technique can also be guided by the type of vocabulary analysis desired. Closed vocabulary text mining relies on pre-defined dictionaries to identify relevant words within the text. This approach is particularly useful in organizational research, where specific constructs and themes are often well-defined (Hickman et al. 2022).

Garg (2021)utilized two common text-to-vector methods for feature extraction, bag-of-words and TF-IDF. In addition to these automated techniques, this study also employed manual feature engineering. This involved extracting specific features from the review data that were deemed relevant for sentiment analysis but might not have been captured by the automated methods. These manually extracted features were then used to create a separate "manual feature" model (Garg 2021). It's important to remember that proper data preprocessing is essential before employing any feature extraction technique or building sentiment analysis models (Garg 2021).

2.11.1 Bag Of Words

Despite rapid advancements in Natural Language Processing (NLP), the bag-of-words (BoW) model remains a cornerstone technique for text classification. Its enduring appeal stems from its simplicity, efficiency, and effectiveness, particularly for classifying short texts where word order plays a less crucial role (Juluru et al. 2021). At its core, BoW represents a document as an unordered collection of words, disregarding grammatical dependencies and word order. Each word acts as an independent feature, quantified by its frequency within the document or simply by its presence/absence (Abubakar et al. 2022) This straightforward approach makes BoW computationally efficient and readily implementable. BoW shines in its ability to effectively handle short texts where word order carries less semantic weight. This makes it well-suited for tasks like sentiment analysis of tweets or classifying short news articles(Lee et al. 2023).. However, BoW does face limitations. By discarding word order and grammar, it loses valuable semantic information inherent in the text's structure(Abubakar et al. 2022). Additionally, frequent words like "the" or "a" can add noise, potentially impacting classification accuracy (HaCohen-Kerner et al. 2020). Finally, the high dimensionality of the feature space can lead to sparsity issues, posing challenges for certain machine learning algorithms (Lee et al. 2023). To address these limitations, several strategies can be employed. Preprocessing steps like tokenization, stop word removal, and stemming/lemmatization can significantly improve BoW's performance (HaCohen-Kerner et al. 2020)

2.11.2 Term Frequency-Inverse Document Frequency

The quest for accurate and efficient information retrieval remains a cornerstone of various computational linguistics tasks. Among the plethora of techniques deployed, Term Frequency-Inverse Document Frequency (TF-IDF) stands out as a versatile and robust approach for quantifying word relevance in documents. This review delves into the intricate workings of TF-IDF, exploring its theoretical underpinnings, applications, and challenges through the lens of relevant literature. Brask and Gellerman (2021) highlighted the inherent tension in text classification between recall (finding all relevant documents) and precision (retrieving only relevant documents). Salton and Buckley (1988) argued that term weighting schemes like TF-IDF play a crucial role in striking

this delicate balance. TF-IDF achieves this by considering both the frequency of a word within a document (TF) and its distribution across the entire corpus (IDF). Terms that appear frequently within a document are deemed more important for its content (Qi 2020). utilizes this principle in hotel review classification, where words like "clean" or "comfortable" carry higher weights as they are prevalent in positive reviews. However, a common word like "the" might appear frequently across many documents, diluting its significance for any specific content. This is where IDF comes in. IDF penalizes terms that occur in many documents, promoting specificity and discriminative power (Gupta 2020 ; Wu et al. 2008).

The versatility of TF-IDF extends beyond the realm of text classification. Harsha Kadam and Paniskaki (2020) leverage it for multi-label email classification, while Haryadi et al. (2022) employed it to classify drug effectiveness based on patient reviews. It finds applications in sentiment analysis as well, with Parmar et al. (2023) employing it for drug quality classification based on review sentiment. Despite its numerous strengths, TF-IDF is not without its limitations. Computational considerations arise when dealing with large datasets, as calculating TF-IDF weights can be resource-intensive (Gupta 2020). Additionally, tuning the TF-IDF parameters for optimal performance requires careful calibration depending on the specific task and data characteristics (Brask & Gellerman 2021). Furthermore, handling stop words effectively is crucial, as their high frequency can skew the weightings and impact accuracy (Haque et al. 2023).

2.12 CONFUSION MATRIX

Confusion matrix is a tabular representation of a model's performance, visualizing correct and incorrect predictions for each class (Sami et al. 2021)

The key elements in a confusion matrix are:

1. True Positives (TP): Correctly predicted positives
2. True Negatives (TN): Correctly predicted negatives
3. False Positives (FP): Incorrectly predicted positives
4. False Negatives (FN): Incorrectly predicted negatives (Zope et al. 2022)

2.13 PERFORMANCE METRICS

Accuracy is the most intuitive metric, measuring the percentage of correctly predicted observations (Joshi & Abdelfattah 2021). However, it can be misleading for imbalanced datasets where one class dominates (Mtetwa et al. 2018). Considering the significant class imbalance, we chose not to rely on accuracy for model evaluation.

$$Accuracy = (TP + TN)/Total\ Observations$$

(Sami et al. 2021)

Precision (also called positive predictive value) measures the proportion of true positives among predicted positives (Zope et al. 2022).It emphasizes how many of the predicted positives are actually correct.

$$Precision\ = TP\ /\ (TP + FP)$$

(Zope et al. 2022)

Recall (also called sensitivity) measures the proportion of true positives correctly identified among all actual positives (Sami et al. 2021). It focuses on how many of the actual positives were correctly captured.

$$Recall\ = TP\ /\ (TP + FN)$$

(Sami et al. 2021)

F1-score is the harmonic mean of precision and recall, balancing both measures to provide a more robust evaluation of performance, especially with imbalanced data (Joshi & Abdelfattah 2021).It's often preferred in scenarios where both precision and recall are important, and where the cost of false positives and false negatives are different (Mtetwa et al. 2018). Therefore, for the purpose of this study, F1-score is the preferred performance metric to assess the performance of the machine learning models.

$$F1\ score\ = 2 \times (Precision \times Recall)$$

(Joshi & Abdelfattah 2021)

Time – the computational speed of each algorithm is also measured in seconds to assess runtime efficiency of the models.

2.14 RELATED WORK

Abbas et al. (2019) reported MNB achieved good performance for sentiment analysis, highlighting its efficiency for classifying short text with categorical features while Hadi et al. (2018) reported combining MNB with rule-based classification improved accuracy in some cases, suggesting potential for hybrid approaches. Meanwhile Kibriya et al. (2005) reported that MNB provided competitive performance compared to other algorithms, showcasing its effectiveness for general text classification. Generally, these past studies highlighted the MNB as an efficient and interpretable model that was robust to feature scaling and handles categorical features well. It also highlighted the challenges associated with MNB, such as its characteristic where it assumes independence of features (may not hold for complex data), is sensitive to priors, and is limited to categorical features.

Joshi and Abdelfattah (2021) reported that the linear SVC outperformed other models for multi-class text classification of drug reviews, demonstrating its versatility and accuracy while Mesa-Jiménez et al. (2022) reported that XGB proved robust to noise and outliers in building management system text data. Joshi and Abdelfattah (2021) compared six machine learning models for classifying online drug reviews by medical condition. Decision trees achieved an accuracy of 78.6%, lower than Linear Support Vector Machines (84.2%) but higher than Multinomial Naïve Bayes (75.3%).

Mtetwa et al. (2018) studied the effects of feature extraction techniques in machine learning models for movie review classifications. Different combinations of text representation methods and machine learning classifiers were evaluated for their performance (Table 2.1). The study found that using tf-idf with a Support Vector Machine (SVM) and bigrams with a Multinomial Naive Bayes (MNB) classifier produced the best results, achieving a score of 0.88 besides also highlighting that the success of a good classifier depended on the feature extraction technique and the machine learning model (Mtetwa et al. 2018). Mtetwa et al. (2018) found RF achieved high accuracy for movie review classification, highlighting its effectiveness for sentiment analysis. RF generally was highly accurate, robust to noise and outliers, handles mixed data types, and provided feature importance insights but it can be

computationally expensive, less interpretable than MNB, and was prone to overfitting if not tuned properly.

Anvekar (2020) investigated classification of online patient reviews based on effectiveness (drugs.com and WebMD.com dataset) using machine learning techniques such as RF, KNN, SVM (Support Vector Machine) and XGBoost classifier. The feature extraction techniques employed were Count Vectorizer (BOW) and TF-IDF with N-gram approach (unigram and bigram). In this study (Anvekar 2020), XGBoost outperformed other models when paired with unigram model and tf-idf vectorization achieving an F1-score of 0.79.

Kwon et al. (2018) used the Binary BOW, Count BOW and tf-idf feature vectors to compare the performance of classifiers in malware detection. In this study, DT, KNN, MLP, MNB and SVM were used as classifiers. Kwon et al. (2018) recognized both the Multilayer Perceptron (MLP) and K-Nearest Neighbours (KNN) algorithms as top classifiers for this dataset. Their consistent performance in terms of AUC score and accuracy stood out regardless of data imbalance and diverse feature extraction techniques (Binary BOW, Count BOW, and TF-IDF).

Bolukbasi et al. (2016) reported that Linear SVC showed promise in reducing bias in word embeddings for sentiment analysis while Novakovic and Veljovic (2011) observed that it performed well for medical diagnosis classification, suggesting the potential of Linear SVC for binary tasks with structured data. A study by Solovyeva and Abdullah (2022) found that Linear SVC achieved good accuracy on various text classification tasks, showcasing its versatility. These studies found that Linear SVC was efficient for linear data, it was highly interpretable, and handled binary classification well. Its performance however can degrade with complex non-linear data, and it may require feature scaling, making it less flexible than Random Forest.

Extreme Gradient Boosting (XGBoost) has emerged as a powerful and versatile algorithm for text classification tasks. Its effectiveness lies in its ability to handle complex relationships between features, leading to highly accurate and interpretable models. Qi (2020) employed XGBoost to classify theft crime data based on textual

descriptions. They achieved an accuracy of 93.8%, outperforming other algorithms like K-Nearest Neighbours, Naïve Bayes, and Support Vector Machines. This suggested XGBoost's capability in handling intricate textual patterns for crime classification. Meanwhile, Hendrawan et al. (2022) compared XGBoost with Naïve Bayes for sentiment classification of online product reviews. They found XGBoost to achieve superior accuracy (F1-score: 0.941) compared to Naïve Bayes (F1-score: 0.915) highlighting XGBoost's effectiveness in capturing sentiment nuances within textual data. Piter et al. (2021) utilized XGBoost for multi-label classification of scientific conference activity information text. They obtained an average hamming score of 79.52% and an F1 score of 85.88%, demonstrating XGBoost's proficiency in handling multi-faceted textual data with diverse labels while Haumahu et al. (2021) implemented XGBoost for classifying fake news articles in Indonesian. They achieved an accuracy of 89% with a precision of 90% and recall of 80%. These findings collectively paint a promising picture of XGBoost's efficacy in text classification. XGBoost consistently delivered high accuracy across various text classification tasks, outperforming other algorithms in several cases. The algorithm adapted well to different text data types, including crime descriptions, product reviews, conference information, and news articles. XGBoost effectively handled tasks with multiple labels associated with each text data point, as demonstrated in the scientific conference activity classification. However, training XGBoost models can be computationally expensive compared to simpler algorithms like Naïve Bayes and XGBoost required careful hyperparameter tuning to achieve optimal performance, which can be challenging for non-experts.

Past studies reported that while decision trees can be effective for text classification, their performance can be improved by using techniques like feature selection, pre-processing, and ensemble methods. Decision trees are often used as a baseline model for comparison with other algorithms due to their simplicity and interpretability. The accuracy of decision trees for text classification varied depending on the dataset and task. DT was able achieve competitive results, especially when combined with other methods. Rahman and Akter (2019) compared decision trees, K-Nearest Neighbours, and Multinomial Naïve Bayes for topic classification on news articles. Decision trees achieved an accuracy of 88.4%, lower than Multinomial Naïve Bayes (91.8%) but higher than K-Nearest Neighbours (83.3%) while Thangaraj and

Sivakami (2018) discussed decision trees in the context of AI for text classification. They mention decision trees' popularity due to their simplicity and interpretability.

A Covid-19 case study by Almomany et al. (2022) proposed an optimized KNN classification algorithm using the DCT-KNN approach on the Intel FPGA platform. The study found that this approach significantly improves the execution time of KNN classification compared to traditional CPU-based implementations, achieving 44 times faster execution on the Intel De5a-net Arria-10 device. While Wen et al. (2022) proposed a weighted ML-KNN approach that used rough sets to identify and address the uncertainty in samples. This approach was found to be effective in multi-label classification tasks, achieving improvements in F1 score compared to several state-of-the-art multi-label classification methods. Zhao et al. (2023) meanwhile explored the use of KNN in conjunction with prompt learning for biomedical document relation extraction. The study found that this approach can improve the learning of document semantic information and achieve improvements in relation F1 score compared to other methods. Huang et al. (2023) investigated the use of ML-KNN for multi-label classification of social media users. The study found that this approach can effectively capture the multi-faceted nature of social media users and outperform existing single-label classification methods. Adhikary and Banerjee (2023) introduced a novel distributed KNN algorithm called Distributed Nearest Hash (DNH) that utilized hash maps and primary key clustering to achieve near-real-time scalability and fast prediction times. The study found that DNH can be 25% faster than state-of-the-art distributed KNN algorithms.

Solovyeva and Abdullah (2022) explored various machine learning algorithms for text classification, including logistic regression. They found that logistic regression performs well on smaller datasets but can struggle with complex, non-linear data. The study by Hassan et al. (2022) compared the performance of different machine learning algorithms, including logistic regression, on two text classification tasks and found that logistic regression achieved competitive accuracy on both tasks, outperforming other models in some cases. Meanwhile, Mesa-Jiménez et al. (2022) investigated the use of machine learning for classifying sensor points in building management systems, comparing several algorithms, including logistic regression, and found that XGBoost

achieved the best performance. However, logistic regression still offered a viable option with good accuracy and efficiency. Overall, these studies suggest that logistic regression is a valuable tool for text classification, particularly for smaller datasets or when computational resources are limited. While it may not always be the most powerful algorithm, it offers a good balance of accuracy, efficiency, and interpretability. Logistic regression is often used as a baseline model for comparison with other, more complex algorithms. Its simplicity and interpretability make it a good choice for understanding the relationships between features and text categories. Logistic regression can be effective for tasks where the data is well-structured and the relationships between features and categories are linear.

Saad et al. (2021) used Extra Trees as one of five machine learning models to classify sentiment in drug reviews, reporting that while Extra Trees performed well, it was not as well as the best-performing model, Logistic Regression with TF-IDF features. Umer et al. (2022) used Extra Trees as part of an ensemble model to classify sentiment in COVID-19 tweets and found that the ensemble model performed better than Extra Trees alone, and that Extra Trees with TF-IDF features achieved the best accuracy among the individual models used in the ensemble. Overall, these findings suggest that Extra Trees can be a good option for text classification, but it may not always be the best-performing model. It can be effective when used in conjunction with other models in an ensemble, and its performance can be improved by using TF-IDF features. Uçar et al. (2020) highlighted the critical role of the sampling method in determining the success of machine learning models, particularly during the training and testing phases. The study also mentioned that training size less than 50% is not favourable as it has negative implications on the results. A summary of related reviewed are as in Table 2.1 below.

In this chapter, the various machines learning approaches and components were reviewed. Past studies that used the machine learning models and techniques employed in the current study were reviewed and discussed to give a background knowledge of machine learning models in drug review data. In the next chapter, the methodology employed for the classification of conditions for the drug review data are stated in a sequence of process.

Table 2.1 Summary of Related Work

No	Past Studies	Task	Dataset	Method	Findings / Conclusion	Gaps
1	Joshi and Abdelfattah (2021)	Multi-class Text Classification Using Machine Learning Models for Online Drug Reviews	Drugs.com & Druglib.com	• MNB, DT, RF, ET, LR, Linear SVC • random split 80/20 • TF-IDF • hyperparameter tuning (gridsearch, randomized search) • F1-score, time	• F1-score Linear SVC - 0.8825 (highest) • Time for Linear SVC and MNB was the shortest (less than a minute) • Others 51, 353,380,490 and 1003 seconds (LR, RF, DT, ET)	• Text preprocessing – beautiful soup • Only six machine machine learning algorithms considered • Effect of split was not considered • Effect of feature extractors was not considered
2	Uddin et al. (2022)	Drug Sentiment Analysis using Machine Learning	drug review dataset	• Binary classification (effective/not effective) • NB, RF, SVC, MLP • Multiclass classification – linear SVC (highly, considerable, moderately, marginally, ineffective) • Sampling method unknown	• Accuracy % RF (94.06) MLP (86.82), SVC (88.63), NB (88.57) • Multiclass classification, linear SVC showed promising result	• Unclear sampling methods and feature extraction methods • Only four ML methods considered

to be continued…

…continuation

3	Mtetwa et al. (2018)	Feature Extraction and Classification of Movie Reviews	Stanford University's ACL IMDB movie review dataset	• Binary sentiment classification • SVM, MNB, RF • TF-IDF, Bigrams, counter vector (BOW), • Accuracy, F1-score, precision, recall • Sampling method was not stated	• MNB + bigrams achieved highest F1-score	• Effect of other machine learning algorithms were not experimented • Other splits were not considered
4	Qi (2020)	The Text Classification of Theft Crime Based on TF-IDF and XGBoost Model	theft crime data	• XGB, KNN, NB, SVM, GBDT , multilabel • TF-IDF • accuracy, recall, F1-score • Sampling method – 80/20 split	• XGBoost -best training model • F1-score -96.6% • lowest SVM -80.8%	• Only five classifiers and one feature extractor considered
5	Bangyal et al. (2021)	Detection of Fake News Text Classification on COVID-19 Using Deep Learning Approaches	COVID Fake News Datase	• SVM, LR, NB, • Adaboost, K-NN, • DT, RF, MLP, • CNN, RNN, • LSTM, GRU • TF-IDF, • 80/20 split	• Bi-LSTM and CNN perform better than other classifiers (accuracy -97%, execution time, nonsensitive to outliers, reduction of noise) • ML models with highest accuracy 97% are (K-NN, MLP, RF).	• Effect of other feature extractors were not considered • Effect of other sampling methods were not considered

to be continued…

…continuation

6	(Garg 2021)	Drug Recommendation System based on Sentiment Analysis of Drug Reviews using Machine Learning	Drug Review Dataset from (Drugs.com)	• LR, MNB, Stochastic Gradient descent, Linear SVC, Perceptron, Ridge Classifier experimented with TF-IDF and BOW • DT, RF, LGBM,CatBoost Classifier on word2vec • random split 75/25 • SMOTE • precision, Recall, F1-score, accuracy	• Linear SVC with TF-IDF outperforms all other classifiers -93% accuracy.	• Effect of other sampling methods were not considered • N-gram feature extractor were not considered
7	Ling (2023)	Bio+ Clinical BERT, BERT Base, and CNN Performance Comparison for Predicting Drug-Review Satisfaction.	UCI ML Drug Review dataset	• classifying patients' drug review sentiment • Classifier: BERT, Bio-Clinical BERT, CNN, word2vec • Sampling method not mentioned • Parameter – Precision, Recall, F1-score	• F1-score Bio + Clinical BERT is 0.81	• Conventional classifiers were not considered • Only one feature extractor experimented
8	Piter et al. (2021)	multi-label classification of scientific conference activity information text	scientific conference activities on the internet.	• XGB • hyperparameter test analysis • TF-IDF, word2vec, • cross validation • precision, recall hamming score • Multi-label classification	• F1 score of 85.88%, • demonstrating XGBoost's proficiency in handling multi-faceted textual data with diverse labels	• Only one classifier was selected • Only 2 types of classifiers selected • Effect of random split sampling were not considered

to be continued…

…continuation

9	Parmar et al. (2023)	Drug Quality Classification Using Sentiment Analysis of Drug Reviews	Drug Review dataset from (Drugs.com)	• NB, LR, perceptron, ridge classifier, DT, RF (only ML classsification method with quick prediction and training cycle were selected) • BOW, TF-IDF, Countvectorizer, • K-fold cross splits, • SMOTE	• Selected ML: MNB, LR, DTC, RFC • Best F1-score- RF: • 094(TF-IDF), 0.93(countvectorizer) • worst F1-score: MNB: • 0.79 (TF-IDF), 0.78 (countvectorizer)	• Only four classifiers were selected • Effect of random split sampling method were not experimented
10	Hendrawan et al. (2022)	Comparison of Naïve Bayes Algorithm and XGBoost on Local Product Review Text Classification	local product review	• XGB, NB • word2vec, TF-IDF, • sampling 80/20	• word2vec + XGBoost F1-score 0.941, • TF-IDF +XGBoost (0.940). • NB+TF-IDF (0.9), NB+word2vec (0.9). • Xgboost classify unbalanced data better than NB	• Only 2 classifiers and 2 types of feature vectors considered • Other sampling methods were not experimened
11	(Haumahu et al. 2021)	Classifying fake news articles in Indonesian	indonesian news websites	• XGB • TF-IDF • cross validation • precision, recall, F1-score, accuracy,	• XGB-accuracy and F1-score 92%	• Only one classifier and one feature extractor were selected • No random sampling method studied

to be continued…

…continuation

12	Rahman and Akter (2019)	Topic Classification from Text Using Decision Tree, K-NN and Multinomial Naïve Bayes	Amazon's product review corpus	• Decision trees, KNearest Neighbours, and Multinomial Naïve Bayes • 75/25, • multiclass • tf-idf, count feature	• F1-score (Tf-IDF): • DT - 0.79, • MNB - 0.918 • KNN - 0.826 • F1-Score (count-feature) • DT: 0.798 • K-NN: 0.590 • NB: 0.905	• Only four classifiers studied • Only one sampling method was considered • Only 2 types of feature extractor considered
13	Abbas et al. (2019)	Multinomial Naive Bayes Classification Model for Sentiment Analysis	dataset of movie reviews	• MNB, TF-IDF	• Accuracy - 90%	• sampling methods are not stated • only one classifier and feature extraction used
14	Mesa-Jiménez et al. (2022)	Machine learning for text classification in building management systems	Building Management Systems data	• multi-class • LogReg, RF, XGB, MNB, Linear SVC • bag-of-words	• XGBoost performs better than the other four except MNB which shows slightly worse results.	• sampling methods were not stated • only one feature technique used
15	Novakovic and Veljovic (2011)	C-Support Vector Classification: Selection of Kernel and Parameters in Medical Diagnosis	Nine datasets from UCI repository database used to compare results of classification with C-SVC and different kernels and parameters in medical diagnosis.	• Classifier - C-Support Vector Classifier	• after optimization of parameters, resuts proved that classification accuracy is consistent for all kernels.	• single class classification

to be continued…

…continuation

16	Solovyeva and Abdullah (2022)	Comparison of Different Machine Learning Approaches to Text Classification	comparative analysis of the main machine learning algorithms to classify texts	• Comparative analysis of: • Word tokenization vs word embedding • NB, LR, Linear SVC, Deep Neural Networks	• Naive Bayes algorithm performs even with limited training datasets • Logistic regression can classify unknown text instantly and perform well when the dataset is linearly separable. sets. However, the assumption of linearity between It has good accuracy for many textual data the dependent and independent variables is a limitation of the algorithm • Support vector machine is stable and efficient in spaces with high machine approach is not very effective for huge datasets. • Deep neural networks efficiently solve many problems like classification, regression, function approximation, clustering, and others and deal with different kinds of data. But they demand a significant load of data to achieve enhanced results than the previous methods.	• Comparative analysis done only on 4 ML models • No comparative model on sampling methods • N-gram method was not analysed • No data on accuracy/F1-score for comparative analysis done

to be continued…

…continuation

17	Hassan et al. (2022)	Analytics of machine learning-based algorithms for text classification	IMDB and SPAM dataset	• comparative analysis of text classification • Support Vector Machine (SVM), k-Nearest Neighbour (k-NN), Logistic Regression (LR), Multinomial Naïve Bayes (MNB), and Random Forest (RF) • Tf-IDF, Bag-of-words • accuracy, precision, recall and f1- score.	• k-NN model outperforms the other models in the Spam dataset with an accuracy of 98.5%. • LR model surpasses the other models in the IMDB dataset with an accuracy of 85.8% • TextBlob tends to show better results for annotation drug reviews	• Drug review dataset was not analysed • sampling method was not stated
18	Saad et al. (2021)	Determining the Efficiency of Drugs Under Special Conditions from Users' Reviews on Healthcare Web Forums	sentiment analysis on drug reviews	• learning-based and lexicon-based methods of sentiment analysis • three sentiment lexicons including AFFIN, TextBlob, and VADER. • three feature engineering approaches TF, TF-IDF, and TF U TF-IDF • logistic regression (LR), random forest (RF), extra tree classifier (ET), AdaBoost classifier (AB), and multilayer perceptron (MP) • hyperparameter setting done • accuracy, precision, recall and f1- score.	• MLP and LR showed good performance when trained on TF-IDF and TF U TF-IDF with TextBlob sentiments	• paper focuses on sentiment analysis of drug review dataset • only 5 ML models used • sampling method was not stated

to be continued…

…continuation

19	Umer et al. (2022)	ETCNN: Extra Tree and Convolutional Neural Network-based Ensemble Model for COVID-19 Tweets Sentiment Classification	COVID-19 tweet dataset from IEEE dataport	Random Forest (RF), Extra Tree (ET), Gradient Boosting Machine (GBM), Logistic Regression (LR), Naive Bayes (NB), Stochastic Gradient (SG) and Voting Classifier (VC), ET-CNN (combines ET and CNN) TextBlob, VADER Tf-IDF, Word2vec hyperparameter setting done	ET model shows the best performance among the machine learning models when TF-IDF features are used.	sampling method was not stated no multiclass text classification

CHAPTER III

METHODOLOGY

3.1 INTRODUCTION

The use of eight different machine learning models and four feature extraction techniques were employed to classify top ten common disease conditions for the drug review data. This study's research methodology are further explained following the steps as shown in Figure 3.1.

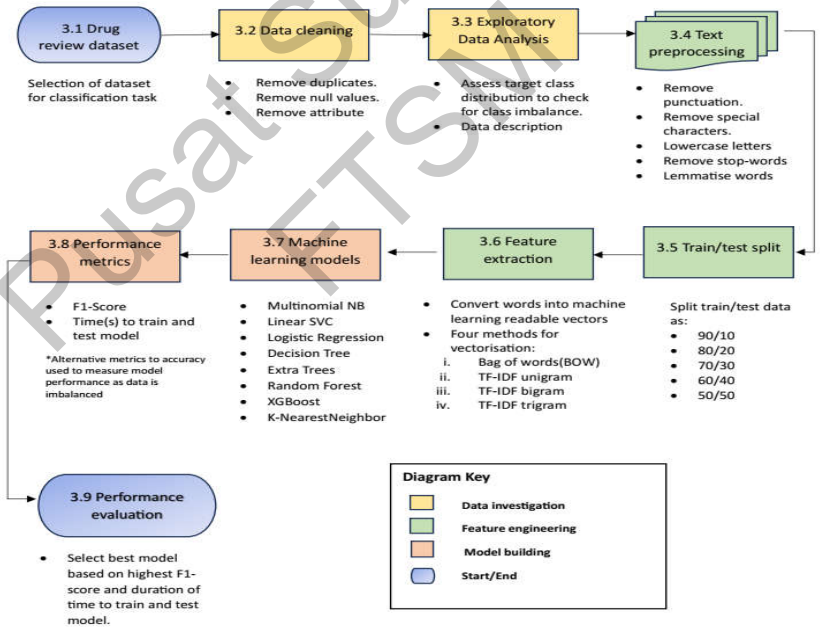

Figure 3.1 Flowchart of drug review classification using machine learning models

3.2 DRUG REVIEW DATASET

The two drug review datasets namely Drugs.com (Surya Kallumadi 2018) and Druglib.com (Surya Kallumadi 2018) were obtained from UCI Machine Learning Repository. The dataset includes reviews of different medications for various medical

conditions, along with user ratings reflecting overall satisfaction and perceived effectiveness. The dataset was chosen as it contains information on side-effect profile and feedback on effectiveness of a drug which was like an adverse drug reaction report. This dataset has been used by other studies as reviewed in Table 2.1. The dataset comprised of 219,206 patient reviews were obtained from UCI Machine Learning Repository, with 215,063 sourced from Drugs.com and the remaining 4,143 reviews obtained from Druglib.com. The drug review data were obtained by crawling online pharmaceutical review sites. The raw sample data of both datasets (druglib.com and drugs.com) are shown in Figure 3.3 and 3.2. Both datasets contained the same conditions. The list of attributes with its corresponding data type are shown in Tables 3.1 and 3.2.

Table 3.1 List of attributes with their types and descriptions for drug review (druglib.com) dataset

Attributes name	Type	Attribute description
Unnamed: 0	numerical	Unique number assigned for each review
urlDrugname	categorical	Name of drug Excluded from classification task
condition	categorical	Name of condition Class attribute
benefitsReview	text	Patient review on benefits Excluded from classification task
sideEffectsReview	text	Patient review on side-effects Excluded from classification task
commentsReview	text	Overall patient comment (includes benefit and side-effects)
rating	numerical	10-star patient rating (1-10) Excluded from classification task
sideEffects	categorical	5 step side effect rating (Mild side effects, no side effects, moderate side effects, severe side effects, extremely severe side effects) Excluded from classification task
		to be continued…
…continuation		
effectiveness	categorical	5 step effectiveness rating (Highly effective, considerably effective, moderately effective, ineffective, marginally effective) Excluded from classification task

```
df_lib.head()
```

	Unnamed: 0	urlDrugName	rating	effectiveness	sideEffects	condition	benefitsReview	sideEffectsReview	commentsReview
0	2202	enalapril	4	Highly Effective	Mild Side Effects	management of congestive heart failure	slowed the progression of left ventricular dys...	cough, hypotension , proteinuria, impotence , ...	monitor blood pressure , weight and asses for ...
1	3117	ortho-tri-cyclen	1	Highly Effective	Severe Side Effects	birth prevention	Although this type of birth control has more c...	Heavy Cycle, Cramps, Hot Flashes, Fatigue, Lon...	I Hate This Birth Control, I Would Not Suggest...
2	1146	ponstel	10	Highly Effective	No Side Effects	menstrual cramps	I was used to having cramps so badly that they...	Heavier bleeding and clotting than normal.	I took 2 pills at the onset of my menstrual cr...
3	3947	prilosec	3	Marginally Effective	Mild Side Effects	acid reflux	The acid reflux went away for a few months aft...	Constipation, dry mouth and some mild dizzines...	I was given Prilosec prescription at a dose of...

Figure 3.2 Raw sample data for Druglib.com

Table 3.2 List of attributes with their types and descriptions for drug review(drugs.com) dataset

Attributes name	Type	Attribute description
uniqueID	string	Unique ID for each review
drugName	categorical	Name of drug Excluded from classification task
condition	categorical	Name of condition Class attribute
review	text	Patient review Excluded from classification task
rating	numerical	10-star patient rating (1-10) Excluded from classification task
date	date	Date of review entry Excluded from classification task
usefulCount	numerical	Number of users who found the review useful. Excluded from classification task

	uniqueID	drugName	condition	review	rating	date	usefulCount
0	206461	Valsartan	Left Ventricular Dysfunction	"It has no side effect, I take it in combinati...	9	20-May-12	27
1	95260	Guanfacine	ADHD	"My son is halfway through his fourth week of ...	8	27-Apr-10	192
2	92703	Lybrel	Birth Control	"I used to take another oral contraceptive, wh...	5	14-Dec-09	17
3	138000	Ortho Evra	Birth Control	"This is my first time using any form of birth...	8	3-Nov-15	10
4	35696	Buprenorphine / naloxone	Opiate Dependence	"Suboxone has completely turned my life around...	9	27-Nov-16	37

Figure 3.3 Raw data sample for drugs.com

3.3 DATA CLEANING

Data cleaning is the process of removing noise and inconsistency to improve the quality of data. In the following section, description on data cleaning for both datasets will be elaborated further. Figure 3.4 shows the information of attributes and number of entries for each dataset. From the figure, we can see some discrepancies in the numbers as highlighted in the image, therefore data cleaning was performed to enhance the data quality.

The discrepancies in the data were as follows:

1. Df_lib dataset has a range of 4143 entries, however the three attributes do not tally with this range and they are:
    a. condition with 4142 entries.
    b. sideEffectsReview with 4141 entries
    c. commentsReview with 4135 entries
2. Df_com dataset has a range of 215063 entries, however one of the attributes, which is, 'condition' has 213869 entries.

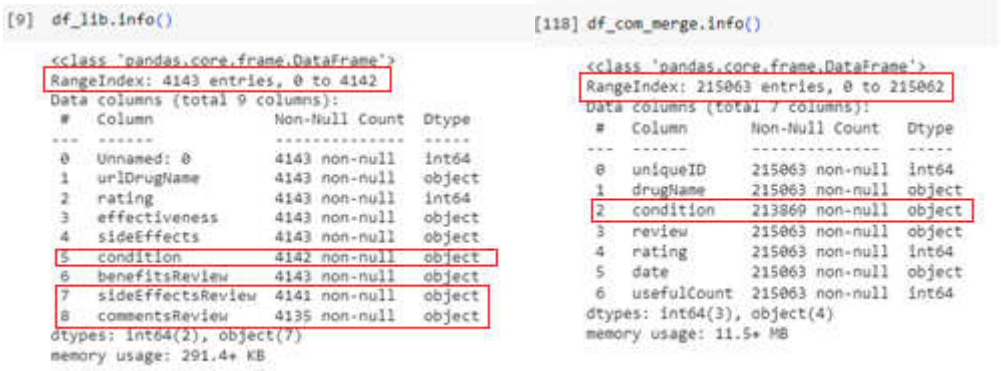

Figure 3.4 Count of entries for each attribute in drug.lib and drug.com datasets before data cleaning

3.3.1 Remove duplicates

Both the dataset was checked for duplicate entries using the unique number assigned for each entry (Unnamed: 0 for druglib.com and uniqueID for drug.com). The output of results shows that there no duplicate values in either dataset.

3.3.2 Remove null values

The drug review dataset was inspected in python Colab tool to detect missing values and the result shows that there was missing values. Since the missing values comprises less than 0.5% (Figure 3.5) of the total data size, it was therefore removed from the data frame.

Figure 3.5 Output of results for missing values in drug.lib and drug.com datasets

```
df_lib_cleaned.info()

<class 'pandas.core.frame.DataFrame'>
Int64Index: 4132 entries, 0 to 4142
Data columns (total 9 columns):
 #   Column             Non-Null Count  Dtype
---  ------             --------------  -----
 0   Unnamed: 0         4132 non-null   int64
 1   urlDrugName        4132 non-null   object
 2   rating             4132 non-null   int64
 3   effectiveness      4132 non-null   object
 4   sideEffects        4132 non-null   object
 5   condition          4132 non-null   object
 6   benefitsReview     4132 non-null   object
 7   sideEffectsReview  4132 non-null   object
 8   review             4132 non-null   object
dtypes: int64(2), object(7)
memory usage: 322.8+ KB
```

```
df_com_merge_null.info()

<class 'pandas.core.frame.DataFrame'>
Int64Index: 213869 entries, 0 to 215062
Data columns (total 7 columns):
 #   Column       Non-Null Count   Dtype
---  ------       --------------   -----
 0   uniqueID     213869 non-null  int64
 1   drugName     213869 non-null  object
 2   condition    213869 non-null  object
 3   review       213869 non-null  object
 4   rating       213869 non-null  int64
 5   date         213869 non-null  object
 6   usefulCount  213869 non-null  int64
dtypes: int64(3), object(4)
memory usage: 13.1+ MB
```

Figure 3.6 Count of entries for each attribute in drug.lib and drug.com datasets after data cleaning

Based on the result output in Figure 3.6, there are no null entries (missing values) in the data frame after data cleaning. Each column has a “non-null count” equal to the total number of entries in the data frames. This indicates that all the values in each column are non-null, and there are no missing values in the dataset. The absence of null entries and duplicates means that the dataset is complete with values for each column, making it easier to work and analyse with.

3.3.3 Remove attribute

Given that the classification of conditions relies on reviews, additional columns from both datasets were excluded before merging them. Figure 3.7 shows the description of attributes selected in both datasets.

```
df_com_merge_cleaned.describe()
```

	condition	review
count	213869	213869
unique	916	128449
top	Birth Control	"Good"
freq	38436	39

```
df_lib_new.describe()
```

	condition	review
count	4132	4132
unique	1804	4053
top	depression	none
freq	302	8

Figure 3.7 Output of results of both dataset after data cleaning

Both these data sets (druglib.com and drug.com) were then combined for the purpose of this study and the description of the data frame is shown in Figure 3.8.

```
#combine df_com_merge dataset and df_lib_new into new dataframe
df = pd.concat([df_com_merge_cleaned, df_lib_new], ignore_index=True)

df.shape
```

```
(218001, 2)
```

```
df.describe()
```

	condition	review
count	218001	218001
unique	2720	132502
top	Birth Control	"Good"
freq	38436	39

Figure 3.8 Description of cleaned drug review dataset comprising of all medical conditions

3.4 EXPLORATORY DATA ANALYSIS

Exploring data is a crucial phase in comprehending and preparing data. This step is essential for gaining a complete understanding of the data's characteristics and for identifying any potential data quality issues. Since the classification on conditions are

based on reviews, the exploratory data analysis will be focused on attributes “condition” and “review”.

Originally, the dataset encompasses reviews of 2720 distinct medical conditions. However, for the experiment, reviews associated with the ten most prevalent medical conditions were isolated to train and test the chosen models. The ten most common medical conditions include:

1. Birth Control
2. Depression
3. Pain
4. Anxiety
5. Acne
6. Bipolar disorder
7. Insomnia
8. Weight Loss
9. Obesity
10. ADHD

Figure 3.9, shows the description of data frame of drug review dataset after cleaning and selection of top 10 medical conditions. The final data frame size of the drug review dataset is 98, 723.

```
conditions = df["condition"].value_counts().sort_values(ascending=False)
conditions[:10]
```

```
Birth Control      38436
Depression         12164
Pain                8245
Anxiety             7812
Acne                7435
Bipolar Disorde     5604
Insomnia            4904
Weight Loss         4857
Obesity             4757
ADHD                4509
Name: condition, dtype: int64
```

Figure 3.9 Selected top 10 conditions and their corresponding counts.

Figure 3.10 Distribution of conditions (classes) in the dataset

The following section will describe the overall distribution of review length (number of characters in the review column) for the selected top 10 medical conditions (Figure 3.10). Figure 3.11 shows the statistics of review length, whereas Figures 3.12 and 3.13 shows the distribution of the review lengths. The total count of reviews in the data frame is 98,723 which is the equal number of total number of entries in the data frame. The average length on the review is about 492.94 characters. The standard deviation is 231.68 which indicates that the review lengths are more spread out from the mean. It shows that the length of review is more dispersed, and there is greater variability in the dataset. The shortest review in the dataset is 3 whereas longest review is 5723. Most of the reviews (50%) fall between 308 and 717 characters. The outliers as shown in Figure 3.12 were not removed as the aim of the study is to experiment on the effects of different features.

```
Review Length Statistics:
count    98723.000000
mean       492.941037
std        231.675630
min          3.000000
25%        308.000000
50%        505.000000
75%        717.000000
max       5723.000000
```

Figure 3.11 Output of review length statistic in final data frame

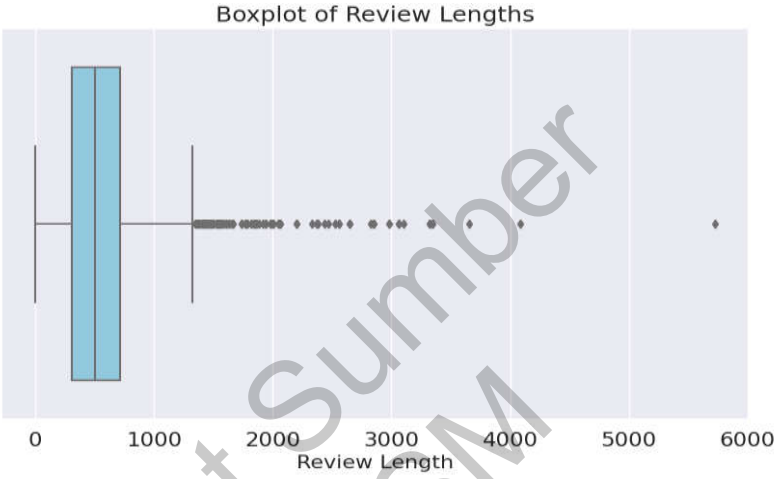

Figure 3.12 Distribution of review lengths shown in box plot chart

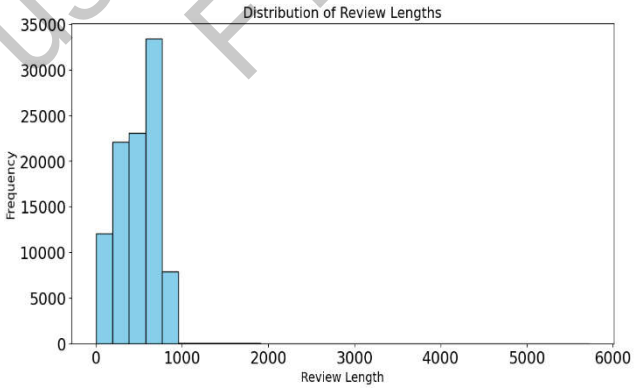

Figure 3.13 Distribution of review lengths in data frame

3.5 TEXT PRE-PROCESSING

In the text pre-processing stage, the unstructured data were converted to structured data. This is an important step as it involves the refinement of input data, which among others include activities such as eliminating unwanted punctuations, removal of stop words, lower casing of the data, removal of links and symbols, and returning the words to its roots by lemmatizing. This is important as data with unwanted links and symbols could block the analysis process and negatively affect the model's accuracy. This stage is crucial in order to maximise the output required from vectorization. In this study, Phyton's "BeautifulSoup" library from the 'bs4' module and the 're' regular expression library was employed to undertake the text pre-processing task. (Figure 3.14).

The overall pre-processing stage was sub-divided into 4 separate sub-processes. In the first sub-processing stage (tokenization) whole sentences within the dataset were deconstructed into individual words. This was followed by the Stop word and punctuation removal sub-process where unimportant non-alphabetical characters were removed. The next sub-process is the lemmatization process where words with identical roots were identified. This step allowed words with different inflections and variations, but with similar roots to be identified and returned to the common base form. Finally, the individual words that have been cleaned and returned to its base form was reconstructed back into sentences in the detokenization pre-processing stage (Figure 3.15).

```python
from nltk.stem import WordNetLemmatizer
import nltk
nltk.download('wordnet')  # Make sure to download the WordNet dataset

# Create an instance of the lemmatizer
lemmatizer = WordNetLemmatizer()

# Your existing code
def review_to_words(raw_review):
    # 1. Remove HTML
    review_text = BeautifulSoup(raw_review, 'html.parser').get_text()
    # 2. Remove non-letters
    letters_only = re.sub("[^a-zA-Z]", " ", review_text)
    # 3. Convert to lowercase and split into words
    words = letters_only.lower().split()
    # 4. Remove stop words
    meaningful_words = [w for w in words if not w in stop]
    # 5. Lemmatization
    lemmatized_words = [lemmatizer.lemmatize(w) for w in meaningful_words]
    # 6. Join the words back into one string, separated by space
    return ' '.join(lemmatized_words)

# Apply the function to create a new column 'review_clean' in your DataFrame
X['review_clean'] = X['review'].apply(review_to_words)
```

Figure 3.14 Text preprocessing done to achieve a clean review data

review	review_clean
My son is halfway through his fourth week of Intuniv. We became concerned when he began this last week, when he started taking the highest dose he will be on. For two days, he could hardly get out of bed, was very cranky, and slept for nearly 8 hours on a drive home from school vacation (very unusual for him.) I called his doctor on Monday morning and she said to stick it out a few days. See how he did at school, and with getting up in the morning. The last two days have been problem free. He is MUCH more agreeable than ever. He is less emotional (a good thing), less cranky. He is remembering all the things he should. Overall his behavior is better. \r\nWe have tried many different medications and so far this is the most effective.	son halfway fourth week intuniv became concerned began last week started taking highest dose two day could hardly get bed cranky slept nearly hour drive home school vacation unusual called doctor monday morning said stick day see school getting morning last two day problem free much agreeable ever le emotional good thing le cranky remembering thing overall behavior better tried many different medication far effective
I used to take another oral contraceptive, which had 21 pill cycle, and was very happy- very light periods, max 5 days, no other side effects. But it contained hormone gestodene, which is not available in US, so I switched to Lybrel, because the ingredients are similar. When my other pills ended, I started Lybrel immediately, on my first day of period, as the instructions said. And the period lasted for two weeks. When taking the second pack- same two weeks. And now, with third pack things got even worse- my third period lasted for two weeks and now it's the end of the third week- I still have daily brown discharge.\r\nThe positive side is that I didn't have any other side effects. The idea of being period free was so tempting... Alas.	used take another oral contraceptive pill cycle happy light period max day side effect contained hormone gestodene available u switched lybrel ingredient similar pill ended started lybrel immediately first day period instruction said period lasted two week taking second pack two week third pack thing got even worse third period lasted two week end third week still daily brown discharge positive side side effect idea period free tempting ala
This is my first time using any form of birth control. I'm glad I went with the patch, I have been on it for 8 months. At first It decreased my libido but that subsided. The only downside is that it made my periods longer (5-6 days to be exact) I used to only have periods for 3-4 days max also made my cramps intense for the first two days of my period, I never had cramps before using birth control. Other than that in happy with the patch	first time using form birth control glad went patch month first decreased libido subsided downside made period longer day exact used period day max also made cramp intense first two day period never cramp using birth control happy patch
Abilify changed my life. There is hope. I was on Zoloft and Clonidine when I first started Abilify at the age of 15.. Zoloft for depression and Clondine to manage my complete rage. My moods were out of control. I was depressed and hopeless one second and then mean, irrational, and full of rage the next. My Dr. prescribed me 2mg of Abilify and from that point on I feel like I have been cured though I know I'm not.. Bi-polar disorder is a constant battle. I know Abilify works for me because I have tried to get off it and lost complete control over my emotions. Went back on it and I was golden again. I am on 5mg 2x daily. I am now 21 and better than I have ever been in the past. Only side effect is I like to eat a lot.	abilify changed life hope zoloft clonidine first started abilify age zoloft depression clondine manage complete rage mood control depressed hopeless one second mean irrational full rage next dr prescribed mg abilify point feel like cured though know bi polar disorder constant battle know abilify work tried get lost complete control emotion went back golden mg x daily better ever past side effect like eat lot
I had been on the pill for many years. When my doctor changed my RX to chateal, it was as effective. It really did help me by completely clearing my acne, this takes about 6 months though. I did not gain extra weight, or develop any emotional health issues. I stopped taking it bc I started using a more natural method of birth control, but started to take it bc I hate that my acne came back at age 28. I really hope symptoms like depression, or weight gain do not begin to affect me as I am older now. I'm also naturally moody, so this may worsen things. I was in a negative mental rut today. Also I hope this doesn't push me over the edge, as I believe I am depressed. Hopefully it'll be just like when I was younger.	pill many year doctor changed rx chateal effective really help completely clearing acne take month though gain extra weight develop emotional health issue stopped taking bc started using natural method birth control started take bc hate acne came back age really hope symptom like depression weight gain begin affect older also naturally moody may worsen thing negative mental rut today also hope push edge believe depressed hopefully like younger

Figure 3.15 Output of review before and after text preprocessing

In order to give a better insight into the cleaned review, word cloud for each condition was generated. Word cloud is an excellent visualization tool to view the frequency of common word used in a review. The size of a word is directly proportionate to the number of times a word appears in a comment. Figures 3.16 -3.25 gives an illustration of commonly used words for each condition or class by reviewers.

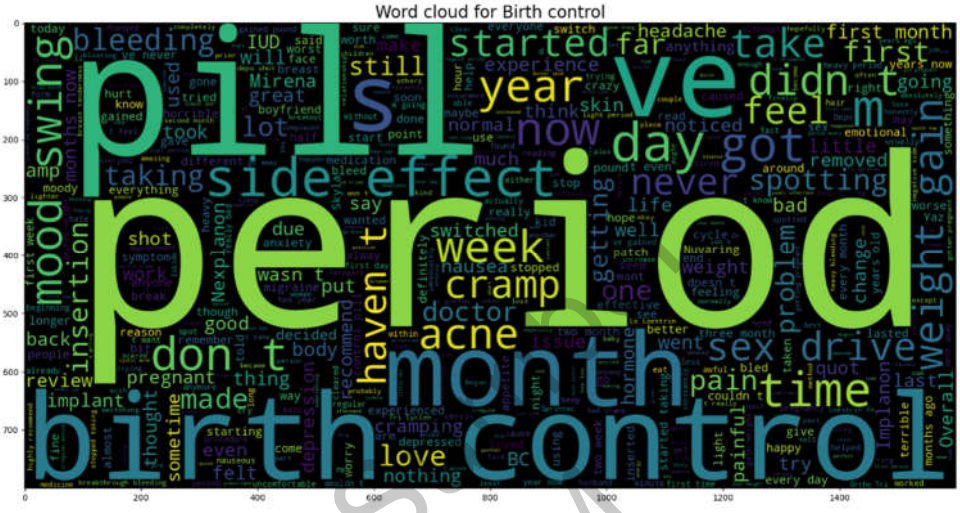

Figure 3.16 Word cloud of birth control

Figure 3.17 Word cloud of depression

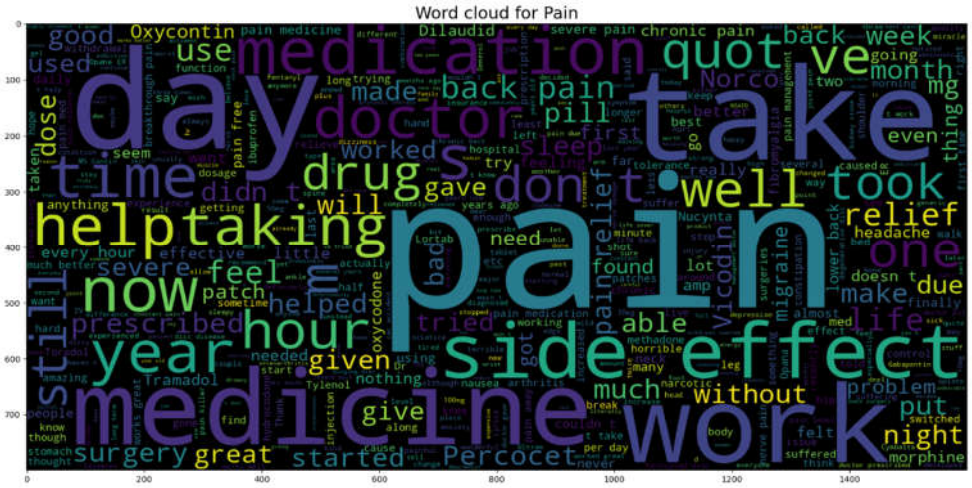

Figure 3.18 Word cloud of pain

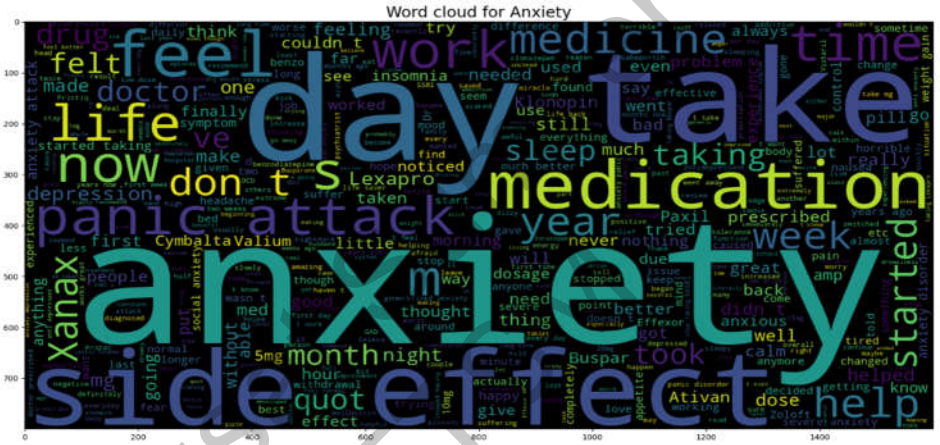

Figure 3.19 Word cloud of anxiety

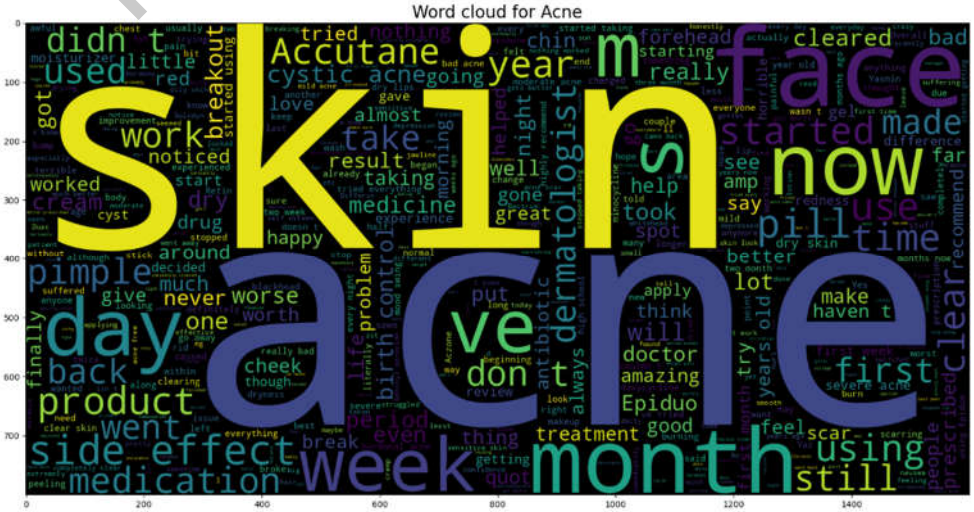

Figure 3.20 Word cloud of acne

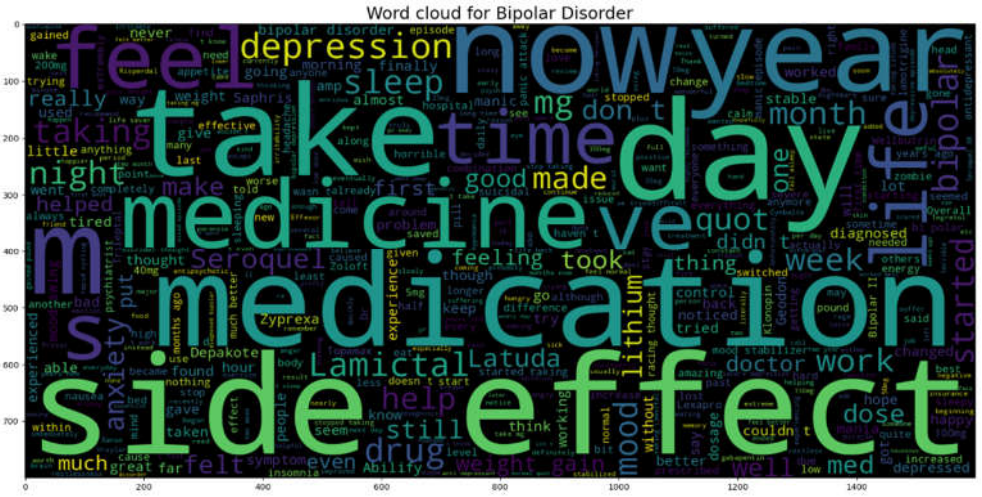

Figure 3.21 Word cloud of bipolar disorder

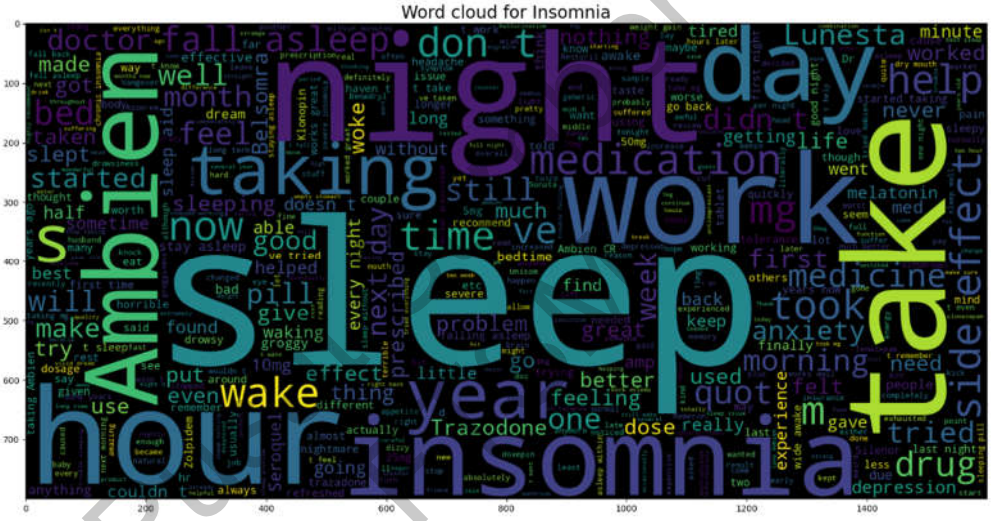

Figure 3.22 Word cloud of insomnia

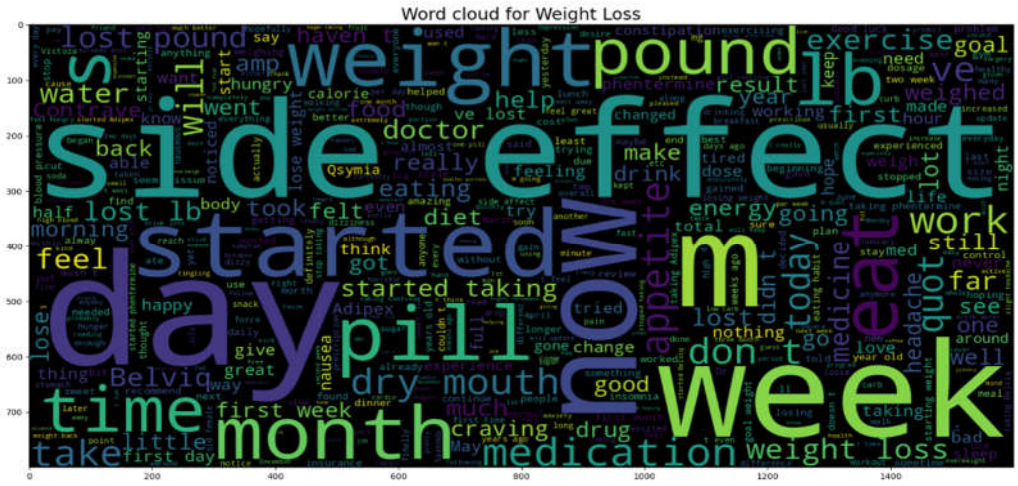

Figure 3.23 Word cloud of weight loss

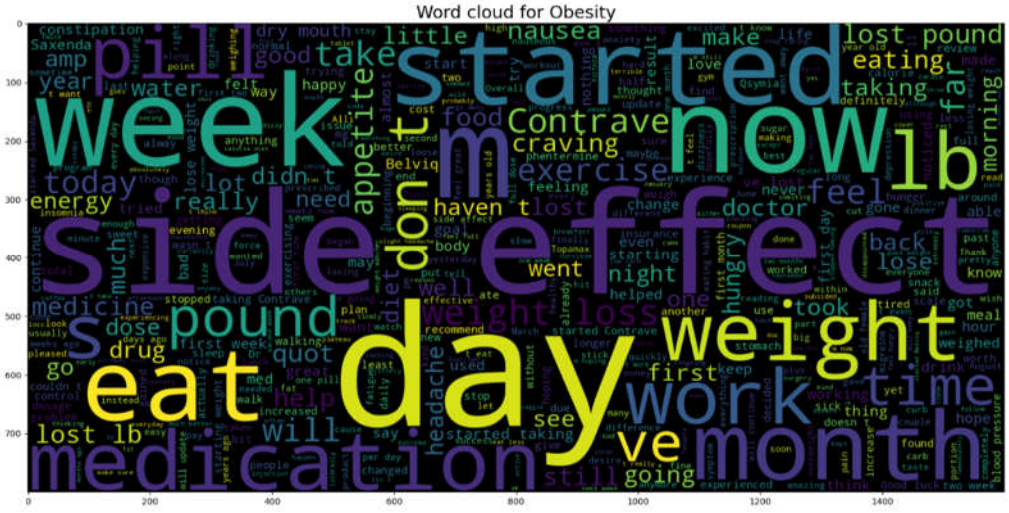

Figure 3.24 Word cloud of obesity

Figure 3.25 Word cloud of ADHD

3.6 TRAIN/TEST SPLITS

Different split techniques were employed in the machine learning model to determine any possible effects on the performance metrics. All machine learning models used for the classification task were subjected to five modes of train and test random splits which were:

1. 90 /10 (train/test) split
2. 80/20 (train/test) split
3. 70/30 (train/test) split
4. 60/40 (train/test) split
5. 50/50 (train/test) split

Figure 3.26 displays extracts of code for the train/test split applied in all the machine learning models within the Python Colab environment. The stratify function was utilized in all the codes as the data is imbalanced. Maintaining similar dataset proportions in the training and testing sets is made easier with the help of the stratify function. This feature helps create an impartial dataset and is especially helpful for handling class imbalances, as observed in the drug review dataset employed in this study.


```
from sklearn.model_selection import train_test_split
X_feat=X['review_clean']
y=X['condition']
```

```
X_train, X_test, y_train, y_test = train_test_split(X_feat, y,stratify=y,test_size=0.1, random_state=0)
```

Figure 3.26 Excerpt from Colab for 90/10 (train/test) split model.

3.7 FEATURE EXTRACTION

Algorithms of machine learning cannot work directly with text. Text must be transformed into numerical values or more precisely vectors of numbers. This step is known as feature extraction which is important in reducing the complexity of text data which aids in the performance of machine learning models. There are few approaches

of feature extraction. Four different feature extraction techniques were used in this investigation for each model, and they are:

1. BOW (Bag of words)
2. TF-IDF unigram
3. TF-IDF bigram
4. TF-IDF trigram

Bag of words is the simplest form of text representation. This method groups the features based on the occurrence of a word in the text disregarding the order of the word in a review. Thus, one potential issue with BOW is that it prioritizes words that appear frequently. As a result, words that appear frequently but are not very important could be given preference. In this experiment, BOW model was implemented using CountVectorizer() function from the Sk-learn library in the Colab environment as shown in Figure 3.27.

```
from sklearn.feature_extraction.text import CountVectorizer
count_vectorizer = CountVectorizer(stop_words='english')

count_train = count_vectorizer.fit_transform(X_train)

count_test = count_vectorizer.transform(X_test)
```

Figure 3.27 Excerpt from Google Colab on the use of count vectorizer to implement BOW as feature extraction

Term frequency-Inverse Document Frequency (TF-IDF) is a feature extraction technique which is based on BOW. However, the TF-IDF method captures not only the occurrence but also the importance of a term in the review. A sample code of the employment tf-idf as feature is shown in Figure 3.28.

```
from sklearn.feature_extraction.text import TfidfVectorizer

# Create TF-IDF vectorizer for unigrams
tfidf_vectorizer = TfidfVectorizer(stop_words='english', max_df=0.8, ngram_range=(1, 1))

# Transform the training and testing data
tfidf_train_2 = tfidf_vectorizer.fit_transform(X_train)
tfidf_test_2 = tfidf_vectorizer.transform(X_test)
```

Figure 3.28 Excerpt from Google Colab on the use of tf-idf vectorizer as feature extraction technique.

3.8 MACHINE LEARNING MODELS

A total of eight machine learning models were selected for the classification task as listed below:

1. Multinomial Naïve Bayes
2. Linear Support Vector Classifier
3. Linear Regression
4. Random Forest
5. Extra Trees
6. Decision Tress
7. Extreme Gradient Boost
8. K-Nearest Neighbour

3.9 PERFORMANCE METRICS

To evaluate the performance of classifier models, performance metrics code were created using the sklearn.metrics in Colab environment. The medical conditions in the drug reviews dataset which is the class attribute is highly imbalanced, therefore the choice of accuracy to assess the performance of the models would be ineffective as it has the tendency to mark a review to the majority class leading to incorrect results. The reason is because accuracy is not a suitable metric to evaluate classifier if the dataset is imbalanced. Hence, F1-score which is a combination of precision and recall was chosen to evaluate the model performance as it punishes the extreme values more. Besides measuring the F1-score, total time taken to train and test the models were also recorded to assess duration of learning time by the classifiers. Nevertheless, for the purpose of record keeping, accuracy, precision and recall were also measured.

3.10 PERFORMANCE EVALUATION

Based on the highest F1-score and time duration to test and train the models, the best machine learning model for the classification was selected.

3.11 TOOLS USED

Colab version 2.0 was used for this study, which is a Python development environment that runs in the browser via Google Cloud. Power BI. Table 3.3 shows the tools used in this study

Table 3.3 Tools Used

Tools	Function
Colab version 2.0	Data cleaning – removal of missing values, checking duplicates
	Exploratory data analysis – data visualization
	Statistics for text review
	Text preprocessing
	Feature extraction
	Perform classification task using machine learning models
	Obtain performance metrics for analysis of classifiers
Excel workbook	To record all the results that was produced by machine learning models
Power BI	Visualisation of all the results that was recorded in excel workbook

CHAPTER IV

RESULTS

4.1 INTRODUCTION

In this section, the results of each model are presented in graph and table format. This section further explains the apparent effect of using different train/split method, feature extractors and the time taken to complete the codes by each machine learning models

4.2 MULTINOMIAL NAÏVE BAYES

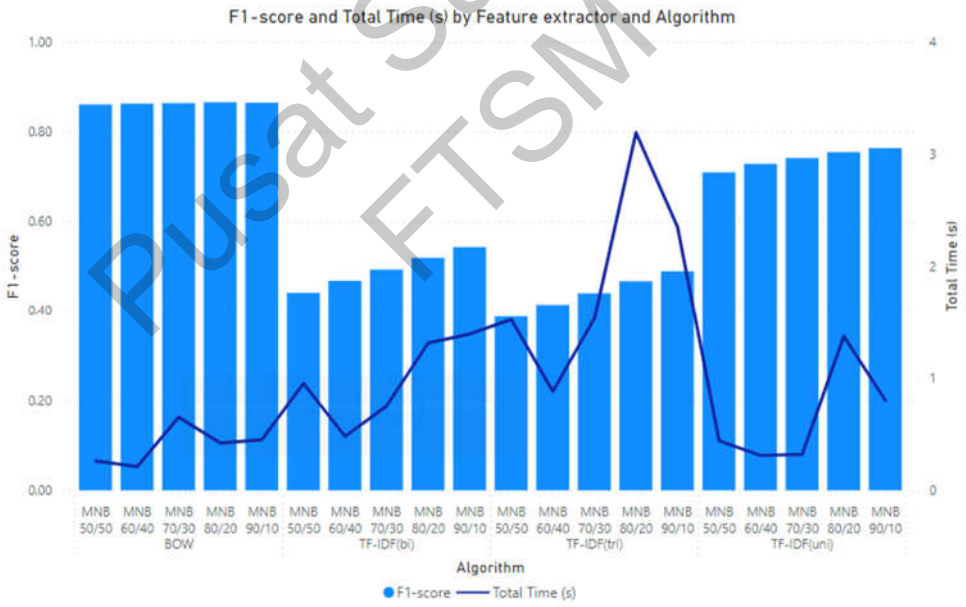

Figure 4.1 F1-score and Total Time(s) by Feature Extractor and Algorithm for MNB.

Table 4.1 F1-score by algorithm and feature extractor for MNB

Feature	Algorithm (train/test ratio)	F1-score	Accuracy	Precision	Recall
BOW	50/50	0.86	0.861	0.866	0.861
	60/40	0.862	0.862	0.866	0.862
	70/30	0.863	0.864	0.867	0.864
	80/20	0.865	0.866	0.868	0.866
	90/10	0.864	0.865	0.867	0.865
unigram	50/50	0.709	0.731	0.792	0.731
	60/40	0.728	0.746	0.801	0.746
	70/30	0.741	0.758	0.807	0.758
	80/20	0.754	0.768	0.813	0.768
	90/10	0.763	0.776	0.818	0.776
bigram	50/50	0.441	0.525	0.719	0.525
	60/40	0.468	0.546	0.73	0.546
	70/30	0.493	0.565	0.736	0.565
	80/20	0.519	0.585	0.741	0.585
	90/10	0.543	0.602	0.747	0.602
trigram	50/50	0.387	0.487	0.722	0.487
	60/40	0.414	0.506	0.735	0.506
	70/30	0.44	0.524	0.735	0.524
	80/20	0.467	0.544	0.738	0.544
	90/10	0.489	0.561	0.744	0.561

Table 4.2 Runtime (s) by algorithm and feature extractor for MNB

Algorithm (train/test ratio)	BOW	bigram	trigram	unigram
50/50	0.26	0.95	1.52	0.44
60/40	0.21	0.48	0.88	0.31
70/30	0.42	1.31	3.19	1.37
80/20	0.42	1.31	3.19	1.37
90/10	0.45	1.39	2.35	0.8

Based on table 4.1, the highest F1-score for the model was recorded under the BOW feature extractor (0.87, 80/20 split) while the 90/10,70/30, 60/40, 50/50 splits of the same feature extractor had the second highest F1-score at 0.86. The TF-IDF trigram with a 50/50 split showed the lowest F1-score at 0.39, followed by the TF-IDF trigram with a 60/40 split at 0.41. Subsequently, TF-IDF trigram (70/30 split) and TF-IDF

bigram (50/50 split) both recorded an F1-score of 0.44. The highest F1-score (0.87) was 55% higher than the lowest F1-score (0.39). Among the feature extractors, BOW had the highest average F1-score (0.86) followed by TF-IDF unigram (0.74) and TF-IDF bigram (0.49). The lowest average of 0.44 was observed under TF-IDF trigram that was 48% lower than the highest feature extractor (BOW), and 40.5% lower than the highest TF-IDF feature extractor (unigram).

Train/test split did not have any noticeable effect on F1-score of BOW feature extractor. It was noted that, apart from BOW, all TF-IDF feature extractors demonstrated an improvement in F1-score as the train/test split increased from 50/50 to a 90/10 split (Figure 4.1). The 50/50 split was consistently the lowest among all TF-IDF feature extractors while the 90/10 split was observed to have the highest F1-score. The difference between maximum and minimum F1-score of the TF-IDF feature extractors were lowest in unigram (7%) followed by bigram (22.7%), with trigram having the highest difference of 25.6%.

From Table 4.2, the top two quickest model runtimes, recorded by the feature extractor BOW, were 0.21 seconds (with a 60/40 split) and 0.26 seconds (with a 50/50 split). Following closely, TF-IDF unigram achieved a runtime of 0.31 seconds for the 60/40 split. Meanwhile, TF-IDF trigram feature extractors exhibited the lengthiest runtimes for the models, with times of 3.19 seconds for 80/20 split, 2.35 seconds for 90/10 split, and 1.53 seconds for 70/30 split being the bottom three. The fastest runtime (0.21s) was 93.4% higher than the slowest model (3.19s). Among the feature extractors, BOW had the fastest average runtime (0.40s) followed by TF-IDF unigram (0.65s) and TF-IDF bigram (0.98s). The slowest average runtime of 1.89s was observed under TF-IDF trigram that was 88.8% lower than the fastest time (0.21s,) and 83.6% lower than the quickest TF-IDF feature extractor (unigram). The effect of split has no noticeable effect on the runtime, however 60/40 split consistently records lowest runtimes for all the feature extractors at 0.21s (BOW), 0.31s (TF-IDF unigram), 0.48s (TF-IDF bigram) and lastly 0.88s (TF-IDF trigram). The difference in time for 60/40 split between the fastest time (0.31s) and slowest time (0.88s) is 64.7%.

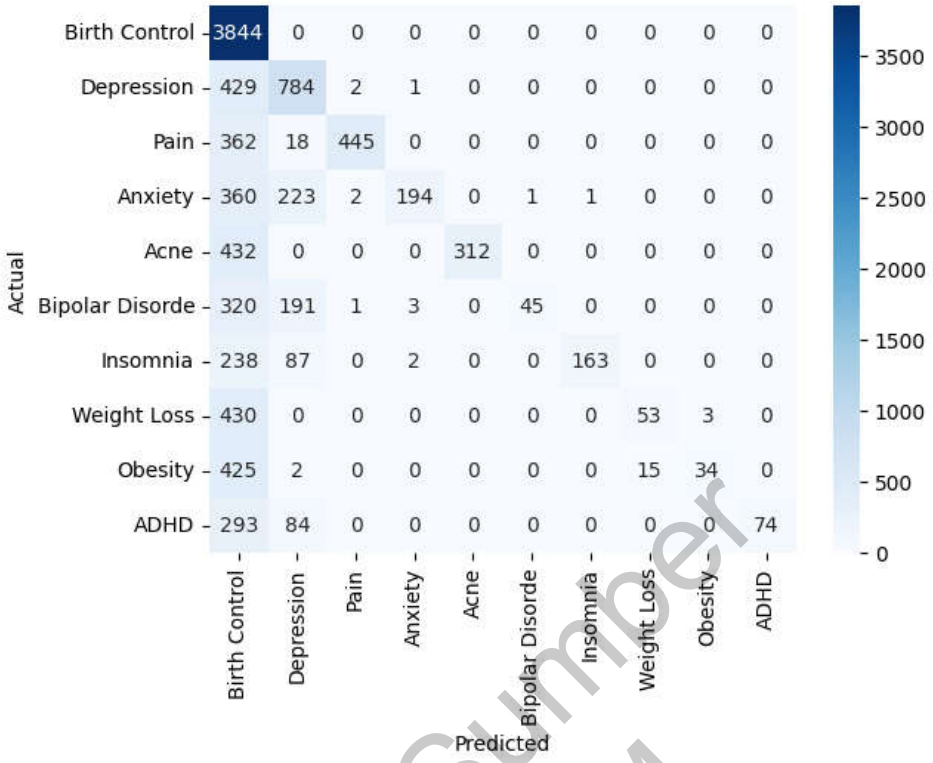

Figure 4.2 Confusion Matrix for MNB 90/10 with bigram

The confusion matrix (Figure 4.2) reveals the model's performance across different classes. Each cell represents the number of instances where the true class (rows) was predicted as the corresponding class (columns). The model excels at identifying "Birth Control," achieving a perfect 100% accuracy. However, it struggles with "ADHD", correctly classifying only 74 out of 451 instances. This could be due to potential limitations where model might assume features are independent, which might not be true for some classes.In addition, because the data is so severely unbalanced, the model with more examples than the others is better able to forecast the dominating class (birth control).

4.3 LOGISTIC REGRESSION

Table 4.3 F1-score by algorithm and feature extractor for Logistic Regression

Feature	Algorithm (train/test ratio)	F1-score	Accuracy	Precision	Recall
BOW	50/50	0.89	0.887	0.887	0.887
	60/40	0.9	0.897	0.897	0.897
	70/30	0.89	0.887	0.887	0.887
	80/20	0.91	0.911	0.911	0.911
	90/10	0.92	0.918	0.918	0.918
unigram	50/50	0.88	0.876	0.877	0.876
	60/40	0.88	0.879	0.88	0.879
	70/30	0.88	0.876	0.877	0.876
	80/20	0.88	0.884	0.884	0.884
	90/10	0.89	0.886	0.887	0.886
bigram	50/50	0.89	0.886	0.889	0.886
	60/40	0.89	0.893	0.895	0.892
	70/30	0.89	0.886	0.889	0.886
	80/20	0.9	0.905	0.906	0.905
	90/10	0.91	0.91	0.911	0.91
trigram	50/50	0.88	0.886	0.888	0.886
	60/40	0.89	0.894	0.896	0.894
	70/30	0.88	0.886	0.888	0.886
	80/20	0.91	0.908	0.91	0.908
	90/10	0.92	0.915	0.917	0.915

Table 4.4 Runtime(s) by algorithm and feature extractor for Logistic Regression

Algorithm (train/test ratio)	BOW	bigram	trigram	unigram
50/50	36.38	154.59	373.61	21.13
60/40	46.24	185.73	475.42	25.93
70/30	62.73	277.46	706.82	34.70
80/20	92.08	361.23	1044.42	49.00
90/10	105.96	435.72	1105.45	54.22

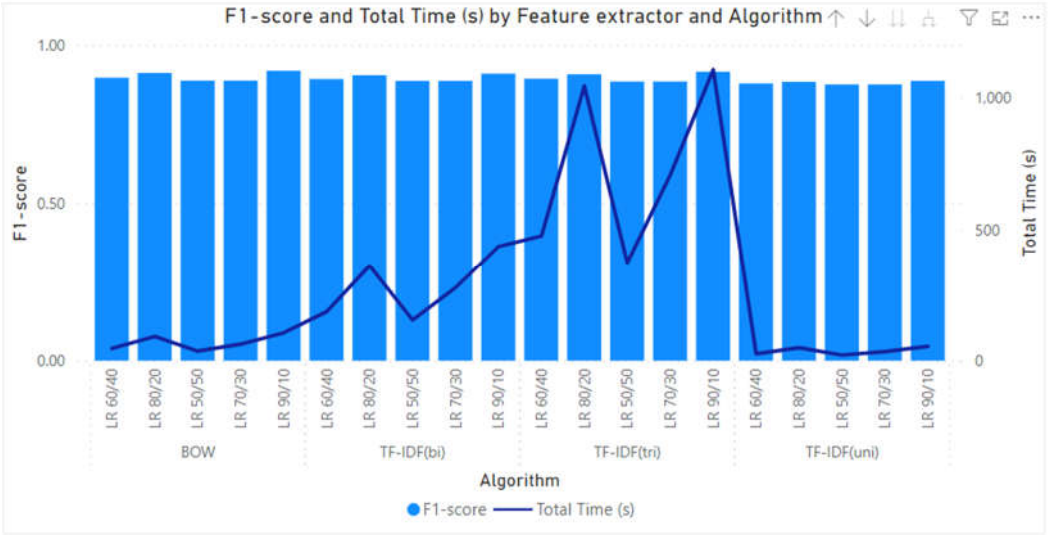

Figure 4.3 F1-score and Total Time(s) by Feature Extractor and Algorithm for Logistic Regression

Figure 4.3 shows the F1-score and Total Time(s) by Feature Extractor and Algorithm for Logistic Regression. LR model 90/10 split for BOW and TF-IDF (trigram) both yielded highest F1-score at 0.92 (Table 4.3). The second highest F1-score was at 0.91 exhibited by BOW (80/20 split), TF-IDF bigram (90/10 split) and TF-IDF trigram (80/20 split). The third highest F1-score was at 0.90, shown by BOW (60/40 split) and TF-IDF bigram (80/20 split). The models which exhibited lowest F1-score at 0.88 were TF-IDF unigram (50/50, 60/40, 70/30, 80/20 splits) while the second lowest F1-score at 0.89 were BOW (50/50, 70/30 splits), TF-IDF unigram (90/10 split), TF-IDF bigram (50/50, 60/40, 70/30 splits) and TF-IDF trigram (60/40 splits). The highest F1-score (0.92) was 4.3% higher than the lowest F1-score (0.88). Except for TF-IDF unigram, the remaining feature extractors (BOW, TF-IDF bigram and trigram) had an average F1-score of 0.90 whereas lowest average F1-score 0.88 was displayed by TF-IDF unigram with a 2.2% difference.

The top three quickest runtime was displayed by TF-IDF unigram feature extractor with times at 21.13s (50/50 split), 25.93s (60/40 split), and 34.70s (70/30 split) (Table 4.4). On the other hand, TF-IDF trigram logged the slowest time at 1105.45s (70/30 split), followed by 1044.22 for 80/20 split and 706.82s for 70/30 split. The fastest runtime (21.13s) was 98.08% higher than the slowest model (1105.45s). Within the groups of feature extractors, TF-IDF unigram recorded shortest average model runtime

at 37.00s followed by BOW at 68.68s and TF-IDF bigram at 282.95s. The longest average runtime was at 741.14s displayed by TF-IDF trigram that was 95% lower than the quickest feature extractor (TF- IDF unigram). Interestingly, all the feature extractors showed increasing trend in runtime as the split increases from 50/50 to 90/10.

The 50/50 split had consistently the shortest runtime among all the splits while the 90/10 split was observed to have the longest runtime. The difference between maximum and minimum runtime of the splits were lowest in TF-IDF unigram (61%) followed by TF-IDF bigram (64.5%), and BOW with 65.7%. TF-IDF had the highest difference with 66.2%.

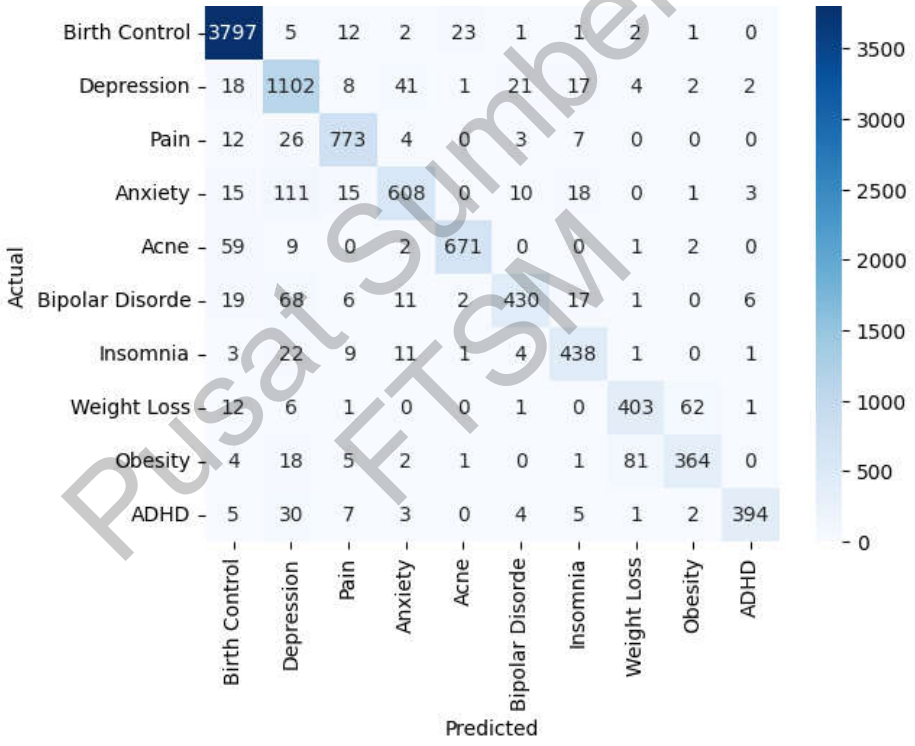

Figure 4.4 Confusion Matrix of LR with 90/10 split and bigram

The confusion matrix (Figure 4.4) reveals the model's performance on different classes. While the model excels at identifying Birth Control with a near-perfect accuracy of 98.7%, it struggles with Obesity, only correctly classifying 77% (364 out of 476 instances) of cases. This lower accuracy compared to other classes might be due to class imbalance with significantly fewer examples of “Obesity” compared to “Birth Control”. This can bias the model towards the dominant class (Birth Control) due to the abundance of training data. Although, Logistic Regression, by its nature, excels at

handling linear relationships, it might struggle with classes requiring capturing non-linear patterns or those with overlapping features. **"**Obesity" and similar classes (like weight loss) might have overlapping features, making it difficult for the model to distinguish them.

4.4 LINEAR SUPPORT VECTOR MACHINE

Table 4.5 F1-score by algorithm and feature extractor for Linear SVC

Feature	Algorithm (train/test ratio)	F1-score	Accuracy	Precision	Recall
BOW	50/50	0.88	0.883	0.883	0.883
	60/40	0.9	0.895	0.895	0.895
	70/30	0.92	0.915	0.915	0.915
	80/20	0.92	0.915	0.915	0.915
	90/10	0.93	0.926	0.925	0.926
unigram	50/50	0.89	0.889	0.889	0.889
	60/40	0.89	0.895	0.894	0.895
	70/30	0.91	0.905	0.905	0.905
	80/20	0.91	0.905	0.905	0.905
	90/10	0.91	0.91	0.909	0.91
bigram	50/50	0.92	0.915	0.915	0.915
	60/40	0.93	0.926	0.926	0.926
	70/30	0.95	0.946	0.946	0.946
	80/20	0.95	0.946	0.946	0.946
	90/10	0.96	0.957	0.956	0.957
trigram	50/50	0.92	0.916	0.916	0.916
	60/40	0.93	0.927	0.928	0.927
	70/30	0.95	0.947	0.947	0.947
	80/20	0.95	0.947	0.947	0.947
	90/10	0.96	0.957	0.957	0.957

Table 4.6 Runtime(s) by algorithm and feature extractor for Linear SVC

Algorithm (train/test ratio)	BOW	bigram	trigram	unigram
50/50	37.33	7.61	15.71	3.27
60/40	42.38	9.21	16.38	3.01
70/30	68.16	12.31	22.54	5.43
80/20	145.95	20.69	36.40	8.12
90/10	127.89	14.30	26.96	5.85

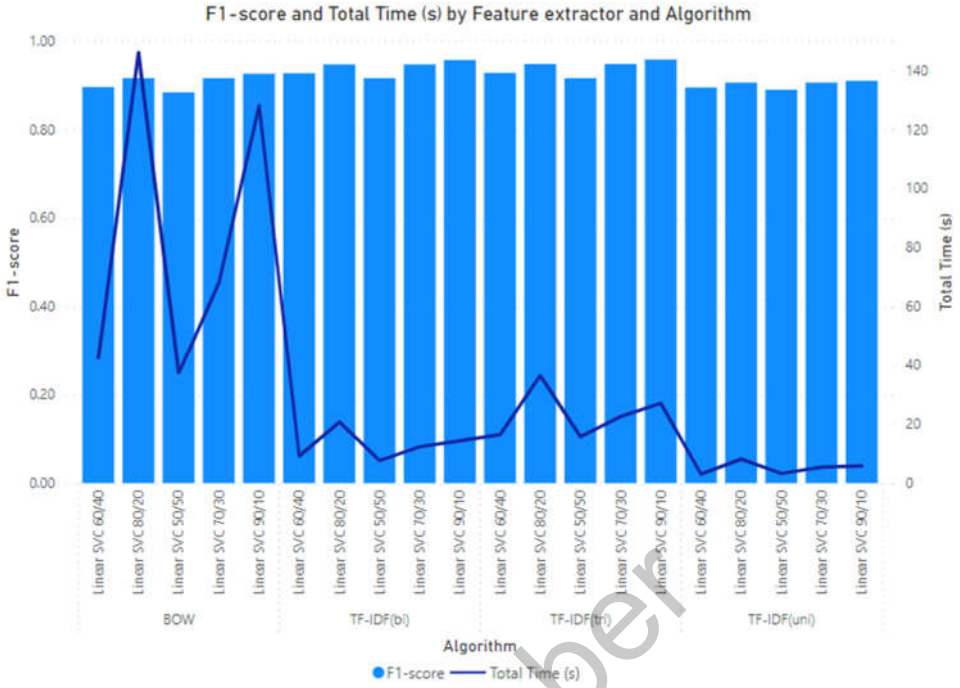

Figure 4.5 F1-score and Total Time(s) by Feature Extractor and Linear SVC splits

Linear SVC model 90/10 split for both TF-IDF (bigram and trigram) yielded the highest F1-score at 0.96 (Table 4.5). The second highest F1-score was at 0.95 exhibited by TF-IDF trigram (80/20, 70/30/ split) and TF-IDF bigram (80/20, 70/30 split). The third highest F1-score was at 0.93, displayed by BOW (90/10 split) and TF-IDF bigram (80/20 split). BOW feature extractor of 50/50 split had the lowest F1-score (0.88), followed by TF-IDF unigram (50/50 and 60/40) at 0.89 and lastly BOW (60/40 split) model recorded third lowest F1-score at 0.90. The highest F1-score (0.96) was 8.3% higher than the lowest F1-score(0.88).

In terms of average F1-score, TF-IDF bigram and trigram feature extraction methods achieved the highest performance, both scoring 0.94. This surpassed the performance of BOW at 0.91 and TF-IDF unigrams at 0.90. Within the feature extractor, the gap between the best and worst performing configurations was 4.3%, with an F1-score range from 0.90 to 0.94. Compared to the 4.3% spread within the feature extractors, the BOW approach, as the second-worst performer, only reached 3.2% below the top F1-score of 0.94 (TF-IDF bigram and trigram).

The effect of split was seen on the F1-score as increment of train/test split model moved from 50/50 to 90/10 (Figure 4.5). The peak average F1-score was observed at 0.94 (90/10 split) while the lowest average F1-score at 0.90 occurred with 50/50 split, indicating a 4.2% difference. The average score of 80/20 and 70/30 split was 0.93 reflecting a decrease of 1.0% lower from the highest split which was 0.94 at 90/10 split. Likewise, 60/40 split produced an average F1-score of 0.91, indicating a 3.2% decrease compared to the highest split average (0.94).

The TF-IDF unigram stands out for its remarkable speed, completing the 60/40 split in a mere 3.01 seconds, making it the leader in terms of runtime efficiency (Table 4.6). The second and third fastest runtime were also clocked-in by TF-IDF unigram at 3.27s and 5.43s. BOW consistently produced slowest runtime across all the splits and the lowest three clock-in time were at 145.95s (80/20 split), 127.89s (90/10 split) and 68.16s for 70/30 split. Feature extractor BOW (80/20 split) was 49 times slower than the fastest TF-IDF unigram (3.01s). The average runtime of TF-IDF unigram is only 5.14s making it the most time efficient feature extractor which is about 94% faster than the slowest feature extractor (BOW) with an average runtime at 84.34s. The TF-IDF bigram clocks at 12.82s average runtime making it the second fastest feature extractor with a difference of 60% from the fastest feature extractor (TF-IDF unigram). Meanwhile, TF-IDF trigram had an average runtime of 23.60s making it 78.2% slower than TF-IDF unigram (5.14s).

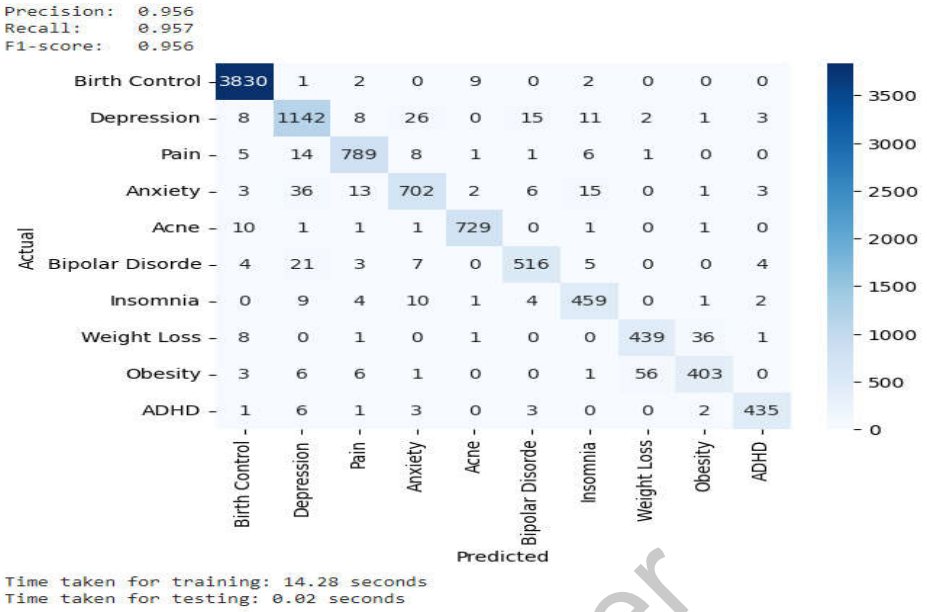

Figure 4.6 Confusion matrix of linear SVC with F1-score and total time at 90/10 split with tf-idf bigram

Figure 4.6 shows the confusion matrix of linear SVC. Overall, the model's performs well across most classes. Birth Control stands out with the highest F1-score of 0.99, indicating exceptional classification accuracy. However, the model struggles slightly with Obesity (F1-score 0.88) and Weight Loss (F1-score 0.89). This could be due to data overlap where Obesity and Weight Loss might have features that overlap with other health conditions. These classes might rely on more subjective features compared to Birth Control. Birth Control data might have clearer, distinct features compared to others, allowing the model to draw a sharper decision boundary between positive and negative cases.

4.5 RANDOM FOREST

Table 4.7 F1-score by algorithm and feature extractor for Random Forest

Feature	Algorithm (train/test ratio)	F1-score	Accuracy	Precision	Recall
BOW	50/50	0.89	0.892	0.893	0.892
	60/40	0.9	0.905	0.906	0.905
	70/30	0.92	0.921	0.921	0.921
	80/20	0.93	0.935	0.935	0.935
	90/10	0.94	0.944	0.944	0.944
unigram	50/50	0.89	0.892	0.892	0.892
	60/40	0.91	0.908	0.908	0.908
	70/30	0.92	0.919	0.92	0.919
	80/20	0.93	0.935	0.935	0.935
	90/10	0.94	0.945	0.945	0.945
bigram	50/50	0.88	0.878	0.88	0.878
	60/40	0.89	0.895	0.897	0.895
	70/30	0.91	0.912	0.913	0.912
	80/20	0.92	0.925	0.926	0.925
	90/10	0.94	0.94	0.941	0.94
trigram	50/50	0.87	0.873	0.875	0.873
	60/40	0.89	0.89	0.892	0.89
	70/30	0.91	0.906	0.908	0.906
	80/20	0.92	0.923	0.924	0.923
	90/10	0.94	0.936	0.936	0.936

Table 4.8 Runtime(s) by algorithm and feature extractor for Random Forest

Algorithm (train/test ratio)	BOW	bigram	trigram	unigram
50/50	153.11	1477.87	3537.47	142.29
60/40	208.40	1815.99	4307.28	200.17
70/30	265.17	2295.58	5855.48	240.53
80/20	352.89	3226.64	7981.62	319.98
90/10	406.43	3322.03	10005.65	357.77

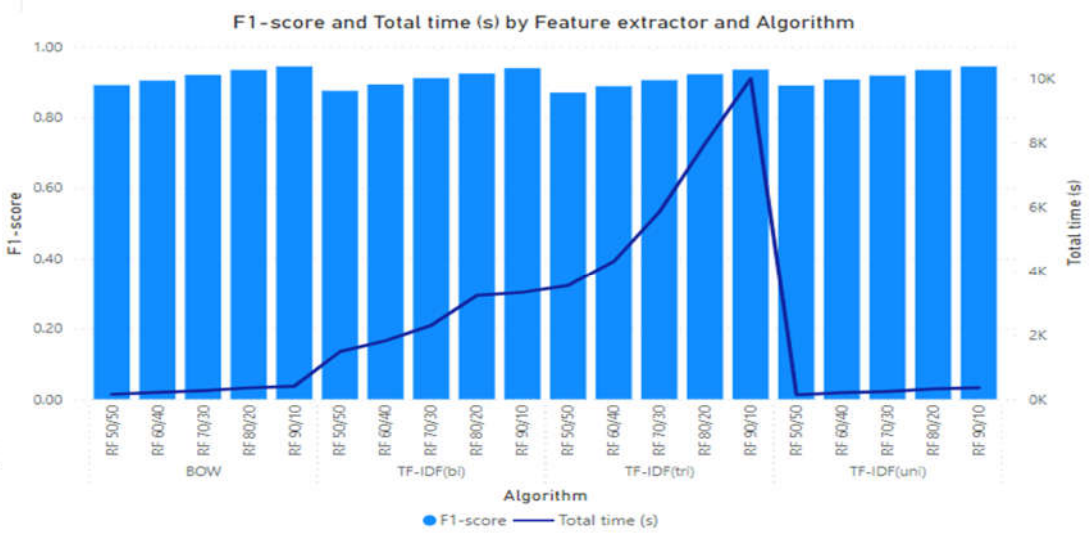

Figure 4.7 F1-score and Total Time(s) by Feature Extractor and Random Forest splits

Based on Table 4.7, Random Forest combined with BOW and TF-IDF (unigram, bigram, trigram) achieved the highest F1-score of 0.94 using a 90/10 data split. This was followed by 80/20 split for BOW and TF-IDF unigram producing F1-score of 0.93. Subsequently, TF-IDF bigram and trigram with 80/20 split, and BOW and TF-IDF unigram with 70/30 split, shared the third highest F1-score of 0.92. Compared to other configurations, TF-IDF trigram and bigram with a 50/50 split underperformed and took the bottom with an F1-score of 0.87 and 0.88 respectively. Following closely at third lowest F1-score of 0.89 were models with 50/50 split (BOW and TF-IDF unigram) and 60/40 split (BOW and TF-IDF trigram). The top performer achieved an F1-score of 0.94, a 7.4% improvement over the weakest performing configuration (0.87). Among the feature extraction techniques, BOW and TF-IDF unigram achieved the highest average F1-score of 0.92. This outperformed TF-IDF trigram, which scored 0.90 (the lowest) and TF-IDF bigram at 0.91 F1-score. There was only a 2.1% difference between the best and worst performing feature extractor. Increasing the training data size demonstrably boosted model performance, with the average F1-score climbing from 0.88 (50/50 split) to 0.94 (90/10 split). This translates to a 6.4% improvement. Compared to the optimal 90/10 split, the 80/20 yielded slightly lower average F1-score of 0.93, reflecting a 1.0% decrease while the 70/30 split achieved an average F1-score of 0.91 with 3.2% decrease. Similarly, the 60/40 split suffered a 4.2% dip in performance with F1-score of 0.90.

Table 4.8 shows that TF-IDF unigram achieved the fastest time in completing the train and test for 50/50 split model with a runtime of 142s. The second shortest runtimes were recorded by BOW (50/50 split) at 153s while the third was recorded by TF-IDF (60/40 split) at 200s. TF-IDF trigram produced longest two runtimes at 4307s with 60/40 split ratio and 3537s with 50/50 split ration. The third longest runtime was at 3322s produced by TF-IDF bigram with 90/10 split ratio.

The feature extractor with the lengthiest runtime (4307s), identified as TF-IDF trigram (60/40), was 30 times slower than the fastest feature extractor, namely TF-IDF unigram with a 50/50 split (142s). The feature extraction technique with the fastest average speed was TF-IDF unigram at 250s which is about 25 times faster than the slowest technique TF-IDF trigram at an average runtime of 6330s. The BOW clocks at 280s average runtime making it the second fastest feature extractor with a difference of 10.7% from the shortest average runtime (250s). Meanwhile, TF-IDF bigram had an average runtime of 2420s making it 89.7% slower than TF-IDF unigram. Every 10% increase in training data translated to a notable rise in the processing time (Figure 4.4). The longest average runtime was observed at 3520s (90/10 split) while the shortest average runtime was by 50/50 split with 1330s giving a 62.2% of time reduction. While more training data generally led to slower training, the performance varied across specific splits. The 60/40 split found a middle ground at 1630 seconds, while the 70/30 and 80/20 splits clocked in at 2160 seconds and 2970 seconds respectively.

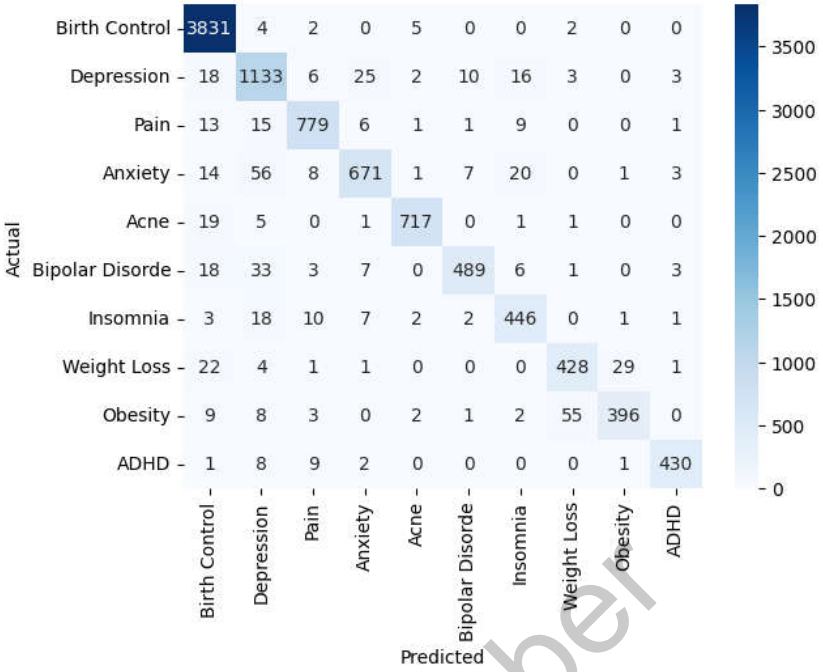

Figure 4.8 Confusion matrix for Random Forest at 90/10 split with bag-of-words

Based on Figure 4.8, the model could identify classes such as birth- control (F1-score : 0.98) and acne (F1-score : 0.97) efficiently. This could be due to clear separation and balanced data in these classes. However, the model struggles to classify classes such as obesity and weight -loss. Random Forests are powerful for complex data patterns, but they might be less efficient than simpler models (like Linear SVC) for well-separated classes.

4.6 DECISION TREE

Table 4.9 F1-score by algorithm and feature extractor for Decision Tree

Feature	Algorithm (train/test ratio)	F1-score	Accuracy	Precision	Recall
BOW	50/50	0.835	0.836	0.835	0.836
	60/40	0.859	0.86	0.859	0.86
	70/30	0.88	0.88	0.879	0.88
	80/20	0.901	0.901	0.901	0.901
	90/10	0.924	0.925	0.924	0.925
unigram	50/50	0.831	0.832	0.831	0.832
	60/40	0.857	0.857	0.856	0.857
	70/30	0.876	0.877	0.876	0.877
	80/20	0.901	0.901	0.901	0.901
	90/10	0.921	0.922	0.921	0.922
bigram	50/50	0.832	0.833	0.832	0.833
	60/40	0.855	0.856	0.855	0.856
	70/30	0.876	0.877	0.875	0.877
	80/20	0.898	0.899	0.898	0.899
	90/10	0.921	0.921	0.921	0.921
trigram	50/50	0.822	0.823	0.822	0.823
	60/40	0.847	0.847	0.847	0.847
	70/30	0.872	0.872	0.872	0.872
	80/20	0.895	0.895	0.895	0.895
	90/10	0.917	0.918	0.917	0.918

Table 4.10 Runtime (s) by algorithm and feature extractor for Decision Tree

Algorithm (train/test ratio)	BOW	bigram	trigram	unigram
50/50	30.82	197.7	659.2	41.15
60/40	36.14	189.55	617.54	46.98
70/30	49.23	289.41	1033.31	56.53
80/20	58.89	405.4	1128.49	97.62
90/10	71.68	378.41	1207.86	79.47

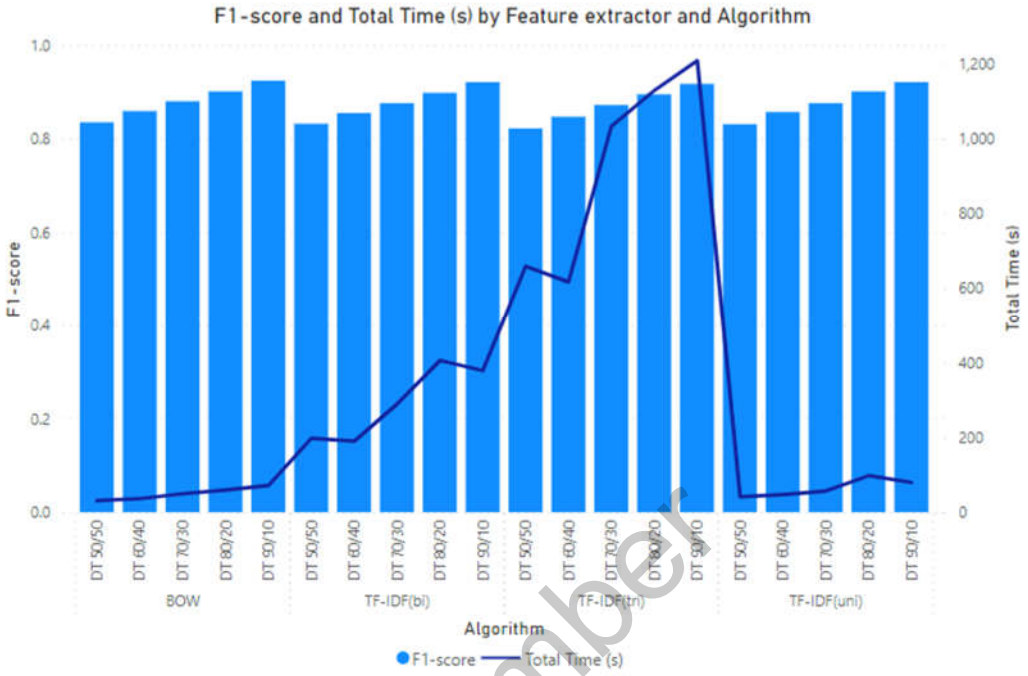

Figure 4.9 F1-score and Total Time(s) by Feature Extractor and DT splits

A combination of Bag-of-Words (BOW) and TF-IDF features (unigrams, bigrams, and trigrams) fed into a Decision Tree classifier yielded a top F1-score of 0.92 on a 90/10 training-testing split (Table 4.9). Following closely, the 80/20 split for BOW and TF-IDF (unigram, bigram, trigram) yielded an F1-score of 0.90. In the third position, TF-IDF (unigram and bigram) and BOW with a 70/30 split shared an F1-score of 0.88. In contrast, all 50/50 split configurations performed less optimally, securing the bottom three positions with F1-scores of 0.82 (TF-IDF trigram), 0.83 (TF-IDF unigram and bigram), and finally, 0.84 (BOW). The top-performing configuration achieved an F1-score of 0.92, marking a 10.9% improvement over the weakest performing configuration (0.82). The average F1-score within the feature extractors did not differ much as BOW and TF-IDF (unigram and bigram) achieved a score of 0.88 while TF-IDF trigram had 0.87 F1-score with a mere difference of 1.1%. Notably, augmenting the size of the training data substantially enhanced model performance, evidenced by the average F1-score escalating from 0.83 (50/50 split) to 0.92 (90/10 split), indicating a notable 9.7% improvement (Figure 4.5). In comparison to the optimal 90/10 split, the 80/20 split resulted in a slightly lower average F1-score of 0.92, indicating a 2.1% decrease. The 70/30 split achieved an average F1-score of 0.88, reflecting a 4.3%

decrease. Similarly, the 60/40 split experienced a 7.6% decline in performance, yielding an F1-score of 0.85. BOW with 50/50 split model demonstrated the fastest completion time for both the training and testing phases, achieving a runtime of 30.82s. The second shortest runtimes were observed with BOW in the 60/40 split model at 36.14s, followed by TF-IDF in the 50/50 split model at 41.15s. TF-IDF trigram recorded the longest runtimes, with 1208s for the 90/10 split ratio and 1128s for the 80/20 split ratio. The third longest runtime, at 1033s, was produced by TF-IDF bigram with a 70/30 split ratio.

The model with the lengthiest runtime (1208s), identified as TF-IDF trigram (90/10), was 39 times slower than the fastest model, specifically TF-IDF unigram with a 50/50 split (31s) (Table 4.10). The feature extraction technique with the fastest average speed was BOW unigram at 49s, approximately 18 times faster than the slowest technique, TF-IDF trigram, with an average runtime of 929s. Tf-IDF unigram had an average runtime of 64s, making it the second fastest feature extractor with a difference of 23% from the shortest average runtime 49s). Meanwhile, TF-IDF bigram had an average runtime of 292s, making it 83% slower than BOW (49s). The influence of the data split was visible in the average runtime, with test proportions ranging from 0.1 to 0.3, the models exhibited an increase in processing time. The split 90/10 had the longest average runtime (434 seconds), followed by 80/20 (423 seconds) and 70/30 (357 seconds). Based on Figure 4.5, the effect is not visible in splits 50/50 and 60/40 since the average runtime was shorter in test proportion 0.5 (232s) compared to 0.6 (223s). The average runtime difference between the two the quickest (60/40) and slowest(90/10) split ratios was 49%.

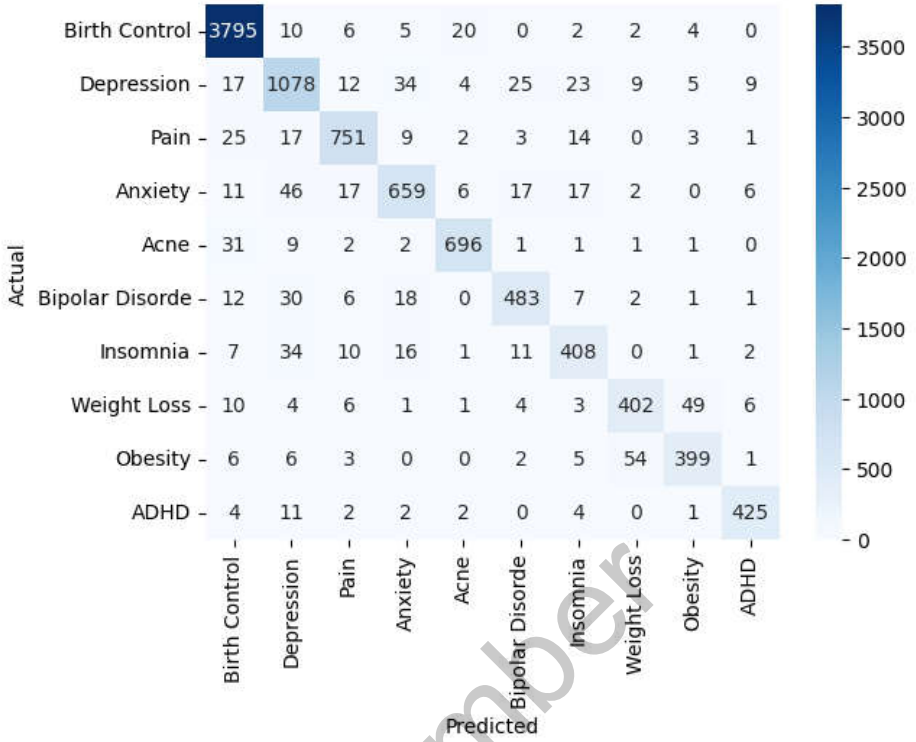

Figure 4.10 Confusion Matrix of Decision Tree with 90/10 split using bigram

The confusion matrix of DT (Figure 4.10) indicated that the model's performance varies across classes, excelling at identifying clear-cut conditions like "Birth Control" and "Acne" but struggling with more nuanced ones like "Anxiety" and "Obesity." This suggests the underlying data characteristics play a significant role in F1-score variations. Decision trees, like the one used here, create simpler decision boundaries compared to models like Random Forests. If a class like "Anxiety" involves complex, non-linear relationships between features, the decision tree might struggle to capture these nuances, leading to misclassifications and lower F1-scores. Decision trees are susceptible to overfitting, especially if not properly regularized. This occurs when the model becomes too focused on memorizing the training data instead of learning generalizable patterns. Overfitting can lead to high accuracy on the training data but poor performance on unseen data, potentially impacting the F1-score for specific classes.

4.7 EXTRA TREE

Table 4.11 F1-score by algorithm and feature extractor for Extra Tree

Feature	Algorithm (train/test ratio)	F1-score	Accuracy	Precision	Recall
BOW	50/50	0.89	0.894	0.895	0.894
	60/40	0.91	0.908	0.908	0.908
	70/30	0.92	0.921	0.921	0.921
	80/20	0.93	0.935	0.935	0.935
	90/10	0.95	0.949	0.949	0.949
unigram	50/50	0.9	0.897	0.898	0.897
	60/40	0.91	0.912	0.912	0.912
	70/30	0.92	0.923	0.923	0.923
	80/20	0.94	0.938	0.938	0.938
	90/10	0.95	0.948	0.948	0.948
bigram	50/50	0.89	0.888	0.89	0.888
	60/40	0.9	0.902	0.903	0.902
	70/30	0.91	0.915	0.915	0.915
	80/20	0.93	0.93	0.93	0.93
	90/10	0.94	0.943	0.943	0.943
trigram	50/50	0.88	0.883	0.884	0.883
	60/40	0.9	0.898	0.899	0.898
	70/30	0.91	0.913	0.913	0.913
	80/20	0.93	0.928	0.928	0.928
	90/10	0.94	0.94	0.941	0.94

Table 4.12 Runtime (s) by algorithm and feature extractor for Extra Tree

Algorithm (train/test ratio)	BOW	bigram	trigram	unigram
50/50	204.63	1645.89	4122.8	189.68
60/40	272.97	2565.94	6792.91	262.57
70/30	317.19	2965.8	7037.61	327.06
80/20	380.63	3346.05	7642.77	379.6
90/10	450.76	4080.03	10515.35	432.59

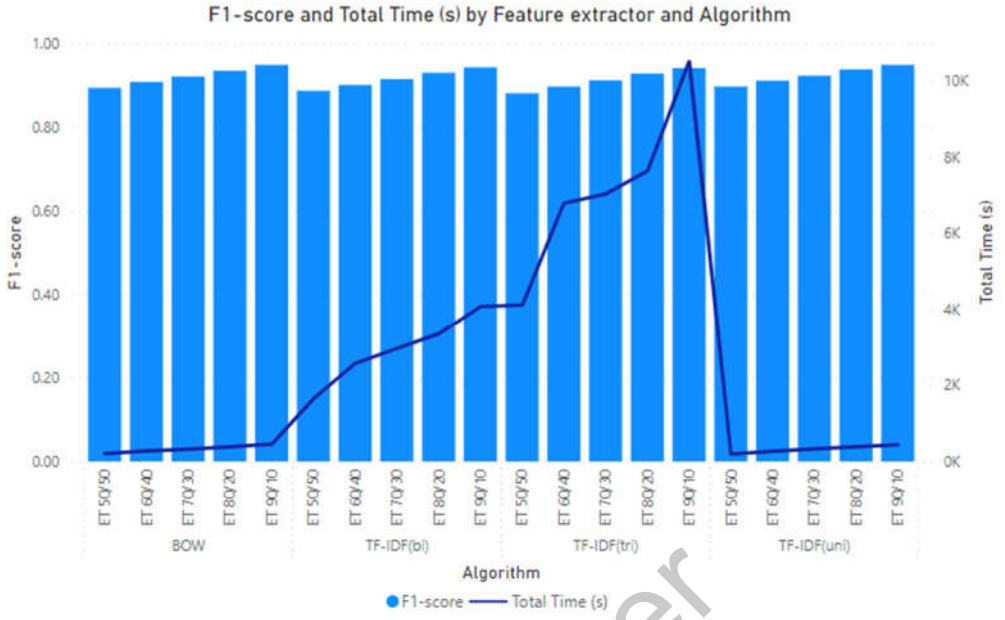

Figure 4.11 F1-score and Total Time(s) by Feature Extractor and ET splits

The greatest F1-score, as shown in Table 4.11, is 0.95. The BOW and TF-IDF unigram feature extractor using the 90/10 method were used to accomplish this. Closely behind, an F1-score of 0.94 was obtained from the 90/10 split for TF-IDF (unigram, bigram, trigram). With an 80/20 split, TF-IDF (bigram and trigram) and BOW shared the third highest F1-score of 0.93. All 50/50 split configurations, on the other hand, did not perform as well as they could have, earning the lowest three F1-scores (TF-IDF trigram: 0.88), (TF-IDF bigram and BOW: 0.89), and (TF-IDF unigram: 0.90). TF-IDF bigram and trigram for 60/40 and Tf-IDF unigram for 50/50 split also displayed the third lowest F1-score of 0.90. With a F1-score of 0.95, the best configuration outperformed the worst, improving on it by 7.4% (F1-score = 0.88). Comparing the feature extraction methods, TF-IDF bigram and trigram had lower average F1-score (0.91), compared to TF-IDF unigram and BOW at 0.92. Separating the top and bottom performing feature extractors at just 1%.

From Figure 4.6, expanding the training data set improved the model's performance, as seen by the average F1-score rising from 0.89 (50/50 split) to 0.94 (90/10 split), a rise of 5.3%. Compared to the ideal 90/10 split, the 80/20 split produced an average F1-score of 0.93, which is 1.0% less than the optimal split. An average F1-

score of 0.92 was obtained by the 70/30 split, indicating a 2% decline. Likewise, the 60/40 split saw a 5% drop in performance, resulting in an F1-score of 0.89.

Based on Table 4.12 and Figure 4.6, the model with fastest runtime was the TF-IDF unigram with 50/50 split which showed completion time at 190s. In the 50/50 split model, BOW had the second-shortest runtimes at 204s, while in the 60/40 split model, TF-IDF unigram had the third shortest runtimes at 262.57s. With 10515s for the 90/10 split ratio and 7642s for the 80/20 split ratio, the TF-IDF trigram had the longest runtimes. The TF-IDF bigram with a 70/30 split ratio generated the third longest runtime, measuring 7037s. TF-IDF trigram (90/10), the feature extractor with the longest duration (10515s), was 55 times slower than TF-IDF unigram with a 50/50 split (190s), which was the fastest feature extractor.

With an average duration of 7220s, TF-IDF trigram was the slowest feature extraction technique, running around 23 times slower than the fastest, TF-IDF unigram, with an average speed of 320s. With a 3% difference from the shortest average runtime (320s), BOW's average runtime of 330s positioned it as the second quickest feature extractor. With an average runtime of 2920s, TF-IDF bigram was 89% slower than the fastest average runtime of 320s for TF-IDF unigram. Runtime analysis revealed that the split data had an effect, with average processing time trending upwards with every 10% increase in training data. While the 50/50 split had the least average runtime at 1540s, indicating a 60% shorter average time compared to the longest average runtime that was recorded at 3870s (90/10 split). Performance differed throughout splits, but in general, training became slower with more data. While the splits of 70/30 and 60/40 clock in average runtimes at 2660s and 2470s, respectively, the 70/30 split finds a medium ground at 2660s.

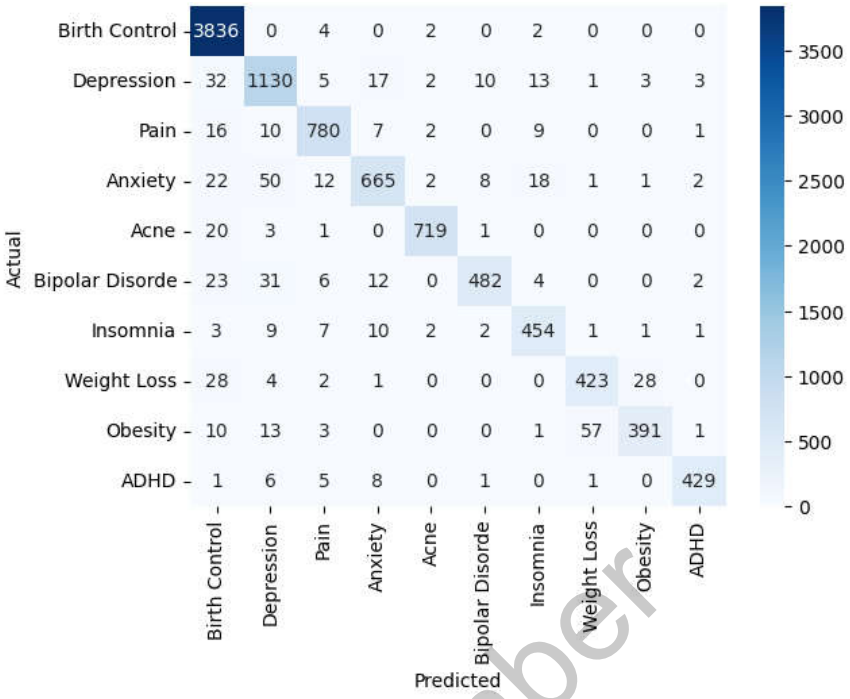

Figure 4.12 Confusion Matrix for Extra Tree with 90/10 split using bigram

From Figure 4.12, we can conclude that Extra Tree excels at classifying straightforward classes like birth control (99.8% accuracy) but struggles with more complex ones like obesity (82% accuracy) similar to Random Forest. While ET introduce randomness in feature selection at each node split to potentially reduce overfitting compared to RF, this difference seems to have minimal impact on F1-score variations across classes in this case. The inherent characteristics of the data likely play a more prominent role in the model's performance.

4.8 EXTREME GRADIENT BOOST

Table 4.13 F1-score by algorithm and feature extractor for XG Boost

Feature	Algorithm (train/test ratio)	F1-score	Accuracy	Precision	Recall
BOW	50/50	0.88	0.881	0.883	0.881
	60/40	0.89	0.885	0.888	0.885
	70/30	0.89	0.887	0.89	0.887
	80/20	0.89	0.889	0.892	0.889
	90/10	0.89	0.889	0.893	0.889
unigram	50/50	0.88	0.884	0.886	0.884
	60/40	0.89	0.889	0.891	0.889
	70/30	0.89	0.892	0.893	0.892
	80/20	0.9	0.897	0.898	0.897
	90/10	0.9	0.897	0.9	0.897
bigram	50/50	0.88	0.883	0.885	0.883
	60/40	0.89	0.888	0.89	0.888
	70/30	0.89	0.892	0.893	0.892
	80/20	0.9	0.896	0.898	0.896
	90/10	0.9	0.898	0.901	0.898
trigram	50/50	0.88	0.88	0.881	0.88
	60/40	0.89	0.886	0.887	0.886
	70/30	0.89	0.89	0.891	0.89
	80/20	0.89	0.895	0.896	0.895
	90/10	0.9	0.897	0.899	0.897

Table 4.14 F1-score by algorithm and feature extractor for XG Boost

Algorithm (train/test ratio)	BOW	bigram	trigram	unigram
50/50	71.8	1813.78	3939.35	404.2
60/40	88.75	1979.03	4484.64	471.66
70/30	84.73	2168.48	4984.9	482.16
80/20	98.67	2420.77	5475.8	530.92
90/10	101.15	2584.91	5916.39	571.8

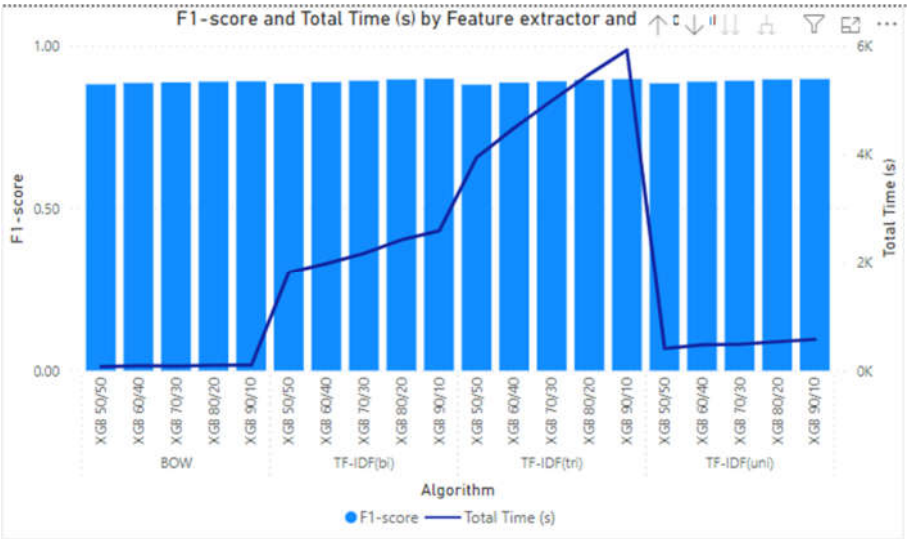

Figure 4.13 F1-score and Total Time(s) by Feature Extractor and XGBoost splits

The F1-score performance of the XGBoost models varied little, as Table 4.13 and Figure 4.13 illustrates. The TF-IDF (unigram, bigram, trigram) splits of 90/10 and 80/20 displayed the highest F1-score of 0.90 (unigram and bigram). These models—90/10 split of BOW, 80/20 split of BOW and TF-IDF trigram, 70/30 split of BOW and TF-IDF (unigram, bigram, trigram), 60/40 split of BOW and TF-IDF unigram, bigram, trigram, and 50/50 split—were not far behind, yielding an F1-score of 0.89. In the meantime, every 50/50 split of TF-IDF and BOW obtained an F1-score of 0.88. In summary, this means that there is only a 2.2% difference between the highest and lowest F1-score, which fall between 0.88 and 0.90. Out of all the feature extraction methods, TF-IDF bigram and BOW had the lowest average F1-score (0.91), while TF-IDF trigram and unigram had the most (0.92). There was just 2.1% difference between the top and bottom performing feature extractors. The XGBoost performance improved when the training data set was increased, as seen by the average F1-score rising by 2.2% from 0.88 (50/50 split) to 0.90(90/10 split). The average F1-score of 0.89 was obtained for the 80/20, 70/30 and 60/40 splits which were only 1.1% lower than that of the optimum 90/10 split.

In terms of runtime (Table 4.14), BOW placed first with 71.80 seconds (50/50 split), second with 84.73 seconds (70/30 split), and third with 88.75 seconds (60/40 split). Conversely, the TF-IDF trigram was consistently the feature extractor, taking the longest time across all splits and finishing in the bottom three with runtimes of 5916s

(90/10 split), 5476s (80/20 split), and 4985s (70/30 split). The feature extractor that logged the shortest duration (71.80s) was BOW (50/50 split), which was 82 times faster than the longest-logging TF-IDF trigram (5916s) with a 90/10 split.

With an average duration of 4960s, TF-IDF trigram was the slowest feature extraction technique, running around 55 times slower than the fastest feature extractor, BOW, with an average speed of 90s. With a 81.6% difference from the shortest average runtime (BOW-90s), TF-IDF's average runtime of 490s positioned it as the second quickest feature extractor. With an average runtime of 2190s, TF-IDF bigram was 95.9% slower than the fastest feature extractor average runtime (BOW). Runtime analysis revealed that the split data had an effect, with processing time trending upwards with every 10% increase in training data. While the 50/50 split had the least average runtime at 1560s, indicating a 31.9% reduction in time, the longest average runtime was recorded at 2290s (90/10 split). Performance differed throughout splits, but in general, training became slower with more data. While the splits of 80/20 and 60/40 clock in at 2130s and 1760 seconds, respectively, the 60/40 split finds a medium ground at 1930 seconds.

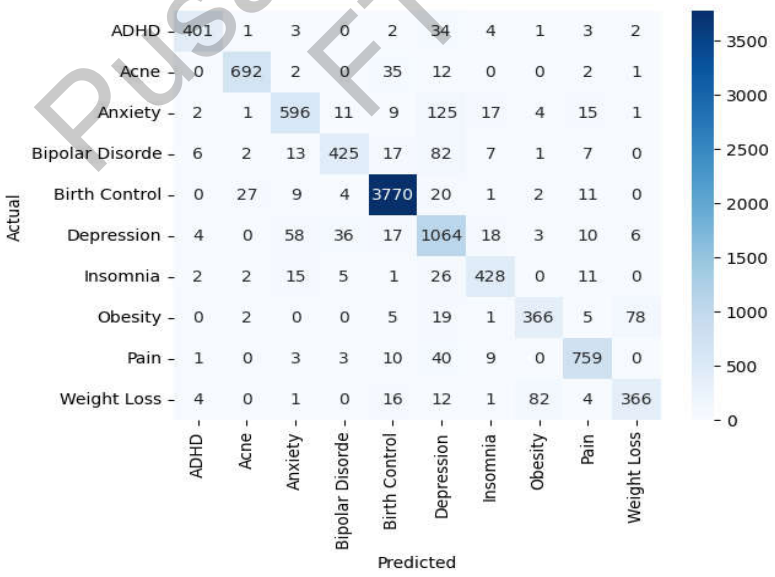

Figure 4.14 Confusion Matrix for XGBoost with 90/10 split using bigram

Based on Figure 4.14, the XGBoost model achieved impressive results for classes like "Birth Control" (F1-score: 0.98) and "Acne" (F1-score: 0.92). However, it struggled with "Obesity" and "Weight Loss," only reaching an F1-score of 0.78 in these categories. This could be be due to ’s k-nearest neighbour

4.9 K-NEAREST NEIGHBOUR (KNN)

Table 4.15 F1-score by algorithm and feature extractor for K-NN

Feature	Algorithm (train/test ratio)	F1-score	Accuracy	Precision	Recall
BOW	50/50	0.65	0.643	0.659	0.643
	60/40	0.65	0.648	0.665	0.648
	70/30	0.66	0.652	0.67	0.652
	80/20	0.66	0.657	0.674	0.657
	90/10	0.66	0.659	0.676	0.659
unigram	50/50	0.21	0.208	0.755	0.208
	60/40	0.07	0.104	0.67	0.104
	70/30	0.3	0.298	0.771	0.298
	80/20	0.32	0.317	0.803	0.317
	90/10	0.17	0.217	0.705	0.217
bigram	50/50	0.17	0.183	0.663	0.183
	60/40	0.02	0.078	0.499	0.078
	70/30	0.26	0.276	0.707	0.276
	80/20	0.28	0.29	0.732	0.29
	90/10	0.1	0.18	0.584	0.18
trigram	50/50	0.17	0.182	0.719	0.182
	60/40	0.01	0.077	0.512	0.077
	70/30	0.26	0.276	0.766	0.276
	80/20	0.27	0.29	0.749	0.29
	90/10	0.1	0.179	0.653	0.179

Table 4.16 Runtime (s) by algorithm and feature extractor for K-NN

Algorithm (train/test ratio)	BOW	bigram	trigram	unigram
50/50	151.62	11006.75	20067.75	3157.72
60/40	140.46	10800.37	16568.28	3037.78
70/30	128.78	9493.36	19034.85	2644.22
80/20	94.50	7423.43	14875.59	2032.04
90/10	77.45	4230.91	8773.08	1162.24

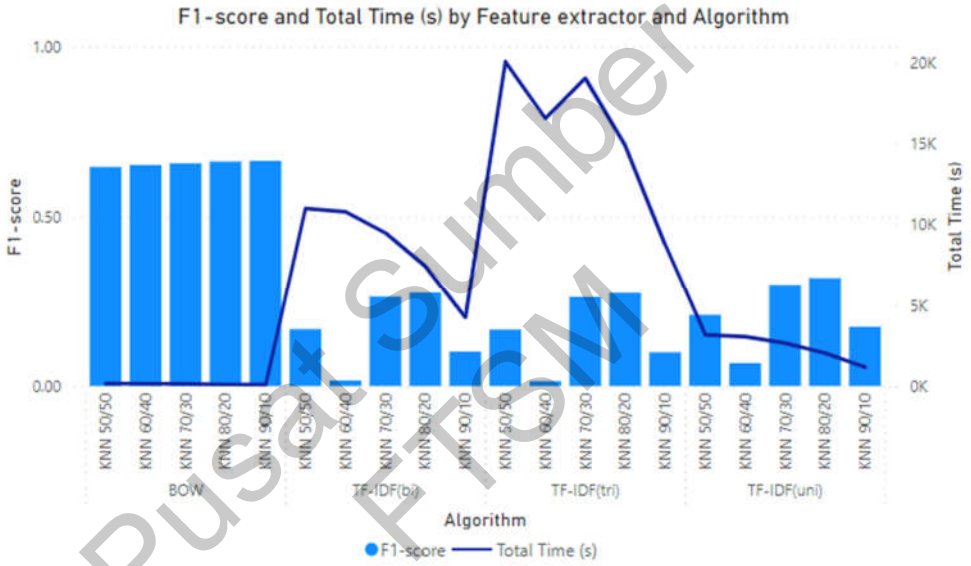

Figure 4.15 F1-score and Total Time(s) by Feature Extractor and KNN splits

Based on Figure 4.15 and Table 4.15, KNN performance showed that an F1-score above 0.5 could only be achieved by using the BOW feature extractor. KNN combined with BOW achieved top highest F1-score at 0.66 for splits 90/10, 80/20 and 70/30. Following closely at the second place, BOW with 60/40 and 50/50 splits achieved F1-score of 0.65. The third highest F1-score, fell low at 0.32, displayed by TF-IDF unigram for 80/20 splits. Compared to other configurations, TF-IDF unigram, bigram, and trigram with a 50/50 split underperformed and took the bottom three for the F1-score with an 0.01 (trigram), 0.02 (bigram) and 0.07 (unigram). The top performer achieved an F1-score of 0.66, a 98.5% improvement over the weakest performing configuration (0.01).

Among the feature extraction techniques, BOW achieved the highest average F1-score of 0.66. This outperformed TF-IDF bigram and trigram, which scored 0.16 (the lowest) and TF-IDF unigram at 0.21 F1-score. There was only a notable difference of 75.8% between the best and worst performing feature extractor. The KNN model's performance did not appear to improve with a larger training data set. For example, the 80/20 split had the highest average F1-score of 0.38, while the 60/40 split had the lowest average F1-score of 0.17. This corresponds to a 55.2% discrepancy. The 70/30 split produced a slightly lower average F1-score of 0.37, indicating a 2.6% decline, compared to the highest average F1-score (0.38) 80/20 split, while the 50/50 split showed an average F1-score of 0.30, indicating a 21% decrease. Likewise, the 60/40 split's F1-score of 0.19 indicated a 50% decline in performance.

Based on Table 4.16, the BOW feature extractor finished first in runtime with 77.45s (90/10 split), second with 94.50s (80/20 split), and third with 128.78s (70/30 split). The 60/40 split, on the other hand, required the most time with TF-IDF (unigram, bigram, trigram) as feature extractors and finished in the bottom three with runtimes of 16568s, 10800s, and 3038s. The model with the shortest runtime (77.45s) was BOW (90/10 split), which was 214 times faster than the slowest model, the TF-IDF trigram (16568s) with a 60/40 split.

The slowest feature extraction approach, TF-IDF trigram, had an average duration of 15860s, which was almost 132 times slower than the fastest feature extractor, BOW, which had an average speed of 120s. The TF-IDF unigram average runtime of 241s was the second quickest feature extractor, with a 50.2% difference from the smallest average runtime (BOW-120s). The TF-IDF bigram average runtime (8590s) was 98.6% slower than the fastest feature extractor average runtime (BOW). The divided data had a minor effect on processing time, with processing time decreasing with every 10% increase in training data (Figure 4.8). The 50/50 split had the longest average runtime at 8600s, while the 90/10 split had the smallest average runtime at 3560s, a difference of 58.6%. While performance varied between split ratios, training was progressively slower as data volume increased. Although the 60/40 and 70/30 divisions have clock times of 7830s and 7640 seconds, respectively, the 80/20 split ran for an average of 6110 seconds.

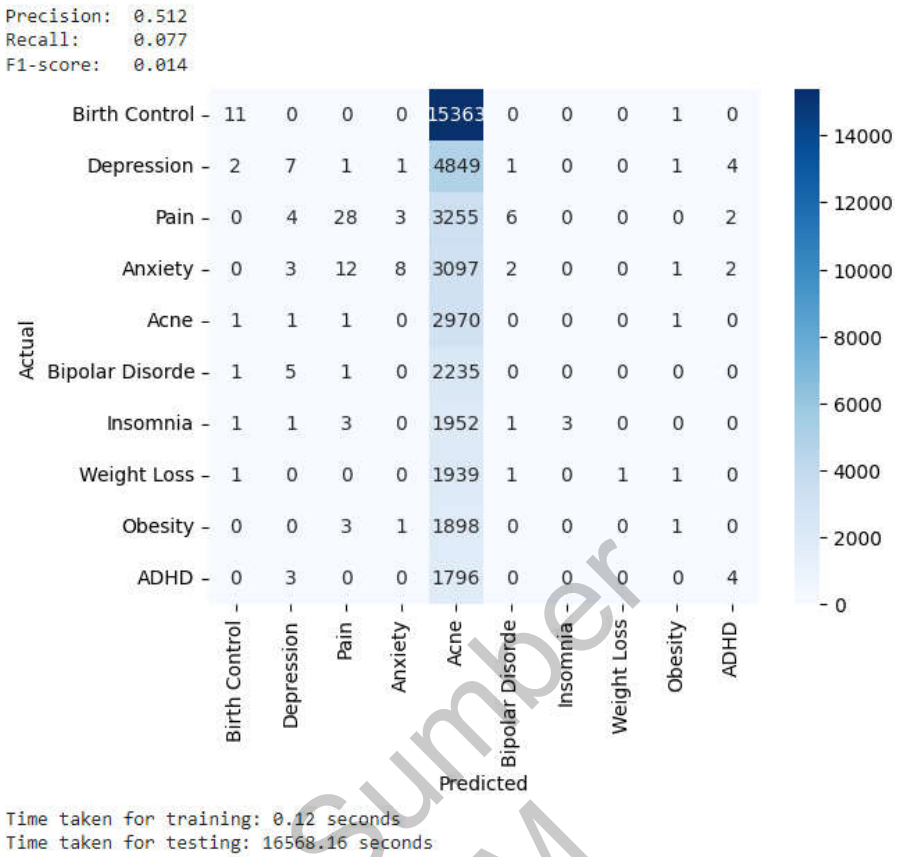

Figure 4.16 Confusion matrix of KNN with F1-score and total time at 60/40 split with tf-idf trigram

Based on Figure 4.16, almost all classes have very low precision, recall, and F1-score, indicating the KNN model is struggling to differentiate between classes effectively. The call value of zero for classes “ADHD”, “Anxiety”, “Birth Control”, “Depression” and “Obesity” indicates that the model is not identifying any true positive instances for those classes. Almost all classes have very low precision, recall, and F1-score, indicating that KNN model is struggling to differentiate between classes effectively. The possible reasons for the low performance may be due to data imbalance, high feature dimensionality or ambiguous features. Since KNN has shown very poor performance, alternative models which are more robust such as Linear SVC could be considered for this dataset.

4.10 CONCLUSION

This section elaborated on the effects and results achieved by eight different machine learning models using four different feature extractors and five train/test splits on drug review dataset. In general, the results showed that the performance of Linear SVC is the best whereas KNN showed the worst performance in terms of F1-score and total processing time.

CHAPTER V

DISCUSSION

5.1 INTRODUCTION

The machine learning algorithms chosen for the multi-class classification of conditions in the drug review data set were Decision Tree, Extra Tree, K-nearest Neighbour, Logistic Regression, Linear SVC, Multinomial Naïve Bayes, Random Forest, and Extreme Gradient Boost. All these models were subjected to four types of feature extraction techniques namely Bag-of-Words (BOW) and TF-IDF (unigram, bigram, and trigram) to investigate effect of feature extraction techniques. Stratified random sampling method with various test sizes (test/train ratio: 50/50, 60/40, 70/30, 80/20, 90/10) were employed to the models following the feature extraction techniques. The split ratios of test and train datasets were experimented to observe the effect of train/test dataset size on the classifier algorithms. The performance metrics chosen to evaluate the classifiers were F1-score and total time taken to train and test the models. The following section summarizes the overall results in terms of mean F1-score and average performance time obtained by the selected models, feature extraction techniques and sampling methods by visualizing into boxplot chart.

5.2 COMPARISON OF F1-SCORE BY ALGORITHM

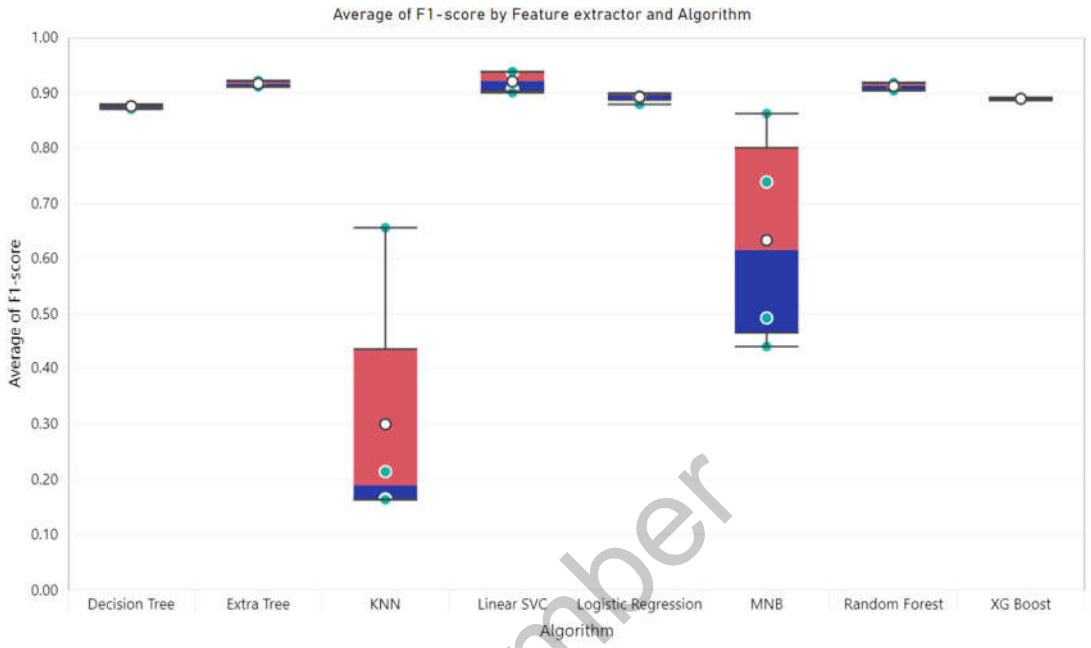

Figure 5.1 Comparison of classifier's F1-score by feature extractor and algorithm.

Table 5.1 Comparison of classifier's F1-score by algorithm

Algorithm	BOW	unigram	bigram	trigram	Mean F1-score of algorithm
Linear SVC	0.91	0.90	0.94	0.94	0.92
Extra Tree	0.92	0.92	0.91	0.91	0.92
Random Forest	0.92	0.92	0.91	0.90	0.91
Logistic Regression	0.88	0.90	0.90	0.90	0.89
XG Boost	0.89	0.89	0.89	0.89	0.89
Decision tree	0.88	0.88	0.88	0.87	0.88
MNB	0.86	0.74	0.49	0.44	0.63
KNN	0.66	0.21	0.16	0.16	0.30
Mean F1-score of feature extractors	**0.86**	**0.79**	**0.76**	**0.75**	

Based on Figure 5.1, the performance based on the F1-score, of the different feature extractors and algorithms varied considerably. Looking at the mean F1-score for the models (Table 5.1), except for KNN and MNB, all other machine learning models had an F1-score of 0.88 and above. The lowest mean F1-score was by KNN (0.30) followed by MNB with 0.63 whereas the highest mean F1-score of 0.92 was displayed by Linear SVC and Extra Tree. Following closely, RF scored a second highest mean F1-score of

0.91. Meanwhile the mean F1-score of XGB and LR were 0.89 followed by DT at 0.88. The difference between the highest (average F1-score: 0.92) and the lowest average F1-score (0.30) was 68%. There were also some variations in the spread of the F1-scores for each feature extractor and algorithm. When looking at the minimum(min) and maximum(max) F1-scores, the model with the highest min-max difference was KNN with 0.50, followed by MNB at 0.42, whereas XGB showed zero min-max difference. The rest of the classifiers only showed minimal difference at 0.01 (ET, DT), 0.02 (RF, LR) and SVC (0.04). The model that achieved the highest maximum average F1-score (all splits) was 0.94 by linear SVC with feature extractor TF-IDF trigram, whereas the lowest average F1-score (0.16) was exhibited by KNN using the bigram and trigram feature extraction technique.

The highest average F1-score of 0.96 was obtained using linear SVC with TF-IDF bi gram and trigram with 90/10 train/test split (Table 4.5). This surpasses the previous benchmark of 0.88 by Joshi and Abdelfattah (2021), who employed a similar dataset (druglib.com and drug.com) and TF-IDF but with a 80/20 split to investigate patient condition classification based on drug reviews. Notably, even the lowest F1-score observed here (0.88) using linear SVC (bag-of-words with a 50/50 split) exceeds Joshi and Abdelfattah (2021) highest score. While both studies share similarities in their overall approach, the observed discrepancy in F1-score could stem from several factors. Potential contributors include differences in the train/test split ratios employed for data partition, the feature extraction techniques utilized (BOW, TF-IDF, Ngram), and the text pre-processing steps implemented. It is noteworthy that this study used both BeautifulSoup and Regular Expressions for text pre-processing, while Joshi and Abdelfattah (2021) only used BeautifulSoup from the Python library. Among other classifiers investigated by Joshi and Abdelfattah (2021) were, MNB, Logistic Regression, Decision Trees, Extra Trees, and Random Forest with MNB and DT taking the bottom two spots. Garg (2021) used the drug review dataset (drugs.com) for sentiment analysis on six machine learning models (LR, MNB, linear SVC, perceptron, ridge, SGD), also found that Linear SVC outperformed all other models with 93% accuracy.

On the other hand, Uddin et al. (2022), upon drug sentiment analysis on drug review dataset, concluded that RF showed the best accuracy (94.06%) compared to Multilayer Perceptron (86.82%), SVC(88.63%) and NB(88.57%) using 80/20 split ratio. In the current study, RF showed promising results as the mean F1-score (0.91) was second to linear SVC. Besides that, Gawich and Alfonse (2022), also concluded that RF showed the best accuracy (0.86) for drug review analysis (drugs.com dataset) comparing 10 different classifiers which includes , MNB, LR, DT, RF, KNN and linear SVC with 70/30 ratio. The highest accuracy value achieved in the current study for RF was 94.5% (90/10 split, TF-IDF unigram) giving a difference of 0.4%. Hossain et al. (2020) compared linear SVC, DT and KNN to assess the accuracy of sentiment analysis in drug review dataset and found that linear SVC performed better with accuracy 83.08% followed by DT (76.79%) and lastly KNN (55.41%). The performance of KNN in this study is the worst (mean F1 -score : 0.30, highest F1-Score : 0.66 (Table 5.1)), followed by MNB (mean F1-score: 0.63, highest F1-score : 0.87 (Table 5.1) when compared to other classifiers. Parmar et al. (2023), also found that MNB performs the worst when compared to LR, DT and RF classifier for drug quality classification using sentiment analysis of drug reviews.

The XGBoost is not as popular as traditional machine learning models (MNB, RF, LR, SVM, Tree classifiers) as it is not widely used in drug review datasets, however it is being employed in text classification tasks. Qi (2020), compared five machine learning models which includes, SVM, KNN, NB and XGBoost for the text classification of theft crime. The results showed that XGBoost achieved highest F1-score of 0.96 followed by NB (0.93), KNN (0.85) and SVM (0.81). However, in this study, the best F1-score of XGBoost was 0.90 and the mean F1-score was 0.89. This shows that XGBoost, with proper hyperparameter tuning has the potential to be used as text classifier of drug review datasets. In summary, the box plot chart provides a good overview of the performance of different feature extractors and algorithms on a classification task. Linear SVC was the best performing algorithm in this case with its best average F1-score at 0.94 (bigram and trigram) (Table 5.1).

5.3 COMPARISON OF AVERAGE TOTAL RUNTIME BY ALGORITHM

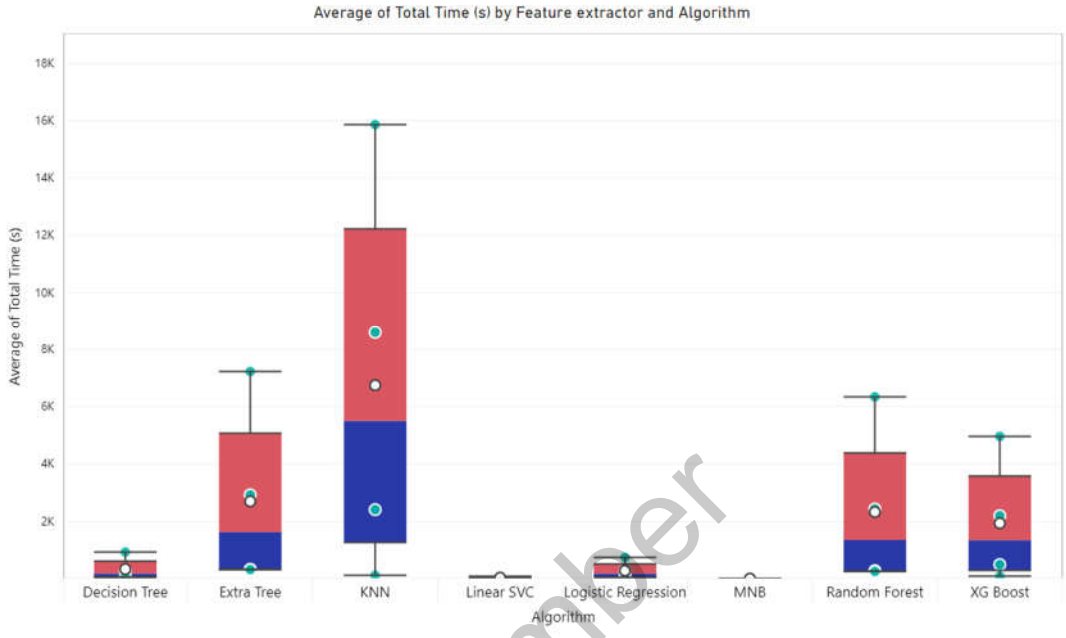

Figure 5.2 Comparison of model's run time by algorithm

Table 5.2 Average total time (s) by feature extractor and algorithm

Algorithm	Decision Tree	Extra Tree	KNN	Linear SVC	Logistic Regression	MNB	Random Forest	XG Boost	Mean runtime by feature extractor
BOW	49.35	325.24	118.56	84.34	68.68	0.40	277.20	89.02	127
unigram	64.35	318.30	2406.80	5.14	37.00	0.65	252.15	492.15	447
bigram	292.09	2920.74	8590.96	12.82	282.95	0.98	2427.62	2193.39	2090
trigram	929.28	7222.29	15863.91	23.60	741.14	1.89	6337.50	4960.22	4510
Mean	333.77	2696.64	6745.06	31.48	282.44	0.98	2323.62	1933.69	

Figure 5.2 and Table 5.2 illustrates the different runtimes taken by the classifiers to train and test the drug review dataset. As shown in the boxplot chart, the MNB is the fastest model to run with average runtime at 1.0s followed by linear svc at 31.5s. This was followed by LR with average runtime at 282s and DT at 333s. LR showed that it is 15% faster than DT. The mean runtime of XGB was 28% faster ET and 17% faster than RF.