Classification of Customer Complaints using Machine Learning Algorithms

Prepared by:
Teballo Madumetja Kgomo
KGMTEB001

Supervisor:
Mzabalazo Ngwenya

Department of Statistics
University of Cape Town

April 29, 2024

Research Submitted to the Department of Statistics at the University of Cape Town in partial fulfillment of the academic requirement for a degree of Masters of Science in Data Science.

Declaration

I, Teballo Madumetja Kgomo, hereby declare that the work presented in this study is my work and does not contain any work that has been submitted in fulfillment of any degree. All information from other's work has been fully cited and acknowledged.

Acknowledgements

The research project would not have been complete without the support and roles played by various people.

Firstly, my esteemed gratitude goes to my supervisor, Mr. Mzabalazo Ngwenya for his pivotal role played in this project. Your guidance, words of encouragement, and commitment given to me can never go unnoticed.

Thank you to the Old Mutual Digital and Data Analytics team managers, Wayne Idas, and Carey-Anne Foulds, without your support, let alone the project, my master's degree would not have been a success. I am very grateful for your thoughtful leadership and efforts to ensure the degree is a success.

Finally, I would love to extend my gratitude to my friends, classmates, siblings, parents, and most importantly my son Phenyo; I am blessed to have you!

Declaration

This research project is dedicated to my loving parents Rasibe and Mmakwena Kgomo, and my siblings (Katlego, Mosima, and Thapelo) for their constant love, support, and encouragement during my studies.

Abstract

Poor handling of customer complaints leads to bad customer experience and impact brand reputation. With an ever-increasing volume of complaints facing customer services team(s), handling customer complaints by service desk agents becomes tedious, especially when pressed with time. For these reasons, many companies have adopted ML technologies to improve their customer services. Technologies like ML text classification have shown great potential in improving customer support. This research proposes an ML text classification approach to categorise customer complaint(s) into one of the thirteen relevant product complaint topics. This technique aims to reduce customer agent desks' customer complaints reading and classifying time.

This research uses five ML algorithms namely: LR, SVM, LightGB, KNN, and CART DT to assess how text classification technology can be used to improve the classification of customer complaints in the financial services industry by assessing how accurately would the algorithms categorize customer complaints data. These algorithms are trained on three different word vectorisation techniques namely: CV, TFIDF, and Word2Vec word-embedding. The algorithms are meant to classify each customer complaint into one of the thirteen possible Products.

Due to imbalanced distributions of the target (Product complaint topics), a balanced-accuracy metric was used to evaluate the model's performance. The results show that LR with TFIDF word vectorisation produced the best model with 87.29 % balanced-accuracy on the OOT dataset. This shows that ML algorithms can be used to improve the customer complaints classification process. Furthermore, the solution can be extended to solve customer complaints emails. This has the potential to improve the company's customer response time and complaint classification from the customer service desk's team.

Keywords: Machine Learning - Text Classification - Natural Language - Artificial Intelligence - Customer experience

Contents

1 Introduction **1**

1.1 Background 1

1.2 Aim 3

1.2.1 Objectives 3

2 Literature Review **4**

2.1 Text Classification 4

2.2 Handling of Customer Complaints 4

2.3 Automatic Text Classification 5

2.4 Customer Support and Service Improvement 6

2.5 Other Text Classification Applications 7

2.6 Multi-class Imbalanced text data 8

2.7 Conclusion 9

3 Research Methodology **10**

3.1 Data Description 10

3.2 Class Imbalance 12

3.3 Training and Validation Dataset . . . 13
3.4 Text Pre-processing . . . 14
3.4.1 Tokenisation . . . 14
3.4.2 Stopwords Removal . . . 14
3.4.3 Lower cases . . . 15
3.4.4 Stemming . . . 16
3.4.5 Lemmatisation . . . 16
3.5 Feature Extraction . . . 17
3.5.1 Vector Space Model: Bag-of-Words . . . 17
3.5.2 Word Embedding . . . 19
3.6 Feature Engineering . . . 20
3.7 Machine Learning Classifier . . . 22
3.7.1 Logistic Regression Classifier . . . 22
3.7.2 K-Nearest Neighbour (KNN) Classifier . . . 23
3.7.3 Support Vector Machine (SVM) Classifier . . . 24
3.7.4 Decision Tree Classifier . . . 25
3.7.5 Light Gradient Boosting (LightGBM) Classifier . . . 27
3.7.6 Evaluation Metrics . . . 28
3.8 Hyperparameter Optimisation . . . 32
3.9 Conclusion . . . 32

4 Results and Discussion **34**

4.1 Results and Analysis . . . 34

4.2 Model Evaluation with Non-Engineered Features 36
4.2.1 Performance on Validation dataset 36
4.2.2 Model Evaluation on OOT Sample Data 39
4.3 Model Evaluation with Engineered Features 43
4.3.1 Performance on Validation dataset 43
4.3.2 Model Evaluation on OOT Sample Data 45
4.4 Final Model Selection 48

5 Conclusion **51**

5.1 Customer Email Resolution Time 51
5.2 Limitation of the study 52
5.3 Future Work and Recommendations 53

Acronyms

AdaBoost Adaptive Boosting. 7

ANN Artificial Neural Network. 6

BoW Bag-of-Words. 4, 8, 9, 14, 17–19, 48

CBoW Continuous Bag of Words. 19

CFPB Consumer Financial Protection Bureau. 2, 5, 10, 53

CNN Convolutional Neural Networks. 5–7

CV Count Vectorize. 1, 17, 18, 35, 36, 38, 40–46, 48, 49

DT Decision Tree. 1, 7, 9, 22, 25, 27, 37, 44

EFB Exclusive Feature Bundling. 28

GOSS Gradient-based One-Side Sampling. 28

IDF Inverse Document Frequency. 18, 19

IE Information Extraction. 16

IR Information Retrieval. 16, 17

KNN K-Nearest Neighbor. 1, 7, 9, 22, 23, 36–38, 40, 42–46, 48, 49

LightGB Light Gradient Boosting. 1, 22, 27

LightGBM Light Gradient Boosting Model. 36–38, 40, 41, 43–46, 49, 52

LLM Large Language Model. 53

LR Logistic Regression. 1, 6, 9, 22, 23, 36–38, 40–42, 46, 48–51

LSTM Long Short-Term Memory. 6, 7

ML Machine Learning. vi, 1–8, 10–15, 17, 20–22, 24, 25, 31–33, 35, 38, 39, 48, 51–53

MLP Multi-layer Perceptron. 7

MMH Maximum Margin Hyperplane. 24

NB Naive Bayes. 6, 7, 9

NLP Natural Language Processing. 6–9, 14, 16, 17, 34, 35, 38, 39, 42, 43, 51

OOT Out-of-Time. viii, 1, 3, 10, 13, 34–36, 38–43, 45, 46, 48–53

RBF Gaussian Radial basis function. 25

RF Random Forest. 6, 7, 9

RNN Recurrent Neural Networks. 5, 6

SMOTE Synthetic Minority Oversampling Technique. 8, 9

SRS Stratified Random Sampling. 13, 34

SVM Support Vector Machine. 1, 6–9, 22, 24, 25, 40, 41, 43, 45, 49, 52

TF Term Frequency. 18, 19

TFIDF Term Frequency Inverse Document Frequency. viii, 1, 4, 7, 8, 14, 17–19, 35–38, 40–44, 46, 48, 49, 51

XGBoost Extreme Gradient Boosting. 27

List of Figures

1.1 CPFB Consumer Complaints volume distribution over time 2

3.1 CPFB Consumer Complaints volume distribution over time 14

3.2 The demonstration of the modelling process for the project (1) 15

3.3 Bag of Words (2) 18

3.4 The Word2Vec word embedding architecture (3) 20

3.5 Word2Vec demonstration on a 3-dimensional plane (4) 21

3.6 Bag of Words with Engineered Feature (2) 21

3.7 KNN classifier during classification in a two-dimensional plane (5) . . 23

3.8 Support Vector machine in a two-dimensional plane (6) 25

3.9 Non-linear data in a two-dimensional plane (7) 26

3.10 Decision Tree Diagram and Terminology (6) 26

3.11 Confusion Matrix for binary and multi-class classification problems . 28

3.12 Machine Learning (ML) text pre-processing and classification process 33

4.1 Boxplot representation (8) 38

4.2 Balanced-accuracy score distribution per model across all feature extraction techniques evaluated on the validation dataset. 39

4.3 Models' balanced-accuracy score performance per feature extraction techniques with no engineered features when evaluated on the Validation dataset. 39

4.4 Models balanced accuracy score on NLP feature extraction techniques with no additional engineered features when evaluated on the OOT data. 41

4.5 Models performance comparison on the balanced-accuracy score per Feature Extraction Technique with no engineered features when evaluated on OOT data. 42

4.6 Models' balanced accuracy score distribution on NLP feature extraction techniques with additional engineered features when evaluated on the OOT data. 44

4.7 Models' performance comparison on the balanced-accuracy score per Feature Extraction Technique with engineered features when evaluated on validation data. 45

4.8 Models balanced accuracy score distribution on NLP feature extraction techniques with additional engineered features when evaluated on the OOT data. 47

4.9 Models' performance comparison on the balanced-accuracy score per Feature Extraction Technique and engineered features when evaluated on OOT data. 47

4.10 Confusion matrix of a Logistic Regression model evaluated on OOT complaints data. 49

List of Tables

3.1 Data Quality Analysis 11

3.2 Distribution of Product by Consumer Complaint Narrative Count . . 12

4.1 Training, Validation, and testing data split by Product 35

4.2 Evaluation results of Count Vectoriser feature extraction on validation data. 36

4.3 Evaluation results of Term Frequency Inverse Document Frequency (TFIDF) feature extraction on validation data. 37

4.4 Evaluation results of Word2Vec feature extraction on validation data. 37

4.5 Evaluation results of Count Vectorizer feature extraction on OOT data 40

4.6 Evaluation results of TFIDF feature extraction on Out-of-Time (OOT) data 40

4.7 Evaluation results of Word2Vec feature extraction on OOT data . . . 41

4.8 Evaluation of Count Vectorizer with engineered features on validation data. 43

4.9 Evaluation of TFIDF with engineered features on validation data across all metrics. 44

4.10 Evaluation of Count Vectorizer with engineered features on OOT data. 45

4.11 Evaluation of TFIDF with engineered features on OOT data across all metrics. 46

4.12 Models' Summary results by Feature Extraction Technique on the OOT data using Balanced-accuracy and a weighted F1-score. 48

4.13 Logistic Regression Classification results for OOT complaints data. . 50

Chapter 1

Introduction

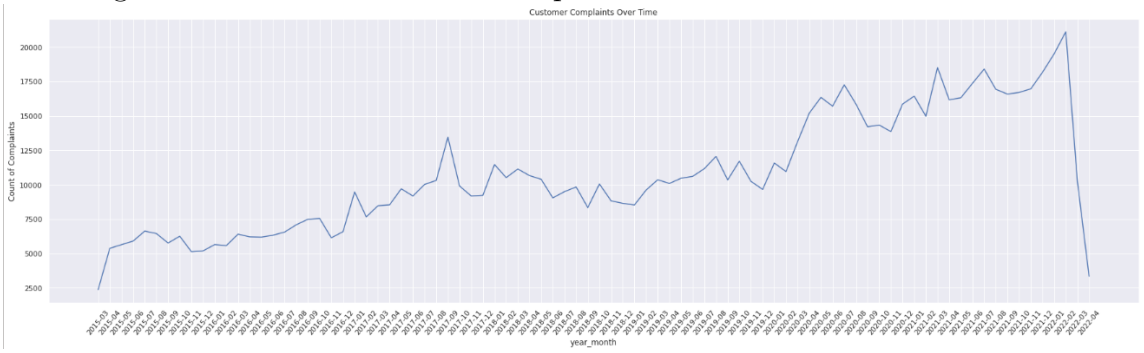
This thesis investigates how ML text classification techniques can be used to improve customer services by efficiently and effectively handling the customer complaint mailbox in the financial services industry. The research starts with an overview of customer complaints and the potential value they may provide to organisations. An overview of how text classification analysis is applied across various industries to improve customer service. Furthermore, this is followed by a brief overview of ML model techniques that were applied to solve a similar problem in various industries.

1.1 Background

Customer complaints in any business forms a key integral part of the customer experience and customer management system. Complaint data if properly analysed can have valuable insights about both businesses and customers (9). As such, businesses in the servicing industry have begun to give importance to developing and maintaining customer complaints management systems(10). Effective handling of customer complaints has shown a direct impact on customer satisfaction, retention rates, the brand's image, and the company's overall performance (11). It is suggested that organizations that can effectively handle customer feedback have on average 5% increase in productivity and become 6% more profitable than their competitors (9). Dissatisfied customers pose a threat to customer retention strategy and are often in problem situations that require urgent attention (12). As such, providing a solution to these customers' needs opens a good relationship between organisations and customers.

With an ever-increasing amount of text data generated from different communication channels, it is challenging for many organisations to handle customer complaints efficiently and effectively (13). This is due to a lack of resources, talents, and skills, and complaints coming across different media channels. This leads to poor turn-around time and customer resolution time taking long. These challenges are attributed to operational costs, lack of resources, and other challenges. For example, customers commonly use text-writing channels like emails or forms on company websites to lodge their complaints (14). In the case of customers complaining via emails, X, and Facebook; the handling of these complaints often means having customer help-desk agents that manually read and forward complaints to the relevant back-end offices. This approach is not scalable- that is, it cannot apply to a higher volume of complaints daily and is not cost-friendly as an increasing volume of complaints often requires more staff, increased cost to the company, and thus results in poor customer complaint resolution turnaround time. This may result in a dire impact on customer satisfaction and thus, give customers a poor service experience which might lead to customer churn.

Figure 1.1: CPFB Consumer Complaints volume distribution over time



It is evident that due to the large volume of customer complaints, an efficient customer complaint management service that targets emails is necessary (15). An efficient and effective complaint management system is a system that allows service providers to listen and empathise, identify the causes of issues, respond to customer complaints promptly, and so on (16). Figure 1.1 shows how complaints volume has been increasing according to the Consumer Financial Protection Bureau (CFPB) from 2015 to April 2022. This shows that there's so much need to scale-up the complaint resolution time for most companies. Hence, the need for big data analytics and ML solution. A drop in trend is the day on which the data was extracted. However, to handle customer complaints data effectively and efficiently without incurring high costs associated with an increasing volume of complaints, the text classification approach is proposed as a solution to improve manual reading and labeling of customer complaints.

1.2 Aim

This thesis seeks to develop a system for handling customer complaints effectively and efficiently by classifying complaints into relevant product type categories and hence enable routing them to the right back-office teams to improve customer agents' labeling, routing, and customer query resolution time.

1.2.1 Objectives

To achieve the aim of the research project, the following objectives are set:

(a) Build and compare various ML classification models to classify customer complaints into thirteen different product type/ categories.

(b) Measure the misclassification errors to understand the efficiency and effectiveness of the models in handling customer complaint mailboxes.

(c) Perform model evaluation analysis to test the robustness of the models by testing how well they generalize on OOT sample.

Chapter 2

Literature Review

This section focuses on text classification and how it applies to solve various problems across different fields of study. The emphasis and focus are on the work done around customer complaints analysis, customer services, and customer experience.

2.1 Text Classification

Text classification is a technique that is used to assign text data such as sentences, paragraphs, queries, and documents into a defined set of categories or labels using ML algorithms (17). These texts can come from various data platforms or sources such as social media, emails, the internet, chatbot data, etc. Normal text is difficult for machines to process in its original form and thus requires data transformation (18). Text data transformation is a technique used to transform raw texts into a structured format, enabling machines to process such data. Some of the commonly known techniques include Word embedding, TFIDF, and Bag-of-Words (BoW).

2.2 Handling of Customer Complaints

Many researchers and businesses have examined customer complaints and their relative importance in the service industries and their impact on customer satisfaction. Companies that invest in complaints-handling processes do so as a way to increase customer commitment and build customer loyalty (19). For example, in the Turkish banking sector, (11) found that quality complaint management is a good indicator

of competitive success in the service industry. The study was administered with 200 questionnaires from the four Turkish banks. The results show that in the service industries, complaint management quality was a huge indicator of competitive success.

Effective handling of customer complaints means acknowledging customer problems, seeking clarity, and identifying the complaint topic (20). As proposed in (12), effective handling of customer complaints has been shown to increase customer satisfaction which in turn produces a good and long-lasting relationship between customers and businesses.(19) evaluated the customers' complaints and the implication that their experience had on company marketing. It was indicated that satisfactory complaint handling is important and is strongly associated with customer trust and commitment to the business.

When (12) studied consumer email complaints, they found that firms that respond to customer complaints result in increased customer satisfaction and purchasing likelihood. It was suggested that companies that aim to increase customer satisfaction with response and perception of the company need to be specific with the problems they are addressing. Furthermore, quick response time was shown to have a direct relationship with customer satisfaction and perception of company concern. The results in (21) which investigated how quickly customers expect companies to respond to emails, showed that the average time it takes a customer to get a response can be as long as 8 days, while almost 46% of customers expect a response in a time not more than 4 hours.

2.3 Automatic Text Classification

With recent advancements in information technology, communication has become more indispensable and instant as people communicate as quickly as possible (22). This overwhelming pace of information generation requires businesses to scale up in complaint-handling technologies and minimize complaint resolution time to remain relevant. In cases where businesses cannot handle fast-paced communication with their clients, auto-response generation becomes necessary. Many industries have adopted the use of ML models to automate processes. This section looks at how ML has been used to automate complaint processes.

A study on customer complaints auto-categorisation is introduced in (23). The study focused on comparing Recurrent Neural Networks (RNN) and Convolutional Neural Networks (CNN) performances in classifying CFPB consumer complaints

data. Two different architectures were used to obtain word vector representations using word2vec embedding and word2vec pre-trained word embeddings. The embedding technique was used to retain syntactic and semantic information from high dimensions. It was found that RNN performs better than CNN with an accuracy score as a metric. Although word2vec pre-trained word embedding did not improve the models' performance, it has improved the training time of both CNN and RNN architectures with CNN training faster than RNN.

In Russia, Natural Language Processing (NLP) algorithms were applied to automate the classification of Russian scientific text (24). Several models such as Artificial Neural Network (ANN), Random Forest (RF), Logistic Regression (LR), and Support Vector Machine (SVM) were compared to determine the model that would be used in the decision support system for the classification of scientific text. The SVM model with word2vec word embedding was used and showed excellent and significant results.

2.4 Customer Support and Service Improvement

To suggest future error sources of medical products, (25) used deep learning models to classify error sources of medical products based on historical complaint text. This study aimed to support complaint management in the medical technology industry. The study used RNN with Long Short-Term Memory (LSTM) cells and a pre-trained word embedding layer of English words to classify textual descriptions of complaints. Naive Bayes (NB) was used as a baseline model. It was shown that the model is correct in over 60% of all the cases presented. However, full automation of the process remained the future work of the authors.

ML algorithms are discriminatory algorithms and operate under the assumption that each item can be classified into one class. However, when this assumption is not met, techniques such as fuzzy methods are introduced. Fuzzy classifiers are methods of learning that use generative learning to measure the degree to which an instance belongs to a class (26). In Malaysia, a government study was conducted to improve the customer complaint process using the Fuzzy approach (17). The study used a classification algorithm that is based on keywords for specific references and ranking. This solution was developed as an alternative approach as the complaint system uses Malay wording. Based on 406 complaints, the fuzzy approach was shown to be more effective in handling the complaint process and classification with 93.35% accuracy. This study demonstrated how efficiently a fuzzy algorithm may improve a customer service.

Another study aimed at improving the handling of emails in the customer contact center compared ML algorithms to find the best model that accurately classifies emails into four classes (27). The study used different NLP techniques like removal of stop words, stemming, TFIDF, and word2vec word embeddings for word vector representation. The results showed that the SVM with word2vec word embedding produced the highest accuracy score. It was concluded that word2Vec is a better option for feature extraction technique when compared to TFIDF on a banking customer contact center email data spanning from 2016 to June 2018.

ML models are mostly used to automate and improve services while reducing business costs. In another study (15), ML algorithms and word embeddings are used in email classification for improved customer support. This study was done in a large telecom industry to improve a manual rule-based algorithm. The study tested different machine learning models like SVM, Adaptive Boosting (AdaBoost), and LSTM in classifying 33 target labels. The performance of the models was evaluated using different NLP techniques of text representation. It was found that the LSTM model performed better than other non-sequential models with an F1 score of 0.91.

An increasing amount of text data on the internet is overwhelming and as such, it may be hard to extract relevant information and categorise it. To help authors/editors reduce the categorisation and classification time of articles, (28) proposed the use of ML to automate text categorisation and classification of articles on suitable topics. The author compared three algorithms namely, SVM, NB, and K-Nearest Neighbor (KNN). It was noted that data pre-processing such as lowering texts, played a significant role in the standardisation of the dataset. Furthermore, SVM has shown to be a better model when classifying articles into suitable topics.

2.5 Other Text Classification Applications

A more recent study (29) used social media data to understand whether an online review was a praise or a complaint. This study aimed to compare different deep learning models against the ensembling approach in classifying extreme opinions. The deep learning models were Dense Neural Network, CNN, and multichannel CNN. Ensembling models were considered from SVM, RF, Decision Tree (DT), and Multi-layer Perceptron (MLP). The study used the AFINN lexicon;- a tool used to assess sentiment in a text, to determine whether an online review is a praise or complaint. The ensembling technique was shown to outperform all other models.

In the public service domain, manually processing citizens' complaint data can be

hard to handle and time-consuming. (18) used ML models and NLP techniques to classify Portugal's citizens' complaints about economic and food safety reasons. The analysis was done on 150,700 complaints related to food and economic surveillance aimed at improving human labour required to ML and deep learning algorithms using the TFIDF bag of words. Although the authors suggested more work around deep learning models, it was shown that the SVM model performs at a satisfactory level.

Many studies have explored the impact of feature extraction techniques on text classification problems. In another study, (30) explored word embedding models for intrusion detection in network traffic. The study was performed on three datasets and showed that word embeddings as a feature extraction technique have great potential in increasing the performance of conventional ML algorithms. It was stated that this technique requires a lot of computational time than other simple techniques but may lead to higher accuracy.

2.6 Multi-class Imbalanced text data

Many ML algorithms assume equal class samples for classification problems. For this reason, many ML algorithms tend to be biased when the equal assumption is not met. A study in (31) was done on dealing with imbalanced data in text classification. The study used data consisting of short text descriptions of work experiences from the Human Resources field and classified them into imbalanced job type categories. The study used resampling techniques like the Synthetic Minority Oversampling Technique (SMOTE) and SMOTE-SVM to handle an imbalanced dataset. Two ML algorithms namely Logistic regression and SVM with linear Kernel were used. After extensive analysis of various text representation techniques, the results showed that SVM with BoW does not seem to be affected by an imbalanced dataset when evaluated on the F1 metric.

In their study, (32) indicates that imbalanced datasets bring challenges when implemented in real-world ML applications. In their paper on the review of methods and application of imbalanced data in classification problems, it is argued that the predictive models get affected when the data is highly imbalanced and the sample size is large. Three methods were explored namely, data, algorithms, and combination of methods. The data approach refers to the random under-sampling and/or over-sampling of class(es) with the lowest records (minority) to match it with the majority class(es). The algorithmic approach is commonly known as the class-weight approach and it uses expert knowledge and approach in handling the ML algorithms

to rebalance classes. It was concluded that the sampling methods are important in this kind of problem but may introduce bias in real-life applications, so hybrid approaches are required.

When dealing with multi-class imbalanced text classification of cyberbullying, (33) used a feature engineering approach to detect cyberbullying on Twitter. This study aimed to classify the severity of the tweet into three categories. The study compared five algorithms, namely NB, SVM, KNN, DT, and RF against two NLP feature extraction techniques BoW and Word2Vec. After the addition of engineered features, the study indicates improvements in results. It was indicated that SMOTE was used to handle the data imbalance. Furthermore, the multinomial LR was used to identify significant features.

2.7 Conclusion

This chapter looked at Text classification applications around different industries and sectors. This is done to gain an understanding of the subject under study.

Chapter 3

Research Methodology

This section describes and explains the data source, text representation techniques, and other ML model approaches used in this thesis. To achieve the goal of this thesis, five ML models are proposed and evaluated on a multi-class text classification problem. The thesis uses real customer complaints data related to financial products and services in the financial services industries.

3.1 Data Description

This thesis is motivated by the need to improve customer satisfaction and experience in one of the large financial services companies. One of the problems faced by customer service companies is to attend and resolve customer complaints effectively and efficiently. According to (13), Customers expect companies to provide fast responses and appropriate services to their problems.

This thesis used real-world customer complaints data from the CFPB data ranging from March 2015 to April 2022 (34). In this data, a customer can complain about a single or multiple product(s) and service(s) they experienced with their financial services company. Table 3.1 shows the summary statistics and data quality. The extracted data had 2,640,478 records (unique Customer ID) with a total of 18 features. For this thesis, only three features are relevant namely *Date received*, *Product*, and *Consumer complaint narrative*. The Date received represents when the complaint was received and in this thesis, it is used to select Training and OOT sample data. All other features are not useful for this task.

Meanwhile, the *Product* variable contains categories of complaints issues that cus-

tomers experienced with their financial services firms. The Consumer complaint narrative feature has the raw verbatim complaint a customer had about the product. A customer can complain multiple times about one of the eighteen *Product* labels because a complaint can only be related to one of the eighteen *Product*. Preliminary data analysis in Table 3.1 shows that the Consumer complaint narrative feature has 65.2% of the data missing. For this reason, all records with missing complaints were dropped. Table 3.2 provides a summary view of Product distribution by consumer complaints narrative by Product after grouping of Product labels.

Table 3.1: Data Quality Analysis

Feature Name	Data Type	Counts	Unique Counts	Missing Percentage
Complaint ID	int64	2,640,478	2,640,478	0.00
Date received	Date	2,640,478	3,803	0.00
Product	Category	2,640,478	18	0.00
Sub-product	Category	2,405,315	76	8.91
Issue	Category	2,640,478	165	0.00
Sub-issue	Category	1,982,680	221	24.91
Consumer Complaint narrative	Text	923,232	820,057	65.04
Company public response	Text	1,096,913	11	58.46
Company	Category	2,640,478	6,364	0.00
State	Category	2,600,909	63	1.50
Zip code	Category	2,600,909	59,103	1.51

There was a Product labeling inconsistency in the data. For example, some labels of the Product feature like *Credit card* and *Credit card or prepaid card* can be found in the *Credit card or prepaid card* labels, so these data investigations necessitated the grouping of some labels, thus resulting in 13 labels. This was required to improve labeling consistencies as that will further help ML algorithms to learn data better. The products with the most complaints are *Credit reporting, repairs, or others* with a contribution of about 45% to the total complaints. The top 4 *Products* with the most complaints contribute to 84% of the complaints. This poses a challenge known as class imbalance in ML problems. Although the imbalance in complaints distribution may reflect the true nature and prevalence of issues in real-world situations, inequality in target classes often poses a big challenge in ML classification problems if not addressed.

Table 3.2: Distribution of Product by Consumer Complaint Narrative Count

Product	Complaint Count	% Distribution
Credit reporting, repair, or other	413,279	44.76
Debt Collection	127,563	18.91
Mortgage	91,085	9.87
Credit Card or Prepaid	90,369	9.79
Checking or saving account	44,349	4.80
Student Loan	30,544	3.31
Money transfer, virtual currency, or money service	22,033	2.39
Vehicle loan or lease	17,343	1.88
Bank account or service	14,885	1.61
Payday loan, title loan, or personal loan	13,523	1.46
Consumer Loan	9,470	1.03
Money Transfer	1,497	0.16
Other Financial service	292	0.03
Total	**876,232**	**100**

3.2 Class Imbalance

Class imbalance can be defined as a large discrepancy between two or more classes of the target variable, where one class is represented by more instances than other(s) (35). This is a problem in ML problems as many ML algorithms assume an equal number of observations in each class (36). However, this assumption is often not met in real-life situations. (37) indicated that the selection of training data is crucial when dealing with imbalanced distribution for improving classification accuracy. Data sampling techniques are known techniques for transforming training datasets to address class-imbalance classification problems. In (38) it is noted that relying on train test split can lead to unrepresentative data, thus leading to poor generalization and unreliable performance estimates of ML models.

Imbalanced data in ML is a problem realized by many scholars and academics. There are several ways in which data imbalance is solved, such as downsampling, upsampling, and class weight optimization (38). Although these techniques aim at modifying the dataset and reducing the model bias, it alter the true underlying distribution of the data and often lead the model(s) to do well during the training and not generalize well on the unseen data. When evaluating the model, the data like the one provided in training must be provided (39). To address this problem,

Stratified Random Sampling (SRS) was introduced to handle the representativeness of training, validation, and OOT data.

Stratified Random Sampling

SRS is a type of sampling in which samples are drawn from the population that is divided into mutually exclusive, non-overlapping groups called strata (40). Each Stratum is sampled independently of all other strata (41). For example, strata will be each class in the Product. Stratified sampling can be divided into 3 parts-stratification, sampling allocation, and sampling. SRS can produce higher precision of estimates when there's high heterogeneity among strata but requires time to plan and implement (40). The allocation is done to reflect the true proportion of the overall data. This is done to have a reflection of the true population distribution of classes per complaint *Product*. This technique is important when dealing with imbalanced datasets to ensure each class is represented accordingly.

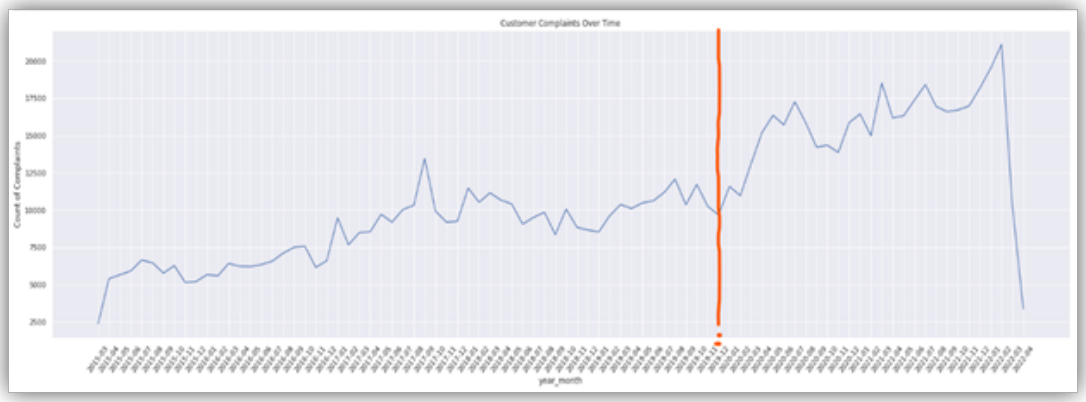
3.3 Training and Validation Dataset

To understand and evaluate the performance of each model vs others, the training, validation, and OOT datasets are created. The dataset from March 2015 to December 2019 was used for creating training and validation datasets, while the data post-December 2019 was used as an OOT performance evaluation dataset. This was designed to emulate the real-world situation where the OOT data will be used to assess how the model would perform with real-time customer complaints. Figure 3.1 shows the volume of complaints from March 2015 to April 2022.

The red line in Figure 3.1 shows the cut-off date used to define Training, validation, and OOT datasets. The training data and OOT samples had 491,082 and 432,150 records respectively. The decision to choose a cut-off date between training and OOT was arbitrary. Furthermore, the training data was divided into training and validation data for ML model-building purposes using a 75:25 split ratio.

Figure 3.2 shows the process followed in the training of models. However, due to the lack of computing resources, the training and OOT data were reduced to a manageable size using a SRS technique respectively. A total of 21,000 complaints were sampled from 491,082 to make training data. The 21,000 training data was split into 75% training and 25% validation datasets. The OOT data was sampled to 8,000 records for a test dataset. The SRS technique was applied to preserve the distribution of the *Product* as seen in the population data.

Figure 3.1: CPFB Consumer Complaints volume distribution over time



3.4 Text Pre-processing

For ML algorithms to understand and process text data, the data needs to be transformed and structured. That is, representing text data in the form of numbers. Text data representation techniques may vary depending on the data type and problems. Available text pre-processing techniques include methods such as vector space models based on TFIDF, BoW, and Word embeddings. The following are techniques used to clean and prepare data.

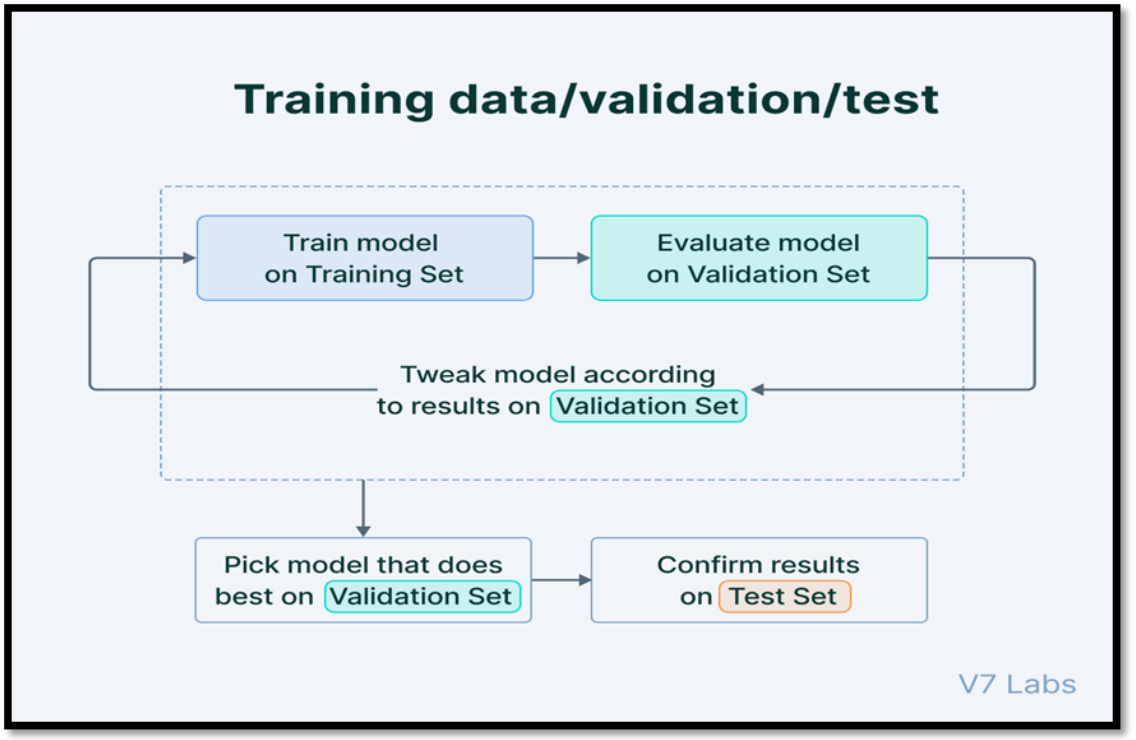
3.4.1 Tokenisation

In NLP, the process of splitting text into a sequence unit of text is known as tokenisation. In the English language, space is used to identify where one word end and the other begin. For this reason, a space delimiter is used to define word tokens. Tokenisation is a necessary step in ML to help machines understand human language by breaking texts into units that are easier to analyze.

3.4.2 Stopwords Removal

In text classification and NLP, stopwords are regarded as common and uninformative words that exist in text documents without much dependence on the associated topic or label (42). In ML text classification, these words are not important as they do not

Figure 3.2: The demonstration of the modelling process for the project (1)



help in differentiating associated labels and thus, require to be removed during the text pre-processing stage (27). (42) further indicated that the removal of stopwords is an important step in text pre-processing as it can improve the statistical importance of each term. This process is known to increase information density(15).

3.4.3 Lower cases

The conversion of all words into lower cases is a necessary step and it works as a word reduction technique. For example, if the words "Wait" and "wait" are not converted to lower cases, machine learning models recognises them as two different words. Converting words to lower cases reduces the dimensionality of the word vectors available from all documents and improves the ML model performance (43).

3.4.4 Stemming

Stemming is an NLP processing technique that produces variants of a base word. In simple words, it reduces all words with the same roots to their stem words (44). This technique has shown great importance in various NLP applications such as text classification, Information Extraction (IE), and Information Retrieval (IR) (45). Although there are many stemming algorithms such as PorterStemmer, LancasterStemmer, SnowballStemmer, and many others, this research used PorterStemmer from the NLTK library in Python. The PorterStemmer is known to be fast, simple, produce the best outputs compared to other stemmers, and has less error rate (46). The following are some examples of words with base root *wait* using PorterStemmer:

- Waits
- Waited
- Waiting

Applying stemming to all words above will result in the same word *wait*. This process has improved many IR systems like search engines (47).

3.4.5 Lemmatisation

Lemmatisation is the process of assembling the inflected part of a word such that it can be recognized as a single word called a lemma (44). Lemmatisation finds the base form of a word by removing inflectional endings and returns the base of a word. The following are examples of lemmatising:

- Rocks — rock
- better — good
- corpora — corpus

Lemmatisation groups words with similar meanings into a single word. thus helping reduce the number of words while preserving the core meaning.

3.5 Feature Extraction

After the text data pre-processing, the resulting data is required to be transformed in a manner that ML models can understand. There are several ways in which text documents can be represented namely, count, predictive, and sequential-based (15). Each approach has its pros and cons. For example, count based approach captures the frequency of the word/term in the document and as such, this approach does not capture context and ignores the order of words. To account for these limitations, a comparison of techniques is used to represent text data namely, BoW, TFIDF, and word embedding.

3.5.1 Vector Space Model: Bag-of-Words

The BoW model is a text representation or feature extraction technique commonly used in NLP and IR problems. The basic idea behind this technique is to transform the occurrence of every word available in a document as a feature for model training and prediction. Although this technique is easy to understand and implement, it disregards the order of words, grammar, and context (48). For example, suppose there are three movie reviews as follow:

- Review 1: This movie is very scary and spooky
- Review 2: This movie is long and slow
- Review 3: This spooky movie is good long

The BoW technique collects individual words from all reviews, ignoring their order and focusing on their presence or absence. Figure 3.3 shows how each review will be represented by the BoW technique.

From Figure 3.3 a review can be regarded (d_i), and each word in the columns as a term (t_i). The BoW text representation assumes that a review d_i in a collection set of reviews $\{d_1, d_2, \ldots, d_m\}$ can be represented by a set of words t_i in $\{t_1, t_2, \ldots, t_n\}$ where n and m denote the total number of words and documents respectively. To understand the relative importance of each t_i to d_i to all other documents, the weighting method is introduced. There are two general weighting methods of the BoW model, namely, Count Vectorize (CV) and TFIDF.

Figure 3.3: Bag of Words (2)

	1 This	2 movie	3 is	4 very	5 scary	6 and	7 long	8 not	9 slow	10 spooky	11 good
Review 1	1	1	1	1	1	1	1	0	0	0	0
Review 2	1	1	2	0	0	1	1	0	1	0	0
Review 3	1	1	1	0	0	0	1	0	0	1	1

Count Vectorizer

The BoW model can be regarded as a feature-generating tool from which different measuring techniques can be derived. The CV is a BoW weight-measuring technique that is used to convert text documents into a vector of words based on the frequency of each word in the entire document(s). This technique gives more importance to words that appear the most in a document(s)

$$\text{Count}(t_i, d_j) = \text{frequency of term } t_i \text{ in document } d_j$$

Where t_i refers to the i-th term in the vocabulary, and d_j refers to the j-th document in the collection of documents. The formula calculates the frequency of the term t_i in the document d_j. This is also known as Term Frequency (TF).

TFIDF

The TFIDF is a statistical technique used to assess the relative importance of a word for a document in a set of documents called corpora. It measures how important a term is within a document relative to a collection of documents (49). The TFIDF works by combining TF and Inverse Document Frequency (IDF).

Inverse Document Frequency

The Inverse Document Frequency (IDF) looks at how common/uncommon a word is amongst the corpus. Words with a small percentage are given higher importance than words that are common across the documents (50). This TFIDF technique operates under the following intuitions:

- The more a term occurs in a document, the more representative it is of the text.
- The more text word occurs in a document, the less significance it carries.

The TFIDF is a product of TF and IDF given as follow:

$$TF\text{-}IDF = TF \times \log\left(\frac{N}{DF}\right)$$

where N is the number of documents in the corpus and DF is the number of terms in a document. The TF and IDF techniques do not account for grammar, context, and order of the words (48), the embedding technique was introduced.

3.5.2 Word Embedding

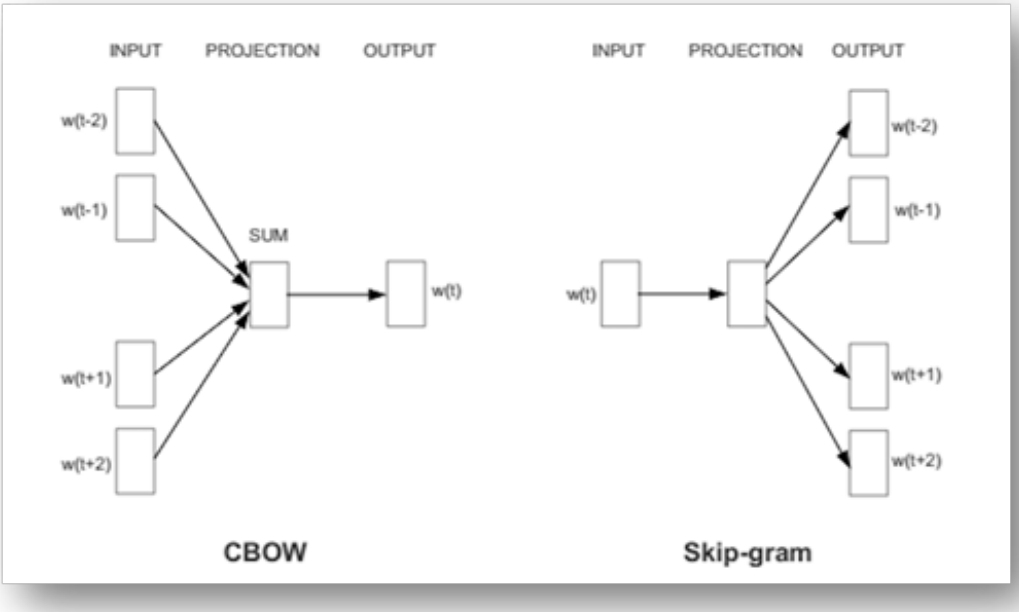
The word embedding technique is introduced to account for the shortfalls and limitations of the BoW text representation methods mentioned above. Word embedding is a collection of statistical language modeling techniques that map words and phrases into a vector of real numbers (51). This technique ensures that words with the same meaning have similar text representation and it captures semantic and syntactic information of words.

Word2vec

Word2vec is a word embedding technique that uses a collection of text documents also known as corpus as inputs and words represented as vectors. The word2Vec word embedding uses either of the two architectures namely the Continuous Bag of Words (CBoW) and the Skip-gram model (51). The CBoW uses context to predict the next word, while Skip-gram uses a word to predict the context target Figure 3.4. The output of this model is a set of word vectors associated with words in the input corpus. This technique helps bring context by associating words with a similar meaning close to another through the created vector space. The resulting word vector space indicates the semantic of the word2vec model.

The Word2Vec works by estimating the likelihood that words will occur together. Although Word2Vec is not a deep neural network model, it does create neural em-

Figure 3.4: The Word2Vec word embedding architecture (3)



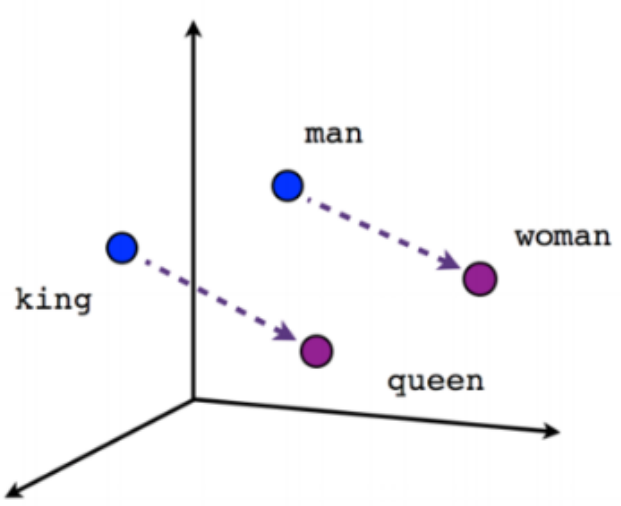
bedding of words that occurs together. Word2vec with Skip-gram can produce more accurate results on large datasets (52).

Figure 3.5 demonstrates the basic underlying assumption of Word2Vec that words with similar context share similar meanings and vice versa, on a three dimensional plane. The words King and man share a similar context. Similarly, woman and queen have a similar context. However, king is closer to queen as they occur more frequently together than king and woman, hence the long distance between words on the graph. Similarly, man is close to woman than it is to queen. However, queen and woman are in the same vector as king and man. This shows that Word2Vec can be used to calculate similarity between words and have them represented as vectors.

3.6 Feature Engineering

ML algorithms generally require input data to learn patterns and generate outputs. These input data are commonly known as features. Feature engineering is a data pre-processing technique used to create additional features from existing raw data.

Figure 3.5: Word2Vec demonstration on a 3-dimensional plane (4)



This technique aims at improving ML model performances. The success of this technique depends on many factors such as problem type, domain knowledge, data transformations, and iterative trial and error process. Figure 3.6 has a column named "Length of the review" which is generated from the presence of words available in a review. This is an example of an engineered feature extending from figure 3.3.

Figure 3.6: Bag of Words with Engineered Feature (2)

	1 This	2 movie	3 is	4 very	5 scary	6 and	7 long	8 not	9 slow	10 spooky	11 good	Length of the review(in words)
Review 1	1	1	1	1	1	1	1	0	0	0	0	7
Review 2	1	1	2	0	0	1	1	0	1	0	0	8
Review 3	1	1	1	0	0	0	1	0	0	1	1	6

The feature engineering technique was explored to assess the impact it can have in improving the model performances. Features like *number of characters*, *Number of stopword*, *Number of punctuations*, *Number of digits*, *Number of word with Capital Letters*, *Number of word with Capital Letters*, *Number of word with Capital Letters* a complaint contained are generated. Feature engineering technique applies to vector space text representation and may add to the curse of dimensionality faced by these vector space representations.

3.7 Machine Learning Classifier

Text Classification can be implemented using various ML classifiers. Five ML algorithms that are commonly used in multi-class text classification were considered namely: Multivariate LR, KNN, SVM, DT, and Light Gradient Boosting (LightGB).

3.7.1 Logistic Regression Classifier

LR is the process of modeling the probability of a discrete outcome given an input variable (53). It is a supervised ML algorithm that is used to solve binary or multi-class classification problems. It can also be extended to model for multiple variables known as multivariate LR. It uses the logistic function to estimate the probability of an event occurring by having the log odds of an event be a linear combination of one or more independent variables (54). Mathematically, the multivariate logistic regression is derived from a linear regression equation is shown as follows:

$$Y = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \beta_3 x_3 + ... + \beta_n x_n + \epsilon \tag{3.1}$$

Where β_0 is an intercept of the model. In this case, Y represents the outcome, and covariates (X) are features derived from the feature representation techniques (55). Mathematically, it can be shown that if you apply logit, a special function defined as logit(x) = log(x) - log(1-x), to an equation in (3.2), the multivariate logistic regression results as follows:

$$logit(P[Y = 1|X_1, ..., X_n]) = logit(\frac{P_i}{1 - P_i}) \tag{3.2}$$

Where P_i is the transformation in the regression model given by:

$$P_i = \frac{e^(\beta_0 + \beta_1 x_1 + \beta_2 x_2 + \beta_3 x_3 + ... + \beta_n x_n)}{1 + e^(\beta_0 + \beta_1 x_1 + \beta_2 x_2 + \beta_3 x_3 + ... + \beta_n x_n)} \tag{3.3}$$

From the above equations, it can be shown that

$$P(Y = 1|X) = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \beta_3 x_3 + ... + \beta_n x_n \tag{3.4}$$

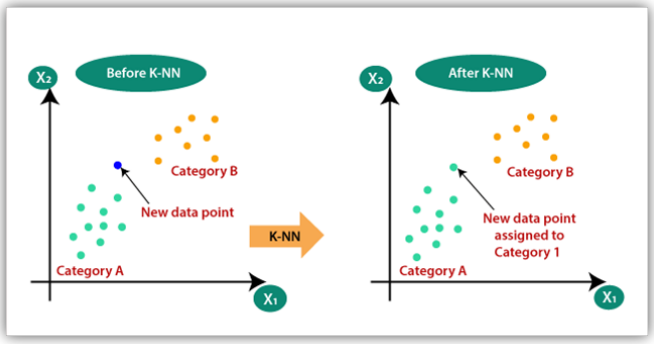
Where $P(Y = 1|X)$ denotes the probability of the positive class given the input features X and model parameters β (54). The model parameters β are estimated

using maximum likelihood estimation (54), and various optimization algorithms, such as gradient descent, can be employed for this purpose (56). The above equation can be extended to a multiclass problem known as multinomial logistic regression. It should be noted that multinomial LR can overfit when the number of features are greater than the number of observations (57). The linearity assumption in multinomial LR can be a challenge when the problem faced is not a linearly problem.

3.7.2 K-Nearest Neighbour (KNN) Classifier

The KNN classifier is a non-parametric and distanced based algorithm that classifies new documents based on how close they are to the seen documents and k neighbours in the training set (58) (5). This algorithm is often regarded as a lazy learner because, during its training, it stores the data points and performs action during the classification time (59). Figure 3.7 shows how the KNN algorithm works on a two-dimensional plane.

Figure 3.7: KNN classifier during classification in a two-dimensional plane (5)



Suppose there are two Products A and B such that a new document d (denoted in blue in Figure 3.7) must be classified into A or B. The KNN algorithm will calculate how close d is to all other seen documents belonging to categories A and B, and assign d to the closest category using k neighborhood points. In Figure 3.7, d is close to category A and thus it is assigned to A.

The KNN algorithm operates in the following steps:

1. Select the number of nearest neighbours k.

2. Calculate the distance between the objects with all data points in the training set.

3. Take k closest objects and count the number of classes belonging to each point.

4. Assign the new data point to the most frequent category.

The closeness of the data point can be calculated using Euclidean distance or Manhattan distance from n feature vectors representing documents (5). However, this can be a problem when dealing with binary or categorical features as such data are non-euclidean data (60). Furthermore, text data has dimensionality since each word is considered a feature and this may lead to overfitting or poor performance (61).

3.7.3 Support Vector Machine (SVM) Classifier

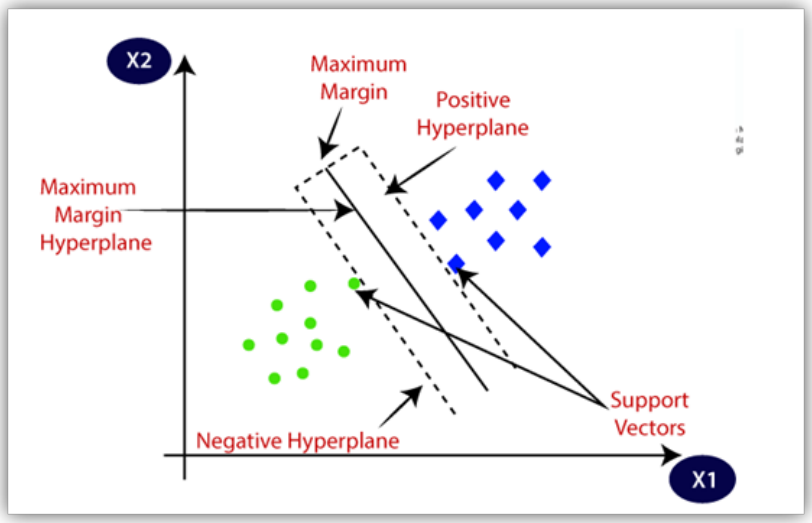
SVM is a supervised ML algorithm that can be used for both regression and classification (59). It is one of the most effective algorithms in handling problems with multi-dimensional data (28). SVMs aim at creating the best decision boundary called hyperplane that can be used to distinguish n-dimensional space into k number of classes. They are also considered the most accurate classifiers for text classification problems (6). It uses extreme points of each class called support vectors to help create hyperplanes. The objective of SVMs is to find the maximum margin distance that obtains good classification results.

Figure 3.7 shows the SVM algorithm when applied to a two-dimensional plane for a binary classification problem. In the cases of three classes, the hyperplane will be a 2-dimensional plane (7). The negative hyperplane is the highest boundary line for the "green' class and the positive hyperplane is the lowest boundary for the "blue" class. The distance between hyperplanes and support vectors is called the margin. SVMs can be extended to multiclass classification problems. Generally, there are two types of classification using SVMs namely; linear and non-linear data classification.

Linear SVM

Linear SVM in an efficient technique for high-dimensional data applications like document classification (62). Linear SVM is an SVM algorithm that is used to classify data that is linearly separable. It uses a Maximum Margin Hyperplane (MMH) to separate two data points (59). Figure 3.7 shows an example of data that is linearly separable.

Figure 3.8: Support Vector machine in a two-dimensional plane (6)



Non-Linear data classification

Data that cannot be separated with linear hyperplane is known as non-linear data (63). Figure 3.8 shows an example of data that is not linearly separable, meaning a straight line cannot be used to separate the two classes. SVM algorithm uses the help of kernel functions to separate non-linear data (59)(63). There exist many types of kernels such as Polynomial, Gaussian Radial basis function (RBF), sigmoid, and many others. This method has to find the right kernel to transform one class of data to a higher dimension and the decision surface to classify data points (59).

3.7.4 Decision Tree Classifier

The DT classifier is an ML-supervised learning technique used for both classification and regression problems. It is a tree-structured classifier, whose internal nodes represent the features of a dataset, sub-trees represent the decision rules, and each leaf node predicts an outcome (64).

Figure 3.10 shows how the decision trees works. It divides data (root node) into smaller sub-trees based on their features until there is only one class left. Each tree has a logical decision like yes or no that divides data into categories at every step. The resulting decision is known as the leaf node. Some of the most popular methods used to choose the best attribute at each node are information gain and gini

Figure 3.9: Non-linear data in a two-dimensional plane (7)

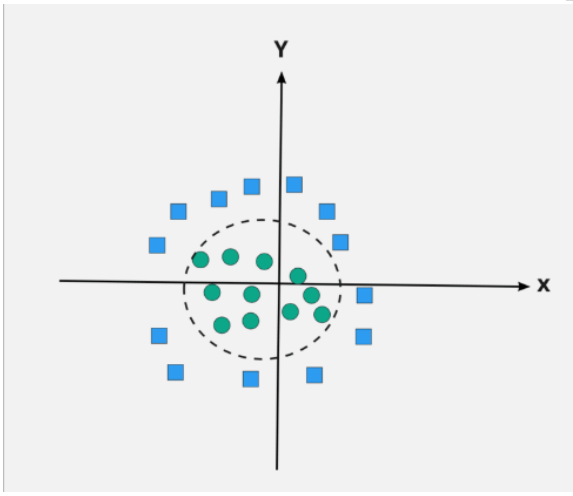
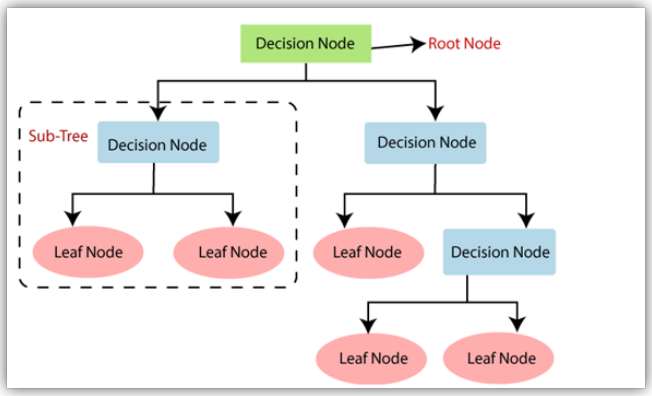


Figure 3.10: Decision Tree Diagram and Terminology (6)



impurity (64)(65). These metrics are important in determining the optimal feature for splitting the data (64).

Entropy

Entropy is a statistical measure denoted by H(D), where D is a given dataset. It measures the impurity or disorder within the dataset elements. In decision trees, entropy gauges the unpredictability of the outcome on the data points class labels. A dataset with no impurity, meaning all elements are of a single class, has an entropy value of 0, indicating total certainty. Conversly, when the dataset shows a uniform distribution across different classes, the entropy reaches its highest value, reflecting maximum disorder. A mathematical formula for entropy is given as follow:

$$H(D) = -\sum_{i=1}^{n} p_i \log_2(p_i) \tag{3.5}$$

Information Gain

Information gain is a statical metric used to determine the effectiveness of a feature in reducing entropy (64). It is essentially the change in entropy that occurs by partitioning the data on an attribute. By calculating the information gain for each attribute, we can determine which feature is the most informative and should be used to split the data at each node in a decision tree. The formula for calculating information gain is as follows:

$$\text{Information Gain} = H(D) - \sum_{v=1}^{V} \frac{|D_v|}{|D|} H(D_v) \tag{3.6}$$

In this formula:

V is the number of values (unique outcomes) the feature can take. Dv denotes the subset of data points for which the feature has the vth value, and —D— denotes the total number of data point in dataset D.

A higher value of information gain indicates a more significant reduction in entropy, meaning the feature provides a clearer distinction between the classes. Decision tree algorithms use this metric to select the sequence of features upon which to split the data, aiming to maximize information gain and thus, create the most effective model. There are types of Decision tree algorithms such as ID3, C4.5, and CART. In this research, the sci-kit learn DT was used, which essentially implements an optimized version of the CART algorithm.

3.7.5 Light Gradient Boosting (LightGBM) Classifier

Boosting Methods is a family of ensemble learning that is known to improve the learning of a model by combining many weak classifiers to build a strong classifier that can predict with more accuracy (66). LightGB is an extension of the Extreme Gradient Boosting (XGBoost) algorithm. Considering an increasing amount of data and many features, LightGBM was proposed to address issues of efficiency, scalability, training computational power, and speed when dealing with high-dimensional

feature space (67). It uses two algorithms namely, Gradient-based One-Side Sampling (GOSS) and Exclusive Feature Bundling (EFB). GOSS handle a small set of samples with large gradients and combine them with other samples with a small gradient to evaluate the information gain (36)(67), while EFB plays a built-in dimensionality reduction and efficiency role. Generally, boosting algorithms are less prone to overfitting, reduce high bias, and can handle missing data.

3.7.6 Evaluation Metrics

Performance evaluation metrics are important and necessary when the aim is to evaluate and compare various classification models. Although there exist many metrics (like accuracy, specificity, and CohenKappa) many of these metrics are derived from the confusion matrix table. However, some metrics can be biased when the class distribution is imbalanced (68) and thus do not reflect the true performance of classifiers.

Confusion Matrix

The confusion matrix is a table used to measure the performance of a classification model on the true labels against the predicted labels. Figure 3.10 below shows the confusion matrix of a binary vs multiclass.

Figure 3.11: Confusion Matrix for binary and multi-class classification problems

		Predicted Class	
		Positive	Negative
Actual Class	Positive	TP	FN
	Negative	FP	TN

(a)

		Predicted Class			
		C_1	C_2	...	C_N
Actual Class	C_1	$C_{1,1}$	FP	...	$C_{1,N}$
	C_2	FN	TP	...	FN
	...	...	...	...	...
	C_N	$C_{N,1}$	FP	...	$C_{N,N}$

(b)

Where:

- **TP** = True Positive, when the model has predicted the positive class as the actual positive class.

- **FN** = False Negative, when the model has predicted a negative class, but in actuality, it is a positive class.
- **FP** = False Positive, when the model has predicted a negative class, but in actuality, it is a negative class.
- **TN** = True Negative, when the model has predicted a negative class, but in actuality, it is a negative class label.

The choice of a metric depends on the kind of problem data hand and the type of data in question. Metrics like weighted accuracy, specificity, sensitivity, and F1-score are considered.

Accuracy

The accuracy metric measures the proportion of correct predictions relative to the total predictions made. The formula for binary classification is given by the following:

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} \tag{3.7}$$

The accuracy metric is often useful when the class distribution is balanced, and classes are of equal importance. In the cases when it is not, accuracy can hide the model's true performance. For example, suppose we have a binary classification to predict whether a tweet is fake or not. If out of 100 tweets, 90 are fake and others are not. If it is equally important to flag a tweet as fake or not. If the model classifies all 100 tweets as fake, the model's performance accuracy would be 90%, this measure wouldn't give a true reflection of how accurately the model is performing. This means the model is as good is predicting every tweet as fake. For this reason, a balanced-accuracy metric is important to look into how well the model is in classifying each class.

Precision

Precision is a metric that measures the number of correct positive predictions. It attempts to answer the question: what proportion of positive instances was actually correct ()? The formula for the binary classification problem is given below:

$$Precision = \frac{TP}{TP + FP} \tag{3.8}$$

The formula can be extended to the multi-class problem. In the scikit-learn package, we use the macro average weighting for multi-class calculations. The macro precision measures the average precision per class and gives a weighted average. Macro precision is given as follows:

$$Macro\text{-}precision = \frac{Precision_1 + Precision_2 + ... + Precision_n}{n} \tag{3.9}$$

Where n is the total number of classes.

Recall

The Recall of a metric also known as sensitivity is a measure of the number of correct positive predictions out of all the positive predictions made by the model. It attempts to answer the question: what proportion of actual positives was identified correctly? The formula for binary classification is as follows:

$$Recall = \frac{TP}{TP + FP} \tag{3.10}$$

Similarly, this formula can be extended to the multi-class problem by taking the arithmetic mean of recalls for all the classes.

$$Macro\text{-}Recall = \frac{Recall_1 + Recall_2 + ... + Recall_n}{n} \tag{3.11}$$

Where n is the number of classes.

Specificity

Specificity is used to evaluate ML model performances. It measures the proportion of true negatives that are correctly identified by the model (69). In a binary classification model, the mathematical formula for specificity is given as follow:

$$Specificty = \frac{TN}{TN + FP} \tag{3.12}$$

Weighted Accuracy

The weighted accuracy commonly known as the balanced-accuracy metric is a useful performance evaluation measure as it avoids inflated performance estimates on the

targeted classes when the data is imbalanced (10). This measure considers the classifier's performance in every class. The balanced accuracy is given as follows:

$$Balanced\text{-}Accuracy = \frac{Specificity + Recall}{2} \tag{3.13}$$

The above measures differentiate the correct classification of class within classes (). The balanced-accuracy for multi-class is a weighted average as follows:

$$Balanced\text{-}Accuracy = \frac{(\frac{Specificity_1 + Recall_1}{2}) + (\frac{Specificity_2 + Recall_2}{2}) + ... + (\frac{Specificity_n + Recall_n}{2})}{n} \tag{3.14}$$

Where n is the number of classes.

F1-Score

The F1-score is an ML metric used to evaluate classification models. It is derived from two metrics namely precision and recall. F1-score is the harmonic mean of precision and recall. Below is its formula:

$$F1\text{-}score = 2 * \frac{Precision * Recall}{Precision + Recall} \tag{3.15}$$

The above metric can be extended to a multi-class problem. The following is a macro F1-score formula:

$$Macro\text{-}F1\text{-}score = \frac{(2 * \frac{Precision_1 * Recall_1}{Precision_1 + Recall_1}) + (2 * \frac{Precision_2 * Recall_2}{Precision_2 + Recall_2}) + ... + (2 * \frac{Precision_n * Recall_n}{Precision_n + Recall_n})}{n} \tag{3.16}$$

Where n is the number of classes. The macro F1-score tries to find the balance between precision and recall in every class. A summary of the above metrics will be presented throughout the results and analysis of the study.

3.8 Hyperparameter Optimisation

ML algorithms require a set of parameters whose values must be set and chosen carefully before the model training (70). This is done to control the algorithm's learning process and improve model performance. These sets of parameters are commonly known as hyperparameters. Hyperparameter optimization is the process of choosing the best parameters for a given algorithm, which significantly impacts the model's performance (70)(71). Hyperparameters play a critical role in analyzing predictive performance in machine learning models. However, this process can be tedious and resource-intensive. In this research, a randomised grid-search technique was used to find the best-performing parameter.

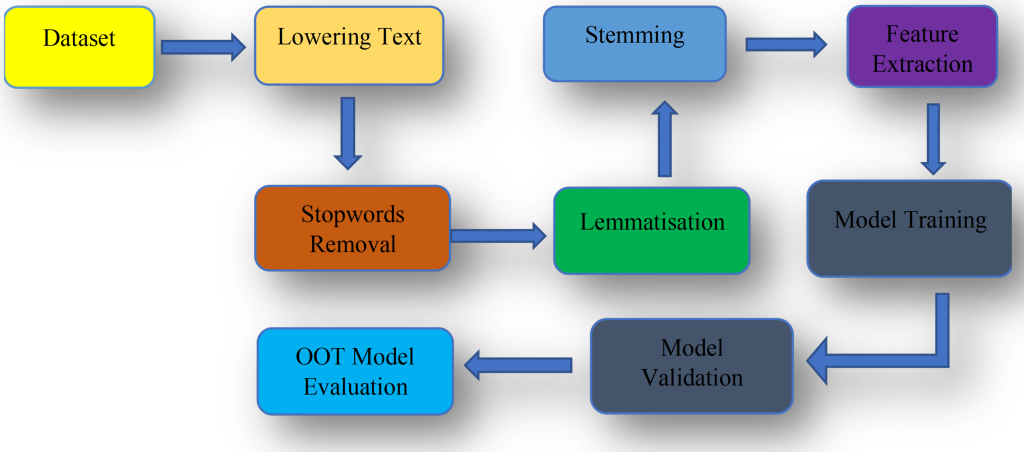
Randomised Grid-Search

A randomised grid search is a hyperparameter optimisation technique used to explore a set of predefined set of hyperparameter values for ML model (71). Unlike the traditional grid-search which makes a complete search of a given subset of hyperparatmeter space, a randomised grid-search overides a complete selection by their random selection (71). A randomised grid search is fast and can outperform traditional grid search, especially when a small number of parameters affect the performance of the ML algorithm.

3.9 Conclusion

This section outlin the steps and techniques followed in the research. Figure 4.2 shows how each step fits into each other.

Figure 3.12: ML text pre-processing and classification process



Chapter 4

Results and Discussion

This section presents the results of five machine learning algorithms applied to three different NLP feature extraction techniques. Furthermore, the performance of models is explored on algorithms trained with engineered features compared to normal feature extraction techniques. The performance of the final model was evaluated using the balanced-accuracy metric on the OOT sample data. The balanced-accuracy metric is used as the first metric of choice. The model with the highest balanced-accuracy score is deemed the best-performing model recommended for deployment.

4.1 Results and Analysis

One of the common techniques to handle large datasets, especially when computing resources are limited is data reduction. There are many ways to reduce data such as random sampling, under-sampling, stratified sampling, a. In this research, the original data of 921,333 valid complaint records was hard to process in an HP Intel Core i7 and 8GB RAM laptop. For this reason, stratified sampling was used to reduce the data based on the *Product* feature name. SRS is important when we try to preserve the distribution of the sampled data according to the original distribution of a feature of interest. Any sampling method aims to create sample data that can represent the true population so that inferences and estimates can be drawn and made about the underlying true population distribution.

The data for this research was reduced from 921,333 to 21,000 record complaints. The sampled data included the training, validation, and testing datasets. A total of 13,000 complaints record (which were selected between 19 March 2015 and

31 December 2019) were used for training and validating the model, while 8,000 complaints (selected from 01 January 2020 to 22 April 2022) were reserved as OOT sample data for testing the models' performance. ML supervised learning algorithms were then trained using these data. All samples were chosen through trial and run by assessing the processing time. I eventually decided to stick with 21,000 records for easy processing of data.

Table 4.1: Training, Validation, and testing data split by Product

Topic	Training Sample	Validation	OOT Sample
Bank account or service	300	94	242
Checking or savings account	393	112	311
Consumer Loan	192	59	154
Credit card or prepaid card	1,047	342	855
Credit reporting, repair, or other	3,489	1,150	2855
Debt collection	2,077	755	1743
Money transfer, virtual currency, or money service	152	59	130
Money transfers	32	8	24
Mortgage	1207	422	1,002
Other financial service	6	2	5
Payday loan, title loan, or personal loan	160	57	134
Student loan	516	151	411
Vehicle loan or lease	179	39	134
Total	**9750**	**3250**	**8000**

Table 4.1 shows the distribution of training, validation, and OOT testing dataset. The training and validation dataset was created using a split ratio of 75:25 from the 13,000 complaints respectively. This was created using the train-test-split function from the sci-kit learn package. The OOT testing sample was chosen to be 8,000 complaints to evaluate the models' performance during the post-training period.

The complaints text was cleaned first before modeling. This was achieved by converting texts to lower cases, removing common words, tokenising, and lemmatising words. Three different feature extraction methods were used namely, TFIDF, CV, and Word2Vec. Therefore, five algorithms were trained and evaluated to see which one performed better with each NLP feature extraction technique.

4.2 Model Evaluation with Non-Engineered Features

In this section, the presentation of three different feature extraction methods with and without the engineered features is analysed. The evaluation of models' performance is done using two datasets namely validation and OOT sampled data.

4.2.1 Performance on Validation dataset

The validation dataset is the data reserved out of the training dataset to evaluate how the model performs during training while tuning the model hyperparameters. This helps provide an unbiased evaluation of the model. The table 4.2 shows the models' performance results against the CV feature extraction technique when evaluated on the validation data. The balanced accuracy is used to identify the best-performing model.

Table 4.2: Evaluation results of Count Vectoriser feature extraction on validation data.

Model Name	Balanced Accuracy	Precision	Recall	F1-Score
LightGBM	0.4608	0.7309	0.7323	0.7299
Logistic Regression	0.4257	0.7042	0.7194	0.7092
LinearSVM	0.4238	0.6946	0.7043	0.6969
Decision Tree	0.3127	0.5981	0.6065	0.6011
KNN	0.0916	0.3993	0.3782	0.2651

Table 4.2 shows that Light Gradient Boosting Model (LightGBM) is the best-performing model according to all five metrics. It achieved a balanced-accuracy of 46.08% and a weighted F1-Score of 72.99%. The second-best performing model is LR with the best score across all metrics. The LR model obtained a balanced-accuracy score of 42.57% and a weighted F1-score of 70.92%. The KNN model is the lowest performing across all metrics. The KNN model achieved the lowest balanced-accuracy of 9.16% and a weighted F1-score of 26.51%. The above models were trained on the CV feature extraction technique with no additional features added. Table 4.3 shows the summary results of the models' performance when trained on the TFIDF feature extraction method with no additional features.

Table 4.3 shows that the LightGBM is best-performing according to the balanced-

Table 4.3: Evaluation results of TFIDF feature extraction on validation data.

Model Name	Balanced Accuracy	Precision	Recall	F1-Score
LightGBM	0.4569	0.7271	0.7363	0.7295
Logistic Regression	0.4369	0.7511	0.7671	0.7479
LinearSVM	0.3668	0.7367	0.7437	0.7129
Decision Tree	0.3134	0.591	0.5932	0.5917
KNN	0.2978	0.6716	0,6215	0.5925

accuracy metric when trained with TFIDF. It achieved 45.69% balanced accuracy and an F1-Score of 72.79%. The LR is the second best-performing model according to balanced accuracy. The LR achieves a balanced-accuracy score of 43.69% and a weighted F1-score of 74.79%. However, the results look different when models are evaluated with a weighted F1-score as the first metric of choice, the LR gets better performance. The KNN model is the least-performing model according to balanced-accuracy metric. It achieved a balanced-accuracy of 29.78%.

Table 4.4 shows the results of a Word2Vec word embedding feature extraction technique on the validation data. The word embedding techniques are meant to curb the curse of the dimensionality problem. For this reason, the feature engineering technique will not apply to this technique.

Table 4.4: Evaluation results of Word2Vec feature extraction on validation data.

Model Name	Balanced Accuracy	Precision	Recall	F1-Score
Logistic Regression	0.3729	0.6782	0.7000	0.6825
LightGBM	0.2631	0.5725	0.6080	0.5866
LinearSVM	0.2432	0.5517	0.6166	0.5732
KNN	0.2161	0.4985	0.5569	0.5112
Decision Tree	0.2017	0.4825	0.4797	0.4808

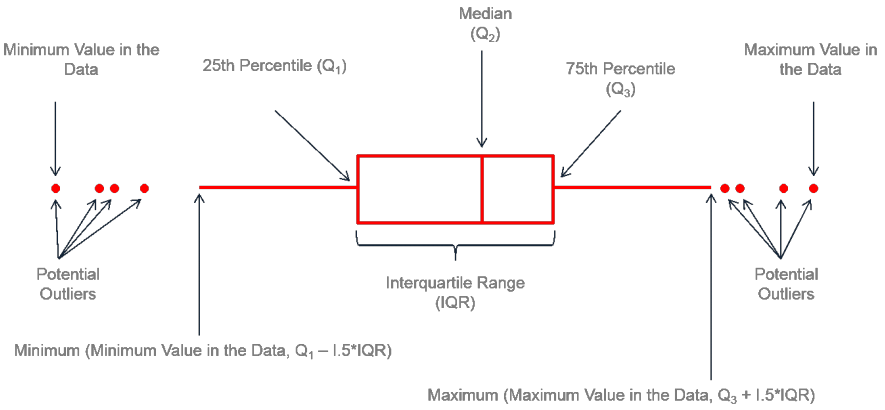
Table 4.4 shows that a LR model performs best according to all five metrics. It achieved a balanced-accuracy score of 37.29% and a weighted F1-score of 68.25%. The LightGBM is the second-best performing model according to balanced-accuracy and weighted F1-score. It achieved a balanced-accuracy of 26.31% and a weighted F1-score of 58.66%. The DT model is performing the least across all metrics.

Box-plot Graph

The box-plot graph is a graphical representation of the distribution of numerical

data. It uses a five-number summary, also known as quartiles. This technique is widely used for detecting outliers and/or comparing distribution between groups, and outliers and/or

Figure 4.1: Boxplot representation (8)



outliers and/or comparing distribution between groups, and visualizing the spread and central tendency of the data.

Figure 4.2 compares the models' results when evaluated on balanced accuracy across the three feature extraction techniques. The LR model scored very high on average compared to all other models across all feature extraction techniques. However, it does perform lower than LightGBM on two out of three occasions of feature extractions applied. The KNN model has relatively lowest performance across all feature extraction techniques. However, it remains to be seen which combination of model and feature extraction technique produces the best performance on the validation dataset. Figure 4.2 shows the models' performance per feature extraction technique when evaluated on the validation dataset using the balanced-accuracy metric.

Figure 4.3 shows that LightGBM is performing better than all models when trained with CV. It achieved a balanced-accuracy score of 46.08%. The LightGBM is the second-best performing model when trained with the TFIDF feature extraction technique. It achieved a balanced-accuracy score of 45.69%. The KNN model is the least-performing model when trained with the CV and NLP feature extraction technique.

The biggest test of many ML models is the performance on the OOT sample data or unseen dataset. This is to gauge whether the algorithms have learned real patterns over noise or whether models are robust enough to withstand time. To accurately

Figure 4.2: Balanced-accuracy score distribution per model across all feature extraction techniques evaluated on the validation dataset.

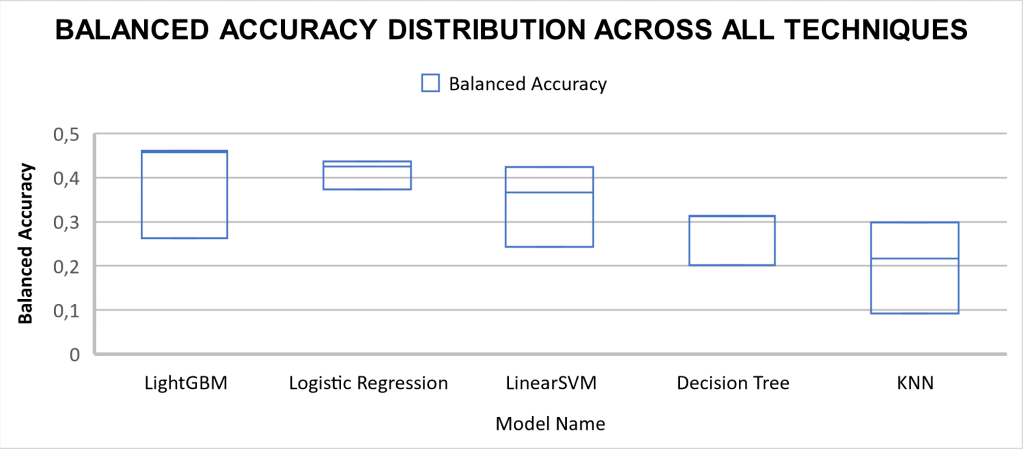
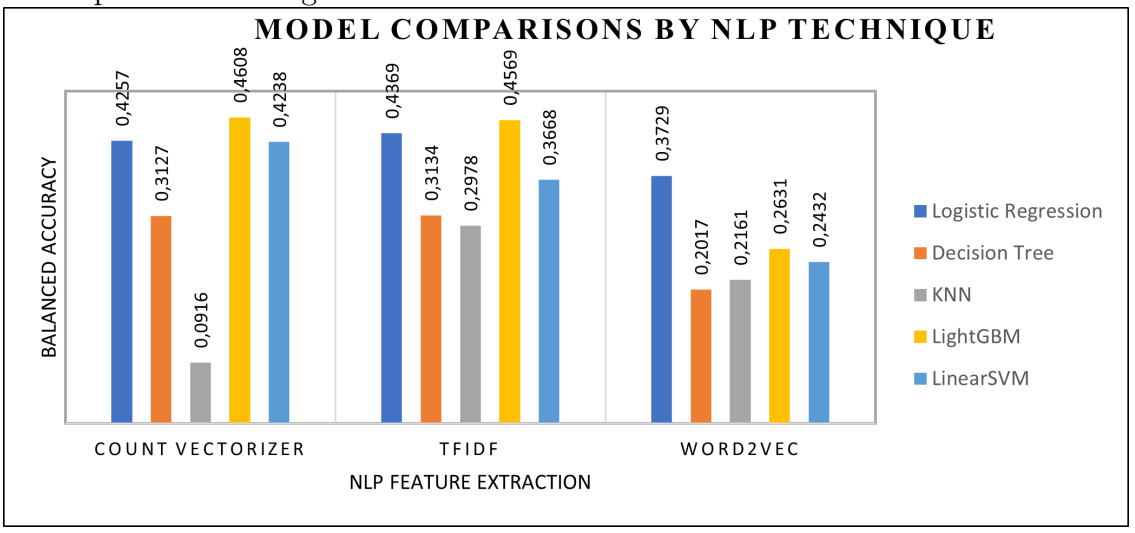


Figure 4.3: Models' balanced-accuracy score performance per feature extraction techniques with no engineered features when evaluated on the Validation dataset.



choose the best model, an evaluation of the five models on each of the three NLP feature extraction techniques is performed on the OOT sample data.

4.2.2 Model Evaluation on OOT Sample Data

The OOT model validation is a technique in ML used to model the time-relevancy of data and the model's application. The test evaluation data contains data from an

entirely different period. Table 4.5 shows the results of models trained on the CV feature extraction technique and evaluated on the OOT complaints data.

Table 4.5: Evaluation results of Count Vectorizer feature extraction on OOT data

Model Name	Balanced Accuracy	Precision	Recall	F1-Score
Logistic Regression	0.8667	0.9184	0.9175	0.9174
LinearSVM	0.8568	0.9131	0.9129	0.9126
LightGBM	0.8314	0.8757	0.8720	0.8724
Decision Tree	0.7168	0.8100	0.8098	0.8091
KNN	0.1829	0.6410	0.4702	0.3849

Table 4.5 shows that LR model is best-performing according to all metrics. It achieved a balanced-accuracy of 86.67% and a weighted F1-score of 91.74%. The second-best performing model is LinearSVM with the second-best score across all metrics. The LinearSVM achieved a balanced-accuracy score of 85.68% and a weighted F1-score of 91.26%. The KNN is the least-performing model across all metrics. It achieved a balanced-accuracy score of 18.29% and 38.49% weighted F1 score. The LightGBM model, which was the top performing model on validation data, has dropped to 3rd with 83.14% balanced-accuracy and 87.24% weighted F1-score. Table 4.6 shows the results of the models that are trained with the TFIDF feature extraction technique when evaluated on the OOT sample data.

Table 4.6: Evaluation results of TFIDF feature extraction on OOT data

Model Name	Balanced Accuracy	Precision	Recall	F1-Score
Logistic Regression	0.8729	0.9384	0.9381	0.9375
LightGBM	0.8337	0.8829	0.8818	0.8816
LinearSVM	0.7545	0.9239	0.9226	0.9207
Decision Tree	0.6916	0.7880	0.7866	0.7870
KNN	0.5001	0.7912	0.7631	0.7525

Table 4.6 shows that LR is a top classifier across all four metrics. It achieved a balanced-accuracy score of 87.29% and a weighted F1-score of 93.75%. It is followed by the LightGBM with the second-best performance on balanced-accuracy of 83.37%. The LinearSVM model is 3rd best-performing classifier when evaluated on balanced accuracy, with a 75.45% score. However, the weighted F1-score of LinearSVM is better than LinearSVM. The KNN model is performing the least with 50.01% balanced accuracy. LinearSVM has improved as compared to the performance it produced on the validation data. Table 4.7 shows the results of models

trained with the Word2Vec feature extraction technique and evaluated on the OOT sample dataset.

Table 4.7: Evaluation results of Word2Vec feature extraction on OOT data

Model Name	Balanced Accuracy	Precision	Recall	F1-Score
Logistic Regression	0.4475	0.6930	0.6899	0.6842
LinearSVM	0.4267	0.7066	0.7210	0.7076
LightGBM	0.3485	0.5685	0.5548	0.5607
Decision Tree	0.3248	0.6374	0.6211	0.5962
KNN	0.2503	0.5560	0.6189	0.5783

Table 4.7 shows that LightGBM is the best-performing model when evaluated on the balanced-accuracy score. It achieved a balanced-accuracy score of 44.75% and a weighted F1-score of 68.42%. The second best-performing model is LR with a balanced accuracy score of 42.67% and a weighted F1-score of 70.46%. The LinearSVM is the least-performing model with a balanced-accuracy score of 25.03%. Figure 4.6 shows the models' balanced-accuracy score distribution per feature extraction technique on the OOT sample data.

Figure 4.4: Models balanced accuracy score on NLP feature extraction techniques with no additional engineered features when evaluated on the OOT data.

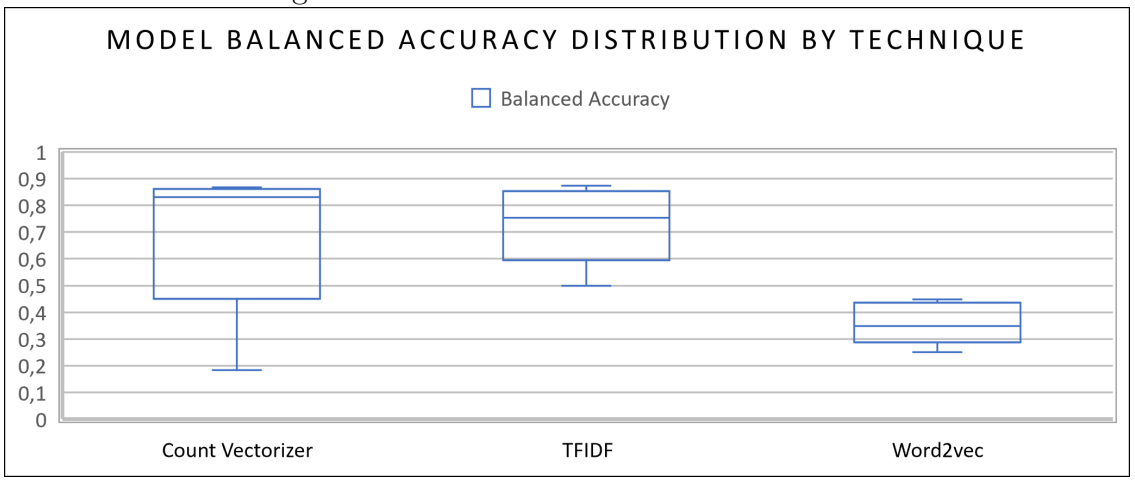


Figure 4.6 shows that the TFIDF models' score distribution is relatively higher on average compared to the other two feature extraction techniques. The models trained with Word2Vec show a relatively low balanced-accuracy score distribution compared to TFIDF and CV. Figure 4.7 shows the models' performance comparison results

on the balanced-accuracy score per feature extraction technique when evaluated on the OOT sample data.

Figure 4.5: Models performance comparison on the balanced-accuracy score per Feature Extraction Technique with no engineered features when evaluated on OOT data.

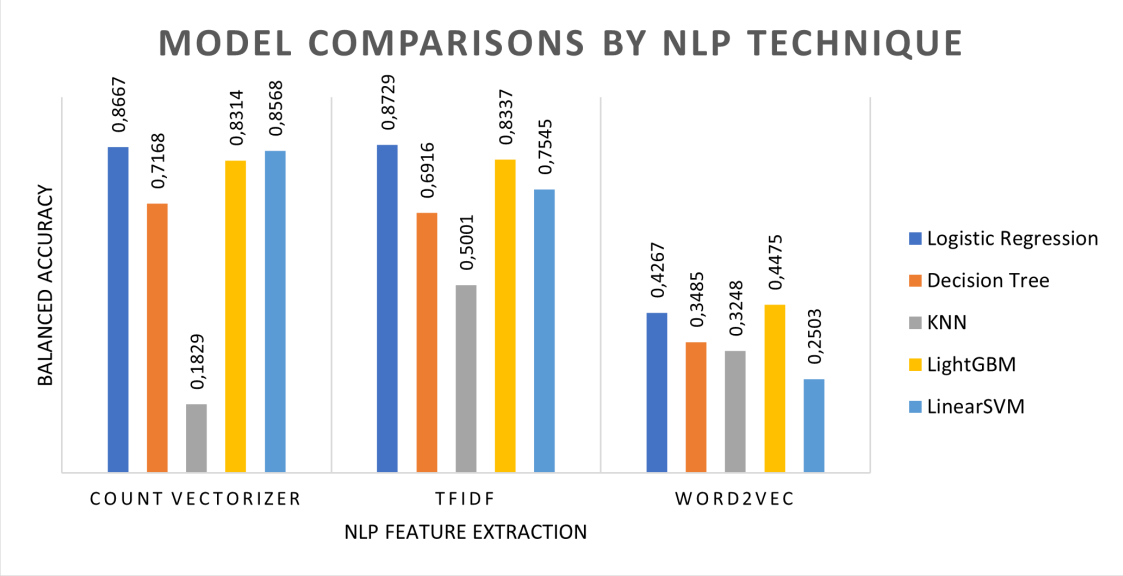


Figure 4.7 shows that LR trained with TFIDF feature extraction technique is the best-performing model according to the balanced-accuracy score when evaluated on the OOT sample data. It achieved a lanced accuracy of 87.29%. The second-best performing model is LR when trained with the CV feature extraction technique and evaluated on the balanced-accuracy metric. It achieves an 86.67% balanced-accuracy score. The least-performing model is KNN according to the balanced-accuracy score. It achieved an 18.29% balanced accuracy score on the OOT sample data.

The main goal is to find the best-performing algorithm when trained with the right NLP feature extraction technique. The above results showed that LR is a very robust model:- it performs well on data outside its training period and performs better than all other models according to a balanced-accuracy score when evaluated on the OOT sample data. In the next section, an investigation on whether the introduction of additional features generated from complaint text may lead to the improved overall performance of the models is presented.

4.3 Model Evaluation with Engineered Features

In this section, a presentation and evaluation of model performance on two datasets namely validation and OOT data is done, when algorithms are trained with additional engineered features. Feature engineering was used only on the two NLP feature extraction techniques namely CV and TFIDF. Although feature engineering aims to improve model performances, it may result in additional features that are less important thus increasing the feature dimensions and thus, model complexity.

4.3.1 Performance on Validation dataset

Table 4.8 below shows the models' performance results when trained on a CV and additional engineered features on the validation dataset.

Table 4.8: Evaluation of Count Vectorizer with engineered features on validation data.

Model Name	Balanced Accuracy	Precision	Recall	F1-Score
LightGBM	0.4518	0.7213	0.7289	0.7220
LinearSVM	0.4260	0.6956	0.7052	0.6977
Logistic Regression	0.3600	0.6702	0.6203	0.6354
Decision Tree	0.2947	0.5468	0.5548	0.5504
KNN	0.0931	0.4161	0.3834	0.2701

Table 4.8 shows that LightGBM model performs better than all models across all the metrics when evaluated on a validation dataset. It achieved a balanced-accuracy score of 45.18% and a weighted F1-score of 72.2%. It is followed by LinearSVM with 42.60% balanced-accuracy and a weighted F1-score 69.77%. The least-performing model is KNN with 9.31% balanced-accuracy and a weighted F1-score of 27.01%.

Table 4.9 shows the models' performance results when trained with the TFIDF feature extraction technique and additional engineered features when evaluated on the validation dataset.

Table 4.9 shows that LightGBM is the best-performing model according to the balanced-accuracy metric. It achieved a balanced-accuracy of 44.55% and a weighted F1-score of 72.06%. The second best-performing model is LinearSVM with a balanced accuracy of 36.72% and a weighted F1-score of 70.76%. The least-performing

Table 4.9: Evaluation of TFIDF with engineered features on validation data across all metrics.

Model Name	Balanced Accuracy	Precision	Recall	F1-Score
LightGBM	0.4455	0.7193	0.7277	0.7203
LinearSVM	0.3672	0.7197	0.7357	0.7076
Logistic Regression	0.3602	0.7026	0.6471	0.6613
KNN	0.3197	0.6619	0.6523	0.6234
Decision Tree	0.2948	0.5602	0.5686	0.5627

model is the DT with a balanced-accuracy score of 29.48% and a weighted F1-score of 56.27%.

Figure 4.6: Models' balanced accuracy score distribution on NLP feature extraction techniques with additional engineered features when evaluated on the OOT data.

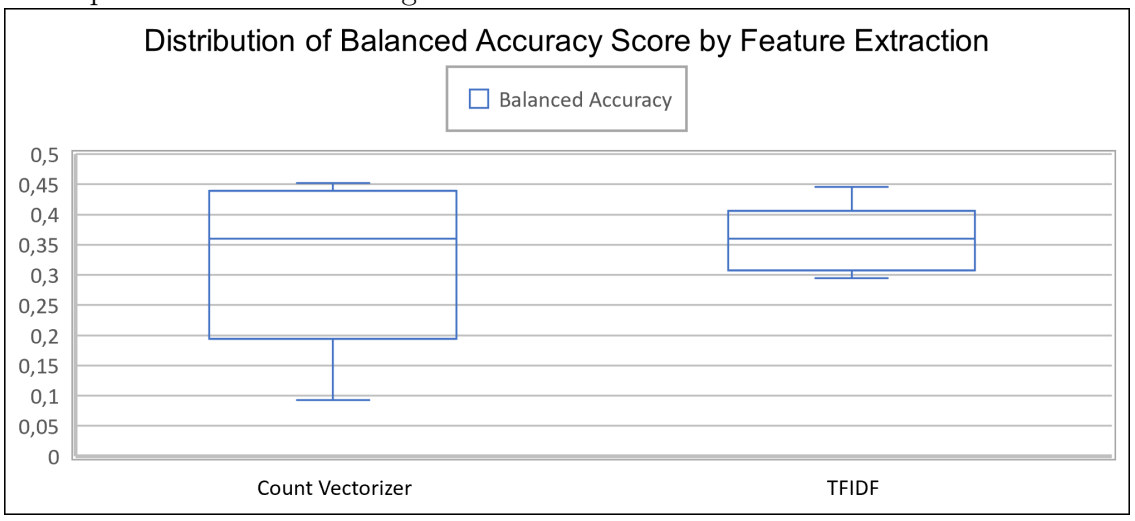
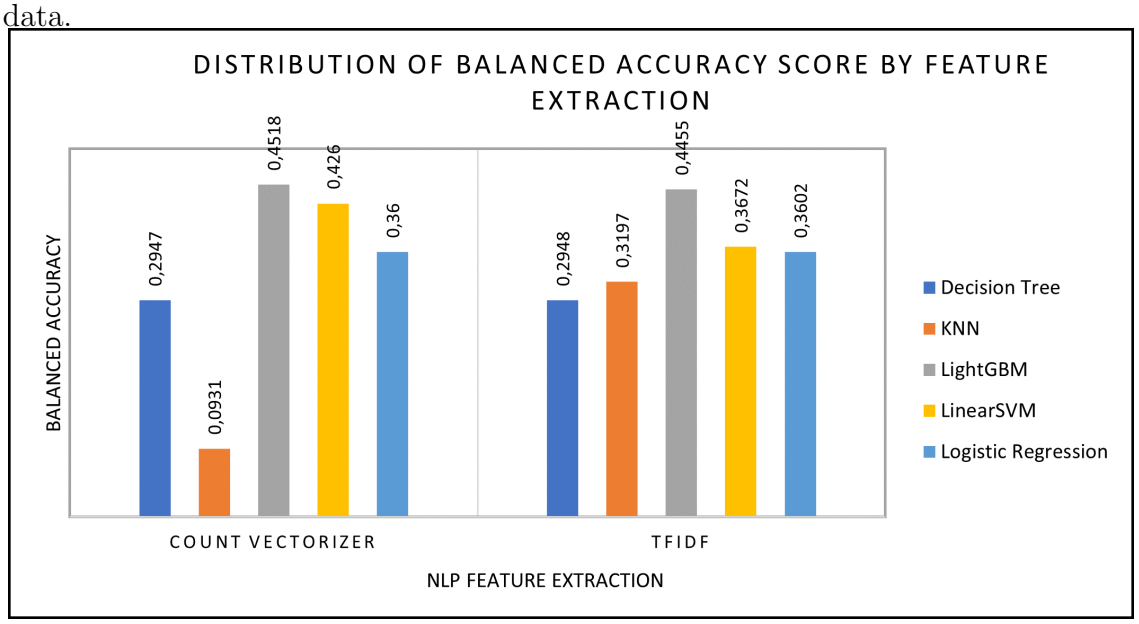


Figure 4.6 shows that models tend to perform on a higher balanced-accuracy score when trained with TFIDF than CV when additional features are all included. This performance is seen in the validation dataset. Figure 4.7 shows which model produces the highest balanced-accuracy score when evaluated with a balanced-accuracy score per feature extraction technique added with engineered features.

Figure 4.7 shows that LightGBM produced the highest balanced accuracy score with 45.18% when trained with CV. The second best-performing model is LightGBM when trained with TFIDF. The least performing model is KNN when trained with a CV. It achieved a balanced-accuracy score of 9.31%.

Figure 4.7: Models' performance comparison on the balanced-accuracy score per Feature Extraction Technique with engineered features when evaluated on validation data.



4.3.2 Model Evaluation on OOT Sample Data

Table 4.10 shows the models' performance results of CV feature extraction and additional engineered features when evaluated on the OOT sample dataset. The LinearSVM model produces the highest balanced accuracy score with 85.63% followed by LightGBM with 83.25%. The KNN comes at the least with 18.89%.

Table 4.10: Evaluation of Count Vectorizer with engineered features on OOT data.

Model Name	Balanced Accuracy	Precision	Recall	F1-Score
LinearSVM	0.8563	0.9137	0.9135	0.9132
LightGBM	0.8325	0.8853	0.8825	0.8827
Logistic Regression	0.8591	0.8947	0.8591	0.8718
Decision Tree	0.6669	0.7903	0.7896	0.7894
KNN	0.1849	0.6495	0.4790	0.3916

There are general improvements in the models' performance when evaluated on the balanced-accuracy in the OOT data across all feature extraction techniques. The LinearSVM model is best-performing with an 85.63% balanced-accuracy score and a weighted F1-score of 91.32%. This performance surpasses the LightGBM

model performance that was leading on the validation dataset. The second best-performing model is LightGBM with a balanced-accuracy of 83.25% and a weighted F1-score of 88.27%. The least-performing model is KNN with a balanced-accuracy of 18.49% and a weighted F1-score of 39.16%. Table 4.11 shows the models' metrics performance results when trained with the TFIDF feature extraction technique and additional engineered features when evaluated on the

Table 4.11: Evaluation of TFIDF with engineered features on OOT data across all metrics.

Model Name	Balanced Accuracy	Precision	Recall	F1-Score
Logistic Regression	0.8569	0.9224	0.9196	0.9198
LightGBM	0.8383	0.8889	0.8869	0.8868
LinearSVM	0.7578	0.9233	0.9219	0.9201
Decision Tree	0.6375	0.7621	0.7605	0.7609
KNN	0.5217	0.7991	0.7869	0.7748

Table 4.11 shows that the Logistic Regression is best-performing model better than all models across all metrics. It achieved 85.69% balanced-accuracy and a weighted F1-score of 91.98%. The second best-performing model is LightGBM with a balanced-accuracy of 83.83% and a weighted F1-score of 88.69%. The least-performing model is KNN with a balanced-accuracy of 52.17% and a weighted F1-score of 77.48%. The LR model improved from the 3rd position on the validation dataset to the first on the OOT sample. The improvement shows how the model can generalize on unseen data. Figure 4.8 demonstrates the distribution of models' balanced-accuracy scores per feature extraction technique when each technique is trained with additional engineered features and evaluated on the OOT dataset.

Figure 4.8 shows that the CV technique is not producing consistent balanced-accuracy scores across different models, producing high variation across all the trained models when evaluated on the OOT data. The TFIDF technique produces less variation across all the models. Figure 4.8 shows the overall performance of models across two feature extraction techniques using balanced-accuracy score metric on the OOT sampled data.

Figure 4.10 shows that the LR model produced the highest balanced-accuracy score when trained with the TFIDF feature extraction and engineered features. It produced a balanced-accuracy score of 85.69%. The least performing model is KNN in both feature extraction techniques.

Figure 4.8: Models balanced accuracy score distribution on NLP feature extraction techniques with additional engineered features when evaluated on the OOT data.

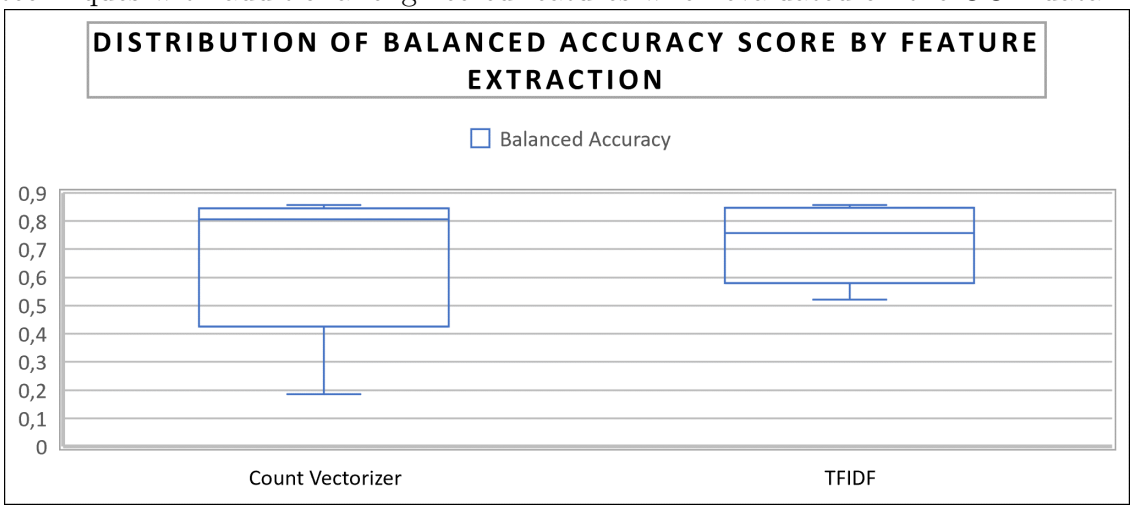
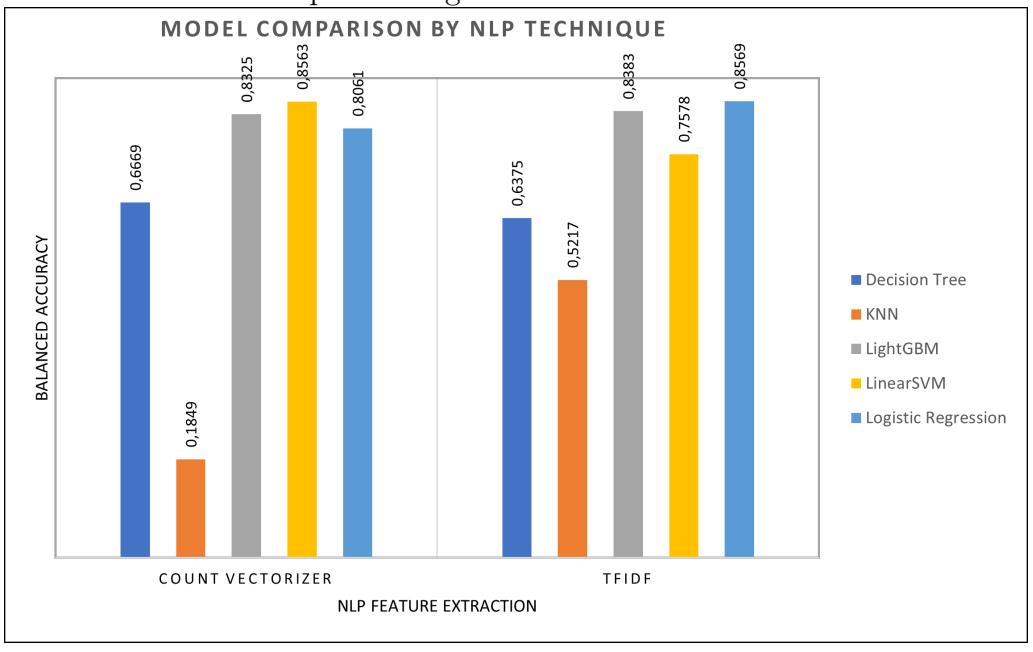


Figure 4.9: Models' performance comparison on the balanced-accuracy score per Feature Extraction Technique and engineered features when evaluated on OOT data.



4.4 Final Model Selection

The main objective of many ML solutions is to find the best-performing model on the OOT sample dataset. In this research, a combination of each of the three feature extraction techniques and each of the five ML algorithms were explored. Feature extraction techniques were used with and without the engineered features. The feature engineering techniques are helpful with BoW techniques as they aim to find additional important features to improve model learning. Therefore, for this reason, feature engineering was introduced to improve model learning and improve performance. Table 4.12 shows the overall models' performance summary results:

Table 4.12: Models' Summary results by Feature Extraction Technique on the OOT data using Balanced-accuracy and a weighted F1-score.

With Added Feature Engineered		Balanced-Accuracy		F1-Score	
Model Name	Feature Extraction	No	Yes	No	Yes
Logistic Regression	TFIDF	0.8729	0.8569	0.9375	0.9198
Logistic Regression	Count Vectorizer	0.8667	0.8061	0.9174	0.8718
LinearSVM	Count Vectorizer	0.8568	0.8563	0.9126	0.9132
LightGBM	TFIDF	0.8337	0.8383	0.8816	0.8868
LightGBM	Count Vectorizer	0.8314	0.8325	0.8724	0.8827
LinearSVM	TFIDF	0.7545	0.7578	0.9207	0.9201
Decision Tree	Count Vectorizer	0.7168	0.6669	0.8091	0.7894
Decision Tree	TFIDF	0.6916	0.6375	0.7870	0.7609
KNN	TFIDF	0.5001	0.5217	0.7525	0.7748
LightGBM	Word2Vec	0.4475		0.6842	
Logistic Regression	Word2Vec	0.4267		0.7076	
Decision Tree	Word2Vec	0.3485		0.5607	
KNN	Word2Vec	0.3248		0.5962	
LinearSVM	Word2Vec	0.2503		0.5783	
KNN	Count Vectorizer	0.1829	0.1849	0.3849	0.3916

Table 4.12 shows that LR trained with the TFIDF feature extraction technique produces the best results when evaluated on the OOT data using a balanced-accuracy metric. The model achieved 87.29% balanced accuracy score and a weighted F1-score of 91.98%. The second best-performing model is still Logistic regression when trained with CV. This model achieves 86.67% balanced accuracy on the OOT sample data and a weighted F1-score of 87.18%. The additional feature engineering technique did not help much. However, it seemed to have improved the model performance of the KNN model. Furthermore, the Word2Vec technique produced

relatively low scores across all evaluation metrics used when compared with other feature extraction techniques.

From the analysis in this section, it can be shown that additional engineered features did not improve any model when combined with CV. Additional features do improve model performances when combined with the TFIDF technique. The LightGBM, KNN, and LinearSVM models have shown a significant improvement when trained with additionally engineered features as compared to other models. However, the improvement was not enough to surpass the LR when tested on the OOT sample data. For this reason, LR with TFIDF feature extraction was selected as the final model.

Logistic Regression with TFIDF

Figure 4.10 shows the confusion matrix of a LR, the best model trained using the TFIDF feature extraction technique with no additional engineered features. The model's performance is evaluated on the OOT sample data. The confusion matrix compares the model's predicted complaint *Product* volumes against the actual complaint *Product* labels. Values along the diagonals represent cases where the model predicted a true complaint *Product*. Ideally, a model with 100% balanced accuracy would have all values along the diagonals. The model consistently achieves precision when predicting *money transfers and other financial services* but may not always recall all complaints related to these products. This phenomenon can be perplexing at times and is best understood by analyzing the precision and recall metrics per *Product*.

Figure 4.10: Confusion matrix of a Logistic Regression model evaluated on OOT complaints data.

	Predicted Product												
Actual Product	Bank account or service	checking or savings account	Consumer Loan	Credit card or prepaid card	Credit reporting, repair, or other	Debt collection	Money transfer, virtual currency, or money service	Money transfers	Mortgage	Other financial service	Payday loan, title loan, or personal loan	student loan	Vehicle loan or lease
Bank account or service	197	21	0	11	3	5	0	0	5	0	0	0	0
checking or savings account	13	276	0	9	6	2	0	0	4	0	1	0	0
Consumer Loan	0	0	127	2	3	9	0	0	7	0	2	2	2
Credit card or prepaid card	3	2	0	810	22	13	0	0	3	0	0	1	1
Credit reporting, repair, or other	0	0	1	11	2744	69	0	0	19	0	0	2	9
Debt collection	0	2	0	10	73	1655	0	0	14	0	1	7	1
Money transfer, virtual currency, or money service	0	4	0	6	4	2	113	0	1	0	0	0	0
Money transfers	0	0	0	1	1	1	2	19	0	0	0	0	0
Mortgage	0	1	0	1	10	8	0	0	980	0	0	2	0
Other financial service	0	0	0	0	0	1	0	0	0	4	0	0	0
Payday loan, title loan, or personal loan	1	1	1	4	2	12	0	0	9	0	94	8	2
student loan	0	0	0	1	12	6	0	0	5	0	0	386	1
Vehicle loan or lease	0	0	3	1	5	2	0	0	3	0	0	0	120

Table 4.13 assesses the model's performance using precision and recall metrics per *Product* complaint. To understand the overall performance of the model, weighted average, also known as balanced accuracy was utilized. Table 4.13 shows that the model achieves 100% precision when predicting both *Money transfers* and *Other financial services*.

Table 4.13: Logistic Regression Classification results for OOT complaints data.

Topic	Precision	Recall	F1-Score
Bank account or service	0.92	0.81	0.86
Checking or savings account	0.90	0.89	0.89
Consumer Loan	0.96	0.82	0.89
Credit card or prepaid card	0.93	0.95	0.94
Credit reporting, repair, or other	0.95	0.96	0.96
Debt collection	0.93	0.94	0.93
Money transfer, virtual currency, or money service	0.98	0.87	0.92
Money transfers	1.00	0.79	0.88
Mortgage	0.93	0.98	0.96
Other financial service	1.00	0.80	0.89
Payday loan, title loan, or personal loan	0.96	0.70	0.81
Student loan	0.95	0.94	0.94
Vehicle loan or lease	0.88	0.90	0.89
Weighted Average	**0.94**	**0.94**	**0.94**

It also does well when predicting *Money transfers, virtual currency, or money issues* and *Consumer loans* with 98%, and 96% precision, respectively. The product that achieves low precision is *Vehicle loan or lease* with an 88% precision score. Many products have a relatively low recall score compared to precision. This shows that the model confuses complaints, and this may be due to the existence of similarity of complaints per product does well when predicting *Money transfers, virtual currency, or money issues* and *Consumer Loan* with 98% and 96% precision. For this reason, it is important to look at the F1-score that balances precision and recall. This helps understand the overall model classification power at a level of a product. The LR model on the OOT has achieved a balanced-accuracy of 87.29% and a weighted F1-score of 94%.

Chapter 5

Conclusion

This research assessed the use of supervised ML algorithms with NLP feature extraction techniques as a solution to classify customer complaints data. Three different NLP feature extraction techniques and five ML classifiers were applied and evaluated on thirteen complaint topics. The results showed that the multinomial LR model with TFIDF feature extraction achieved the highest balanced-accuracy score of 87.29%. Furthermore, the classification results show that different feature extraction techniques can impact the model's performance as algorithms learn differently depending on the technique used. ML-supervised learning was proposed as a solution to handle customer complaints emails in the financial service industries. The best-performing model would be deployed in real-time to categorise complaints feedback.

This solution can improve complaints processes by saving customer agents time to manually classify and flag emails to the relevant teams, thus improving customer query resolution time and satisfaction. The solution's high accuracy score on the OOT sample indicates that the multinomial LR model is good enough to withstand different complaints over time.

5.1 Customer Email Resolution Time

The results discussed above showed that financial services companies or any other industry with customer complaints data can use ML text classification as a solution to classify complaints. This can improve customer agent reading and routing time. For example, if it takes on average 2.5 minutes for a customer agent to read and route

a complaint, it will require 2 hours to read and route 50 customer complaints. This can generally be reduced with more customer agents but at a cost to the company. However, when the volume of complaints increases it is likely to result in more cost to the company in both monetary value and customer experience. These costs can be reduced by using an ML solution that would accurately classify complaints and route them to the right back-end office with 87.29% balanced-accuracy.

In the cases when agents have 8,020 customer complaints. This will require 334 hours of work that may be split around customer agents. This solution will not only be important in reducing the time and work of customer agents but may help the business respond to customers promptly. Thus, improving the customer experience and ensuring the company is focusing on improving customer response time and customer experience. The fast response time means that the company will be decreasing customer query resolution time. However, in the few cases when the model's prediction is wrong, the back-end office team will correct the model's prediction by re-routing the complaints which in turn may lead to a good model monitoring framework. This will further test how robust is the model's predictions with time.

During the model validation, LightGBM and LinearSVM were performing better than most models across all feature extraction techniques when evaluated on balanced-accuracy. However, these models have dropped performance when evaluated on OOT. This phenomenon may be caused by overfitting and models' prediction power degrading with time.

5.2 Limitation of the study

The research has had its limitations. The data on which the models were developed is not purely email complaints data, as such, the data pre-processing required for email complaints data may not necessarily be as structured as the data used in this research. I suggest looking for customer complaint data and see if this solution will still stand. However, similar text processing may be applied and adapted to other domains. Despite many models showing good performance according to balanced-accuracy metric on the OOT sample, the training data was reduced to less than 1% of the total data due to a lack of computational resources. I suggest increasing the sample size and seeing if that will not improve the overall performance of this model or others.

Although the final model has shown good performance when evaluated on the OOT data (87.29% balanced-accuracy), it would be interesting to see how the model

would perform when trained on a larger sample. The training and evaluation data was created using Stratification random sampling, it would help assess how the model would perform under different sampling techniques. Techniques such as cross-validation could not be tested due to lack of computational power of the machine. Time was not measured in this study because the aim was to develop a solution. In the future, I'd track the training time of the model and how long it takes to score 8,000 records on the OOT.

5.3 Future Work and Recommendations

This research focused on and tested basic ML algorithms and feature extractions. Extending this problem to some of the state-of-the-art deep learning algorithms such as Large Language Models (LLMs) and feature extractions such as GloVe, FasText, and so on, may improve the accuracy of the existing model(s). The complaints data comes from CFPB which may differ from the actual complaint email text data structure of the organization in question. This may suggest different text pre-processing approaches depending on the data type in question. Finally, the complaints topic and model's prediction monitoring are required as new complaints arise over time. Some complaint topics are more prevalent than others and new emerging topics may occur post-model deployment. This suggests the need to monitor the topics while the model is in production. I suggest developing the models on a balanced dataset and assess the model improvement as unequal distribution tends to be biased toward the majority class.

Currently, new issues that are not known are assigned under *Other financial service*. However, state-of-the-art techniques like LLM can be used and help in detecting new issues as stand-alone topics. During the model training, the data has been reduced to a lower number of records to deal with issues arising with processing power and the curse of dimensionality. I would recommend training on a larger sample of data and performing hyper-parameter grid-search. The OOT data was sampled using stratification techniques. It would be interesting to see how the model would perform on the data sampled using simple random sampling.

Bibliography

[1] V. Labs, "Train test validation split: How to best practices," *V7 Labs Blog*, 2023. [Online]. Available: https://www.v7labs.com/blog/train-validation-test-set

[2] P. Huilgol. (2024) Quick introduction to bag-of-words (bow) and tf-idf for creating features from text. Accessed: 2024-04-22. [Online]. Available: https://www.analyticsvidhya.com/blog/2020/02/quick-introduction-bag-of-words-bow-tf-idf/

[3] user70394, "Question about continuous bag of words," Cross Validated Stack Exchange, 2015, accessed: 2024-04-22. [Online]. Available: https://stats.stackexchange.com/questions/140377/question-about-continuous-bag-of-words

[4] L. Burdick, "Word embeddings and how they vary," https://michiganaiblog.github.io/2018/07/23/Word-Embeddings-and-how-they-vary/, 2018, accessed: 2024-04-23.

[5] Javatpoint, "K-nearest neighbor (knn) algorithm for machine learning," *Javatpoint*, Year. [Online]. Available: https://www.javatpoint.com/k-nearest-neighbor-algorithm-for-machine-learning

[6] GeeksforGeeks, "Naive bayes vs. svm for text classification," *GeeksforGeeks*, 2024. [Online]. Available: https://www.geeksforgeeks.org/naive-bayes-vs-svm-for-text-classification/

[7] S. Awasthi, "Seven most popular svm kernels," *Dataaspirant*, 2018. [Online]. Available: https://dataaspirant.com/svm-kernels/

[8] L. S. Corporation, "Box plot with minitab," *Lean Sigma Corporation*, N/A. [Online]. Available: https://leansigmacorporation.com/box-plot-with-minitab/

[9] F. V. Ordenes, B. Theodoulidis, J. Burton, T. Gruber, and M. Zaki, "Analyzing customer experience feedback using text mining: A linguistics-based approach," *Journal of Service Research*, vol. 17, no. 3, pp. 278–295, 2014.

[10] M. Sokolova, N. Japkowicz, and S. Szpakowicz, "Beyond accuracy, f-score and roc: a family of discriminant measures for performance evaluation," in *Australasian joint conference on artificial intelligence*. Springer, 2006, pp. 1015–1021.

[11] S. S. Bengül and C. Yılmaz, "Effects of customer complaint management quality on business performance in service businesses: An application in turkish banking sector," *Boğaziçi Journal*, vol. 32, no. 2, pp. 77–100, 2018.

[12] J. Strauss and D. J. Hill, "Consumer complaints by e-mail: an exploratory investigation of corporate responses and customer reactions," *Journal of Interactive Marketing*, vol. 15, no. 1, pp. 63–73, 2001.

[13] J. Brownlee. (2019) A gentle introduction to the bag-of-words model. [Online]. Available: https://machinelearningmastery.com/gentle-introduction-bag-words-model

[14] J. Smith, *Book Title*. City, Country: Publisher Name, 2023.

[15] A. Borg, M. Boldt, O. Rosander, and J. Ahlstrand, "E-mail classification with machine learning and word embeddings for improved customer support," *Neural Computing and Applications*, vol. 33, no. 6, pp. 1881–1902, 2021.

[16] Pipefy, "Effective complaint management," https://www.pipefy.com/blog/effective-complaint-management/, 2024, accessed: 2024-04-26.

[17] M. D. N. Arusada, N. A. S. Putri, and A. Alamsyah, "Training data optimization strategy for multiclass text classification," in *2017 5th International conference on information and communication technology (ICoIC7)*. IEEE, 2017, pp. 1–5.

[18] J. Filgueiras, L. Barbosa, G. Rocha, H. L. Cardoso, L. P. Reis, J. P. Machado, and A. M. Oliveira, "Complaint analysis and classification for economic and food safety," in *Proceedings of the Second Workshop on Economics and Natural Language Processing*, 2019, pp. 51–60.

[19] S. S. Tax, S. W. Brown, and M. Chandrashekaran, "Customer evaluations of service complaint experiences: implications for relationship marketing," *Journal of marketing*, vol. 62, no. 2, pp. 60–76, 1998.

[20] S. Lovetta, "Customer satisfaction and customer complaint: Case ayaba hotel bamenda," 2021.

[21] S. Macdonald. (2023) 7 ways to reduce customer service response times. [Online]. Available: https://www.superoffice.com/

[22] S. Behzadifard, R. Farzadnia *et al.*, "Business process discovery from emails: Text classification and process mining-a case study of procurement process," 2022.

[23] V. Hayuningrum, "Customer complaints auto-categorization: Performance comparison of recurrent and convolutional neural networks," Ph.D. dissertation, Tilburg University, 2021.

[24] A. Romanov, K. Lomotin, and E. Kozlova, "Application of natural language processing algorithms to the task of automatic classification of russian scientific texts," *Data Science Journal*, vol. 18, pp. 37–37, 2019.

[25] P. Hake, J.-R. Rehse, and P. Fettke, "Supporting complaint management in the medical technology industry by means of deep learning," in *Business Process Management Workshops: BPM 2019 International Workshops, Vienna, Austria, September 1–6, 2019, Revised Selected Papers 17*. Springer, 2019, pp. 56–67.

[26] H. Liu, P. Burnap, W. Alorainy, and M. L. Williams, "A fuzzy approach to text classification with two-stage training for ambiguous instances," *IEEE Transactions on Computational Social Systems*, vol. 6, no. 2, pp. 227–240, 2019.

[27] M. Putong and S. Suharjito, "Classification model of contact center customers emails using machine learning," *Adv. Sci. Technol. Eng. Syst. J*, vol. 5, pp. 174–182, 2020.

[28] T. T. Dien, B. H. Loc, and N. Thai-Nghe, "Article classification using natural language processing and machine learning," in *2019 International Conference on Advanced Computing and Applications (ACOMP)*. IEEE, 2019, pp. 78–84.

[29] S. Khedkar and S. Shinde, "Deep learning and ensemble approach for praise or complaint classification," *Procedia Computer Science*, vol. 167, pp. 449–458, 2020.

[30] H. N. K. AL-Behadili, "Decision tree for multiclass classification of firewall access," *International Journal of Intelligent Engineering and Systems*, vol. 14, no. 3, pp. 294–302, 2021.

[31] C. Padurariu and M. E. Breaban, "Dealing with data imbalance in text classification," *Procedia Computer Science*, vol. 159, pp. 736–745, 2019.

[32] G. Varoquaux and O. Colliot, "Evaluating machine learning models and their diagnostic value," *Machine Learning for Brain Disorders*, pp. 601–630, 2023.

[33] B. A. Talpur and D. O'Sullivan, "Multi-class imbalance in text classification: A feature engineering approach to detect cyberbullying in twitter," in *Informatics*, vol. 7, no. 4. MDPI, 2020, p. 52.

[34] C. F. C. Bureau. (2023) Consumer complaint database. [Online]. Available: https://www.consumerfinance.gov/data-research/consumer-complaints/#download-the-data

[35] T. Wongvorachan, S. He, and O. Bulut, "A comparison of undersampling, oversampling, and smote methods for dealing with imbalanced classification in educational data mining," *Information*, vol. 14, no. 1, p. 54, 2023.

[36] J. Tanha, Y. Abdi, N. Samadi, N. Razzaghi, and M. Asadpour, "Boosting methods for multi-class imbalanced data classification: an experimental review," *Journal of Big Data*, vol. 7, pp. 1–47, 2020.

[37] J. Sadaiyandi, P. Arumugam, A. K. Sangaiah, and C. Zhang, "Stratified sampling-based deep learning approach to increase prediction accuracy of unbalanced dataset," *Electronics*, vol. 12, no. 21, p. 4423, 2023.

[38] M. Merrillees and L. Du, "Stratified sampling for extreme multi-label data," *CoRR*, vol. abs/2103.03494, 2021. [Online]. Available: https://arxiv.org/abs/2103.03494

[39] F. Zhang, M. Petersen, L. Johnson, J. Hall, and S. E. O'Bryant, "Hyperparameter tuning with high performance computing machine learning for imbalanced alzheimer's disease data," *Applied Sciences*, vol. 12, no. 13, p. 6670, 2022.

[40] "Chapter 4: Forest management," https://cals.arizona.edu/classes/rnr321/Ch4.pdf, accessed on March 28, 2024.

[41] T. D. Nguyen, M.-H. Shih, D. Srivastava, S. Tirthapura, and B. Xu, "Stratified random sampling from streaming and stored data," *Distributed and Parallel Databases*, vol. 39, pp. 665–710, 2021.

[42] Y. Fan, C. Arora, and C. Treude, "Stop words for processing software engineering documents: Do they matter?" in *2023 IEEE/ACM 2nd International Workshop on Natural Language-Based Software Engineering (NLBSE)*. IEEE, 2023, pp. 40–47.

[43] Y. HaCohen-Kerner, D. Miller, and Y. Yigal, "The influence of preprocessing on text classification using a bag-of-words representation," *PloS one*, vol. 15, no. 5, p. e0232525, 2020.

[44] D. Khyani, B. Siddhartha, N. Niveditha, and B. Divya, "An interpretation of lemmatization and stemming in natural language processing," *Journal of University of Shanghai for Science and Technology*, vol. 22, no. 10, pp. 350–357, 2021.

[45] J. Singh and V. Gupta, "A systematic review of text stemming techniques," *Artificial Intelligence Review*, vol. 48, pp. 157–217, 2017.

[46] GeeksforGeeks, "Introduction to stemming," *GeeksforGeeks*, N/A. [Online]. Available: https://www.geeksforgeeks.org/introduction-to-stemming/

[47] P. Jacob Murel and E. Kavlakoglu, "What are stemming and lemmatization?" *IBM*, N/A. [Online]. Available: https://www.ibm.com/topics/stemming-lemmatization

[48] J. Brownlee, "A gentle introduction to the bag-of-words model," *Machine Learning Mastery*, 2019. [Online]. Available: https://machinelearningmastery.com/gentle-introduction-bag-words-model/

[49] F. Karabiber, "Tf-idf — term frequency-inverse document frequency," *LearnDataSci*, N/A. [Online]. Available: https://www.learndatasci.com/glossary/tf-idf-term-frequency-inverse-document-frequency/

[50] ——, "Tf-idf — term frequency-inverse document frequency," *LearnDataSci*, 2018. [Online]. Available: https://www.learndatasci.com/glossary/tf-idf-term-frequency-inverse-document-frequency/

[51] A. Vaishnav, "Understanding nlp: Word embeddings & text vectorization," *Towards Data Science*, 2018. [Online]. Available: https://towardsdatascience.com/understanding-nlp-word-embeddings-text-vectorization-1a23744f7223

[52] A. Arseniev-Koehler, "Theoretical foundations and limits of word embeddings: what types of meaning can they capture?" *Sociological Methods & Research*, p. 00491241221140142, 2022.

[53] T. W. Edgar and D. O. Manz, *Research methods for cyber security*. Syngress, 2017.

[54] D. W. Hosmer, S. Lemeshow, and R. X. Sturdivant, *Applied Logistic Regression*. John Wiley & Sons, 2013.

[55] S. W. Grant, G. L. Hickey, and S. J. Head, "Aortic valve replacement in the elderly: a systematic review and meta-analysis of outcomes," *European Journal of Cardio-Thoracic Surgery*, vol. 55, no. 2, pp. 179–186, 2019.

[56] C. M. Bishop, *Pattern Recognition and Machine Learning*. Springer, 2006.

[57] GeeksforGeeks, "Advantages and disadvantages of logistic regression," 2023, last Updated: 10 Jan, 2023. [Online]. Available: https://www.geeksforgeeks.org/advantages-and-disadvantages-of-logistic-regression/

[58] J. Sreemathy and P. Balamurugan, "An efficient text classification using knn and naive bayesian," *International Journal on Computer Science and Engineering*, vol. 4, no. 3, p. 392, 2012.

[59] S. U. Hassan, J. Ahamed, and K. Ahmad, "Analytics of machine learning-based algorithms for text classification," *Sustainable operations and computers*, vol. 3, pp. 238–248, 2022.

[60] AspiringYouths, "Advantages and disadvantages of algorithms," 2023, last Accessed: 25 Apr, 2024. [Online]. Available: https://aspiringyouths.com/advantages-disadvantages/algorithms/

[61] Editorial. (2022) Pros and cons of the k-nearest neighbors (knn) algorithm. Accessed: 25 Apr, 2024. [Online]. Available: https://roboticsbiz.com/pros-and-cons-of-the-k-nearest-neighbors-knn-algorithm/

[62] V. K. Chauhan, K. Dahiya, and A. Sharma, "Problem formulations and solvers in linear svm: a review," *Artificial Intelligence Review*, vol. 52, no. 2, pp. 803–855, 2019.

[63] S. Kumar, "Svm: Difference between linear and non-linear models," *AITUDE*, 2020. [Online]. Available: https://www.aitude.com/svm-difference-between-linear-and-non-linear-models/

[64] "Decision tree algorithms," https://www.geeksforgeeks.org/decision-tree-algorithms/, accessed: 2024-04-22.

[65] IBM. (2023) Decision trees. [Online]. Available: https://www.ibm.com/topics/decision-trees

[66] TowardsMachineLearning, "What are boosting algorithms and how they work," *TowardsMachineLearning*, 2024. [Online]. Available: https://towardsmachinelearning.org/boosting-algorithms/

[67] G. Ke, Q. Meng, T. Finley, T. Wang, W. Chen, W. Ma, Q. Ye, and T.-Y. Liu, "Lightgbm: A highly efficient gradient boosting decision tree," *Advances in neural information processing systems*, vol. 30, 2017.

[68] P. Kumar, R. Bhatnagar, K. Gaur, and A. Bhatnagar, "Classification of imbalanced data: review of methods and applications," in *IOP conference series: materials science and engineering*, vol. 1099, no. 1. IOP Publishing, 2021, p. 012077.

[69] A. Kumar, "Machine learning – sensitivity vs specificity differences, examples," *Vitalflux*, November 2023. [Online]. Available: https://vitalflux.com/ml-metrics-sensitivity-vs-specificity-difference/

[70] B. Bischl, M. Binder, M. Lang, T. Pielok, J. Richter, S. Coors, J. Thomas, T. Ullmann, M. Becker, A.-L. Boulesteix *et al.*, "Hyperparameter optimization: Foundations, algorithms, best practices, and open challenges," *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, vol. 13, no. 2, p. e1484, 2023.

[71] P. Liashchynskyi and P. Liashchynskyi, "Grid search, random search, genetic algorithm: a big comparison for nas," *arXiv preprint arXiv:1912.06059*, 2019.