# Text Content Classification on News Articles

Sean Soutar STRSEA001[a], Fabio Fehr FHRFAB001[b]

[a] *UCT Statistical Science Honours, Cape Town, South Africa*

[b] *UCT Statistical Science Honours, Cape Town, South Africa*

## Abstract

The increased volume of electronic journalism and other text media has driven the processing of the natural language to gather meaning from large bodies of text. Text classification is the process of classifying a body of text into a predefined set of classes. This process can be computationally expensive. This research provides a case study of classifying online News24 news articles into a variety of single label classes. Machine learning techniques are used to classify the news articles with the objective to balance classification accuracy and computational efficiency. The text vectorization techniques Term Frequency Inverse Document Frequency and Naive Bayes are contrasted while the classifier models: Naive Bayes classifier,k-Nearest Neighbours, Support Vector Machines and decision tree forests are compared by computation time and accuracy. It was found that the Naive Bayes vectorisation method produced models that were the most accurate and computationally efficient. The k-NN, SVM and decision tree forest models trained on Naive Bayes vectorised data were deemed to be good choices for text classification.

*Keywords:* text classification, machine learning, natural language processing

*JEL classification*

*Email addresses:* `sean.soutar@gmail.com` (Sean Soutar STRSEA001), `FHRFAB001@myuct.ac.za` (Fabio Fehr FHRFAB001)

## 1. Acknowledgements

We would like to thank Stefan Britz & Nicolaas Johannes Odendaal for the supervision and guidance. We also would like to thank the analytics team at Media24 and Christine de Kock for the support as well as providing the data.

| ***Glossary*** | |
|---|---|
| ANN | Artificial Neural Networks |
| Bagging | Bootstrap Aggregation |
| BoW | Bag of words |
| k-NN | K-Nearest Neighbours |
| LDA | Latent Dirichilet Allocation |
| LSI | Latent Semantic Indexing |
| NB | Naive Bayes |
| NLP | Natural Language Processing |
| PCA | Principal Component Analysis |
| RCNN | Recurrent Convolutional Neural Network |
| SVM | Support Vector Machine |
| TF-IDF | Term Frequency Inverse Document Frequency |

# 2. Introduction

Text classification falls under the umbrella of natural language processing (NLP) which encompasses all techniques used in the synthesis of language and speech. NLP is present in various disciplines and has been used in spam filtering of emails, language identification, genre classification of literary works, category-based sentiment analysis of social media posts and classifying research articles into sub-disciplines (Jain and Mandowara 2016 & Fragos, Belsis, and Skourlas (2014)). Some other fields under NLP include speech processing and sentiment analysis. Manning and Schutze (1999) mentions that the purpose of NLP is to answer the questions: "What kinds of things do people say" and "What do these things request about the world". Text classification helps us attain meaning from what people are saying.

Text classification is distinct from topic modelling in the following way: text classification is supervised learning on labelled data. Text classification seeks to place documents in a corpus (large and structured set of texts) into a predefined set of classes (Basu, Walters, and Shepherd 2003) whereas topic modelling is unsupervised learning which seeks to derive cluster based classes from a corpus (Purver et al. 2006). For this study, the focus will be on text classification on Media24's news articles. Correctly classified news articles will allow for better content recommendation to Media24's users which result in greater user satisfaction.

The research objective is to maximise classification accuracy and computational efficiency for multiclass, single label text documents using machine learning techniques. The multiclass, single label aspect refers to the assumption that there are numerous possible ($K > 2$) predefined classes, but the documents are classified by only one class e.g. "crime", "cricket" or "economy". Text classification accuracy is defined to be how well a given model classifies documents into their appropriate classes. Computational efficiency will be evaluated by the length of time it takes to vectorise the data, train the model and generate predictions. This study seeks to balance accuracy and computational efficiency by carefully selecting, tuning and ensembling various machine learning methods to classify news articles.

This research begins with Chapter 3 by exploring the relevant literature around text classification. In Chapter 4 the data is analysed briefly which leads into Chapter 5 which describes the methodology for the study. This includes: data cleaning, transformation, vectorisation, feature selection, model training and evaluation. This chapter also touches on the statistical models used. In Chapter 6 the results are discussed. Finally, in Chapter 7 we present our conclusions and directions for future work.

## 3. Literature Review

In this chapter a selection of popular techniques within text classification literature will be analysed. These include text vectorisation techniques and the statistical models used to classify the vectorised text data.

Fragos, Belsis, and Skourlas (2014) defines two broad categories of text classification models namely: probabilistic and discrimination. Some examples of probabilistic models include Latent Dirichlet Allocation (LDA) developed by Blei, Ng, and Jordan (2003), Maximum Entropy and Naive Bayes (NB) classification. Fragos, Maistros, and Skourlas (2005) explain that these models seek to estimate parameters of a joint probability distribution of a single document $D$ and a set of defined categories $K$. Discrimination models include algorithms such as artificial neural networks (ANN), k-Nearest Neighbours (k-NN), decision trees, multinomial logistic regression and support vector machines (SVM). These models seek to learn general patterns from the text within documents in order to classify a given document to a category (Fragos, Belsis, and Skourlas 2014). All of these statistical models require the data to be in a numerical format. A common problem is how to convert text to numbers such that it preserves information intrinsic to the text.

A popular approach to viewing language is through the “Bag-of-Words” (BoW) principle. BoW states that the ordering of words in a document has no material effect on its meaning. This is an assumption of exchangeability of words within a document (Aldous 1985). This assumption states that it is only the composition of words in a document that is important and that word order can be neglected. The BoW approach has been criticised for ignoring fundamental characteristics of natural language such as associations between words (Zelikman 2018). Using the BoW principle, documents are vectorised into one vector where each entry is a single word (unigram). Other representations exist where pairs (bigrams) or triplets (trigrams) of words form each entry in the document vector. Wang and Manning (2012) performed sentiment classification using unigram and bigrams. It was concluded from their research that bigrams tended to improve the accuracy of sentiment classification but topic classification accuracy did not improve when using bigrams over unigrams. A possible reason for this is that when topic classification is the objective, certain keywords tend to be indicative of the topic alone (Wang and Manning 2012). Various methods have been developed to attribute numerical values to each entry in the document vector. The Term Frequency - Inverse Document Frequency (TF-IDF) method was one of them.

Salton and Buckley (1988) developed a term weighted approach to text vectorisation in 1988. Their approach up-weights words likely to be useful and down-weights words likely to be irrelevant. Terms that are frequently mentioned in individual documents are hypothesized to be useful in classifying that document. However, it was found that if these high frequency words were also frequent in other documents, the performance of the TF representation decreased. A modification to the TF representation was proposed. The IDF modifies the TF to take into account the prevalence of a given word in other documents in a corpus (Salton and Buckley 1988). Zelikman (2018) argues that a very large corpus is needed for TF-IDF to be effective. TF-IDF is still a very popular representation of text based on BoW. An estimated 70% of all text recommendation systems use TF-IDF due to its easy implementation (Beel et al. 2016). Alternative measures include using Mahalanobis Distances as well as Smooth Inverse Frequencies which is a dimension reduced weighted average of term frequencies (Zelikman 2018). While the TF-IDF vectorisation is popular it presents the issue of high dimensionality and sparsity (Blei, Ng, and Jordan 2003). Jurafsky and Martin (2009) states that this high dimensionality can be reduced through the processes of stop word removal, stemming, pruning and latent semantic indexing (LSI).

Removing stopwords, stemming and pruning are methods of reducing dimensionality and improving classification accuracy by reducing noise of the data (Sedding and Kazakov 2004). Stopwords are words considered not to convey any meaningful information. Stopword lists differ per context and the researcher's personal choice, but typical include "the" , "and", "while" etc. Miao et al. (2009) states it is common practice to remove these words. Stemming is the process of reducing words to their root-form usually through the removal of suffixes. This allows for words meaning the same thing to be reduced to the same linguistic form (Sedding and Kazakov 2004). For example, "operational", "operate" and "operating" would be reduced to "operat". Pruning is the removal of words which occur frequently or infrequently. Sedding and Kazakov (2004) argues that these words are either too frequent such that they do not add any value or too infrequent such that these words are unlikely to occur in other documents. Furthermore, Blei, Ng, and Jordan (2003) suggest using LSI to further reduce overall dimensionality. LSI uses singular value decomposition to reduce feature variables down to a linear subspace of TF-IDF feature components. These components capture the majority of the variation. A common way of doing this is through Principal Component Analysis (PCA). Blei, Ng, and Jordan (2003) mentions that the proponents of LSI state that this method helps to capture synonymy or polysemy of natural language. Blei, Ng, and Jordan (2003) remain unconvinced that this is the case. In fact, Sedding and Kazakov (2004) attempted to enrich BoW representations by tagging each word with it's appropriate parts of speech. Their results showed that this did not improve the accuracy of the models. Isa et al. (2008) & Fragos, Belsis, and Skourlas (2014) suggest an alternative vectorisation to TF-IDF. This technique uses a NB classifier for the vectorisation process and acts as a more supervised dimension reduction technique compared to PCA.

The NB classifier is a probabilistic model. It attempts to estimate the conditional distribution of each document belonging to a particular class. NB can act as both vectoriser and a classifier. This model assumes that any feature in a class is independent of any other feature in that class. The "naive" element is that it presumes each word occurs independently of any other word. One might expect that the words "money" and "equities" might occur more frequently together. The NB classifier does not factor this into account. McCallum and Nigam (1998) supposes that even though this assumption may be incorrect, this model can achieve impressive results. Isa et al. (2008) experimented with both TF-IDF and NB vectorised data with the SVM, decision tree forest, ANN and k-NN statistical models. A typical decision rule would be to classify the document to the category with the highest predicted posterior probability. Isa et al. (2008) found that training other statistical models on top of the NB vectorised data proved to be more accurate than using NB alone. They also found that the NB vectorised models proved to be more accurate than the TF-IDF models. The NB-SVM combination yielded the most accurate results. Two methods of vectorising text data, TF-IDF and NB, have been surveyed. LDA is another popular probabilistic classifier in NLP research.

Blei, Ng, and Jordan (2003) developed the notable LDA as a generative probabilistic model of a corpus. This model is an unsupervised method which seeks to derive its classes from the data. The major advantage of this model over other supervised models is that it allows a single document to be a mixture of more than one class (Blei, Ng, and Jordan 2003). While this was a big contribution to the field, it is not applicable in this context as all of the data are strictly single labelled. Thus the advantage of specifying documents as mixtures of topics is moot. Discrimination based models will now be examined. In particular, the SVM is a popular choice for text classification and has been shown by the researchers Isa et al. (2008) and Joulin et al. (2016) to be effective in classification.

SVM's are an early and widely used discrimination model for text classification. Cortes and Vapnik (1995) developed the algorithm SVM in 1995 and with the use of kernel functions it is able to find non-linear classification boundaries (Hastie and Tibshirani 2013). In later years, Basu, Walters, and Shepherd (2003) performed a text classification comparison between SVM, NB and decision tree algorithms. The results indicated that SVM is preferable for classification of shorter text documents while decision trees and NB where competitive for larger documents. This is of particular interest as the news article data used in this study is typically quite small around 300-500 words (see Chapter 4). Basu, Walters, and Shepherd (2003) also compared the use of SVM against neural networks. It was found that the SVM achieved more accurate results in comparison to the neural networks when controlled for the same vocabulary size, but for large vocabulary sizes neural networks were competitive in accuracy. However, more recent research surrounding recurrent convolutional neural networks (RCNN) by Lai et al. (2015) suggests that more modern implementations outperform BoW based models.

Various forms of RCNN were compared to BoW implementations of SVM and logistic regression models in the work of Lai et al. (2015). One of the datasets included a 4 class English news articles dataset. The results showed that the neural networks all outperformed the BoW based models. However, the computational aspects of the models were not compared and the Media24 news data to be used for this study has many more classes at a total of 26. X. Zhang, Zhao, and LeCun (2015) also applied RCNN to English news data. The same conclusion was reached that these neural networks outperformed the BoW based models. In competition with neural network implementations, Joulin et al. (2016) developed the "fastText" algorithm in 2016. This "fastText" model achieved comparable classification accuracy to the neural network model of X. Zhang, Zhao, and LeCun (2015) however, the fastText algorithm was far quicker to train. This is an important consideration as the study seeks to balance the accuracy of the model with computational aspects. The “fastText” model is a proprietary ensemble of k-NN and SVM models trained on BoW vectorised data. This motivates further investigation into more “traditional” classification models. Hastie and Tibshirani (2013) shows that simple classifiers such as k-NN can achieve impressive classification results in minimal training time.

Cover and Hart (2006) describe the nearest neighbour model as an algorithm which assigns an unclassified sample point to the class most heavily represented by the closest $k$ data points. Miao et al. (2009) used k-NN for document classification. They found that when the training set is too large the k-NN becomes computationally infeasible, but the classifier is simple and can yield accurate predictions. The last discrimination model which will be considered is the decision tree forest. It is an attractive model to use due to its prominent classification performance on high dimensional data and its algorithmic simplicity (B. Xu et al. 2012).

The decision tree classification model originally suffered from high prediction variance with changes in training data. Extensions to this model such as bootstrapped aggregation (bagging) and the Random Forest by Breiman (2001) resulted in greater prediction accuracy than just one decision tree (Hastie and Tibshirani 2013). B. Xu et al. (2012) argues that Random Forest’s can miss important information within the features if the data has high dimensionality. They developed a modified version of the Random Forest which predicted more accurately than the original Random Forest, SVM, NB and k-NN. This motivates for the use of decision tree based methods for text classification. Various probabilistic and discrimination models have been explored. An avenue of research exists where these models are combined or ensembled in order to achieve better prediction accuracy than using a single model.

Miao et al. (2009) states that hybrid classifiers try to negate the weaknesses and combine the strengths of the individual classifiers for enhanced results. A specific example from Miao et al. (2009) uses a k-NN classifier to classify new documents which lie on the the boundary region where the other classifiers predict documents incorrectly. Isa et al. (2008) found that using NB to vectorise the data as opposed to TF-IDF improved classification accuracy when using SVM. The NB-SVM ensemble outperformed the NB classifier and proved to be a robust performer in many contexts. Bennett, Dumais, and Horvitz (2002) also combines classifiers, but includes ‘Reliability Indicators’ which are ‘context sensitive’ meta variables to help classify documents. These could improve classification accuracy through added features such as document length, word variety and average sentence lengths.

The objective for this study is to maximise classification accuracy and computational efficiency for text documents. The representation of text in literature is dominated by the BoW framework due to it’s simple and effective implementation. Unigrams will be used as there is no clear advantage in using bigrams or trigrams for content classification. TF-IDF text vectorisation is commonly used, but suffers from high dimensionality. Fragos, Belsis, and Skourlas (2014) & Isa et al. (2008) mention that using a NB classifier to vectorise data could be more effective in dimensionality reduction and improve classification accuracy as opposed to TF-IDF. The literature has shown that both discrimination and probabilistic methods should be considered. This is because they are not often accessed for both model accuracy and computational efficiency concurrently. Furthermore, prediction accuracy appears to be very much related to the type of text data and vectorisation method used.

This study seeks to add value to the NLP field by conducting a thorough analysis on the computational and prediction accuracies of both the text vectorisation methods and a breadth of statistical models. The models selected for this study include SVM, k-NN, decision tree forests and NB. These algorithms were selected as the literature suggests they should achieve good prediction accuracy for text classification. These algorithms are simple as they contain few hyperparameters to tune, unlike neural networks. Efficient software implementations exist in `R` for these models. The news articles data for this study will now be explored.

## 4. Data

The data consists of news articles that have been hosted on the News24.com website. Important fields of interest include: article ID, title of article, article body, keywords and publication date. The keywords field refers to various tags that are associated to each article assigned by the authors. An article can have one or many tags associated with it which could include places, people or activities. An example of these tags could be “Cape Town”, “Donald Trump” or “Soccer”. Since the research objective of this study deals with document class classification, tags that relate to the article topic are of special interest.

The original data set includes 163305 articles. Not every article had a topic tag associated with it and others had more than one topic tag associated to them. For this study we consider single labels for classification therefore articles with more than one topic tag or with no topic tag are excluded. These following pairs appeared almost always with one another such as equities - markets, celebrities - local celebrities and motorsport - F1.

These pairs were collapsed such that the smaller categories were merged with the larger categories in the pairs. Merging these categories allowed for more data to be used. The distributions of all single labelled articles will now be examined.

The single labelled data set consists of 47170 articles. The distribution of these articles over time is important. This is because a particularly small publishing Year-Month may result in many low prevalence classes not featuring in the validation set. Figure 4.1 shows that the majority of articles published were in 2017 and there are very few articles for April 2017.

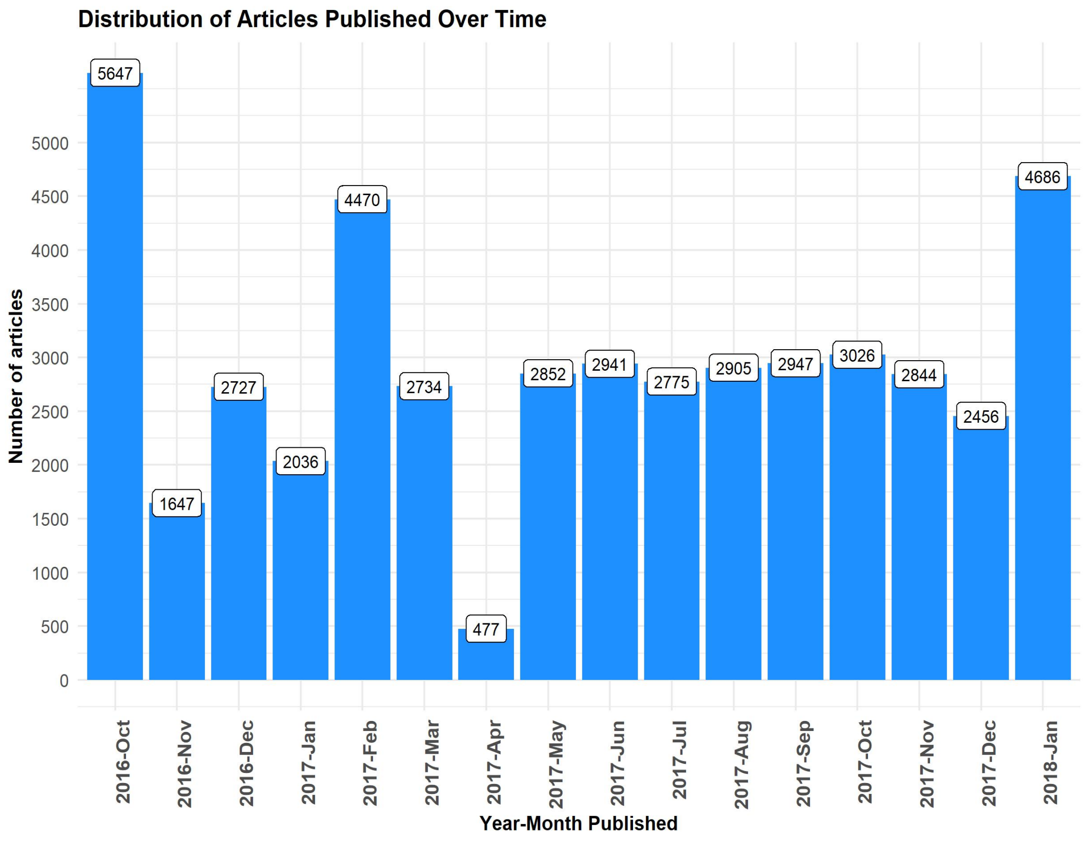


Figure 4.1: Distribution of articles published per year

In order to confirm the appropriateness of training-validating models on monthly data, the class proportions per month from October 2016 to January 2018 are evaluated. This is to determine for example if ‘cycling’ articles are only published earlier on in the year and not during any other time. This is shown in a Year-Month terms in Figure 4.2.

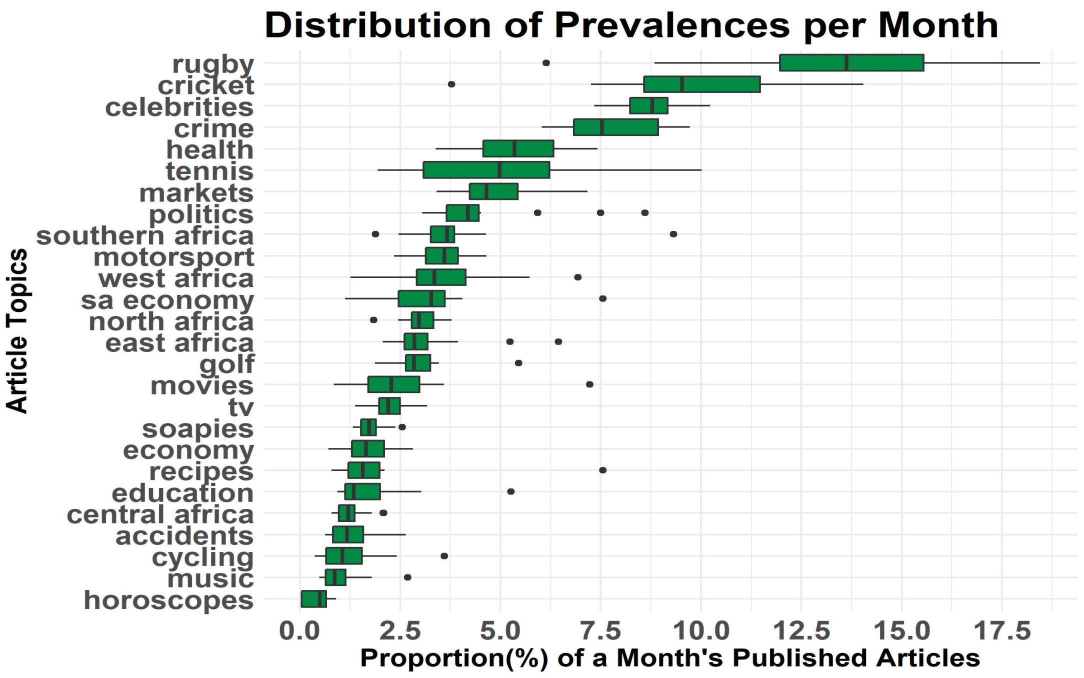


Figure 4.2: Distribution of Topic Prevalences over Time

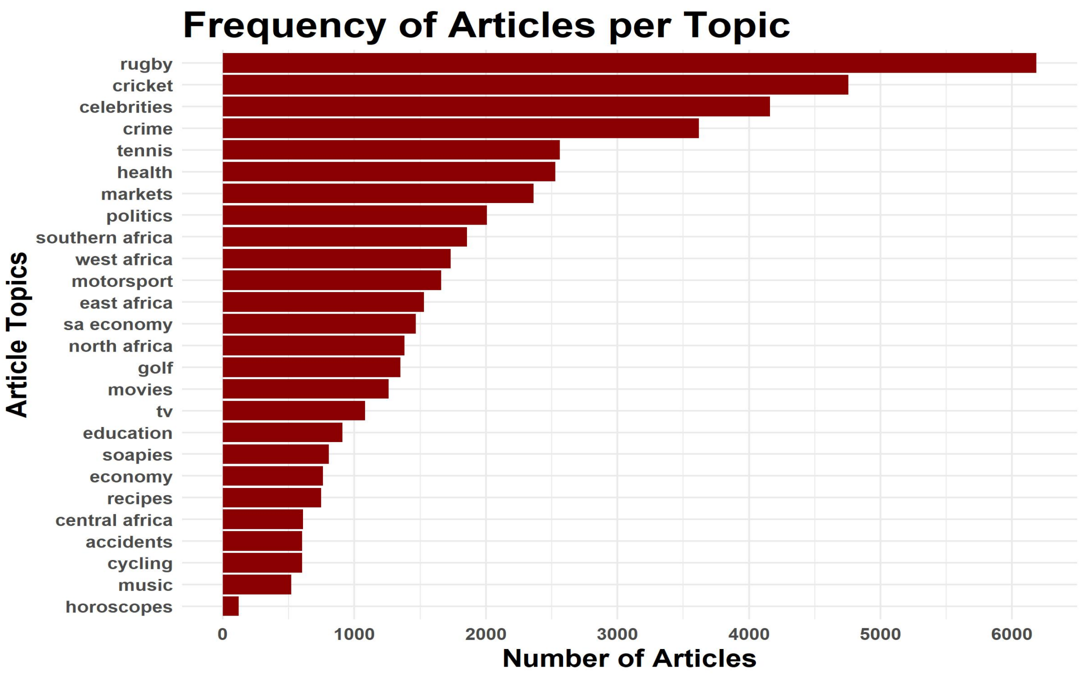


Figure 4.3: Topic Frequencies for Single Labelled Topics

It is clear that the classes are imbalanced. This can also be seen for the global class prevalences in Figure 4.3. It can be seen Figure 4.2 that the more prevalent classes such as 'tennis' and 'rugby' experience the largest variation in prevalence per month. This is not an issue as those classes are guaranteed to feature in a validation set. The class of 'horoscopes' is the only class which runs the risk of not featuring in a validation set. However, this class is the least prevalent as seen in Figure 4.3. This class does not play a huge role in the data and has no greater classification importance than the other classes. Finally, the article lengths per topic are examined. The article lengths for all topic classes are shown in Figure 4.4.

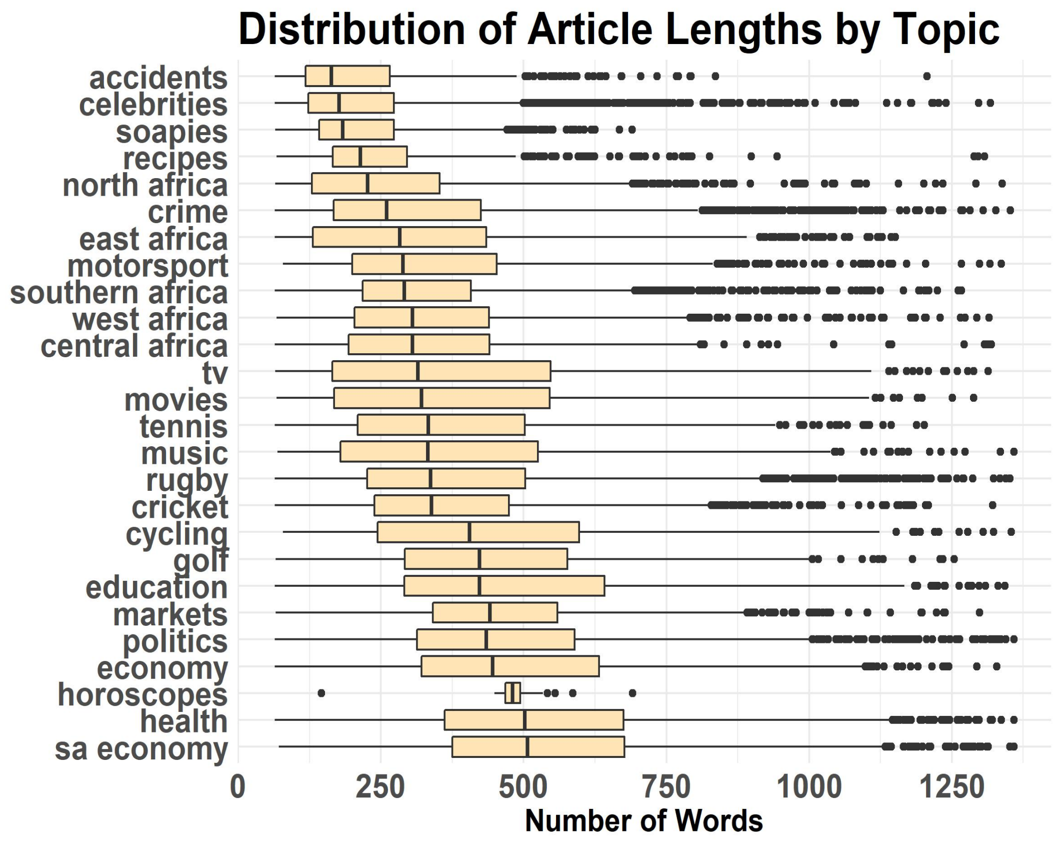


Figure 4.4: Distributions of Article Lengths

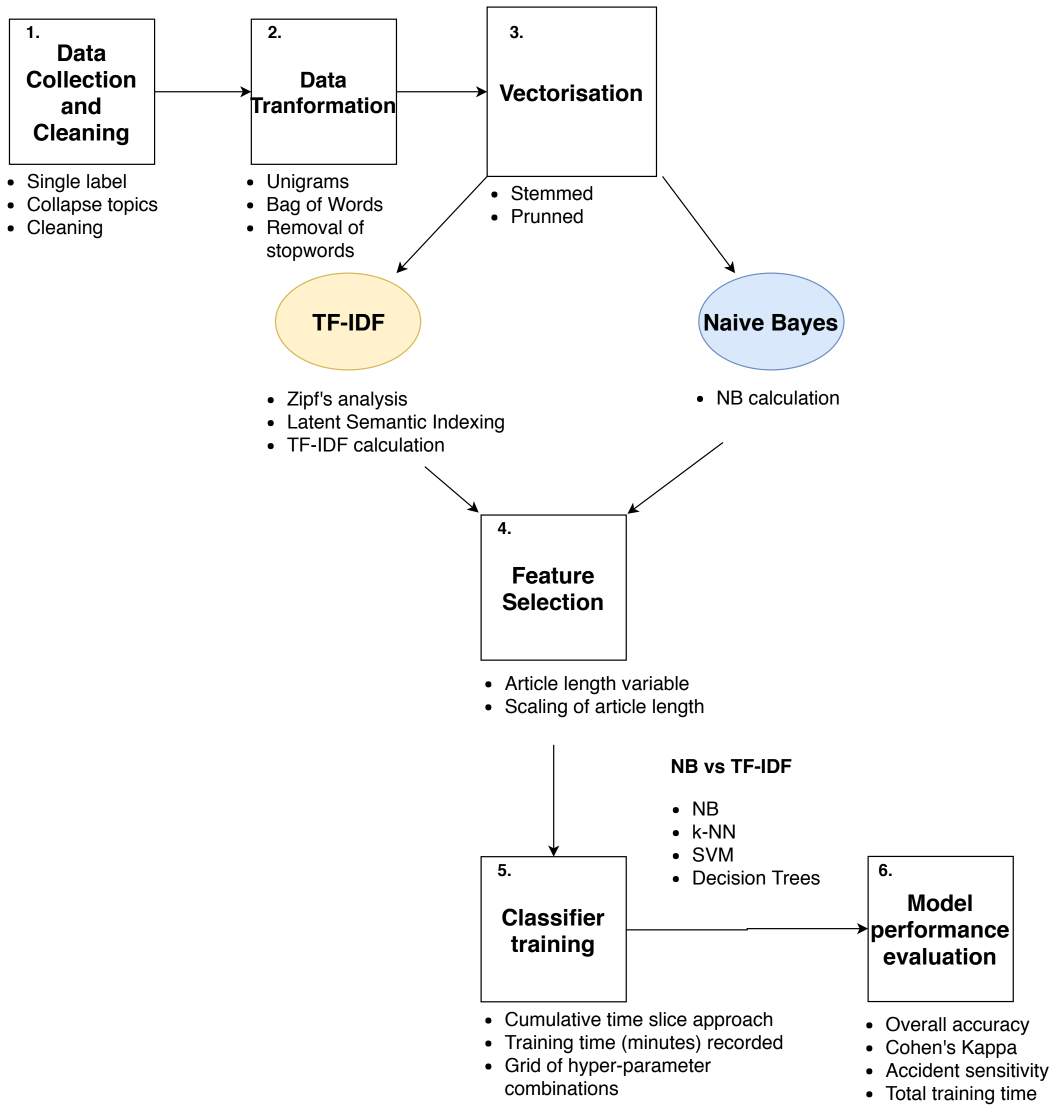
It can be seen that there is a fair amount of variation among the length of topics. This motivates the use of article length as a reliability indicator as mentioned previously in Chapter 3. This will be further explored in the section 5.4 in the following chapter.

In summary, the data consists of a large number of 26 classes and these classes have been shown to be imbalanced in prevalence. It has also been seen that all classes are guaranteed to appear in each Year-Month time slice with the exception of the very small class of 'horoscopes'. This motivates the viability of a Year-Month training-validation model testing procedure. Finally, the article lengths could be used as a reliability indicator to enrich the data. The next chapter will elaborate on the modelling process undertaken using this article data.

## 5. Methodology

Jain and Mandowara (2016) proposes a six-step workflow for approaching text classification problems. This workflow is visualised by Figure 5.1 and will act as a guideline for this study.

Figure 5.1: Methodology flow chart



The language R is used due to prior experience and high quality package implementations of the chosen algorithms. The statistical models are taken from `ranger` (Wright and Ziegler 2017),`class` (Venables and Ripley 2002) and `e1071` (Meyer et al. 2018) with `tidytext` (Silge and Robinson 2016) performing much of the text cleaning. The model object was stored using `caret` (Jed Wing et al. 2018) including all metrics. The modelling will be done using the Google Cloud Compute Engine. The virtual machine has a 8-Core Intel Sandy Bridge processor and 50 Gb of RAM. The data is first collected and cleaned.

## *5.1. Data collection and cleaning*

The data was received from Media24 via a JSON file. The article bodies are all converted to lowercase. Word contractions are split such as "don't" to "do not" and HTML tags are removed. Finally, numerical and monetary entries such as "R1000" are converted to "one thousand rand". The next step is to transform the article body text chunks into vectors of text.

## *5.2. Data transformation*

In keeping with the BoW approach, each article body was split into a vector of single words (unigrams). The vectorisation techniques described in section 5.3.1 and section 5.3.2 require the text to be in this format. There are certain structural words in language which carry no meaning by themselves. Examples of these words include 'the', 'shall', 'my' and 'will'. As mentioned in Chapter 3 it is common practice to remove these words known as stopwords. These words are removed from the text vectors. The composition of a stopword list is dependent on the problem. A general set of stopwords from the `tidytext` package is used. In addition, the word 'Fin24' is also removed. This is because this word only appeared in the "equities" topic and acted as an implicit label for the "equities" topic. The cleaned data is now in a vector form where each entry is a word. To model this data, numerical quantities need to be attached to these vectors as statistical models cannot work with text directly. This is where the TF-IDF and NB vectoriser methods are used. The TF-IDF vectorisation will now be examined.

## 5.3. *Vectorisation*

### 5.3.1. *Term Frequency - Inverse Document Frequency*

$$\text{Term Frequency}(t_i) = \text{TF}(t_i) = f_{t_i,d_j} \Bigg/ \sum_{t' \in d_j} f_{t',d_j} \tag{1}$$

$$\text{Inverse Document Frequency}(t_i) = \text{IDF}(t_i) = \log \frac{N}{n_t} = -\log \frac{n_t}{N} \tag{2}$$

$$\text{TF-IDF}(t_i) = \text{TF} \times \text{IDF} \tag{3}$$

The TF-IDF measure for a single word $t_i$ in a document $D_j$ is detailed above. TF-IDF is the product of two separate quantities as shown in Equation 3. Equation 1 details term-frequency which is the count of a specific word in a document that as been normalised by the length of the document. The quantity $f_{t_i,d}$ is the count of term $t_i$ in document $d_j$ and $t'$ indicates the compliment of term $t$. Equation 2 calculates the inverse document frequency. This is the inverse ratio of the prevalence rate of that specific word in all documents in a corpus. This ratio is often log normalised (Blei, Ng, and Jordan 2003). $N$ represents the number of documents in the corpus and $n_t$ is the number of documents where the given word is present. This metric tries to associate high TF-IDF scores to words that are deemed to be important in a document. For example, if a word appears frequently within a document but is not present in many others, it is likely that it is a defining word for said document and thus it will have a higher TF-IDF score. As mentioned in Chapter 3, the main issue with TF-IDF is that it often results in high dimensionality. Various techniques exist to mitigate this issue. The first of which is word stemming.

Stemming is the process of standardising words suffixes such that only the root of the word remains (Miao et al. 2009). For example, “Computer”, “Compute” and “Computation” are all stemmed to “Comput”. This greatly reduces the number of unique words within the corpus by combining words with similar connotations. It is important to recognise that all three of these words are essentially referring to the same thing. While stemming does reduce the number of unique words, the resultant root word may give rise to situations where it is difficult to understand that word’s meaning in the original text (Miao et al. 2009). This study is concerned with prediction accuracy and not inference so this will not be a concern moving forward. Jurafsky and Martin (2009) further supports stemming by concluding that it is an essential procedure in reducing the dimensionality. In addition to stemming, pruning of non-informative words can be done to reduce the dimensionality of the data.

Word pruning is motivated by the phenomenon known as Zipf's law. Manin (2008) states that if the words of a corpus are ranked in order of decreasing frequency, then the frequency should be inversely proportional to the rank.

$$\text{Zipf's Law} : f_k \propto k^{-B} \tag{4}$$

In Equation 4 above, $f_k$ represents the frequency of the word with rank $k$ and $B \approx 1$. Manin (2008) notes that the words that deviate from the inverse proportionality values at the low-rank and high-rank ends add little information. These words may be removed from the data. Jurafsky and Martin (2009) argues that words that only appear under a low frequency threshold, such as 10 occurrences, add noise and should also be pruned. The observed versus expected frequencies of the words are shown in Figure 5.2

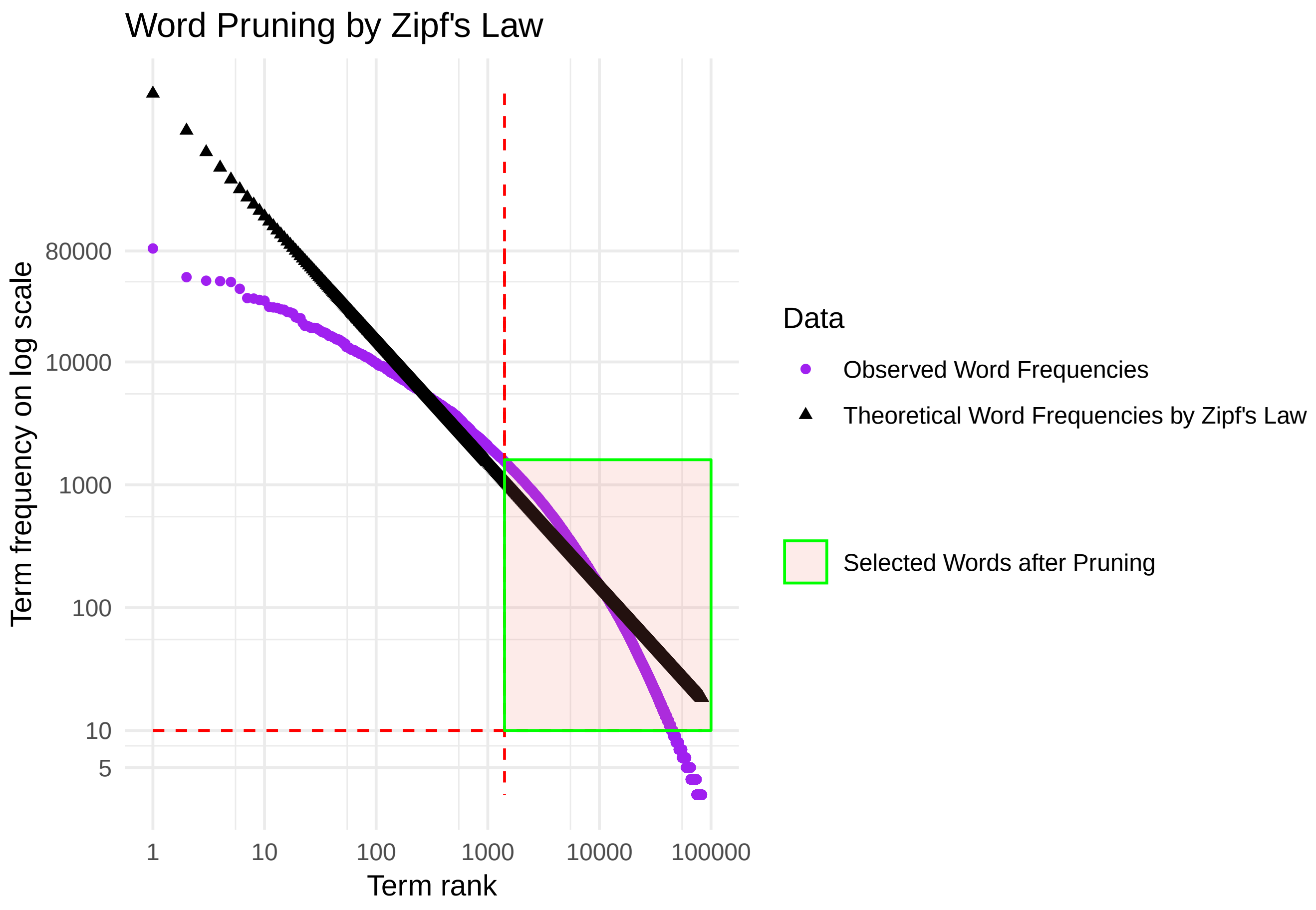


Figure 5.2: Word Pruning

Words with a frequency less than 10 occurrences across the articles account for 82% of the total unique words. By removing these words, the data will become less noisy and this allows for reduced dimensionality. The top 5% of words, shown by all data to the left of vertical line at rank 1400, are also removed. Upon inspection, many of the words which comprised the top 5% included JavaScript or HTML tags which evaded the cleaning performed in section 5.1. The resulting number of unique words reduced from 202097 to 42425. This resulting subset lies within the green box in Figure 5.2. The observed frequencies are also closer to the expected frequencies as per Zipf's law within this subset compared to the entire set of unique words. The words have been stemmed, stopwords have been removed and "noisy" words have been pruned. The TF-IDF values for the remaining words are calculated as per Equations 1-3. Finally, Latent Semantic Indexing through PCA is applied to the TF-IDF scores to further reduce dimensionality.

This study performed PCA on the TF-IDF data such that the calculated components explain an approximate 50% level of variation of the original data. This reduced dimensionality to a more manageable level without losing too much information. The final output of the TF-IDF method is a matrix where each column represents a component of weighted TF-IDF scores. To contrast the TF-IDF vectorisation process, the NB vectorisation is introduced.

### *5.3.2. Naive Bayes*

The NB classifier is traditionally used as a classification model however it has also been used as a text vectoriser in Isa et al. (2008) & Fragos, Belsis, and Skourlas (2014). The NB classifier essentially performs dimension reduction based off empirical word distributions of the text in a document. A NB classifier produces posterior predicted probabilities which result in a dimension reduction.

The values for each of the $K$ features represent the NB predicted probability that the article belongs to each category. A subsequent model such as a SVM could be trained on the predicted probabilities in order to find optimal decision rules for classification. This would contrast the more naive approach of simply classifying the document to the category with the greatest predicted NB probability. Isa et al. (2008) found that using a NB vectorisation was superior to a TF-IDF approach and particularly when using a SVM model.

Fragos, Belsis, and Skourlas (2014) found a text classification specific implementation of this vectorisation. This method assumes that the words within a document are conditionally independent given the class. This implies that the probability of a category given a document is the mean of the conditional probabilities of each word in a document pointing to that category. This is seen in Equation 5. The following equations outline this estimation.

$$Pr(c_k|d_j) = \frac{\sum_i Pr(c_k|w_{ij}) \times R_{ij}}{N_j} \tag{5}$$

$$Pr(c_k|w_i) = \frac{P(w_i|c_k).P(c_k)}{P(w_i)} \tag{6}$$

$$P(c_k) = \frac{\text{Total Word Count Across All Articles in } c_k}{\text{Total Word Count in Corpus}} \tag{7}$$

$$P(w_i) = \frac{\text{Total Word i Count Across All Categories}}{\text{Total Word Count in Corpus}} \tag{8}$$

Equation 6 shows the probability of a class given a word where $c_k$ is the category variable for the $k^{th}$ class and $d_j$ document variable for the $j^{th}$ document.The $w_i$ term is the $i^{th}$ unique word found in the corpus, $w_{ij}$ is the specific word $i$ within document $j$. $R_{ij}$ is the number of times the $i^{th}$ word appears in the $j^{th}$ document and $N_j$ is the total sum of all words in $d_j$. The end result is a vector of posterior probabilities for each document such that the probabilities are associated to each $K$ classes. The text used in this model has been filtered for stopwords, stemmed and pruned as per section 5.3.1. The data has now been vectorised into a numerical format using either TF-IDF or NB. The next step in the modelling process is engineering informative features into this data.

## *5.4. Feature selection*

The reliability indicator of the article length is added to the vectorised data. This is the count of all words in an article. The usage of these indicators are motivated by Bennett, Dumais, and Horvitz (2002). The article lengths are normalised to a $[0; 1]$ range as to not bias the model learning in algorithms such as k-NN. This feature seeks to provide additional information about the article topic that the text does not provide. Other common reliability measures include diversity of vocabulary and average sentence length. Bennett, Dumais, and Horvitz (2002) justifies using article length specifically because performance of classifiers can in some cases be correlated with document length. The article lengths for all topics were shown in Figure 4.4. It can be seen that there is a fair amount of variation among the topics. This is additional evidence for article length to feature in the data. Both the NB vectorisation and TF-IDF vectorised data shall incorporate the normalised article length moving forward. The text data has been converted into numerical values and has been enriched with the article length feature. The next section will explore the statistical models which will be used on this data.

## 5.5. *Statistical Models*

### 5.5.1. *Decision Tree Forests*

A binary classification tree is a data structure where each internal node forms a test to split the data. Branches represent the outcome of the test. In classification trees, the most commonly occurring class in the terminal node determines the classification label (Hastie and Tibshirani 2013). Classification trees are grown recursively and attempt to split the feature space into mutually exclusive regions according to some split rule. Binary recursive splitting is the process of selecting a feature $X_j$ and its corresponding cutpoint $s$ such that it minimises total terminal node impurity. This is done by recursively splitting the feature space into regions like $R_1$ and $R_2$ in Equations 9 & 10 until a stopping criterion is reached.

$$R_1(j,s) = \{X | X_j < s\} \tag{9}$$
$$R_2(j,s) = \{X | X_j \geq s\} \tag{10}$$

$$G_m = \sum_{k=1}^{K} \hat{p}_{mk}(1 - \hat{p}_{mk}) \tag{11}$$
$$G = \sum_{k=1}^{K} \hat{p}_{R_1 k}(1 - \hat{p}_{R_1 k}) + \sum_{k=1}^{K} \hat{p}_{R_2 k}(1 - \hat{p}_{R_2 k}) \tag{12}$$

This splitting rule is commonly the reduction in node impurity (measured by entropy or Gini index) by some threshold at each split (Hastie and Tibshirani 2013). Equation 11 shows the Gini index where $\hat{p}_{mk}$ is the proportion of observations in the $m^{th}$ region from the $k^{th}$ class. Equation 12 is the overall Gini index across the regions $R_1$ and $R_2$. Each split selects the feature and the appropriate cut off value which produced the greatest reduction in the overall Gini index. Geurts, Ernst, and Wehenkel (2006) developed another splitting rule to contrast the aforementioned greedy approach. This approach is known as "Extratrees" (extremely randomised trees) as it randomly selects a cut-point when constructing classification trees. This approach is not greedy as the classification tree no longer minimises node impurity at each split.

A drawback of growing a large single decision tree is overfitting. This occurs when the model fits on the noise of the training data and does not generalize well to out of sample data. The model is then said to have a high variance, meaning on different samples of data the tree structures vary significantly. This changes the terminal nodes making it difficult to use the tree for reliable prediction (Hastie and Tibshirani 2013). To reduce this sampling variation bagged trees were introduced as mentioned in Chapter 3.

Breiman (1996) explains that bootstrapping uses repeated sampling with replacement of the training data. For each bootstrap sample a decision tree is grown and the predicted category for all trees is aggregated. This process is known as "Bagging". For classification problems, the prediction for a particular observation is the majority vote over all bootstrap samples (Liaw and Wiener 2001). Breiman (1997) shows, in Equation 13, that a combination of classifiers $h_m(\vec{x})$ having vote $c_m$ can be used to classify new observations. $I$ denotes the indicator function and $\vec{x}$ is classified into the class such that $\text{argmax}_j s(j, \vec{x})$ is the maximum vote.

$$
\begin{aligned}
s(j, \vec{x}) &= \sum_m c_m I(h_m(\vec{x}) = j) \qquad (13) \\
m &= 1, \ldots, M
\end{aligned}
$$

Equation 13 can be interpreted as a voting procedure. An example would be document variable $d_j$ would be classified to the class $c_k$ for which the majority of all bootstrapped classification trees vote. Breiman (1996) found using bagging led to improvements in accuracy and reduces overfitting.

Random Forests decorrelate bagged trees by taking random subsets $l$ of the total feature space $p$ (Chen, Liaw, and Breiman 2004). This is important as if there is a particular group of important features the bagged trees will not find subtle relationships between alternative features. This random subset $l$ is denoted as "Mtry" in the model parameters in Chapter 9. The NB vectorised data contains probabilities meaning that taking random subsets of features would exclude the possibility of an article belonging to class outside of the selected subset. For NB vectorised data the bagged tree forest shall be used and Random Forests will be used for the TF-IDF vectorised data.

### *5.5.2. k-Nearest Neighbours*

The k-NN classifier first calculates the distance between points. It then uses these distances for prediction by assigning an unclassified point to the class most heavily represented by the closest $k$ points (Cover and Hart 2006). Hastie and Tibshirani (2013) explains that the choice of $k$ has drastic effect on the classifier obtained. For small $k$ the decision boundary is flexible and varies with new training data. This suggests a high variance and low bias. For large $k$ more points are considered and forms a more generalised decision boundary suggesting a lower variance and higher bias. The number of points to consider is a parameter of the model that will be tuned in training.

There exist many metrics for calculating the distance to the nearest point such as: Euclidean distance, Manhattan distance and Mahanalobis. The distance calculated between vectors $\vec{x_i}$ and $\vec{x_{i'}}$ are shown in the equations below.

Euclidean distance

$$d(\vec{x_i}, \vec{x_{i'}}) = \sqrt{(x_{i1} - x_{i'1})^2 + (x_{i2} - x_{i'2})^2 + \cdots + (x_{ip} - x_{i'p})^2} \tag{14}$$

Manhattan distance

$$d(\vec{x_i}, \vec{x_{i'}}) = \sum_{j=1}^{p} |x_{ij} - x_{i'j}| \tag{15}$$

Mahanalobis distance

$$d(\vec{x_i}, \vec{x_{i'}}) = \sqrt{(\vec{x_i} - \vec{x_{i'}})^T S^{-1} (\vec{x_i} - \vec{x_{i'}})} \tag{16}$$

Mahanalobis distance requires the covariance matrix $S$ to be found between the vectors and inverted as seen in Equation 16. Mahanalobis distance is a measure of dissimilarity between vectors assuming the same distribution and covariance matrix. In this context, each vector represents a document vector that has been vectorised via TF-IDF or NB. Weinberger, Blitzer, and Saul (2006) notes that in the absence of prior knowledge the Euclidean distance is most commonly used, but does not capture any statistical subtleties of the data and should be adjusted according to the specific problem. For this study only Euclidean distance shall be considered.

### *5.5.3. Support Vector Machines*

Hastie and Tibshirani (2013) define a hyperplane in a dimensional space $p$ as a flat subspace with its dimension being $p - 1$. For example in two dimensions a hyperplane is a single dimensional line. Equation 17 shows the equation for the hyperplane. This means for any vector $(x_{i1}, x_{i2} \ldots x_{ip})$, for which Equation 17 holds, is a point on the hyperplane.

$$\beta_0 + \beta_1 x_{i1} + \beta_2 x_{i2} + \cdots + \beta_p x_{ip} = 0 \tag{17}$$

Consider the case where only two class labels exist such that $y_1, y_2 \ldots y_p \in$ {-1,1}. Suppose it is possible to construct a separating hyperplane to perfectly separate observations according to class labels. Hastie and Tibshirani (2013) define Equation 18 to show the property of the separating hyperplane and how it leads to a linear decision boundary. This classifier classifies a test observation depending on which side of the hyperplane it lies.

$$
\begin{aligned}
& y_i(\beta_0 + \beta_1 x_{i1} + \beta_2 x_{i2} + \cdots + \beta_p x_{ip}) > 0 && (18)\\
& \forall \ i = 1, \ldots n
\end{aligned}
$$

The margin is defined to be the minimum distance from the hyperplane to the training observations. A maximal margin classifier uses a separating hyperplane where the margin is maximised (Hastie and Tibshirani 2013). The margin determines the distance between the separating hyperplane and the support vectors. Theses $p$-dimensional vectors "support" the maximal margin hyperplane as if these points are moved the maximal margin hyperplane will follow. Hastie and Tibshirani (2013) explains that movement in any other vectors would have no effect on the separating hyperplane.

Often data is non-separable or it could be worthwhile to misclassify a few training observations to do a better job in classifying the remaining observations. A soft margin classifier could be used. The equations below outline the optimisation problem for the soft margin classifier.

$$
\begin{aligned}
& \text{maximise } M && (19)\\
& \text{subject to } \sum_{j=1}^{p} \beta_j^2 = 1, && (20)\\
& y_i(\beta_0 + \beta_1 x_{i1} + \beta_2 x_{i2} + \cdots + \beta_p x_{ip}) \geq M(1 - \epsilon_i), && (21)\\
& \epsilon_i \geq 0, \ \sum_{i=1}^{n} \epsilon_i \leq C && (22)
\end{aligned}
$$

The objective is to maximise $M$, the width of the margin. The slack variables $\epsilon_1, \epsilon_2 \ldots \epsilon_n$ allow observations to be misclassified, giving it the name of "soft" classifier. The term $C$ is a non-negative tuning parameter which bounds the slack variables. $C$ is known as the "budget" and is tuned using the cost hyperparameter seen in the appendix Chapter 9. If $C = 0$ then there is no budget for misclassifications and the optimisation problem uses the maximal margin classifier. As $C$ increases the restraint on Equation 22 allows the slack variables to increase. Hastie and Tibshirani (2013) explains that this allows for violations to the margin by making the margin wider. They further deduced that the soft margin objective improves overall classification performance of the maximal margin classifier.

A key improvement on this algorithm was the enlarging of the feature space using a kernel approach, allowing for non-linear decision boundaries between features (Hastie and Tibshirani 2013). The kernel approach attempts to find a hyperplane in a higher dimension to correctly segregate the feature space. A variety of kernel functions exist such as the linear, radial, polynomial and sigmoid kernels. The kernel $K$ is defined as a function of the following vectors $\vec{x_i}$ ,$\vec{x_{i'}}$ and alters Equation 21 as seen below in Equation 23.

$$y_i(K(\vec{x_i}, \vec{x_{i'}})) \geq M(1 - \epsilon_i), \tag{23}$$

Equation 24 shows a linear kernel function as the sum of the product of the vectors $\vec{x_i}$ and $\vec{x_{i'}}$ over the feature space $p$. Equation 25 shows a radial kernel where $\gamma$ is a positive constant which can be tuned in training.

$$K(\vec{x_i}, \vec{x_{i'}}) = \sum_{j=1}^{p} x_{ij} x_{i'j} \tag{24}$$

$$K(\vec{x_i}, \vec{x_{i'}}) = exp(-\gamma \sum_{j=1}^{p} (x_{ij} - x_{i'j})^2) \tag{25}$$

Radial kernels are preferred for non-linear problems due to their computational advantages over other kernel functions. The radial kernel shall be used for this study. If computation becomes too expensive a linear kernel shall be used. The research objective involves multiclass classification thus a one-versus-one approach may be used. This creates $\binom{K}{2}$ SVM's and then compares a pair of classes using the method as seen before (Hastie and Tibshirani (2013)). The following section explains how these statistical models will be trained on the news article data.

## *5.6. Classifier training*

Each article has a publication date attached to it. The oldest article was published on the 12/10/2016 and the most recent on the 31/01/2018. This results in the training-validation running from 12/10/2016 to 31/01/2018. This set contains 15 months worth of data. An interesting aspect which will be explored is how the accuracy of the models changes with time. For each of the 15 months in the train set, 15 train-validate cycles occurred.

The number of months used for training increases with each training cycle and the validation set is always the next month's articles. For the first train-validate cycle, the month of October 2016 is used for training and the validation set is November 2016. The second cycle trains on October-November 2016 and is validated on December 2016. This continues for all of the cycles. This process is visualised in Figure 5.3.

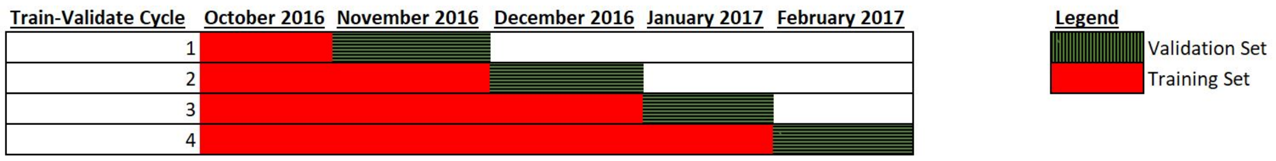


Figure 5.3: Visualisation of first 4 Train-Validate Cycles

At each cycle, validation set prediction accuracy is recorded in a `caret` confusion matrix object which calculates a variety of accuracy metrics such as sensitivity, overall accuracy and Cohen's Kappa. The training time in minutes is also recorded at each cycle. All new words seen in the validation sets are excluded from the prediction phase of the month where they were first encountered, but then they form part of the training set for the next cycle.

A simple grid search of hyperparameters for the k-NN, SVM and the decision tree forests is used. The ranges of the hyperparameters are guided by empirical "rules-of-thumb" by researchers and the recommendations in the appropriate R packages. The `e1071` package is used for the SVM and the NB model trained on TF-IDF data, `ranger` for the bagged/random forests and `class` for k-NN. This study seeks to balance accuracy and computational efficiency of both the models and vectorisation methods. The following section will define how these characteristics will be measured.

## *5.7. Model performance evaluation*

Three metrics will be used to measure model accuracies. These include overall accuracy, Cohen's Kappa and 'accident' class sensitivity. The theory and motivations for why they will be used in this text classification context will be illustrated for each metric.

### *5.7.1. Overall Accuracy*

Overall accuracy is the proportion of total articles in which the models classified correctly. This is a simple measure of accuracy that is easily understood. However McHugh (2012) argues that this measure runs the risk of overstating the true accuracy of a model. This is due to the fact that agreement between the actual class labels and the predicted class labels could be due to pure chance and not due to the reliability of the model. Cohen's Kappa is introduced to take this into account.

### *5.7.2. Cohen's kappa and Accident Sensitivity*

Cohen's Kappa assesses the level of agreement between two raters while removing the agreement caused by chance. It is traditionally used in biostatistics as a measure of the quality of data collected between human observers otherwise known as raters. In this context, the one rater is the actual class labels of the articles and the other rater is the predicted classes. Two quantities play a role in calculating the level of agreement between the predicted and actual classes. The first is the actual raw agreement rate which is the same as overall accuracy. The other quantity is the expected or per chance agreement rate. The expected agreement rate, as shown in Equation 26, is the theoretical overall accuracy rate of a model that classifies articles randomly into classes. Equation 27 shows the difference of these two quantities standardised to form Cohen's Kappa ($\kappa$).

$$\text{Expected Agreement Rate} = \frac{\sum_{j=1}^{k} \frac{\text{predicted}_j \times \text{actual}_j}{N}}{N} \tag{26}$$

$$\kappa = \frac{\text{Raw Agreement Rate} - \text{Expected Agreement Rate}}{1 - \text{Expected Agreement Rate}} \tag{27}$$

The quantity $N$, in Equation 26, is the total number of articles used, $\text{predicted}_j$ is the number of articles that a model predicted to belong to the j$^{th}$ class, $\text{actual}_j$ is the actual number of articles which belong to the j$^{th}$ class and $k$ is the total number of classes which exist. The value of kappa ranges between negative one to positive one. Like the Pearson correlation coefficient, kappa is typically squared to be an accuracy measure between the raters (McHugh 2012).

While Cohen's Kappa does take into account the probability that a model classifies correctly by chance, McHugh (2012) argues that the main drawback is that it cannot be interpreted directly unlike overall accuracy. There are guidelines which exist for interpreting Cohen's Kappa. These are are drawn from the work of McHugh (2012).

***Squared Cohen's Kappa Agreement Guidelines***

| Value of Kappa | Agreement Level |
|---|---|
| 0 - 0.2 | None |
| 0.21 - 0.39 | Minimal |
| 0.4 - 0.59 | Weak |
| 0.6 - 0.79 | Moderate |
| 0.8 - 0.9 | Strong |
| Above 90 | Almost Perfect |

Despite the interpretability drawback, Jeni, Cohn, and De La Torre (2013) motivates that Cohen's Kappa is a good metric to use when faced with imbalanced class data. The imbalanced nature of the article data is seen in Figure 4.3. This is important to keep in mind as classifiers have a tendency to ignore low prevalence classes in favour of more common classes in maximising overall accuracy (Jeni, Cohn, and De La Torre 2013). This is of particular importance in this context as a low prevalence article about 'accidents' should not be classified as a more prevalent class such as 'rugby'. Through prior modelling experiments with this article data, it was noticed that often the 'accidents' class would be incorrectly classified as 'crime'. The number of 'accident' articles seen in each training cycle is shown in Figure 5.4. Since Cohen's Kappa controls for overall class imbalances only, it would be prudent to track the performances of models on the 'accident' class specifically.

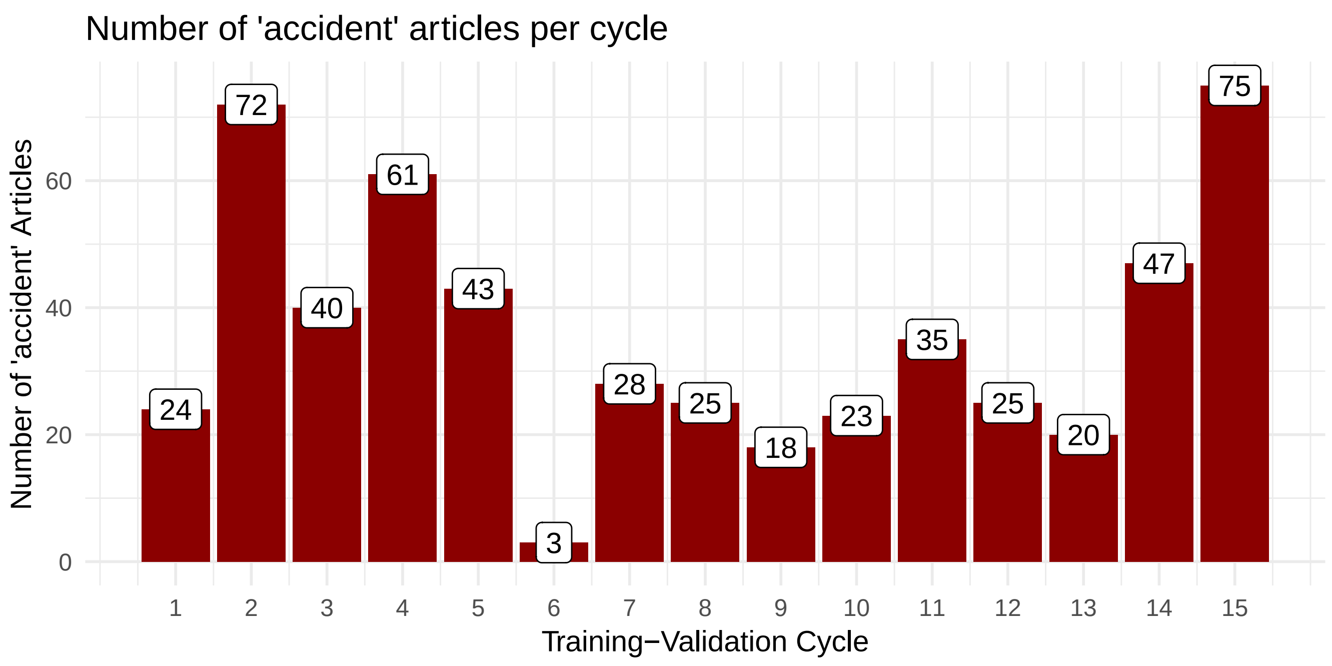


Figure 5.4: Number of Accident Articles Published per cycle

The metric used to measure ‘accident’ class accuracy is sensitivity. Sensitivity is the proportion of ‘accident’ articles predicted correctly out of the total number of actual ‘accident’ articles in a validation set. These three metrics together provide a holistic view on the accuracy of the models. In conjunction with model accuracy, the computational efficiency of the models are assessed as per the research objective.

### *5.7.3. Computational Efficiency*

Computational efficiency will be measured via the total training time. The total training time is the sum of the vectorisation and training time. The vectorisation time is the time that the TF-IDF or NB vectorisation methods take to convert the data in its original text form to its vectorised numerical format. The training time is the time that each model takes to train on the vectorised data and perform predictions on the validation sets.

Both the NB and TF-IDF vectorised data was used on all the statistical models defined in section 5.5 across all 15 training-validation cycles. These methods will be assessed in the following chapter using the metrics defined previously.

# 6. Results

In assessing the accuracy of the models, three metrics will be used. These include the overall accuracy, Cohen’s Kappa and “accident” class sensitivity. The best model configurations will be determined by the greatest mean values of each accuracy metric. The median and the standard deviation over the 15 training cycles will also be reported. Afterwards, the computational efficiency of the optimal accuracy models will be evaluated via the total training times in minutes per training cycle.

## *6.1. Accuracy*

The Tables 6.1, 6.2 and 6.3 reflect the performances of the models over 15 training-validation cycles for each vectorisation technique. The hyperparameters for the optimal models per accuracy metric is also shown in the appendix Chapter 9.

Table 6.1: Overall Accuracy

| ***Accuracy by Overall Accuracy*** | | | | |
|---|---|---|---|---|
| Vectorisation | Model | Mean | Median | Standard Deviation |
| Naive Bayes Vectorisation | Decision Tree Forest | 93.28 | 93.29 | 1.00 |
| Naive Bayes Vectorisation | k-NN | 92.96 | 93.08 | 1.08 |
| Naive Bayes Vectorisation | SVM | 92.77 | 93.16 | 1.22 |
| Naive Bayes Vectorisation | Naive Bayes | 86.70 | 86.75 | 1.56 |
| TF-IDF Vectorisation | SVM | 87.04 | 87.44 | 1.53 |
| TF-IDF Vectorisation | Decision Tree Forest | 84.31 | 85.00 | 2.09 |
| TF-IDF Vectorisation | k-NN | 73.55 | 75.10 | 3.66 |
| TF-IDF Vectorisation | Naive Bayes | 41.50 | 41.49 | 1.86 |

Table 6.2: Kappa

| ***Accuracy by Cohen's Kappa*** | | | | |
|---|---|---|---|---|
| Vectorisation | Model | Mean | Median | Standard Deviation |
| Naive Bayes Vectorisation | Decision Tree Forest | 92.80 | 92.76 | 1.07 |
| Naive Bayes Vectorisation | k-NN | 92.45 | 92.49 | 1.14 |
| Naive Bayes Vectorisation | SVM | 92.25 | 92.69 | 1.31 |
| Naive Bayes Vectorisation | Naive Bayes | 85.66 | 85.75 | 1.64 |
| TF-IDF Vectorisation | SVM | 86.11 | 86.55 | 1.62 |
| TF-IDF Vectorisation | Decision Tree Forest | 83.16 | 83.94 | 2.20 |
| TF-IDF Vectorisation | k-NN | 71.67 | 73.50 | 3.82 |
| TF-IDF Vectorisation | Naive Bayes | 38.50 | 38.43 | 1.95 |

The decision tree forest and k-NN were the top two models for NB vectorised models across all accuracy metrics. For the TF-IDF vectorisation, the SVM and decision tree forests were the two top models. The mean and median values for all models were similar to each other for Cohens Kappa and overall accuracy. The standard deviations of overall accuracy and Kappa for the top models were just over 1% as seen in Table 6.1 and Table 6.2. This implied that the model accuracy as per these metrics proved to be quite stable across training cycles. The mean values for Cohen's Kappa were typically around 1% less than that of overall accuracy. This meant that the models did not neglect the low prevalence classes significantly as a whole across training-validation cycles.

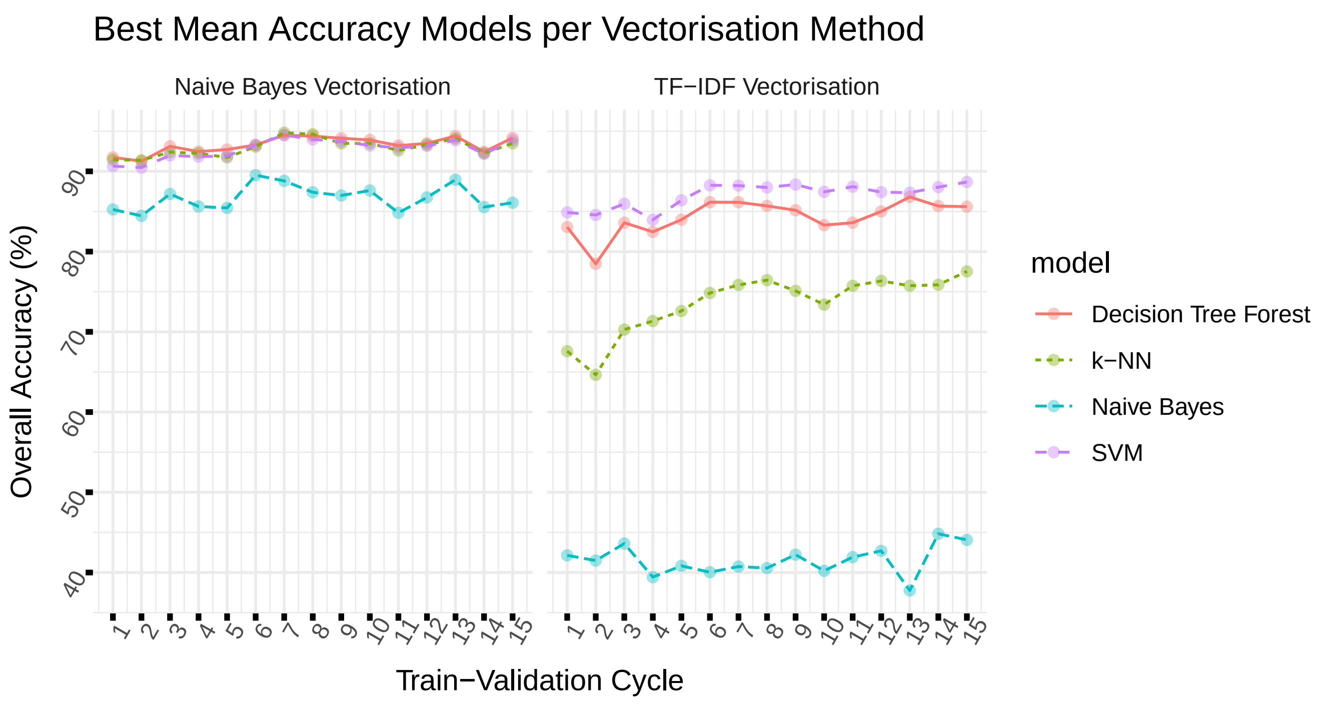


Figure 6.1: Overall Accuracy Over Training-Validation Cycles

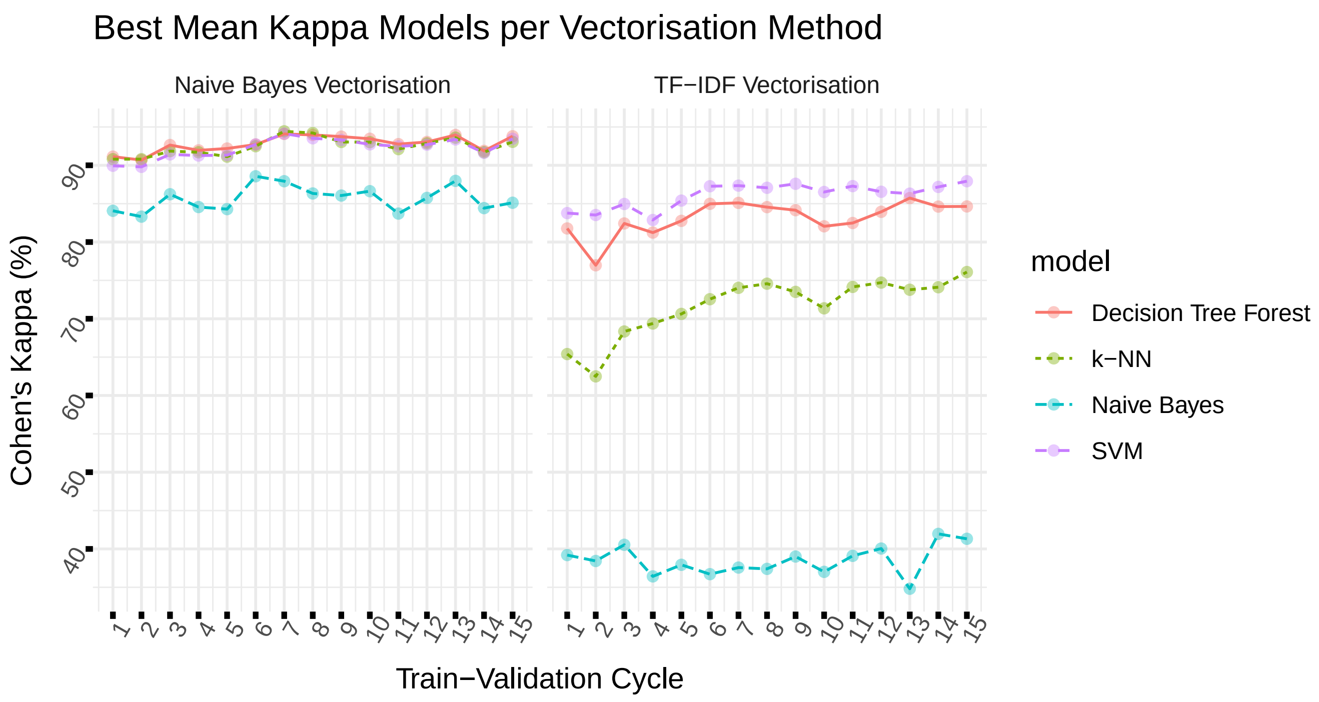


Figure 6.2: Kappa Over Training-Validation Cycles

Figures 6.1 and 6.2 suggested a positive trend between accuracy and training cycles for the TF-IDF vectorisation methods. There was only a very slight positive trend between overall accuracy and training cycles for the NB vectorised models. Noting that each training cycle is an accumulation of data from previous months, this lack of a clear trend with naive bayes vectorised models supposed that the models did not necessarily perform better with more data. However, for the TF-IDF vectorised models, the models tended to perform better on average with each successive training cycle with the exception of the naive bayes model which performed poorly across all cycles.

Despite the positive trend experienced by the TF-IDF vectorised models it was clear that on average, models performed better in both Kappa and overall accuracy with the NB vectorisation. It is not certain as to whether or not this would still be the case with more data due to the upward trend of TF-IDF vectorised models, but this may be an avenue for further research. While the similarity between the Kappa and overall accuracy measures suggested that the models did not neglect the low prevalence classes in general, the low prevalence 'accident' class was monitored specifically.

Table 6.3: "Accident" Sensitivity

| ***Accuracy by 'Accident' Sensitivity*** | | | | |
|---|---|---|---|---|
| Vectorisation | Model | Mean | Median | Standard Deviation |
| Naive Bayes Vectorisation | Decision Tree Forest | 68.19 | 75.00 | 20.26 |
| Naive Bayes Vectorisation | k-NN | 64.63 | 69.77 | 16.11 |
| Naive Bayes Vectorisation | SVM | 64.22 | 71.43 | 20.31 |
| Naive Bayes Vectorisation | Naive Bayes | 9.69 | 8.00 | 9.62 |
| TF-IDF Vectorisation | SVM | 61.98 | 62.30 | 11.69 |
| TF-IDF Vectorisation | SVM | 61.98 | 62.30 | 11.69 |
| TF-IDF Vectorisation | SVM | 61.98 | 62.30 | 11.69 |
| TF-IDF Vectorisation | Decision Tree Forest | 53.27 | 53.57 | 16.01 |
| TF-IDF Vectorisation | k-NN | 40.79 | 44.19 | 15.36 |
| TF-IDF Vectorisation | Naive Bayes | 37.10 | 38.89 | 7.08 |

The sensitivity of the class 'accident' varied significantly over training-validation cycles. The standard deviations of all the mean sensitivities were far greater than those of Kappa or overall accuracy. The Figure 6.3 illustrated this through the highly variable sensitivity rates across training cycles. There was no clear trend of sensitivity improving across training cycles. The NB vectorised decision tree forest had the greatest mean 'accident' sensitivity yet it also had the highest standard deviation of 20.26%. The best model for TF-IDF models was a linear kernel SVM at different cost parameters. This SVM model had a lower mean sensitivity at 61.98% , but its standard deviation was significantly smaller at 11.69%.

Although the top two NB vectorised models did outperform the TF-IDF vectorised models in both mean and median, this was not nearly as dominant as seen in overall accuracy or Kappa. Even though high volatility was present in the sensitivity results, training cycle 6 is of particular interest with majority of models not predicting any of the 'accident' articles correctly. This can be seen by the dip in sensitivity rate at cycle 6 in Figure 6.3. This dip was a result of the fact that only 3 'accident' articles were written during that month as shown in Figure 5.4. Only the NB vectorised k-NN model, TF-IDF SVM and TF-IDF with a NB classifier managed to predict 1 out of those 3 articles correctly.

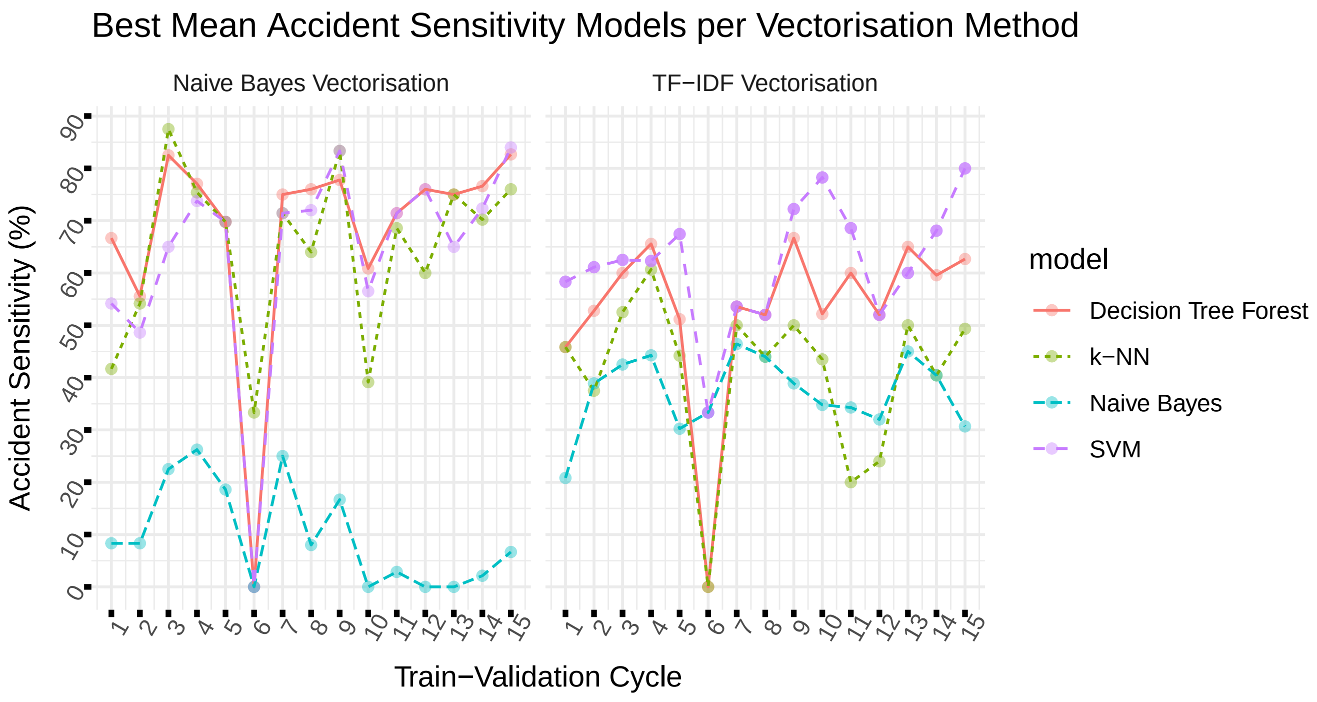


Figure 6.3: Accident Sensitivity Over Training-Validation Cycles

The NB vectorisation tended to yield more accurate models overall across metrics with the decision tree forest and k-NN proving to be good performers. The SVM proved to be the most accurate TF-IDF vectorised model. The computational efficiency of the vectorisation methods and the statistical models will now be assessed.

The duration of vectorisation for the NB and TF-IDF methods will now be examined as well as the training times for the models. The total training time is defined to be the addition of the vectorisation time and the model training time. The mean, median and standard deviations of the total training times in minutes are shown in Table 6.4.

Table 6.4: Total Training Times in minutes

| Vectorisation | Model | Mean | Min | Max | Median | Standard Deviation |
|---|---|---|---|---|---|---|
| Naive Bayes Vectorisation | Naive Bayes | 1.79 | 0.58 | 3.26 | 1.77 | 0.81 |
| Naive Bayes Vectorisation | k-NN | 1.83 | 0.59 | 3.37 | 1.81 | 0.83 |
| Naive Bayes Vectorisation | SVM | 2.11 | 0.60 | 4.08 | 2.03 | 1.07 |
| Naive Bayes Vectorisation | Decision Tree Forest | 2.26 | 0.65 | 4.30 | 2.19 | 1.12 |
| TF-IDF Vectorisation | Naive Bayes | 39.99 | 3.20 | 97.19 | 43.12 | 26.92 |
| TF-IDF Vectorisation | Decision Tree Forest | 44.49 | 3.45 | 108.56 | 47.07 | 30.16 |
| TF-IDF Vectorisation | k-NN | 73.35 | 3.71 | 230.21 | 72.29 | 60.36 |
| TF-IDF Vectorisation | SVM | 101.16 | 5.01 | 286.52 | 84.80 | 82.75 |

The NB statistical models for both vectorisation techniques yielded the lowest mean **total training times** as seen in Table 6.4, but as seen in Figures 6.1 and 6.2 in the previous section, these models did not predict well compared to others. The most accurate NB vectorised models were the decision tree forest and k-NN, had a mean total training times of 2.26 minutes and 1.83 minutes respectively. The most accurate TF-IDF models, SVM and decision tree forest, experienced far greater mean total training times at 101.16 minutes and 44.49 minutes respectively. The most efficient models were all trained using NB vectorised data. It is clear, from Table 6.4, that the NB vectorisation method completed its vectorisation more efficiently than the TF-IDF method.

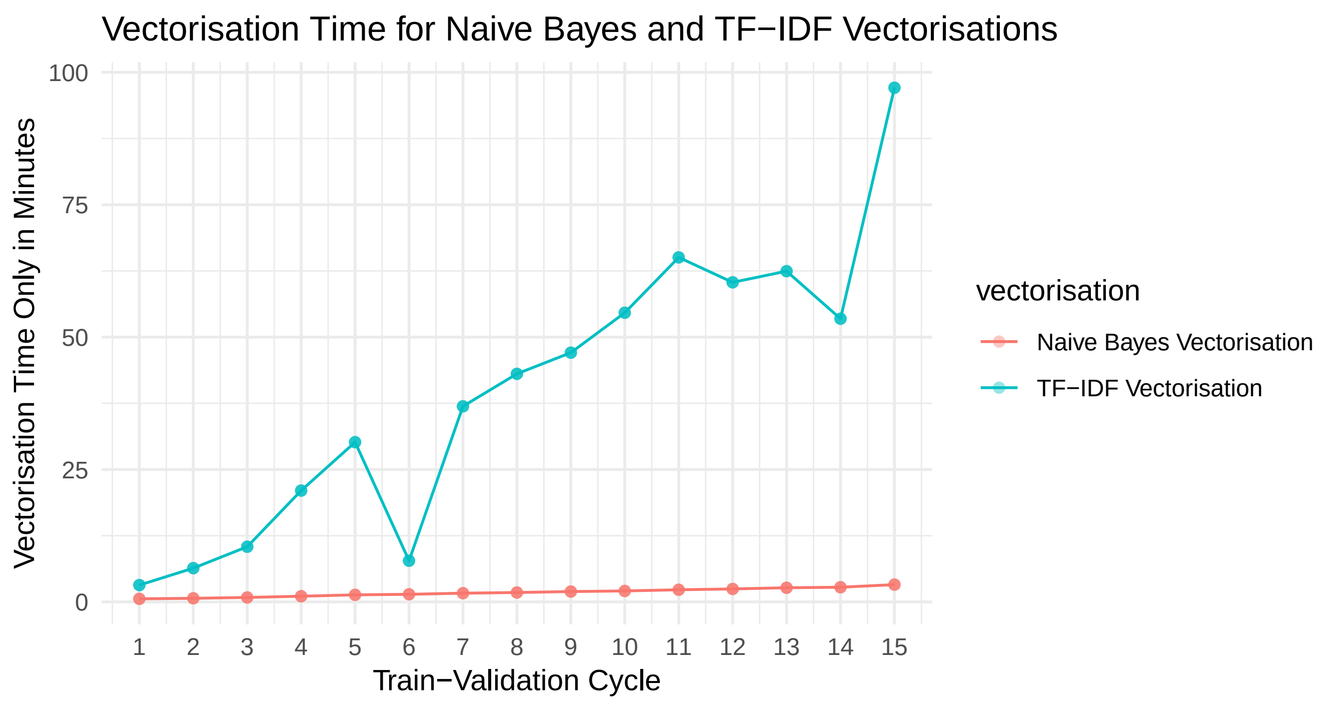


Figure 6.4: Vectorisation times plot

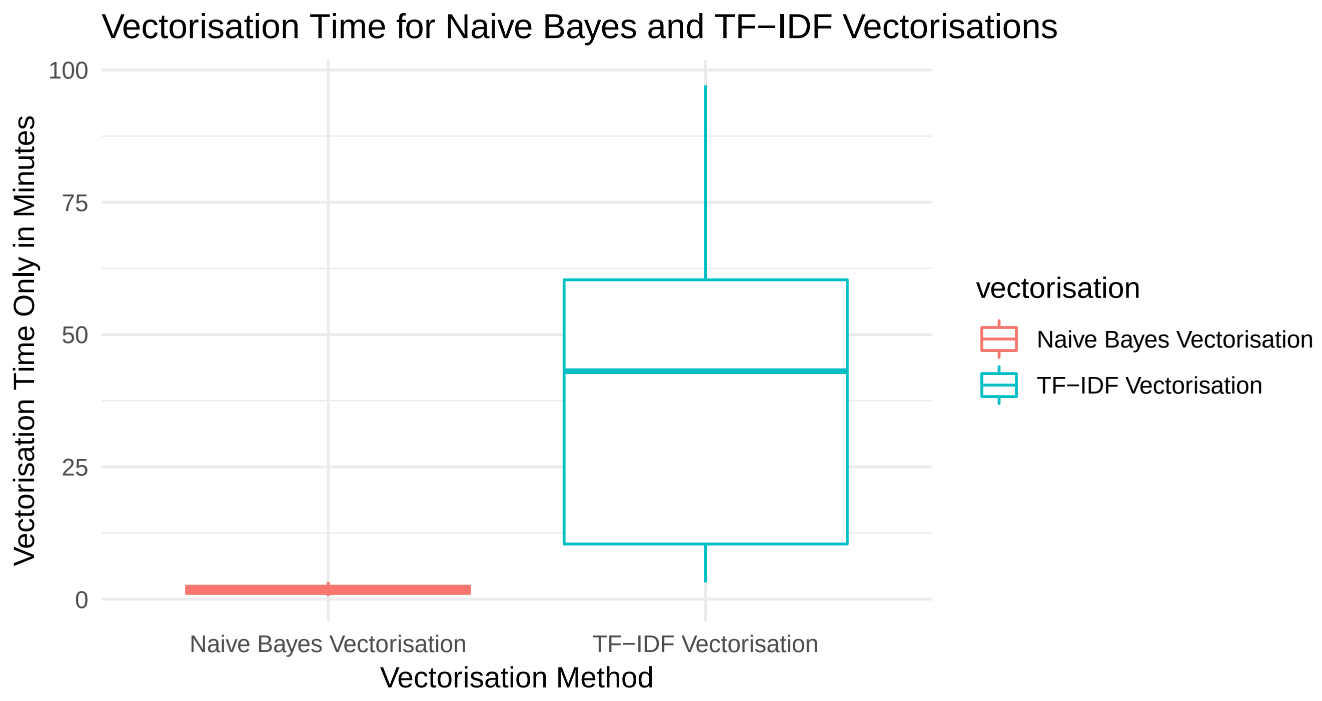


Figure 6.5: Vectorisation times distribution

The TF-IDF times in Figure 6.4 showed a clear upward trend with training cycles. A possible reason for this strong trend was the PCA implementation taking exponentially longer at larger training set sizes. The range of TF-IDF vectorisation times ranged from 3.17 minutes to 97.1 minutes with a median value of 43.1 minutes. The NB vectorisation times ranged between 0.581 minutes to 3.26 minutes with a median value of 1.77 minutes. These values are visualised via boxplots in Figure 6.5. NB was clearly the faster vectorisation method. The total training times per model were also examined to help inform model choice.

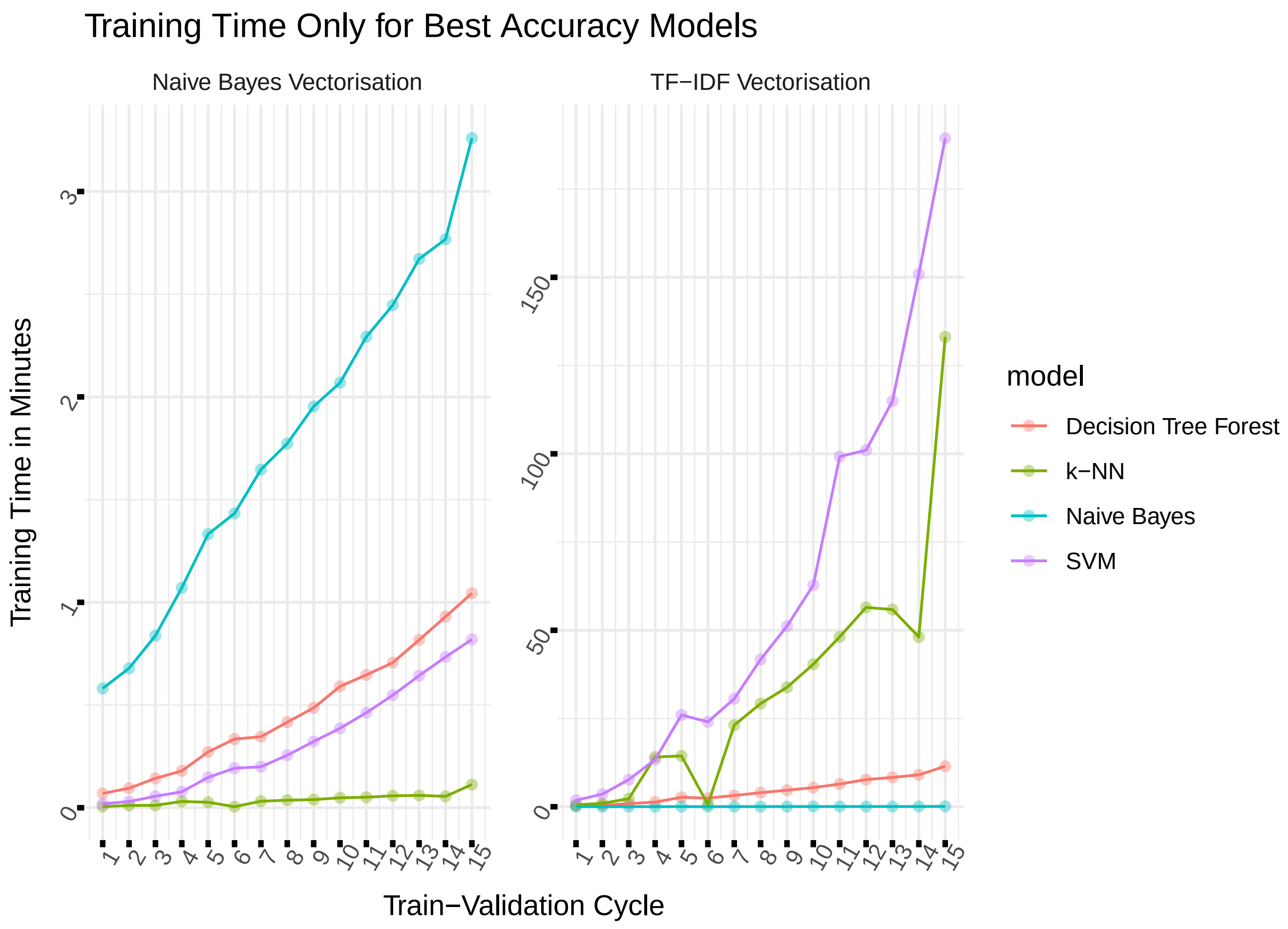


Figure 6.6: Training Times of Models

The models trained on the NB vectorised data as shown in Figure 6.6, took substantially less time than their TF-IDF counterparts. This could be attributed to the difference in dimensionality between the two forms of vectorised data. The number of features in the NB vectorised data was 27 whereas the TF-IDF data ranged between 490 to 1460 across training cycles. The dimensionality of the NB data was determined by the total number of classes which was 26 and the article length reliability indicator which added an additional dimension. Therefore the increased training times in NB vectorisation was a result of increasing sample size with each cycle, but the dramatic increased training times for TF-IDF was mostly caused by increased dimensionality. The increased sample size would of had comparatively small effect for the TF-IDF models. The TF-IDF vectorised dimensionality was determined by the number of components calculated using PCA. The number of components were calculated such that approximately 50% of the original TF-IDF data variation was explained. Since new vocabulary was introduced with each training cycle, the number of components required to meet this threshold increased with each training cycle. This could explain why the models took longer to train on TF-IDF data due to the greater dimensionality.

This difference in TF-IDF vectorised dimensionality seemed to have affected the models differently. The SVM and k-NN models experienced the greatest increases in training times with increased dimensionality cause by successive training cycles. The NB and decision tree forest models were not affected as severely. A multitude of techniques and software implementations exist for dimension reduction so perhaps the TF-IDF vectorisation time could feasibly be improved. The next chapter shall draw conclusions as to which methods and models are superior.

# 7. Conclusions

The results strongly suggested the clear advantage of using a NB vectorisation over TF-IDF vectorisation. All of the selected models used had better accuracy on average with this form of the vectorised data and took substantially less time to vectorise the data as well as train the models. In selecting the best models which balanced accuracy and computational efficiency, only the NB vectorised models were considered. This is because the most accurate models were all trained using this vectorisation and this vectorisation occurred in nominal time.

The decision tree forest, k-NN and SVM models all performed similarly across accuracy metrics as seen in Tables 6.1, 6.2 and 6.3. They all predicted with high overall accuracies of over 90% and fared the best in predicting the low prevalence ‘accident’ class. These models were also all trained within similar times. The mean total training times in minutes for these models were all below 2.5 minutes with each successive training cycle negligibly increasing the training times in absolute terms. This study motivates the usage of NB vectorisation for text classification coupled with SVM, decision tree forest or k-NN models because of their low total training times and accuracy levels.

## 8. Further Areas of Research

Different dimension reduction techniques should be investigated regarding TF-IDF vectorisation. This may improve the accuracy of the models as well as vectorisation times making TF-IDF more competitive. While the models were evaluated across time through 15 training-validation cycles, time was not explicitly modelled in the prediction procedure. Research which explicitly accounts for time may be useful. Relating to the previous point, new vocabulary is introduced into the models with time. Some models may have their prediction accuracies affected more severely than others with the introduction of new vocabulary. This may be prudent to research as some text data may have this as a dominant effect over time. Finally, a different training procedure should be explored besides a cumulative time slice approach. A fixed sliding training window of 3 months worth of data is another possibility that could be explored.

## 9. Appendix

The following tables expand on the hyperparameters for the best models found in Chapter 6. The order of the tables below correspond to the order and rank of Tables 6.1, 6.2 and 6.3. Where NB is used as a classifier only, the predicted class is the class with the greatest posterior predicted probability. Various neighbourhood sizes $k$ were used for k-NN. Decision tree forests' "Mtry" is the size of the random subset of features to consider at each split. Forests with Mtry $= p$ are the bagged forests. Where $p$ is the number of features. Other options explored included Mtry as $\sqrt{p}$ or $\frac{p}{3}$. These are the suggested defaults as per the `ranger` package. The split rules used were Gini and Extratrees. The number of trees grown was indicated by forest size. SVM's used radial or linear kernel functions for NB and TF-IDF vectorisation respectively as outlined in section 5.5.3. Cost's used ranged from 0.5 - 1.5 and included $\frac{1}{p}$ as suggested by the `e1071` package default.

The code for this research is available on request from the authors.

Table 9.1: Parameters for best Overall Accuracy models

| Vectorisation | Model | Hyperparameters |
|---|---|---|
| NB | Decision Tree Forest | Mtry =27;Splitrule = ExtraTrees;Forest Size = 1000 |
| NB | k-NN | Nearest Neighbours = 10 |
| NB | SVM | Radial Kernel;Gamma = 0.05;Cost = 1 |
| NB | Naive Bayes | Max Probability Decision Rule |
| TF-IDF | SVM | Linear Kernel; Cost = 0.5 |
| TF-IDF | Decision Tree Forest | Mtry = p/3; Splitrule = ExtraTrees;Forest Size = 250 |
| TF-IDF | k-NN | Nearest Neighbours = 5 |
| TF-IDF | Naive Bayes | Max Probability Decision Rule |

Table 9.2: Parameters for best Cohen's Kappa models

| Vectorisation | Model | Hyperparameters |
|---|---|---|
| NB | Decision Tree Forest | Mtry =27;Splitrule = ExtraTrees;Forest Size = 1000 |
| NB | k-NN | Nearest Neighbours = 10 |
| NB | SVM | Radial Kernel;Gamma = 0.05;Cost = 1 |
| NB | Naive Bayes | Max Probability Decision Rule |
| TF-IDF | SVM | Linear Kernel; Cost = 0.5 |
| TF-IDF | Decision Tree Forest | Mtry = p/3; Splitrule = ExtraTrees;Forest Size = 250 |
| TF-IDF | k-NN | Nearest Neighbours = 5 |
| TF-IDF | Naive Bayes | Max Probability Decision Rule |

Table 9.3: Parameters for best 'Accident' class sensitivity models

| Vectorisation | Model | Hyperparameters |
|---|---|---|
| NB | Decision Tree Forest | Mtry =27;Splitrule = ExtraTrees;Forest Size = 100 |
| NB | k-NN | Nearest Neighbours = 1 |
| NB | SVM | Radial Kernel;Gamma = 0.05;Cost = 1 |
| NB | Naive Bayes | Max Probability Decision Rule |
| TF-IDF | SVM | Linear Kernel; Cost = 0.5 |
| TF-IDF | SVM | Linear Kernel; Cost = 1.25 |
| TF-IDF | SVM | Linear Kernel; Cost = 1.5 |
| TF-IDF | Decision Tree Forest | Mtry = p/3 ;Splitrule = Gini;Forest Size = 250 |
| TF-IDF | k-NN | Nearest Neighbours = 1s |
| TF-IDF | Naive Bayes | Max Probability Decision Rule |

## 10. References

Aldous, David J. 1985. "Exchangeability and Related Topics." In *École d'Été de Probabilités de Saint-Flour Xiii — 1983*, edited by P. L. Hennequin, 1–198. Berlin, Heidelberg: Springer Berlin Heidelberg.

Basu, A., C. Walters, and M. Shepherd. 2003. "Support Vector Machines for Text Categorization." In *36th Annual Hawaii International Conference on System Sciences, 2003. Proceedings of the*, 7 pp. doi:10.1109/HICSS.2003.1174243.

Beel, Joeran, Bela Gipp, Stefan Langer, and Corinna Breitinger. 2016. "Research-Paper Recommender Systems: A Literature Survey." *International Journal on Digital Libraries* 17 (4): 305–38. doi:10.1007/s00799-015-0156-0.

Bennett, Paul N., Susan T. Dumais, and Eric Horvitz. 2002. "Probabilistic Combination of Text Classifiers Using Reliability Indicators: Models and Results." In *Proceedings of the 25th Annual International Acm Sigir Conference on Research and Development in Information Retrieval*, 207–14. SIGIR '02. New York, NY, USA: ACM. doi:10.1145/564376.564413.

Blei, David M., Andrew Y. Ng, and Michael I. Jordan. 2003. "Latent Dirichlet Allocation." *Journal of Machine Learning Research* 3: 993–1022.

Breiman, Leo. 1996. "Bagging Predictors." *Machine Learning* 24 (2). Springer: 123–40.

———. 1997. "Arcing the Edge." Technical Report 486, Statistics Department, University of California at Berkeley.

———. 2001. "Random Forests." *Machine Learning* 45 (1): 5–32. doi:10.1023/A:1010933404324.

Chen, Chao, Andy Liaw, and Leo Breiman. 2004. "Using Random Forest to Learn Imbalanced Data." *University of California, Berkeley* 110: 1–12.

Cortes, Corinna, and Vladimir Vapnik. 1995. "Support-Vector Networks." *Machine Learning* 20 (3): 273–97. doi:10.1023/A:1022627411411.

Cover, T., and P. Hart. 2006. "Nearest Neighbor Pattern Classification." *IEEE Trans. Inf. Theor.* 13 (1). Piscataway, NJ, USA: IEEE Press: 21–27. doi:10.1109/TIT.1967.1053964.

Fragos, Kostas, Petros Belsis, and Christos Skourlas. 2014. "Combining Probabilistic Classifiers for Text Classification." *Procedia - Social and Behavioral Sciences* 147: 307–12.

doi:https://doi.org/10.1016/j.sbspro.2014.07.098.

Fragos, Kostas, Yannis Maistros, and Christos Skourlas. 2005. "A Weighted Maximum Entropy Language Model for Text Classification." In *NLUCS*.

Geurts, Pierre, Damien Ernst, and Louis Wehenkel. 2006. "Extremely Randomized Trees." *Machine Learning* 63 (1): 3–42. doi:10.1007/s10994-006-6226-1.

Hastie, T., and R. Tibshirani. 2013. In *An Introduction to Statistical Learning*, 337–60.

Isa, Dino, Lam Hong Lee, V. P. Kallimani, and Rajprasad Rajkumar. 2008. "Text Document Preprocessing with the Bayes Formula for Classification Using the Support Vector Machine." *IEEE Transactions on Knowledge and Data Engineering* 20: 1264–72.

Jain, Aaditya, and Jyoti Mandowara. 2016. "Text Classification by Combining Text Classifiers to Improve the Efficiency of Classification." In.

Jed Wing, Max Kuhn. Contributions from, Steve Weston, Andre Williams, Chris Keefer, Allan Engelhardt, Tony Cooper, Zachary Mayer, et al. 2018. *Caret: Classification and Regression Training*. https://CRAN.R-project.org/package=caret.

Jeni, Laszlo A., Jeffrey F. Cohn, and Fernando De La Torre. 2013. "Facing Imbalanced Data–Recommendations for the Use of Performance Metrics." doi:10.1109/ACII.2013.47.

Joulin, Armand, Edouard Grave, Piotr Bojanowski, and Tomas Mikolov. 2016. "Bag of Tricks for Efficient Text Classification." *CoRR* abs/1607.01759. http://arxiv.org/abs/1607.01759.

Jurafsky, Dan, and James H Martin. 2009. "Speech and Language Processing: An Introduction to Natural Language Processing, Computational Linguistics, and Speech Recognition." *Prentice Hall Series in Artificial Intelligence*. Prentice Hall, Pearson Education International.

Lai, Siwei, Liheng Xu, Kang Liu, and Jun Zhao. 2015. "Recurrent Convolutional Neural Networks for Text Classification." In *AAAI*, 333:2267–73.

Liaw, Andy, and Matthew Wiener. 2001. "Classification and Regression by Randomforest" 23 (November).

Manin, Dmitrii Y. 2008. "Zipf's Law and Avoidance of Excessive Synonymy." *Cognitive Science* 32 (7). Wiley Online Library: 1075–98.

Manning, and Schutze. 1999. *Foundations of Statistical Natural Language Processing*. MIT press.

McCallum, Andrew, and Kamal Nigam. 1998. "A Comparison of Event Models for Naive Bayes Text

Classification." In *ACL*.

McHugh, Mary L. 2012. "Interrater Reliability: The Kappa Statistic." *Biochemia Medica* 22 (3): 276–82.

Meyer, David, Evgenia Dimitriadou, Kurt Hornik, Andreas Weingessel, and Friedrich Leisch. 2018. *E1071: Misc Functions of the Department of Statistics, Probability Theory Group (Formerly: E1071), Tu Wien*. https://CRAN.R-project.org/package=e1071.

Miao, Duoqian, Qiguo Duan, Hongyun Zhang, and Na Jiao. 2009. "Rough Set Based Hybrid Algorithm for Text Classification." *Expert Systems with Applications* 36 (5): 9168–74. doi:https://doi.org/10.1016/j.eswa.2008.12.026.

Purver, Matthew, Thomas L. Griffiths, Konrad P. Körding, and Joshua B. Tenenbaum. 2006. "Unsupervised Topic Modelling for Multi-Party Spoken Discourse." In *Proceedings of the 21st International Conference on Computational Linguistics and the 44th Annual Meeting of the Association for Computational Linguistics*, 17–24. ACL-44. Stroudsburg, PA, USA: Association for Computational Linguistics. doi:10.3115/1220175.1220178.

Salton, Gerard, and Christopher Buckley. 1988. "Term-Weighting Approaches in Automatic Text Retrieval." *Information Processing & Management* 24 (5). Elsevier: 513–23.

Sedding, Julian, and Dimitar Kazakov. 2004. "WordNet-Based Text Document Clustering." In *Proceedings of the 3rd Workshop on Robust Methods in Analysis of Natural Language Data*, 104–13. ROMAND '04. Stroudsburg, PA, USA: Association for Computational Linguistics. http://dl.acm.org/citation.cfm?id=1621445.1621458.

Silge, Julia, and David Robinson. 2016. "Tidytext: Text Mining and Analysis Using Tidy Data Principles in R." *JOSS* 1 (3). The Open Journal. doi:10.21105/joss.00037.

Venables, W. N., and B. D. Ripley. 2002. *Modern Applied Statistics with S*. Fourth. New York: Springer. http://www.stats.ox.ac.uk/pub/MASS4.

Wang, Sida I., and Christopher D. Manning. 2012. "Baselines and Bigrams: Simple, Good Sentiment and Topic Classification." In *ACL*.

Weinberger, Kilian Q, John Blitzer, and Lawrence K Saul. 2006. "Distance Metric Learning for Large Margin Nearest Neighbor Classification." In *Advances in Neural Information Processing Systems*, 1473–80.

Wright, Marvin N., and Andreas Ziegler. 2017. "ranger: A Fast Implementation of Random Forests for High Dimensional Data in C++ and R." *Journal of Statistical Software* 77 (1): 1–17.

doi:10.18637/jss.v077.i01.

Xu, Baoxun, Xiufeng Guo, Yunming Ye, and Jiefeng Cheng. 2012. "An Improved Random Forest Classifier for Text Categorization" 7 (December).

Zelikman, Eric. 2018. "Context Is Everything: Finding Meaning Statistically in Semantic Spaces." *CoRR* abs/1803.08493. http://arxiv.org/abs/1803.08493.

Zhang, Xiang, Junbo Jake Zhao, and Yann LeCun. 2015. "Character-Level Convolutional Networks for Text Classification." *CoRR* abs/1509.01626. http://arxiv.org/abs/1509.01626.