Data Science Project

Final Group Report

GROUP #5 – TOPIC: "CRUDE"

H.B. Arslangedikli (2713453)

G.J. Bouwens (2701442)

X.Y. Huang (2690243)

N.M. Soekhlal (2747471)

June 30, 2022

E_EOR2_DSPT

Bachelor Econonometrics & Operations Research

Contents

Abstract

The increasing availability and volume of (digital) financial articles in the last decade has stimulated the use of machine learning (ML) algorithms to automate the classification of said documents. In this paper, we aim to classify documents of the Reuter-21578 data set as either having 'crude' or not as their topic. To this end, we remove documents with no label or content, we transform the remaining articles to remove stop words and additional irrelevant terms, to unify strongly related words that provide equivalent content, through stemming, and to obtain a TF-IDF matrix. We derive and keep 40 features from this data representation via Latent Semantic Analysis. We use logistic regression as a first classifier, as well as other conventional methods, namely k-NN and naive Bayes (NB), which we expect to perform reasonably well overall due to their proven track record in similar applications. Finally, we combine a set of NB classifiers via the AdaBoost algorithm, which we expect to outperform the single NB classifier.

Keywords: *Document classification, Logistic regression, k-Nearest Neighbors, Boosting*

1 Introduction

Financial news is a critical source of information for profitable trading on financial markets. In order to respond quickly to a news article, it is highly useful to have an algorithm that can tell us whether the article in question concerns the topic of interest, which, in our case will be 'crude'. Globally, crude oil is one of the most important fuel sources for both corporates and private citizens. It is thus crucial to recognise news articles that provide relevant information for the price of this oil type. The research question that we study is hence: "How can financial news articles be automatically classified appropriately as belonging to the topic crude or not based on the documents' body and title?"

We first extract the documents of interest from the used data set, Reuters-21578, Distribution 1.0, by selecting only those articles that have at least one topic and at least a title or body. Said three attributes are also the only ones we keep, as the remaining elements serve no purpose in our analysis. Subsequently, we merge the title and body per article and we clean the resulting texts by removing terms that appear in well-established lists of stop words and additional terms we manually classified as such. We use the WordNet database to detect other words that lack meaning for our purpose and which we can hence leave out. In addition, we apply stemming to rid the texts of unnecessary repetitions of the same base form of a word, with different derivational affixes. Next, we obtain a TF-IDF matrix, which we split into a train and test set, using stratified sampling. To reduce the column dimension (the number of terms) of said matrix, we perform Latent Semantic Analysis (LSA), an alternative to Principal Component Analysis (PCA), which supports applications to (sparse) TF-IDF matrices. From the resulting 200 components, we selected the 40 that explain more than average variation in the original data as features. We start our modelling phase with logistic regression, using a fine tuned threshold to turn the model's fitted probabilities into binary predictions. Subsequently, we adopt the conventional text classifier k-NN, with an optimised number of nearest neighbours, as well as the naive Bayes (NB) classifier. Finally, we

leverage the power of multiple NB classifiers in the form of boosting. Section 1 discusses the current literature on said classifiers as well as more advanced alternatives. Section 2 provides a formal description of the classification task at hand, section 3 treats the original data set and the way in which we transformed it to a representation that can be handled by our classifiers, whereas section 4 reveals our approach to dimension reduction of the TF-IDF matrix, section 5 describes the selected classification algorithms in detail, section 7 reflects on our methodology and results, providing suggestions for future improvements, and finally, section 8 lists the software that we used to implement our methods and techniques.

Related Literature

The literature provides a wide variety of classification algorithms that can be applied to the Reuters data set. Kim et al. (2000) employ boosting methods based on NB classifiers to classify the articles of Associated Press (AP) News and Financial Times (FT) from 1988-1990, respectively 1992-1994. They conclude that said methods yield substantial improvements in the considered metrics compared to previously existing boosting algorithms, which typically used binary features. Moldagulova et al. (2017) use k-NN with the Euclidean distance metric to classify text documents. Indra et al. (2016) turn to the logit model for the classification of tweets. A more recently developed method is the unified C-LSTM model as proposed by Zhou et al. (2015), which aims to combine the advantages of both Convolutional Neural Networks (CNNs) and Recurrent Neural Networks (RNNs). The model has been used in Sentiment Classification and Question Type Classification, where it led to impressive results. In this paper, we also investigate the relative performance of (boosted) NB classifiers, k-NN and logit. However, we do so by making use of the Reuters data set. The C-LSTM method is omitted due to computational complexity.

Many research papers have already used the Reuters data set to study the performance of machine learning algorithms for document classification. Apte et al. (1998) investigate decision trees (DTs) for classifying a variant of the Reuters-21578 collection and they find substantial improvement in the performance when combining decision trees adaptive resampling. Joachims (1998) opts for Support Vector Machine (SVMs) to classify articles in the Reuters data set. He concludes that SVMs achieve considerable improvements over conventional methods such as k-NN classifiers. However, DTs and SVMs fall outside the scope of this paper.

2 Classification Task

The mathematical notation used from this section onward is for a major part in accordance with that of Sebastiani (2002). The document classification task can be described with the following five stages:

1. Document Classification:

Let d_j denote the j-th document and assign $\{0, 1\}$ to each $d_j \in \mathcal{D}$, where $\mathcal{D}$ denotes the set of all documents. Let c indicate the topic of interest and $\bar{c}$ its complement. Our goal is to build a

classifier $\Phi(d_j, c) \in \{0, 1\}$ that mimics $\hat{\Phi}(d_j, c)$, the "expert classifier", that is, we wish to mimic the decision-making procedure of the experts that labeled the documents to begin with, due to the supervised nature of our case.

2. Training, Testing & Validation:

Let the initial corpus, the set of all relevant documents with which we start the procedure, be defined as $\Omega = \{d_1, ..., d_{|\Omega|}\} \in \mathcal{D}$, where $|\Omega|$ is the number of documents in the corpus. Define a training and validation set as TV $= \{d_1, ..., d_{|TV|}\}$, as well as a testing set, namely Te $= \{D_{|TV|+1}, ..., d_{|\Omega|}\}$ (testing). TV splits into a training set Tr $= \{d_1, \ldots, d_{|Tr|}\}$ and a validation set Va $= \{d_{|Tr|+1}, .., d_{|TV|}\}$. Note that in practice, the training, validation and testing observations should be (pseudo) randomly drawn from the original data set and the second split can be replaced with a cross-validation on TV, as introduced in subsection 5.1.

3. Document Indexing:

In this stage, each document d_j is represented by a vector of weights, that is, $\overrightarrow{d_j} =< w_{1j}, ..., w_{|\tau|j} >$, where $|\tau|$ is the total number of words/terms appearing at least once in at least one document of the corpus. In our case, said vectors are such that they form the so-called term-frequency times inverse document-frequency (TF-IDF) matrix when lined up together. The purpose of using TF-IDF rather than the tokens' raw frequencies of occurrence in a given document is to scale down the impact of tokens that occur very frequently in a given corpus and that are thus less informative than terms that occur in a relatively limited proportion of the training corpus. The exact manner in which TF-IDF realises this is explained in subsection 3.6.

4. Dimensionality Reduction (DR):

Since $|\tau|$ is often too large for many classification algorithms to properly handle, we wish to (drastically) reduce the dimension of our data set to $|\tau'| << |\tau|$, effectively reducing the risk of overfitting. We can achieve said dimensionality reduction using a variety of methods, ranging from term selection and term extraction to PCA. The features that we are left with at the end of this stage can be denoted by $\overrightarrow{\omega_j} =< \omega_{1j}, \omega_{2j}, ..., \omega_{|\tau'|j} >$ for document j.

5. Machine Learning Classification:

A machine learning classifier can be seen as a mathematical function, denoted by CSV (categorisation status value), that maps from the vector space of all documents to numbers between 0 and 1. More formally, CSV: $\mathcal{D} \rightarrow [0, 1]$ in case of soft classification and CSV: $\mathcal{D} \rightarrow \{0, 1\}$ for hard classification. Note that any soft CSV can be converted to a hard CSV by thresholding, e.g. if CSV$(d_j) \geq \psi$, let CSV$^* = 1$ and if CSV$(d_j) < \psi$, let CSV$^* = 0$, for $\psi \in (0, 1)$. In our case, this conversion is necessary as we wish to determine whether document j has crude as one of its topics, meaning that we are interested in predicting cr_j, which takes on the value 1 if document j has crude as its topic and 0 otherwise. A wide range of classifiers is at our disposal, but our research is limited to the ones mentioned in section 1.

3 Data set & Pre-Processing Steps

3.1 Data Extraction

The used Reuters data set is based on 22 Standard Generalized Markup (SGM) files, which contain a range of details on 21578 financial news articles in total. However, for our research, the only elements of interest are the title, body and topics of the articles, meaning that these pieces are pilled out of the SGM files for each article, whereas other components, such as dates, datelines and places are omitted. Moreover, we require each article to have at least one topic and to have either a body, a title or both, for otherwise we lack the true label and/or the information with which we aim to predict the label. After filtering the documents based on said requirement, we are left with 11305 articles. Each article's list of topics is converted to a binary value, based on whether the topic of interest was present in the list. It turns out that only 627 of the remaining articles belong to our class of interest, namely the class of documents having crude as their topic. Thus, we are dealing with a single class imbalanced classification task, which must be taken into account when making decisions in the following stages.

3.2 Tokenization

The body and title of each article are merged to obtain one text per article, with no distinction between parts belonging to the body and title. To represent the text in a form that can be processed by classification algorithms, we make use of the so-called bag-of-words model, which is the de-facto standard approach in natural language processing. Within said model, we consider the text of each article to be a collection ("bag") of words, disregarding grammar and word order, but keeping track of how often each word appears in the text. Since a word is simply a sequence of characters in-between two white spaces/punctuation marks, we extract the words from the text by splitting each text in a set of elements with white space as the separator and by subsequently removing the punctuation marks from said elements. This process is known as tokenization.

3.3 Data Cleansing

At this point, the bag-of-words representations of the articles still contain terms that are completely or partially made up of digits, even though we do not deem digits valuable for predicting whether a given article has crude as its topic. We thus remove the (parts of) elements that are digits as well as the characters that connect digits to other parts of terms, e.g. the character '-'. Since capitalization is of no concern for our purpose, we convert our texts to lowercase. Subsequently, we remove all occurrences of manually selected and several predefined lists (Bird et al., 2009, Weischedel, Ralph, et al., 2013) of so-called stop words, words that are irrelevant for text classification, but whose presence in texts is imposed by the semantics of the English language. To filter out the remaining words with little to no relevance, we remove the ones that do not appear in the WordNet database (Princeton University "About WordNet.", 2010), which are thus assumed to not be meaningful.

3.4 Stemming

Subsequently, we cut off the ends of the remaining words to reduce inflectional forms and derivationally related forms of a word to a common base form, a process known as stemming. More specifically, we opt for the Porter stemmer as introduced by Porter (1980), because it outperforms the most promising alternative, the Lancaster stemmer, in Razmi et al. (2021) and it is deemed a reliable option for our application area, financial text mining, by Inzalkar and Sharma (2015).

3.5 Descriptive Statistics

To acquire insights into the distribution of the number of words in the articles and the (key)words that appear frequently in the articles, we provide a couple of descriptive statistics.

Table 1: Descriptive statistics for the number of words in the articles, including the titles (after removing stop words and stemming).

	Min	Max	Mean	Standard Deviation	Skewness	Kurtosis
#Words	7	2173	194.418	208.222	2.482	7.859

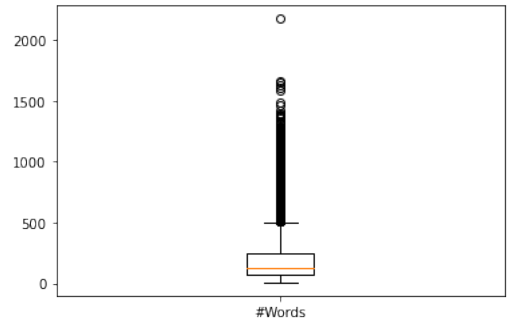

Figure 1: Boxplot of the number of words in the articles, including the titles (after removing stop words and stemming).

Table 1 reveals that there is quite a large spread in the number of words appearing in the documents, as the shortest one only has 7 words, whereas the longest article has 2173 words. The mean is much closer to the former and the distribution of the number of words appears to have a relatively heavy right tail, given the skewness of 2.482. The kurtosis of 7.859 suggests that the tails of said distribution are heavier than those of the normal distribution, which has a kurtosis of 3. Figure 1 illustrates the aforementioned right-skewed behaviour well, as there seem to be a reasonable number of outliers on the right side of the interquartile range, whereas no outliers below the first quartile are present. Given said observations, it is no surprise that the mean is substantially larger than the median, the latter of which turns out to be 127.
Figure 2 in Appendix A gives an indication of the most frequently occurring keywords in the financial news articles and most of them are indeed strongly associated with the financial markets,

the banking system and corporate life. Figure 3, on the other hand, is not limited to key words and hence also largely displays certain words, such as 'said' that are less associated in the financial world, but simply very common in texts in general.

3.6 TF-IDF

Finally, we convert the bags of words to the aforementioned TF-IDF matrix, where variables containing up to three original words are permitted, as long as the resulting terms occur no less than 5 times in total and in less than 80% of the articles. The latter restrictions are imposed to prevent classification based on terms that are too common to provide an appropriate basis for discrimination of crude and non-crude documents. This yields 28843 terms (columns) in total. The formula used for the conversion of a term t of a document d_j is tf-idf(t, d_j) = tf(t, d_j) $\cdot$ idf(t), where the tf component is set to 1 and the idf component is calculated as idf(t) $= log[(1+n)/(1+df(t))]+1$, where n denotes the total number of documents and df(t) is the document frequency of t, which is the number of documents that contain the term t. The reason for adding "1" to the idf component is to ensure that terms with zero idf (terms occurring in all documents) are not ignored completely. The constant '1' is added to the numerator and denominator of the idf component, meaning that we essentially act as if we observed an additional document containing every term that is present in the data set exactly once, preventing zero divisions. Finally, the rows of the TF-IDF matrix are normalised such that the Euclidean norm of each vector is 1.

3.7 Sample Article

To illustrate the challenges that we faced throughout the pre-processing steps, we provide the raw body (as given in the original SGM file) of an article named *"SAUDI RIYAL DEPOSIT RATES REMAIN FIRM"*, which has (only) the topic crude, below:

"Saudi riyal interbank deposits were steady at yesterday's higher levels in a quiet market. Traders said they were reluctant to take out new positions amidst uncertainty over whether OPEC will succeed in halting the current decline in oil prices. Oil industry sources said yesterday several Gulf Arab producers had had difficulty selling oil at official OPEC prices but Kuwait has said there are no plans for an emergency meeting of the 13-member organisation. A traditional Sunday lull in trading due to the European weekend also contributed to the lack of market activity. Spot-next and one-week rates were put at 6-1/4, 5-3/4 pct after quotes ranging between seven, six yesterday. One, three, and six-month deposits were quoted unchanged at 6-5/8, 3/8, 7-1/8, 6-7/8 and 7-3/8, 1/8 pct respectively. The spot riyal was quietly firmer at 3.7495/98 to the dollar after quotes of 3.7500/03 yesterday. REUTER &#3;"

Aside from the Reuter signature with which each article ends, the sample has many irrelevant terms that we can expect in text data in general, including numbers written out and terms containing a combination of digits and the characters '-' and '/'. The term '13-member' is an example of the latter which even contains a word. Said types of unwanted terms are removed as described in

subsection 3.3, as our manually chosen stop words include numbers written out. In addition, the article contains the words 'quiet' and 'quietly', which have the same stem, meaning that they are reduced to the same base form by the Porter stemmer. Finally, this example illustrates an important complication of text mining with the abbreviation 'prc' of 'percent'; different texts can use various conventions regarding for instance abbreviations, which is why it is important to have a solid notion of the types of words present in the considered texts.

4 Feature Engineering

Before applying any method to construct features, we create a train set of 80% and a test set of 20% of the original samples. To ensure sufficient representation of our topic of interest (the minority class) in each subset, we split the full data set in a stratified fashion, sampling proportionally from the articles with and without crude as their topic for each desired subset. As a result of said split, the TF-IDF matrix for the train data contains 9044 rows (articles), while the matrix for the test data encompasses the remaining 2261.

In many cases, PCA could be an appropriate way to construct a limited number of features out of this large number of initial variables. However, due to the sparsity of the TF-IDF matrix, we opt for an alternative method that can deal with such data matrices efficiently, namely Truncated SVD (Halko et al., 2009). This is due to the fact that said compression technique carries out linear dimensionality reduction via truncated singular value decomposition where the data is not centered prior to the calculation of the singular value decomposition, as opposed to PCA. When applied to TF-IDF matrices, Truncated SVD is known as Latent Semantic Analysis (LSA). We employ LSA by computing 200 components (potential features), for each of the ten folds resulting from a 10-fold stratified cross-validation on the train set, where the number of folds was chosen in accordance with the cross-validation used to tune hyperparameters of the classification algorithms (see section 5). Subsequently, we set the number of components to keep from the full train set equal to the rounded average of the number of components that explain an above average portion of the variance in each of the ten folds. This procedure leaves us with 40 components (features) for the modelling phase. Figure 4 in Appendix A displays the correlations between these components, which reveals that the correlation between the first and third is strongly negative (close to -1), whereas the other correlations are mostly close to 0, with a few positive and negative outliers. To prevent data leakage (and a potential mismatch between the features and model parameters), the aforementioned number of components is treated as an estimated parameter and thus kept fixed when performing LSA on the test set to obtain the same number of features to predict the test labels with.

5 Classification Algorithms

5.1 Validation

To tune the hyperparameters of each of the following models, we use K-fold stratified cross-validation, where K is set to 10, as proposed by the naive rule of Jung (2018), based on the training set of 9044 documents and 40 features. This means that in each of 10 runs, we use 9 folds to train the model and we use the remaining one to evaluate its performance with the current value of the hyperparameter. The metric used to evaluate the performance of the cross-validated model on the validation set is the F1 score. This performance metric was proposed by Zou et al. (2016) for single class imbalanced classification problems, such as the one at hand.

5.2 Logistic Regression

As a first benchmark, we employ a logit model, due to ease of implementation, interpretability of the estimated parameters and the proven reliability of logistic regression in a wide variety of scenarios throughout the history of classification modelling, though the remarkable vulnerability of the model to large dimensional data can potentially lead to suboptimal results with the number of features that we use.

To classify documents with a linear parametric model, we could define a new binary variable y_j denoting whether document j has topic crude and model it as follows:

$$y_j = \beta_0 + \beta_1 \cdot \omega_{1j} + \beta_2 \cdot \omega_{2j} + ... + \beta_{|\tau'|} \cdot \omega_{|\tau'|j} + e_j.$$

The e_j can be assumed to have a logistic distribution and the parameters $\beta_0, ..., \beta_{|\tau'|}$ can be estimated as $\hat{\beta}_0, ..., \hat{\beta}_{|\tau'|}$ with Maximum Likelihood Estimation (MLE). The resulting predictions

$$\hat{y}_j = \hat{\beta}_0 + \hat{\beta}_1 \cdot \omega_{1j} + \hat{\beta}_2 \cdot \omega_{2j} + ... + \hat{\beta}_{|\tau'|} \cdot \omega_{|\tau'|j},$$

can be seen as (estimated) probabilities of the corresponding documents having topic crude. However, the $\hat{y}_j$ are not guaranteed to be sensible probabilities between 0 and 1. Thus, we instead define the fitted probabilities $\hat{p}_j$ as a logistic transformation of the y_j, i.e.

$$\hat{p}_j = \frac{1}{1 + e^{-\hat{y}_j}} \in (0, 1).$$

Finally, as logit is a soft classifier, it is necessary to set a threshold $\phi \in (0, 1)$ for the p_j to obtain the desired binary predictions, as introduced in section 2. This hyperparameter was optimised using the aforementioned cross-validation in a range of 50 values, ranging from 0.5 to 0.99, because a threshold below 0.5 in order to predict the minority class is deemed unconventional and intuitively unreasonable in scenarios where the severity of the class imbalance is comparable to ours.

5.3 K-nearest Neighbours

The non-parametric k-NN model is generally perceived as a rather simple text classifier that can nonetheless yield predictive performance similar to that of more advanced algorithms.

Its purpose is to take a prediction point, $\overrightarrow{\omega_0}$, which contains features of a new observation (document), and to compute a prediction, $\hat{f}(\overrightarrow{\omega_0})$, of the value of the dependent variable, cr_0, of this new observation using the K closest observations among our training set, where closeness is measured as a function of the observations' features. Given a value for the parameter K and a prediction point $\overrightarrow{\omega_0}$, KNN identifies the K training observations that are closest to $\overrightarrow{\omega_0}$, represented by $\mathcal{N}_0$. If the dependent variable is categorical, we speak of KNN classification and we simply take the category with highest frequency among the K nearest training observations as a prediction for the new point, i.e.

$$\hat{f}(\overrightarrow{\omega_0}) = \text{mode}_{cr_j}(\mathcal{N}_0). \tag{1}$$

Choosing the value of K implies a trade-off between the bias and the variance of the classifier, as an increase in K generally reduces the bias and simultaneously increases the variance of the k-NN predictions. Said hyperparameter is thus optimised in the range of integers between 1 and 50 via the cross-validation. The other hyperparameter that we tune is the distance measure with which the K nearest neighbours of prediction points are determined. The comparison study of Mulak and Talhar (2015) favours Manhattan distance over Euclidean distance and Chebychev distance. However, due to the difference in application, we include the distance measure as a tuning parameter in the cross-validation and consider said three measures to determine whether Manhattan distance is the best option in our case as well.

5.4 Naive Bayes

Another relatively simple yet popular benchmark in the literature is Naive Bayes (NB). Despite their typically incorrect assumptions, NB classifiers have proven their worth in a wide range of applications, in particular text classification. The conditional independence of the features that is assumed, ensures that we can estimate each one dimensional distribution separately, which reduces the negative effects of the Curse of Dimensionality. Zhang (2004) provides further theoretical justifications for the usefulness and reliability of NB classifiers.

The goal of NB is to compute the probability of c given the features of document j, $\overrightarrow{\omega_j}$. To this end, we apply Bayes' theorem as follows:

$$P(c|\overrightarrow{\omega_j}) = \frac{P(c) \cdot P(\overrightarrow{\omega_j}|c)}{P(\overrightarrow{\omega_j})}.$$

The problem is that we cannot compute $P(\overrightarrow{\omega_j}|c)$ and $P(\overrightarrow{\omega_j})$, as there are too many possible realisations of $\overrightarrow{\omega_j}$. Thus, we 'naively' assume that ω_{ij} is independent of ω_{kj} whenever $i \neq k$, that is, we assume that each feature of a document is independent from the other features of that same

document. Then,

$$P(\overrightarrow{\omega_j}|c) = \prod_{k=1}^{|\tau'|} P(\omega_{kj}|c),$$

where the terms of the product have a(n) assumed closed form expression, depending on the nature of the ω_{kj}. If for instance $\omega_{kj} \in 0, 1$ and if we let p_k denote $P(\omega_{kx} = 1|c)$, then

$$p(\omega_{kj}|c) = p_k^{w_{kj}} \cdot (1 - p_k)^{1-w_{kj}},$$

where p_k, the probability of term k, can be estimated directly from the data as the fraction of documents from the training set with the topic of interest, c, where term k is present. This approach corresponds to the binary independence classifier (Robertson and Sparck Jones, 1976). For generic ω_{kj}, since $P(\overrightarrow{\omega_j})$ is a constant given the features at our disposal, we have:

$$P(c|\overrightarrow{\omega_j}) \propto P(c) \cdot \prod_{k=1}^{|\tau'|} P(\omega_{kj}|c),$$

which means that we can obtain prediction, $\hat{c}$, of the document's label as follows:

$$\hat{c} = argmax_c P(c) \prod_{k=1}^{|\tau'|} P(\omega_{kj}|c),$$

where $P(c)$ and $P(\omega_{kj}|c)$ can be estimated by Maximum A Posteriori (MAP) estimation. $P(c)$ is then simply the fraction of documents with the topic of interest in the training set.

The primary difference between the various NB classifiers lies in the assumptions made concerning the distribution of $P(\omega_{kj}|c)$. In our case, the ω_{kj} are elements of the components resulting from the LSA, which lie between -1 and 1. Hence, we opt for the Gaussian Naive Bayes (GNB) classifier, which supports said domain of the features and imposes a Gaussian likelihood:

$$P(\omega_{kj}|c) = \frac{1}{\sqrt{2\pi\sigma_c^2}} e^{-\frac{(\omega_{kj}-\mu_c)^2}{2\sigma_c^2}},$$

where σ_c and μ_c are estimated using MLE.

Before training or testing the model, we apply the Yeo-Johnson power transformation (Yeo and Johnson, 2000) to the features in order to make them more Gaussian-like. The only hyperparameter we tune via cross-validation is the variance smoothing parameter, which is the fraction of the largest variance of all features that is added to all variances to improve the computational stability of the calculations. The grid of values we consider for the smoothing parameter contains 100 evenly spaced numbers on a log scale from 10^{-9} to 1.

We augment GNB by means of boosting, as this approach was rated favourably by Kim et al. (2000). In particular, we implement the AdaBoost algorithm introduced by Freund and Schapire (1995). Its fundamental idea is to fit a set of weak learners, in our case NB classifiers, on modified versions of the data. Subsequently, the predictions from the weak learners are combined via a

majority vote to yield the final predictions. The modified versions of the data set are obtained by assigning a weight, α_i, to each of the $|Tr|$ observations per iteration of the boosting algorithm. The weights are initialised as $1/|Tr|$, meaning that the first iteration trains the classifier on the original training set. However, for each following iteration, every weight is altered and the classifier is trained using the newly weighted samples. An important aspect of this procedure is that for each iteration, the observations that were incorrectly classified in the previous iteration receive a larger weight, whereas the correctly classified observations obtain a smaller weight. Thus, subsequent classifiers are forced to focus more on the observations that their predecessors could not correctly classify. The number of weak learners to combine is a hyperparameter, which we thus tune using our cross-validation and a grid of values containing 5, 10, 20 and 50.

6 Performance Evaluation

6.1 Metrics

The predictive performance of the considered models was evaluated using the test set of $|Te| = 2261$ samples. We use a variety of metrics to describe different aspects of each classifier's performance, most of which are derived from the components of the generic confusion matrix, which we present as Table 3 in Appendix B) and from which we use the notation in the remainder of this section.

The simplest performance measure is the accuracy, which we define as the fraction of all classified documents that are classified correctly, i.e.

$$Accuracy = \frac{TN + TP}{|Te|} \in [0, 1]. \tag{2}$$

The accuracy is an intuitive measure of a classifier's performance, but particularly in cases with severe data imbalance, such as the classification task at hand, the accuracy may not be sufficient, due to the fact that it can be dominated by correct classifications of the majority class (true negatives), such that even a classifier that always predicts this class, neglecting all information provided by the features, can achieve a high accuracy, whilst incorrectly classifying every observation belonging to the minority class. In such cases, we wish to additionally consider metrics that measure the extent to which a classifier can correctly assign the positive class to observations.

One such metric is the recall, which is defined as the fraction of samples belonging to the positive class that are correctly classified as such, i.e.

$$Recall = \frac{TP}{FN + TP} \in [0, 1]. \tag{3}$$

To arrive at the limitation of recall as a performance measure, we consider a classifier that labels each observation as being a member of the positive class. In that case, the recall of the classifier will be 1, as there are no (false) negatives. However, such a classifier is far from favourable given that most observations do not belong to the positive class. Recall can thus be seen as a measure

of quantity: it solely considers the fraction of true positives that were correctly classified, without taking into account how biased towards the positive class the classifier in question needs to be in order to obtain many true positives. This brings rise to the need for a metric that considers the quality of the positive classifications.

Precision realises this by measuring the number of correct positive classifications as a fraction of the total number of positive classifications, i.e.

$$Precision = \frac{TP}{FP + TP} \in [0, 1]. \tag{4}$$

Precision is on its turn a less appropriate measure for classifiers that are rather conservative when it comes to classifying observations as members of the positive class, as a classifier that only assigns a positive label to an observation when it is extremely confident will have relatively few positives, most of which will likely be correct classifications. This will yield a high precision as this metric only considers the observations that obtain a positive label, ignoring the fact that such a conservative classifier may fail to recognise many positive observations as such. Depending on the application and the needs of stakeholders, either recall or precision could be a more suitable metric, but in our case we deem a false crude label and a false non-crude label to be equally harmful.

Hence, we consider a metric that conveys a balance between precision and recall, namely the F1 score, which is defined as follows:

$$F1 = 2 \cdot \frac{Precision * Recall}{Precision + Recall} \in [0, 1]. \tag{5}$$

Note that the F1 score can only be 1 if both the precision and accuracy are 1, which is the ideal case. A value close to 0 indicates that either precision or recall is poor (or both).

In addition, we take into account the ROC AUC scores of the classifiers, which measure the Area Under the Receiver Operating Characteristic Curve (ROC AUC) from the predictions. An ROC curve is a plot which displays the performance of a classifier as the threshold that determines the way in which observations are discriminated is varied. The curve is generated by plotting the fraction of true positives out of all positives (true positive rate) against the fraction of false positives out of all negatives (false positive rate), at varying threshold values (Fawcett, 2006). The AUC measures the classifier's ability to discriminate between positive and negative observations, effectively summarising the ROC curve. As the AUC increases, the classifier becomes better able to distinguish between positive and negative samples.

6.2 Results

In addition to evaluating the performance of the aforementioned classifiers, we also consider a so-called dummy classifier, which always predicts the majority class, in our case the non-crude label. This classifier serves as a baseline for the classifiers that do make use of the derived features. The evaluation results are based on a test set of 2261 documents, 125 of which have crude as their

topic.

Table 2: Performance measures for each of the considered classifiers. (Hyper)parameter values that are of particular interest for the interpretation/explanation of the performances are provided between brackets for the corresponding classifiers.

	Accuracy	Recall	Precision	ROC AUC	F1
Dummy	0.945	0.000	0.000	0.500	0.000
Logit(threshold=0.73)	0.935	0.272	0.374	0.623	0.315
k-NN(metric=Euclidean, k=5)	0.949	0.112	0.737	0.555	0.194
GNB	0.932	0.000	0.000	0.493	0.000
Boosted GNB(n_estimators=5)	0.924	0.024	0.058	0.501	0.034

6.3 Discussion

Table 2 reveals the performance of the classifiers based on the introduced metrics. As expected, the dummy classifier reaches a rather high accuracy, namely 0.945, due to the fact that the consistently chosen majority class covers such a large proportion of the test set, whereas the recall, precision and hence F1 score are 0 since there are no positive classifications. The Dummy classifier's ROC AUC score of 0.500 can be taken as a suitable baseline for the other classifiers. The logit model yields slightly lower accuracy than the dummy classifier, namely 0.935, whereas the recall and precision are substantially larger (0.272 and 0.374, respectively), which is intuitive, given that the Logit model classifies a document as having topic crude when the fitted probability of this event exceeds the optimised threshold, which is typically low enough to yield at least some positive classifications for the observations for which the model is confident enough. In our case, the optimal threshold of 0.73 is much higher than the standard value of 0.5, which is sensible given the small proportion of the test observations having crude as their topic. The reasonable increase in ROC AUC score from 0.500 to 0.623 and the sizable increase in the F1 score from 0.000 to 0.315 also indicates that the logit model performs considerably better than the dummy classifier.

Regarding the k-NN classifier, we first note that the optimal distance measure turns out to be Euclidean distance, contrary to the study of Mulak and Talhar (2015), where Manhattan distance was preferred. This underlines the importance of the specific classification task at hand and the accompanying data set for the choice of hyperparameters. The optimal value of k was set to 5, which is rather moderate, perhaps due to the relatively large amount of features. The accuracy of the k-NN classifiers is higher than that of both the logit and dummy models. The recall is substantially lower than that of the logit model, whereas the precision is much larger. The difference in the recall values indicates that the logit model is preferable in terms of the number of crude documents it can 'detect', whereas a crude label from the k-NN model is more reliable due to the higher precision. A stakeholder who is specifically interested in one of said capabilities might thus prefer the corresponding classifier. We note that the relatively high precision of k-NN enables it to outperform the dummy classifier in terms of accuracy whilst rendering positive classifications as opposed to the latter, for the dummy classifier guarantees correct classification of all non-crude

documents, meaning that k-NN must be somewhat precise in its positive classifications in order to achieve a higher number of correct classifications overall. The balancing of recall and precision that the F1 reflects favours the combination of the former metrics obtained by the logit model as opposed to k-NN, because the latter attains a lower F1 score of 0.194. The ROC AUC score of the k-NN is also inferior, namely 0.555, but still higher than that of the dummy baseline.

The performance results of the GNB classifier are disappointing, as they are completely inferior to those of the dummy classifier: the recall, precision and F1 have remained 0, whereas the accuracy and ROC AUC of the GNB classifier are slightly lower (0.932 and 0.493, respectively). This implies that the GNB classifier only differs from the Dummy classifier in that it has a few false positives: some non-crude documents are wrongfully classified as having crude as their topic, which keeps the recall, precision and F1 score 0, as there are no true positives, whereas the accuracy decreases compared to the Dummy classifier due to said false positives.

The Boosted GNB classifier based on the optimised number of 5 estimators does manage to correctly classify a few crude documents, as the recall and precision are marginally above 0 (0.024 and 0.058, respectively), which yields a positive but relatively small F1 score of 0.034. This does, however, go hand in hand with a further decline in accuracy, leaving the Boosted GNB with the lowest accuracy of all considered models, namely 0.924. The ROC AUC increased minimally to 0.501 compared to the dummy model. These performance measures do not give us sufficient reason to deem the Boosted GNB classifier a substantial improvement from the dummy classifier, despite the larger (computational) complexity.

7 Conclusion

We started with a review of the relevant literature and proceeded to prepare our initial data set for the modelling phase. We selected the documents that have both a label and a predictor and omitted all article attributes that do not fall under said two categories. From said documents of interest we removed words that were not relevant for our purpose, after which we stemmed the remaining terms. The stemmed words in the bag-of-words model were transformed to a TF-IDF matrix. Train and test sets were created from said matrix and we extracted features from them by performing LSA.

We used the derived features to analyse logit, k-NN, GNB and Boosted GNB classifiers, where the hyperparameters of each model were optimised using 10-fold stratified cross-validation. The accuracy of each classifier is well above 0.9, as expected due to the fact that even a dummy classifier can obtain such a high accuracy score by simply classifying each document as having crude as its topic. The other metrics reveal the considerable differences between the classifiers in terms of predictive performance, based on which we formulate an answer to our research question, which is twofold. The Logit model, with the threshold parameter set to 0.73, turns out to have the highest recall and F1 score out of the considered classifiers. It is thus preferable in case detecting as many crude documents whilst maintaining a high accuracy and F1 score is desired. The k-NN classifier, however, yields the highest precision as well as the highest accuracy and the second

highest F1 score, meaning that it is the best option out of the analysed methods for a stakeholder who wishes to detect a number of crude documents, but who is particularly interested in being confident that the positively classified documents truly have crude as their topic. The GNB based classifiers turn out to be less appealing in this case: the standard GNB classifier is revealed to be inferior to the dummy classifier, whereas the Boosted GNB performs only marginally better on all metrics except accuracy. We would thus not favor these probabilistic methods for this particular application.

We realise that the pre-processing and feature engineering stages have an immense influence on the predictive performance that can be attained by the classifiers we subsequently opt for and we would thus suggest fellow researchers to investigate possibilities to further clean the data before extracting features, for instance by removing irrelevant words that appear in the word cloud, to try alternatives for the Porter stemmer, such as the Lancaster stemmer or lemmatizers, and to consider alternative methods for dimensionality reduction, such as term clustering and high correlation filters. Regarding classification algorithms, future research could expand on the boosting methods that we used and explore increasingly complicated algorithms that require additional computer power, such as the C-LSTM method discussed in section 1. The next step would be to leverage the strengths of different types of classifiers via ensemble methods.

8 Software

The implementation of the discussed methodology was done in the Python computer language. To navigate the raw data we used the BeautifulSoup library (Richardson, 2007), whereas data cleansing and stemming were performed with the Natural Language Toolkit (NLTK) (Bird et al., 2009). To implement the algorithms from the feature engineering and classification stages, as well as the involved data splitting and cross-validation, we used the 'scikit-learn' (Pedregosa et al., 2011) library. The 'Pandas' (The Python Development Team, 2021), 'Matplotlib' (Hunter, 2007) and 'NumPy' (Harris, Millman, van der Walt et al., 2020) libraries were used for data manipulation, visualisation and numerical computing, respectively.

A First Appendix

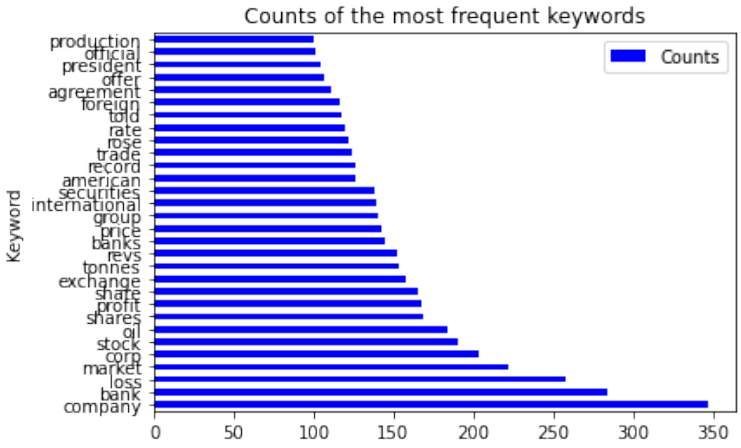

Figure 2: Counts of the most frequent keywords for a stratified random sample of 1000 articles from the full set of articles left after stemming.

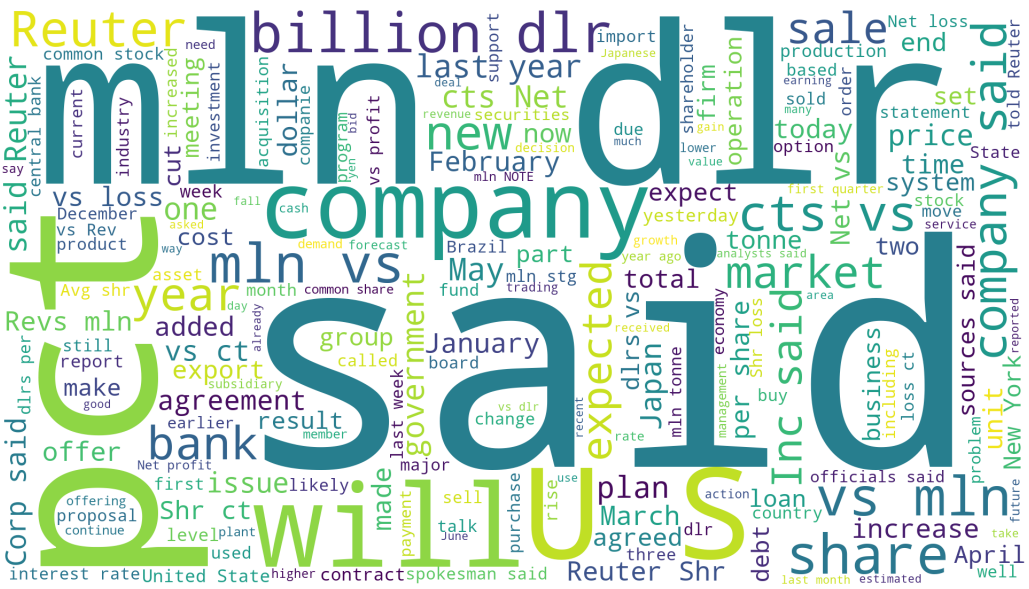

Figure 3: Word cloud corresponding to a stratified random sample of 1000 articles from the full set of articles before stemming.

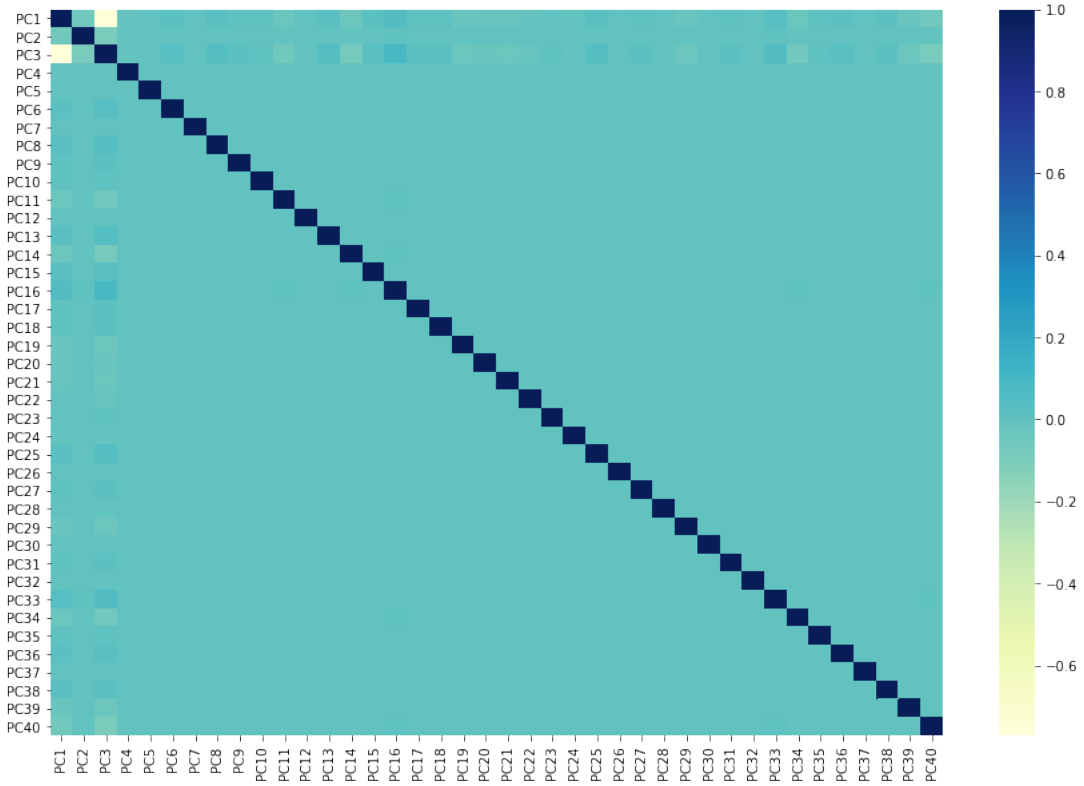

Figure 4: Heatmap of the correlations between the components resulting from the LSA.

B Second Appendix

Table 3: Generic confusion matrix.

		Predicted topic	
		Not Crude	**Crude**
True topic	**Not Crude**	#True Negatives (TN)	#False Positives (FP)
	Crude	#False Negatives (FN)	#True Positives (TP)

C References

Apte, C., Damerau, F., Weiss, S. M., Apte, C., Damerau, F., and Weiss, S. (1998). Text mining with decision trees and decision rules. In *Proceedings of the Conference on Automated Learning and Discovery, Workshop 6: Learning from Text and the Web.*

Joachims, T. (1998). Text categorization with support vector machines: Learning with many relevant features. In *European conference on machine learning*, pages 137–142. Springer.

Princeton University "About WordNet." WordNet. Princeton University. 2010.

Zou, Q., Xie, S., Lin, Z., Wu, M., Ju, Y. (2016). Finding the best classification threshold in imbalanced classification. Big Data Research, 5, 2-8.

Dua, D. and Graff, C. (2019). UCI Machine Learning Repository [http://archive.ics.uci.edu/ml]. Irvine, CA: University of California, School of Information and Computer Science.

D. Lewis. Reuters 21578 data set. URL=http://www.research.att.com/lewis/- reuters21578.html.

Sebastiani, Fabrizio. (2002). Machine Learning in Automated Text Categorization. ACM Computing Surveys. 34. 1-47. 10.1145/505282.505283.

Kim, Y.H., Hahn, S.Y., and Zhang, B.T. 2000. Text filtering by boosting naive Bayes classifiers. In *Proceedings of SIGIR-00, 23rd ACM International Conference on Research and Development in Information Retrieval* (Athens, Greece, 2000), 168–175.

Zhou, C., Sun, C., Liu, Z., Lau, F. (2015). A C-LSTM neural network for text classification. *arXiv preprint arXiv:1511.08630*.

Nedungadi, P., Harikumar, H., Ramesh, M. (2014, February). A high performance hybrid algorithm for text classification. In *The Fifth International Conference on the Applications of Digital Information and Web Technologies (ICADIWT 2014)* (pp. 118-123). IEEE.

Yoonsuh Jung (2018) Multiple predicting K-fold cross-validation for model selection, Journal of Nonparametric Statistics, 30:1, 197-215. https://doi.org/10.1080/10485252.2017.1404598

Moldagulova, A., Sulaiman, R. B. (2017, May). Using KNN algorithm for classification of textual documents. In 2017 8th international conference on information technology (ICIT) (pp. 665-671). IEEE.

Indra, S. T., Wikarsa, L., Turang, R. (2016, October). Using logistic regression method to classify tweets into the selected topics. In 2016 international conference on advanced computer science and information systems (icacsis) (pp. 385-390). IEEE.

Inzalkar, S., Sharma, J. (2015). A survey on text mining-techniques and application. International Journal of Research In Science Engineering, 24, 1-14.

Porter, M. "An algorithm for suffix stripping." Program 14.3 (1980): 130-137.

Razmi, N. A., Zamri, M. Z., Ghazalli, S. S. S., Seman, N. (2021). Visualizing stemming techniques on online news articles text analytics. Bulletin of Electrical Engineering and

Informatics, 10(1), 365-373.

Mulak, P., Talhar, N. (2015). Analysis of distance measures using k-nearest neighbor algorithm on kdd dataset. Int. J. Sci. Res, 4(7), 2319-7064.

Robertson, S. E. and Sparck Jones, K. 1976. Relevance weighting of search terms. J. Amer. Soc. Inform. Sci. 27, 3, 129–146. Also reprinted in Willett [1988], pp. 143–160.

H. Zhang (2004). The optimality of Naive Bayes. Proc. FLAIRS.

I.K. Yeo and R.A. Johnson, "A new family of power transformations to improve normality or symmetry." Biometrika, 87(4), pp.954-959, (2000).

Y. Freund, R. Schapire, "A Decision-Theoretic Generalization of on-Line Learning and an Application to Boosting", 1995.

Christopher D. Manning, Prabhakar Raghavan and Hinrich Schütze (2008), Introduction to Information Retrieval, Cambridge University Press, chapter 18: Matrix decompositions & latent semantic indexing

Fawcett, T., 2006. An introduction to ROC analysis. Pattern Recognition Letters, 27(8), pp. 861-874.

Steven Bird, Ewan Klein, and Edward Loper (2009). Natural Language Processing with Python. O'Reilly Media Inc. https://www.nltk.org/book/

Weischedel, Ralph, et al. OntoNotes Release 5.0 LDC2013T19. Web Download. Philadelphia: Linguistic Data Consortium, 2013.

Scikit-learn: Machine Learning in Python, Pedregosa et al., JMLR 12, pp. 2825-2830, 2011.

Seabold, Skipper, and Josef Perktold. "statsmodels: Econometric and statistical modeling with python." *Proceedings of the 9th Python in Science Conference.* 2010.

Reback, J., McKinney, W., jbrockmendel, den Bossche, J. V., Augspurger, T., Cloud, P., gfyoung, Hawkins, S., Sinhrks, Roeschke, M., Klein, A., Petersen, T., Tratner, J., She, C., Ayd, W., Naveh, S., Garcia, M., Schendel, J., patrick, Hayden, A., Saxton, D., Jancauskas, V., McMaster, A., Gorelli, M., Battiston, P., Seabold, S., Dong, K., chris-b1, h-vetinari, and Hoyer, S.: Pandas-Dev/Pandas: Pandas 1.2.2, Zenodo [code], https://doi.org/10.5281/zenodo.4524629, 2021.a

J. D. Hunter, "Matplotlib: A 2D Graphics Environment", Computing in Science Engineering, vol. 9, no. 3, pp. 90-95, 2007.

Harris, C.R., Millman, K.J., van der Walt, S.J. et al. *Array programming with NumPy.* Nature 585, 357–362 (2020). DOI: 10.1038/s41586-020-2649-2. (Publisher link).

Richardson, L. (2007). Beautiful soup documentation. April.

Bird, S., Klein, E., Loper, E. (2009). Natural language processing with Python: analyzing text with the natural language toolkit. " Ox27;Reilly Media, Inc."

Halko, et al. (2009). "Finding structure with randomness: Stochastic algorithms for constructing

approximate matrix decompositions”