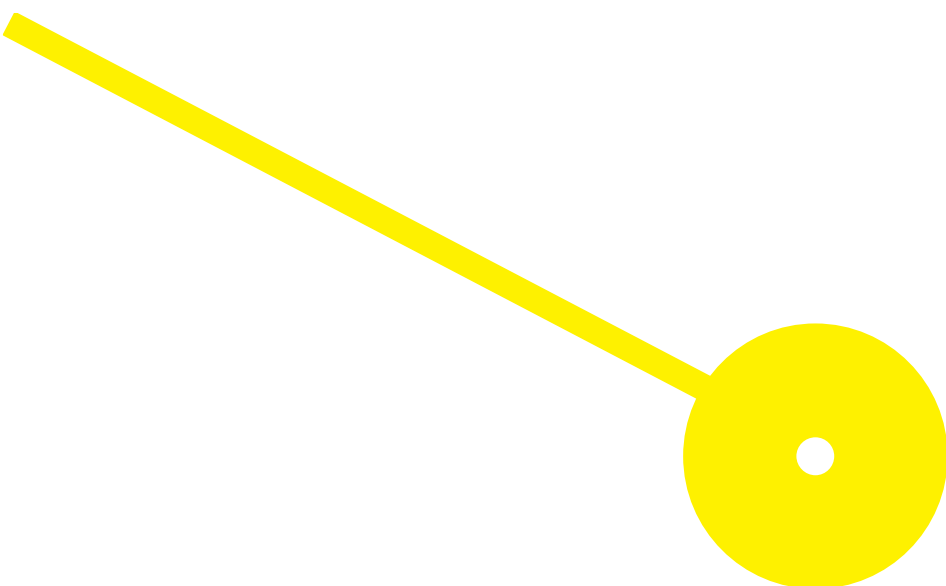




Employing Text Classification to Facilitate Economic and Food Safety Law Enforcing

Gustavo Magalhães

09/2019





**ESCOLA
SUPERIOR
DE SAÚDE**



Employing Text Classification to Facilitate Economic and Food Safety Law Enforcing

Autor

Gustavo Manuel Pinto de Magalhães

Orientador(es)

Professora Doutora/Brígida Mónica Faria/ Escola Superior de Saúde – Instituto Politécnico do Porto

Professor Doutor/Luís Paulo Reis/ Faculdade de Engenharia da Universidade do Porto

Dissertação apresentada para cumprimento dos requisitos necessários à obtenção do grau de Mestre em **Bioestatística e Bioinformática Aplicadas à Saúde** pela Escola Superior de Saúde do Instituto Politécnico do Porto.

Menção de apoio financeiro

Dissertação realizada com o apoio do projeto “IA.SAE – Inteligência Artificial na Segurança Alimentar e Económica”, financiado pelo FCT/MCTES através de fundos nacionais (PIDDAC), inserindo-se no Programa Iniciativa Nacional Competências Digitais e.2030 – INCoDe.2030, inscrito no Plano Nacional de Reformas.



Agradecimentos

Gostaria de agradecer aos meus orientadores, Professora Doutora Brígida Mónica Faria e Professor Doutor Luís Paulo Reis, pelo apoio e orientação ao longo desta jornada.

Ao Laboratório de Inteligência Artificial e Ciência de Computadores, na pessoa do Professor Doutor Luís Paulo Reis, pela oportunidade de investigação sem a qual não seria possível o presente trabalho.

À minha família pelo esforço e incentivo para completar mais um capítulo e em especial aos meus pais por todo o apoio e por serem a roda e a inspiração que tornou tudo isto possível.

A todos aqueles que me ajudaram ao longo desta caminhada cujos nomes não foram mencionados, mas não menos importantes.

Acknowledgements

I would like to thank my advisors, Professor Brígida Mónica Faria and Professor Luís Paulo Reis, for the support and guidance over the course of this journey.

To the Artificial Intelligence and Computer Science Laboratory, represented by Professor Luís Paulo Reis, for the investigation opportunity without which the current work would not be possible.

To my family for the support and incentive to complete another chapter and in particular to my parents for all the support and for being the wheel and inspiration and inspiration which made this possible.

To all those that have helped me through this journey whose names have not been mentioned but no less important.

Resumo

A categorização de textos é uma tarefa de aprendizagem supervisionada que visa atribuir rótulos a documentos com base no resultado previsto sugerido por um classificador treinado num conjunto de documentos rotulados. Com a crescente disponibilidade de informação textual que acompanha o crescimento cada vez maior da internet e dos dados disponíveis online, a capacidade de executar de forma automatizada tarefas demoradas para um ser humano, bem como a capacidade de encontrar padrões ou extrair informações valiosas dos dados, é incrivelmente ponderosa. Assim, a associação da classificação do texto para facilitar a rotulagem de relatórios e reclamações nos campos económicos e de saúde pode ter um impacto tremendo na velocidade com que estas são processadas e, portanto, diminuindo o tempo necessário para agir sobre estas reclamações e relatórios.

Neste trabalho, avaliamos o desempenho da capacidade de classificação de 9 algoritmos em diferentes níveis do fluxo metodológico de classificação textual: pré-processamento, normalização de palavras, extração de características, selecção de características, otimização de hiper-parâmetros e avaliação. Estes algoritmos são: Naïve Bayes Complementar, Naïve Bayes de Bernoulli, Naïve Bayes Multinomial, K-Vizinhos mais próximos, Árvores de Decisão, Florestas Aleatórias, Máquinas de Suporte Vectorial, AdaBoost e Regressão Logística. Os principais resultados revelam que foram atingidos níveis de taxa de acerto elevadas, na casa dos 67% e 85% em dois conjuntos de dados com alvos de rotulagem diferentes. Foi também observado que os classificadores lineares focados (máquina de suporte vectorial e regressão logística) permitiram a obtenção de, para além de valores de acerto mais altos, valores da métrica f1 mais altos do que os restantes. Para além disto, foram observadas algumas situações de documentos que não foram bem classificados por falta de características únicas e, pelo facto das categorias destes documentos terem uma representação baixa nos dados. Este trabalho permite concluir que o uso destes algoritmos é mais adequado para os conjuntos de dados em questão e que é possível a aplicação de métodos de classificação de texto para facilitar e auxiliar o processamento de denúncias e reclamações, levando a uma ação mais rápida por parte das autoridades competentes. Assim, a aposta na classificação textual de denúncias pode influenciar positivamente a quer a prevenção de crimes económicos quer a melhoria da saúde pública, neste caso, por meio da fiscalização alimentar.

Palavras-chave: Classificação de texto; Classificação de denúncias; Seleção de características; Aprendizagem computacional; Classificadores de texto.

Abstract

Text categorization is a supervised learning task which aims to assign labels to documents based on the predicted outcome suggested by a classifier trained on a set of labeled documents. With the increase availability of text accompanying the evermore prevalent growth of the internet and online data, the ability to automatically perform tasks that are time-consuming for a human as well as the ability to find patterns or extract valuable information from data is incredibly powerful. In fact, the association of text classification to facilitate labeling reports and complaints in the economic and health related fields can have a tremendous impact in the speed at which these are processed, and therefore, lowering the required time to act upon these complaints and reports.

In this work we evaluate the classification performance of 9 algorithms at different levels of the text classification flow: Preprocessing, word normalization, feature extraction, feature selection, classifier's hyper-parameters' optimization and evaluation. These algorithms are: Complement Naïve Bayes, Bernoulli Naïve Bayes, Multinomial Naïve Bayes, K-Nearest Neighbors, Decision Tree, Random Forest, Support Vector Machine, AdaBoost and Logistic Regression. Our principle findings reveal that we've achieved high levels of accuracy, roughly around 67% and 85%, in two datasets with different target labels. It was also observed that the linear classifiers (support vector machine and logistic regression) allowed us to obtain higher f1-measure values than the other classifiers in addition to the high accuracy values revealed. Furthermore, some of documents that were misclassified were observed to lack unique features or their representation was low in our data. This work allowed us to conclude that the use of these algorithms is more adequate for the data selected and the applying text classification methods can facilitate and help the denunciation and reports processing which, in turn, leads to a swifter action by authorities in charge. Thus, relying on text classification of reports and complaints can have a positive influence in either economic crime prevention or in public health, in this case, by means of food-related inspections.

Keywords: Text classification; Complaints classification; Feature selection; Machine learning; Text classifiers.

Abbreviations and Acronyms

AB – AdaBoost

ASAE – Economic and Food Safety Authority

BNB – Bernoulli Naive Bayes

Chi² – Chi-square

CNB – Complement Naive Bayes

CV – Cross-validation

DT – Decision Tree

HTML – Hyper Text Markup Language

IE – Information Extraction

IR – Information Retrieval

KDD – Knowledge Discovery in Databases

K-NN – K-Nearest Neighbor

LIACC – Artificial Intelligence and Computer Science Laboratory

LR – Logistic Regression

MNB – Multinomial Naive Bayes

NB – Naive Bayes

RF – Random Forest

SVM – Support Vector Machine

TF – Term frequency

TF-IDF – Term frequency-inverse document frequency

TSM – Tensor Space Model

VSM – Vector Space Model

Table of Contents

1.	Introduction	1
2.	Background and State of the Art.....	3
2.1.	Knowledge Discovery in Databases.....	3
2.2.	Text Mining.....	4
2.3.	Feature Extraction	5
2.3.1.	Bag of words	6
2.3.2.	Tensor Space model.....	7
2.3.3.	Word2vec.....	7
2.4.	Feature Selection.....	8
2.4.1.	Term frequency.....	8
2.4.2.	Term Frequency-inverse Document Frequency.....	9
2.4.3.	Chi-square.....	10
2.4.4.	Information Gain	10
2.4.5.	ANOVA F-test.....	10
2.5.	Data Mining – Classifiers	11
2.5.1.	Naïve Bayes.....	11
2.5.2.	Support Vector Machine	13
2.5.3.	Nearest Neighbor	15
2.5.4.	Decision Tree	16
2.5.5.	Ensemble Learning.....	16
2.5.6.	Logistic Regression.....	17
2.6.	Evaluation Metrics	17
2.7.	Model Validation – Cross-validation	18
2.8.	Related work	20
3.	Methodology	23
3.1.	Software and Programming Languages for Knowledge Discovery	23
3.2.	Experimental Structure	24
3.2.1.	Preprocessing and Word Normalization	25
3.2.2.	Feature extraction, feature selection and evaluation.....	28
4.	Results and Discussion.....	30
4.1.	Exploratory Analysis	30
4.2.	Preliminary Results about Jurisdiction and Infractions.....	36

4.2.1.	Discussion on preliminary results.....	39
4.3.	Word Normalization	39
4.3.1.	Discussion on word normalization	43
4.4.	Feature Extraction	43
4.4.1.	Discussion on feature extraction methods	49
4.5.	Feature Selection.....	50
4.5.1.	Discussion on feature selection and feature sizes	60
4.6.	Hyper-parameter optimization	61
4.6.1.	Discussion on hyper-parameters optimization	67
4.7.	Final Evaluation.....	68
4.7.1.	Discussion on the final results	72
5.	Conclusions and Future Work.....	74
5.1.	Main Contributions and Conclusions	74
5.2.	Limitations and Future work	75
5.3.	Final Notes	76
	References.....	78
	Annex.....	87

Figures

Figure 1- A hypothetical problem separated in a two-dimensional space (adapted).....	14
Figure 2 – K-fold cross validation (adapted).	19
Figure 3 – Stratified K-Fold cross validation (adapted).....	20
Figure 4 – Most commonly used programming languages in data science as of 2018 in a study by JetBrains.	24
Figure 5 – Exemplified of a tokenized and a non-tokenized sample.....	26
Figure 6 – Examples of portuguese stop words (shortened).	26
Figure 7 – Example of a stemmed sentence sample.....	27
Figure 8 – Example of a lemmatized sentence sample.....	27
Figure 9 – Diagram representing the preprocessing steps performed on the dataset.	28
Figure 10 – Diagram representing the procedures involved in the classification task.....	29
Figure 11 – Boxplot representing document's length for each jurisdiction label.	31
Figure 12 – Boxplot representing documents length for the 95 quantiles regarding each jurisdiction.	32
Figure 13 – Distribution of documents length for the entire document. The length is measured in characters and not words.....	32
Figure 14 – Boxplot representing documents length for each infraction category.....	34
Figure 15 – Boxplot representing documents length for the 95 quantiles regarding each infraction category.	35
Figure 16 – Plotted accuracy values for MNB using ANOVA, Chi2, TF and TF-IDF. The results also show the difference between different feature ranges. Results obtained from the jurisdiction dataset.....	51
Figure 17 – Plotted accuracy values for MNB using ANOVA, Chi2, TF and TF-IDF. The results also show the difference between different feature ranges. Results obtained from the infraction dataset.....	51
Figure 18 – Plotted accuracy values for BNB using ANOVA, Chi2, TF and TF-IDF. The results also show the difference between different feature ranges. Results obtained from the jurisdiction dataset.....	52
Figure 19 – Plotted accuracy values for BNB using ANOVA, Chi2, TF and TF-IDF. The results also show the difference between different feature ranges. Results obtained from the infractions dataset.	52
Figure 20 – Plotted accuracy values for CNB using ANOVA, Chi2, TF and TF-IDF. The results also show the difference between different feature ranges. Results obtained from the jurisdiction dataset.....	53
Figure 21 – Plotted accuracy values for CNB using ANOVA, Chi2, TF and TF-IDF. The results also show the difference between different feature ranges. Results obtained from the infractions dataset.	53
Figure 22 – Plotted accuracy values for SVM using ANOVA, Chi2, TF and TF-IDF. The results also show the difference between different feature ranges. Results obtained from the jurisdiction dataset.	54

Figure 23 – Plotted accuracy values for SVM using ANOVA, Chi2, TF and TF-IDF. The results also show the difference between different feature ranges. Results obtained from the infractions dataset.....	54
Figure 24 – Plotted accuracy values for LR using ANOVA, Chi2, TF and TF-IDF. The results also show the difference between different feature ranges. Results obtained from the jurisdiction dataset.....	55
Figure 25 – Plotted accuracy values for LR using ANOVA, Chi2, TF and TF-IDF. The results also show the difference between different feature ranges. Results obtained from the infractions dataset.	55
Figure 26 – Plotted accuracy values for K-NN using ANOVA, Chi2, TF and TF-IDF. The results also show the difference between different feature ranges. Results obtained from the jurisdiction dataset.	56
Figure 27 – Plotted accuracy values for K-NN using ANOVA, Chi2, TF and TF-IDF. The results also show the difference between different feature ranges. Results obtained from the infractions dataset.....	56
Figure 28 – Plotted accuracy values for DT using ANOVA, Chi2, TF and TF-IDF. The results also show the difference between different feature ranges. Results obtained from the jurisdiction dataset.....	57
Figure 29 – Plotted accuracy values for DT using ANOVA, Chi2, TF and TF-IDF. The results also show the difference between different feature ranges. Results obtained from the infractions dataset.	57
Figure 30 – Plotted accuracy values for RF using ANOVA, Chi2, TF and TF-IDF. The results also show the difference between different feature ranges. Results obtained from the jurisdiction dataset.....	58
Figure 31 – Plotted accuracy values for RF using ANOVA, Chi2, TF and TF-IDF. The results also show the difference between different feature ranges. Results obtained from the infractions dataset.	58
Figure 32 – Plotted accuracy values for AB using ANOVA, Chi2, TF and TF-IDF. The results also show the difference between different feature ranges. Results obtained from the jurisdiction dataset.....	59
Figure 33 – Plotted accuracy values for AB using ANOVA, Chi2, TF and TF-IDF. The results also show the difference between different feature ranges. Results obtained from the infractions dataset.	59
Figure 34 – Normalized confusion matrix for LR (jurisdiction dataset).....	69
Figure 35 – Normalized confusion matrix for LR (infractions dataset).....	69
Figure 36 – Normalized confusion matrix for SVM (jurisdiction dataset).....	71
Figure 37 – Normalized confusion matrix for SVM (infractions dataset).....	71
Figure 38 – Normalized confusion matrix for RF (jurisdiction dataset).....	72
Figure 39 – Normalized confusion matrix for SVM (infractions dataset).	72

Tables

Table 1 – Term frequency weighting (example).....	8
Table 2 – Term frequency-inverse document frequency weighting (example).....	9
Table 3 – Sample distribution per label (jurisdiction). The sample distribution is measured in the number of documents each label has assigned.....	30
Table 4 – Descriptive metrics for documents length regarding the jurisdiction attribute.....	31
Table 5 – Most frequent bigrams for each jurisdiction label. Only the top 15 bigrams are represented (from 0 to 14).	33
Table 6 – Sample distribution per label (infraction). The sample distribution is measured in the number of documents each label has assigned.....	34
Table 7 – Descriptive metrics for documents length regarding the infractions attribute.....	35
Table 8 – Average weighted accuracy for each classifier with 10-fold stratified cross-validation for the jurisdiction dataset with no preprocessing (Ds1), dataset with only stop words removed (Ds2) and fully processed dataset(Ds3).....	36
Table 9 – Average training times in seconds for each classifier with 10-fold stratified cross-validation for the jurisdiction dataset with no preprocessing (Ds1), dataset with only stop words removed (Ds2) and fully processed dataset(Ds3).....	37
Table 10 – Average weighted accuracy for each classifier with 10-fold stratified cross-validation for the infractions dataset with no preprocessing (Ds1), dataset with only stop words removed (Ds2) and fully processed dataset (Ds3).....	38
Table 11 – Average training times in seconds for each classifier with 10-fold stratified cross-validation for the infractions dataset with no preprocessing (Ds1), dataset with only stop words removed (Ds2) and fully processed dataset(Ds3).....	38
Table 12 – Average weighted accuracy for each classifier with 10-fold stratified cross-validation for the jurisdiction dataset with a fully preprocessed dataset (Ds3), stemmed dataset(Ds3-S) and lemmatized dataset (Ds3-L).	40
Table 13 – Average training times in seconds for each classifier with 10-fold stratified cross-validation for the jurisdiction dataset with a fully preprocessed dataset (Ds3), stemmed dataset(Ds3-S) and lemmatized dataset (Ds3-L).	41
Table 14 – Average weighted accuracy for each classifier with 10-fold stratified cross-validation for the infractions dataset with a fully preprocessed dataset (Ds3), stemmed dataset(Ds3-S) and lemmatized dataset (Ds3-L).	42

Table 15 – Average training time in seconds for each classifier with 10-fold stratified cross-validation for the jurisdiction dataset with a fully preprocessed dataset (Ds3), stemmed dataset(Ds3-S) and lemmatized dataset (Ds3-L).	42
Table 16 – Average weighted accuracy for TF and TF-IDF feature extraction with unigrams for the jurisdiction dataset.....	44
Table 17 – Average weighted accuracy for TF and TF-IDF feature extraction with unigrams for the infractions dataset.....	45
Table 18 – Average weighed accuracy for each classifier with TF-IDF feature extraction considering unigrams, bigrams and trigrams, the combination of unigrams and bigrams and the combination of unigrams, bigrams and trigrams, applied to the jurisdiction dataset.....	46
Table 19 – Average train times for each classifier with TF-IDF feature extraction considering unigrams, bigrams and trigrams, the combination of unigrams and bigrams and the combination of unigrams, bigrams and trigrams, applied to the jurisdiction dataset.....	47
Table 20 – Average weighed accuracy for each classifier with TF-IDF feature extraction considering unigrams, bigrams and trigrams, the combination of unigrams and bigrams and the combination of unigrams, bigrams and trigrams, applied to the infractions dataset.	48
Table 21 – Average train times for each classifier with TF-IDF feature extraction considering unigrams, bigrams and trigrams, the combination of unigrams and bigrams and the combination of unigrams, bigrams and trigrams, applied to the infraction dataset.	49
Table 22 – Hyper-parameters' searching space for K-NN.	62
Table 23 – Average weighted accuracy for train and test sets with different values of K (K-NN).....	62
Table 24 – Hyper-parameters' searching space for LR.....	63
Table 25 – Average weighted accuracy for train and test sets with different values of C (LR).....	63
Table 26 – Average weighted accuracy for train and test sets with different multi-class schemes (LR)..	63
Table 27 – Average weighted accuracy for train and test sets with different penalties (LR).	64
Table 28 – Hyper-parameters' searching space for RF.....	64
Table 29 – Average weighted accuracy for train and test sets with different number of estimators (RF).	64
Table 30 – Average weighted accuracy for train and test sets with different minimum samples split values (RF).....	65
Table 31 – Average weighted accuracy for train and test sets with different minimum samples leaf values (RF).....	65
Table 32 – Average weighted accuracy for train and test sets with different maximum depth values (RF).	66
Table 33 – Hyper-parameter's searching space for the SVM.....	66

Table 34 – Average weighted accuracy for train and test sets with different values of C (SVM).	67
Table 35 – Average weighted accuracy for train and test sets with different penalties (SVM).....	67
Table 36 – Evaluation metrics for LR: Accuracy, precision, recall and f1-measure (macro).....	69
Table 37 – Evaluation metrics for SVM: Accuracy, precision, recall and f1-measure (macro).....	70
Table 38 – Evaluation metrics for RF: Accuracy, precision, recall and f1-measure (macro).	71

1. Introduction

Economic and Food Safety Authority (ASAE) is Portugal's national authority with administrative autonomy whose aim is to prevent and oversee the compliance with the current law regulating legislation regarding economical activities in the food and non-food sectors, as well as reporting possible food chain-related risks. It also provides a link to other European and international entities that act with the same scope.

From its most remote and similar concept and creation in 1893 by royal decree of D. Carlos I up until now, ASAE has mostly served the same purpose of acting against economical delicts and improving public health. In fact, in 1943 the General Intendence of Supplies (IGA) was created. Its purpose was to help oversee and manage rationing of some food items, consequence of the Second World War. After the war and since the need to ration supplies was ending, IGA's focus was ever more geared towards prevention and repression of economic crimes and public health risks. Although it was restructured several times over the last century, it was in 2006 that ASAE was created.

As mentioned above, ASAE acts in a proactive or reactive way. It acts over possible detected irregularities in food genders sold to the consumers, emergency related situations such as food intoxications or food crises, information retrieved from information exchange system such as the Rapid Alert System (RASFF) and the Alert System RAPEX. The Rapid Alert System's basis was defined on article 50 of the European Communities (EC) Nº 178/2002 and aims to be a system that allows the member states to rapidly exchange information regarding potential dangers towards food and feed (1). It is based on a notification system which encompasses alert notifications, information notifications, border rejection notifications, original notifications and follow-up notifications and rejected and withdrawn notifications (2). While RASFF is the alert system for food and feed products, RAPEX is the system in place for non-food products. It covers products that pose a serious risk to health and safety not including pharmaceutical and medical devices. It has also been expanded under several directives to encompass work related issues as well (3). The Economic and Food Safety Authority also acts on complaints and reports that are submitted directly to ASAE through their website, mail or fax, telephone contact or even presential complaints or reports.

Every year, from 2014 to the time of writing, this authority receives around twenty thousand complaints. These complains are stored in a database and are later analyzed by ASAE's inspectors. For the inspectors to act on this information, the complaints need to be, first and foremost, readable and the inspectors should be able to extract important information such as the possible infraction and the economic entity that is mentioned on them. This is not always the case as some complains have no valid information or do not mention the economic agent involved. In addition, some of these complains are not of ASAE's jurisdiction and should be directed towards other entities, roughly one third of the complains submitted to ASAE. This constant review of data is time consuming and has a heavy toll on the speed at which these complains are

followed on besides being, at the current time, a completely manual process. The possibility of helping inspectors process this information with machine learning tools is of great value to ASAE as it will allow its inspectors to act swiftly on any information received.

In fact, this is one of the objectives of the Artificial Intelligence in Food and Economic Safety (IA.SAE) project. It is a joint effort between ASAE and the Artificial Intelligence and Computer Science Laboratory (LIACC). IA.SAE's most overarching objective is the development of risk analysis models and the selection of economic agents to inspect with the use of state-of-the-art artificial intelligence techniques as well as machine learning methods.

One of IA.SAE's modules aims to extract valuable information from electronic reports and complains but also prioritize complains and reports by rank of importance as well as separate them according to the information retrieved, such as the economic entity or infraction.

This dissertation work is based on the problem created by the huge amount of data that ASAE receives and is incorporated on this module of the IA.SAE project. It involves categorizing several text-based documents present on ASAE's database and labeling them according to their infraction and jurisdiction. Throughout this dissertation, several classification algorithms were tested: Naive Bayes variants, Support Vector Machines, Logistic Regression, ensemble methods such as Random Forest and AdaBoost, k-Nearest Neighbors and Decision Trees. The process of categorizing documents by their respective categories is of utmost importance for a faster reports and complaints process flow as well as indirectly improving public health by facilitating data handling. The faster a complaint or report is addressed, the faster a possible health problem is detected, and thus swifter follow-up actions are done from ASAE's inspectors and competent organs.

This dissertation is organized in 5 chapters. The current chapter, the first chapter, contains an introduction to the problem at hand and a brief explanation of how it will be tackled. The second chapter aims to present a background and state-of-the-art research on the topic of text mining and its derived text classification problem, as well as related work on the subject. Chapter 3 contains the methodology employed on this dissertation and information on what was done on each step. The fourth is the results section where a description of the results obtained for each section of the methodology and discussion are presented after each topic before moving to the next, for readability and understanding purposes. The final chapter aims to conclude the work with all the information obtained on this dissertation as well as the due conclusions that can be established from the analysis performed, in addition to limitations and future work.

2. Background and State of the Art

2.1. Knowledge Discovery in Databases

In 1989, the estimated number of databases were five million, albeit most of them were small compared to today's size (4). Huge quantities of data are generated each day and the rate at which it grows does not drop but rise ever more. This growth makes it so that humans can not cope with the huge influx of data and thus require automated means of extracting and analyzing this information (5). The automated and intelligent analysis, information crossing and retrieval is a growing trend that allows companies and individuals the opportunity to generate value out of it. As Frawley, William stated in 1992, *"there is a growing realization and expectation that data (...) will be a valuable resource to be used for a competitive advantage"* (4). This remains true today and is, indeed, ever more prevalent.

Knowledge Discovery in Databases (KDD) is a set of chained processes that allow for the discovery of useful knowledge and meaningful knowledge from data. It embraces several domains ranging from machine learning, database technology, statistics and mathematics and is the overall arching process behind extracting knowledge and making sense of data (6). KDD is achieved when a knowledge-discovery system is applied on a database to retrieve yet to be discovered relationships or pattern between data (7). The KDD model can be summarized to five steps which can have several branched multiple sub-steps, from raw data selection to knowledge extraction. These five steps are described below (8).

Selection. The first step of the KDD model is to select proper data for the problem presented. This involves understanding the application domain, understanding the business model and identifying the data mining goal. This step can then be summarized in two different but interconnected sub-steps: Identifying the goal of the KDD process and creating a target data set to which knowledge extraction is to be applied (6).

Preprocessing. Preprocessing is one of the most important steps in the KDD process as improperly cleaned data can lead to bad results and thus, the quality of the data might not be able to support the knowledge discovery process (9). Preprocessing includes procedures such as outliers or noise removal, handling missing data, handling data types, collecting information to model or account for noise, accounting for time sequence information, removal of duplicates and feature reduction (6,8,10).

Transformation. The third step of the KDD process aims to form a proper data structure that the data mining algorithms can work on. Data reduction and projection are included on this step. Their aim is to find features that best represent the problem at hand and reduce the dimension of the data that is fed to the model (8,11).

Data Mining. In this step, an appropriate data mining algorithm(s) is chosen that better fits the end goal. In fact, there are several data mining methods such as summarization, clustering, classification, regression, sequence modeling, etc. This step also includes searching for the best parameters for the algorithm(s) chosen and matching the particular method with the overall objective of the KDD process (6).

Evaluation and Interpretation. This is the last step of the KDD process. The user performs an evaluation and validity of the model(s) output. This must be matched to the objective defined on step one. In addition, if the need arises, a return to step 2–4 can also take place in order to perform different methodologies to achieve the best results. Furthermore, the evaluation and interpretation step can also include visualization of extracted patterns and knowledge using the models selected (8,11).

2.2. Text Mining

Text is present everywhere in our everyday life. It is present in the books we read, the e-mails we receive, the text we write and in other situations. Text can be considered as data in form of words or alphabetical characters, a connected piece of work that aims to convey a message. Textual data grows by the day and, according to Simon *et al.* (12), biomedical literature grows by almost a million new publications each year, at the time the article was written. These large corpora of data contain valuable information that can be extracted and read by a machine although not directly.

Text mining is defined as a field between linguistics and data science whose goal is to extract interesting and non-trivial information and knowledge from unstructured and semi-structured text data. In its raw form, text is a collection of unstructured data that is connected by a set of rules to form cohesive information in the form of natural language (13). These set of rules have been ever evolving for thousands of years and form the basis of how humans communicate, creating what we call language. Despite these set of rules being well depicted in today's society, humans find ways to bend and work around them, creating a huge variety in textual information (14). In addition, to be able to apply methodologies that allow us to extract information and establish relations between the data, it is necessary to transform the unstructured data to structured data. On one hand, structured data is represented by numerical values and resides in a record or file contained in a relational database and in spreadsheets. On the other hand, unstructured data does not reside in these traditional databases and is free of form, for instance in text documents (15). Text data can also be in a semi-structured form, such as in hypertext markup language (HTML) files, widely used in web pages (16).

In fact, text mining can have several applications such as information retrieval, information extraction, categorization and natural language processing (15). These will be described further below.

Information retrieval (IR) refers to the association and retrieval of information from a large set of text documents. IR's aim is to find useful information that users want from a large database. This is often the case of web search engines which sip through huge amounts of unstructured data to find the best match for the keywords provided (15,17).

Information extraction (IE) is a field in text data mining that computes statistics over large corpus of text to discover useful patterns identifying key words and relationships within the text (15,17). One such example is named entity recognition (NER) which involves searching specific objects like organizations or

people's names (18). One of the problems information extraction faces is the lack of information regarding the targeted words in each sample of text, reducing the capability of the training models, as Alan Ritter *et al.* reported (19).

Another case of IE is web scrapping which is the process of extracting structured data from unstructured or semi-structured structures such as web pages. This process allows for the extraction of information such as prices, authors, items sold and other information from noisy text data with meta-text, such as HTML.

Much like with structured data, text can also be clustered based on document similarities or classified based on a given set of keywords. This is extremely helpful to automate tasks such as indexing (20).

On one hand, text classification aims to identify the theme of a document by placing it in a set of pre-defined topics. The unstructured text is transformed into a bag of words which the algorithm then tries to classify according to the frequency of each word present on the document (21). Text categorization is a form of supervised machine learning where there is previous knowledge about which document belongs to each class. The model is trained with this information and it is then tested on a set of documents with no knowledge of their respective labels. Furthermore, text classification can also be single-label or multilabel. In single-label classification, each element of a given set of documents can only be fit to one class, whereas in multilabel classification, each element can be assigned to different categories and thus have overlapping categories. Even though the first problem is of binary classification, the algorithms employed can also be used for a multilabel problem by treating multiple labels as multiple binary classification problems. This requires the assumption that each label is independent of each other (22).

On the other hand, text clustering's objective is to group documents by their content without previous knowledge about what they represent (unsupervised learning). Thus, unsupervised methods do not require any training data and can be directly applied (23).

Categorization methods will be further explored on the topic "Data Mining – Classifiers".

Natural Language Processing (NLP) is a field whose goal is to allow for a better understanding of natural language by computers. Much like text mining, NLP is an interdisciplinary field between linguistics, computer science, cognitive science and artificial intelligence (24). As A. Kao *et al* puts it, "*NLP is the attempt to extract a fuller meaning representation from free text*", while text mining consists of a set of procedures to help reach that objective (25).

2.3. Feature Extraction

In order to extract any knowledgeable information, the text must be processed so that the raw unstructured data can be read by a machine whose ability to process data is only impaired by the inability to directly extract information from text. Feature extraction aims to turn this collection of raw unstructured documents into machine-readable formats such as vectors of features.

An appropriate set of keywords needs to be selected to better represent the document, reduce dimensionality and clutter. Each of these keywords are considered features and will be used by the classifiers to predict the outcome of the input. Different algorithms have been developed to represent text in ways that data mining algorithms can read and understand. Some of these methods are summarized below.

2.3.1. Bag of words

The bag of words method is a vector space model (VSM) that considers every word to be a feature. This is one of the most common methods of feature extraction and is the one we will be focusing on this dissertation. It is usually combined with different methods of feature selection and dimensionality reduction, because it tends to wield very large sets of corpora. It is the simplest approach for feature extraction and every document is represented as a vector of words without consideration for their position or meaning. The only property that the bag of words methods captures is word frequency.

To reduce the impact caused by only considering word frequency as a method of feature extraction, several key steps are also applied to this method. As a first way of dealing with this issue, stop words are often removed. These words are often conjunctions, articles, prepositions and other common words (e.g. "a", "the", "I") that are present in almost every sentence. This can be considered as a **filtering** step (26). There have been some interesting studies related to commonly occurring words in text such as the well-known *Zipf's law*. In rough terms, *Zipf's law* states that a small set of words are used very often while the majority are used seldomly (27). Zipf observed that word frequency is an inverse power law. The frequency of a word in a corpus is inversely proportional to its rank (28). This means that if the most frequent word in a corpus has a frequency of 1, then the second most occurring word has a frequency of 0.5. The equation below shows this simplified version of *Zipf's law*.

$$P(n) = \frac{1}{n} \quad (1)$$

n stands for the rank of the word and thus its frequency $P(n)$ is inversely proportional to its rank.

Zipf's law was validated across non-English languages as well as the language whose majority of the textual content in this dissertation falls to, Portuguese. A study by (29) indicates that *Zipf's* first law of frequency in relation to rank was validated.

Since the most frequent words tend to be stop words, these rapidly add up to the total amount of features present in a bag of words representation and thus the filtering step is essential as they provide little to no information. This step can also be complemented with stemming, lemmatization and other forms of dimensionality reduction.

There are several readily available algorithms to stem or lemmatize English words such as Python NLTK's stemmer (Porter's stemmer 1) (30,31), and Snowball's stemmer (Porter's stemmer 2) (30), spaCy's lemmatizer (32), WordNet's lemmatizer (33), among others. The same applicability for the Portuguese

language is harder to find and often pales in comparison to the more common English counterparts. Stemmers are more common than lemmatizers for the Portuguese language as the latter only has two major lemmatizers readily available in Python – spaCy's and Hunspell's. LemPort's lemmatizer (34) is only available in the Java programming language.

2.3.2. Tensor Space model

The tensor space model was proposed by (35) and aims to represent documents by high order tensors. Tensors are used to describe vectors and matrices as a 1-order tensor represents a vector structure, a 2-order tensor a matrix-like structure and a 3-order tensor a cube-like matrix structure. This can be further expanded to other dimensions.

The tensor space model uses a 3-order tensor to represent the 26 letters of the English alphabet. Each letter of the alphabet is denoted by its position on the alphabet and words that do not correspond to the alphabet are represented by a "_" which equals to 0. Thus, the documents are represented by a $27 \times 27 \times 27$ 3-order tensor. For instance, the string "Economic and Food Safety" can be decomposed into:

"eco,con,ono...,and,nd_,d_f,..."

This means that for example, "eco" is equal to (5,3,15) while "nd_" is equal to (14,4,_). Thus, a 4-order tensor is a document collection and is represented by $27 \times 27 \times 27 \times n$ with n being the number of documents in the collection.

The authors reported that with the use of the Tensor Space Model (TSM), F1-metric scores were higher than the standard Bag of Words method. The authors also established comparisons with reduced dimensions of the TSM method: 26^4 , 20^4 and 10^4 . The 26^4 order tensor performed better than the 27^4 order tensor and both performed better than the VSM model. The 10^4 dimension model had much lower results than the VSM model but its performance is better and thus can offer a better solution to large VSM models (35).

2.3.3. Word2vec

Word2vec was introduced by (36) and is a group of shallow, two-layer neural networks that produces word embeddings unlike the bag of words model. Using these neural networks, word2vec learns word representation as vectors when it is fed a large enough corpus. Word2vec finds word similarities and groups words based on the context in which they appear, thus words that are most similar in their meaning tend to be the closest neighbors in the vector space. The authors proposed two architectures, the continuous bag-of-words model (CBOW) and the continuous skip-gram model.

On one hand, the CBOW model is a closer approximation to the bag of words model as it does not take in consideration the sequence of the word's context and thus, does not impact the prediction of the nearest

neighbor words on the vector but instead using a continuous distributed representation of the context. On the other hand, the skip-gram model predicts the current word by using information of other words present in the sentence. The words that are predicted further away from the current word are given a lesser weight because they tend to not represent the current word as well as words that appear closer. This is similar, in theory to the term frequency-inverse document frequency feature selection implementation described in the next topic.

In addition, the authors observed that despite a higher number of features leading to a better accuracy score, the model trained with the CBOW architecture had diminishing returns when adding more features and training data when the vocabulary remained the same. This was tackled by increasing the dimensionality which proved to be more efficient, accuracy-wise.

2.4. Feature Selection

Feature selection is the process that aims to select features that provide important information contained in the text and useful for the data mining steps. In this step, feature reduction can also be considered to reduce dimensionality and remove features that provide close to none or no valuable information (37).

2.4.1. Term frequency

Term frequency (TF) is credited to be first used by (38) and stands for the total amount of times that a term appears throughout a document. This means that a document is a vector of terms so that it contains each term in the vocabulary and that for each time a term occurs in the document, a weight of 1 is added to it, being then divided by the total number of words in that document. Let two documents be:

- Document 1: "This complaint is about food safety risks"
- Document 2: "This complaint is not about food safety"

The vectors for each document can be represented as follows (Table 1):

Table 1 – Term frequency weighting (example)

	This	complaint	is	not	about	food	safety	risks
D1	0.14	0.14	0.14	0	0.14	0.14	0.14	0.14
D2	0.14	0.14	0.14	0.14	0.14	0.14	0.14	0

In the example above we only observe that the words appear once through the documents but if the same word appears more than once then its weight is proportional to its frequency in the document. This means

that terms that are important are tied to the notion that they appear more often than those that are not. However, this means that terms that are frequently used and are not representative of the document will have more impact on the training data. In order to counterbalance this, inverse document frequency is often employed.

2.4.2. Term Frequency-inverse Document Frequency

The well-known and simple method (TF) is frequently used in conjunction with inverse document frequency (IDF) which translates to term frequency-inverse document frequency (TF-IDF).

Inverse document frequency (39) is represented in equation 3 where t stands for the term and N for the total number of documents in a document collection.

$$idf(t) = \log \frac{N}{df(t)} \quad (2)$$

The IDF of a frequent term is low whereas the IDF of a low frequency term (rare) is high.

Term frequency-inverse document frequency is the product of the document frequency of any given term t by the inverse document frequency of that term as follows:

$$tfidf(t, d) = tf(t, d) \times idf(t) \quad (3)$$

Thus, a weight is assigned to any given term t in a document that is: lowest when the document occurs in most documents of the document collection; highest when that term occurs often within a small number of documents and lower when the given term occurs fewer times in a document or in many documents (37,39). For a better representation of this method, let two documents be the ones mentioned on the subsection above. We get the data in Table 2:

Table 2 – Term frequency-inverse document frequency weighting (example)

	This	complaint	is	not	about	food	safety	risks
TF(D1)	1	1	1	0	1	1	1	1
TF(D2)	1	1	1	1	1	1	1	0
IDF	0	0	0	Log (2/1)	0	0	0	Log (2/1)
TF-IDF(D1)	0	0	0	0	0	0	0	0.30
TF-IDF(D2)	0	0	0	0.30	0	0	0	0

We can see that only the word “risks” is truly representative for document 1 while the word not is the only describing factor in document 2. Although the example is short, the possibility in a much wider document collection for a true representation is indeed better than with term-frequency.

Other methods have also been employed for the task of feature selection such as chi-square (40,41) and information gain (40,42), among others.

2.4.3. Chi-square

Chi-square(X^2) feature selection is based on the popular statistical method Chi-square which is used to determine the independence of two events. In feature selection it measures the lack of independence between a term t and a category C . These are independent if:

$$P(tC) = P(t)P(C) \quad (4)$$

Chi-square is a measure of how much counts E are expected and the actual observed counts N deviate from each other and can be represented by the equation below:

$$X^2(ID, t, c) = \sum_{e_t \in \{0,1\}} \sum_{e_c \in \{0,1\}} \frac{(N_{e_t e_c} - E_{e_t e_c})^2}{E_{e_t e_c}} \quad (5)$$

If the X^2 statistic has a value of 0 then the given term t and the class C are independent and thus the feature does not belong to the class (43).

2.4.4. Information Gain

Information Gain allows the categorization of a certain term by knowing whether it is present or absent from a document and measuring its entropy.

In (40), the authors found that information gain (IG) was able to remove several unique terms from the analysis without damaging the classifier’s performance.

Forman, George (44) concluded that to reduce a dataset’s feature size without hampering the classification performance, a rare word cutoff should be set to low and feature selection should be linear in time according to the size of the dataset.

2.4.5. ANOVA F-test

The ANOVA F-test was proposed by (45) reduce dimensionality and increase accuracy classification. The authors used this feature selection method to recognize important features that were shown to represent e-mail classification spam. The F-test allows for the comparison between models and evaluate if the difference is statistically significant. The ANOVA F-test allows for the estimation of sample variances: the total sum of squares, the sum of squared errors and the sum of treatment squares. The sum of squares is given by:

$$SS = \sum_{i=1}^N \hat{u}^2 \quad (6)$$

Where $\hat{u}^2$ represents the squared errors. Thus, we can also represent the equation above as:

$$SS = \sum_{i=1}^N (x_i - \bar{x})^2 \quad (7)$$

The total sum of squares is given by:

$$SS_{\text{Total}} = SST + SSE \quad (8)$$

SST represents the total sum of squares of the treatment while SSE is the total sum of squared errors or residuals. With the measures above, the variance is given by the proportion between the total sum of treatment squares over the total sum of squares which gives us the F value:

$$F = \frac{\frac{SST}{(I-1)}}{\frac{SS_{\text{total}}}{(df)}} = \frac{\text{Variance between treatments}}{\text{Variance within treatments}} \quad (9)$$

The “df” notation concerns the degrees of freedom given by the subtraction of I (number of treatments) to the total number of cases.

The higher the variance proportion the feature or the feature sets can explain in the data, the better that feature or set of features represent the target label. Thus, these are retained for analysis as they prove to have an impact on the categorization of the target label.

2.5. Data Mining – Classifiers

Data mining is the field that aims to extract valuable patterns in data from large volumes of it. Unlike text mining, data mining methods expect to be fed highly structured data in numerical form, which text is not (46). Thus, the range of functions that turn this unstructured data into structured data enable the application of data mining methods to textual data. In the context of this dissertation, these data mining methods or, more precisely, text classification methods aim to assign a pre-defined class to text documents.

For this objective, there are several algorithms in literature that have been successfully applied to classify a set of documents. Some of the most commonly applied algorithms are: Naïve Bayes Classifier, Nearest Neighbor Classifier, Decision Trees, Support Vector Machines, Random Forest Classifier and Logistic Regression (20,47).

2.5.1. Naïve Bayes

Naïve Bayes (NB) is one of the most used, if not the most popular model of probabilistic classifiers (48). It is capable of achieving reasonable performances even though it is simplistic on its methodology and based on an unrealistic independence assumption (49).

The Naïve Bayes classifier learns the probability of each attribute A_i , given the label C from the training data and then calculates the probability of C given the instance of $A_1, \dots, A_n$, predicting the class with the

highest probability. All features A_i are independent given C . This means that the probability for each instance to occur given a label C is strictly independent from each other when $P(C) > 0$ (50).

The Multinomial Naïve Bayes (MNB) implements a Naïve Bayes algorithm for multinomial distributed data. This is a generalization of the binomial distribution. In the binomial distribution the possible outcome of any given trial is either success or failure, a Bernoulli trial. In the multinomial distribution, the outcome is not fixed and thus can take on more values than the Bernoulli's outcomes of two possible results. For text classification problems, each document (D_i) is usually represented by a vector of features (f_j). By applying the Bayes theorem, the probability of class (C_k) is given by

$$P(C_k) = \frac{D_{i,k}}{D_i} \quad (10)$$

so that the probability of each class is the number of times a document D_i is of class C_k ($D_{i,k}$) over the total number of documents in the document collection. The probability that each document belongs to a class (C_k):

$$P(C_k | D_i) = \frac{P(C_k)P(D_i | C_k)}{P(D_i)} \quad (11)$$

$P(C_k | D_i)$ stands for the probability of C_k occurring given D_i . $P(D_i)$ is the normalizing constant, same for all classes and thus can be eliminated from the equation as every document occurrence is independent of the class. Thus, we get that:

$$P(C_k | D_i) = P(C_k) \prod_{j=1}^l P(f_j | C_k) \quad (12)$$

$P(f_j | C_k)$ stands for the probability of the feature f_j given the class C_k . The probability of the document belonging to the class is equal to the product of the probabilities of the features belonging to that same class. The probability of a feature belonging to the given class is given by:

$$P(f_j | C_k) = \frac{n_{j,k} + 1}{n_k + l} \quad (13)$$

$n_{j,k}$ stands for the number of times the feature belongs to the given class, n_k is the total number of features for the class while l represents the entire vocabulary.

The Bernoulli Naïve Bayes classifier (51), contrary to the multinomial variant, ignores term frequency and replaces it by term presence. This multivariate event model thus considers the document to be a binary vector over the space of words. This means that given the vocabulary of the document, each word is treated as a binary event ("0" and "1" for whether the word is not present or present on the given document set, respectively). The probability of each word occurrence is independent of other words in the document. As the authors in (52) state, a document is then considered a "collection of multiple Bernoulli experiments". The Bernoulli Naive Bayes decision rule is thus calculated by:

$$P(D_i | C_k) = P(f_j | C_k) D_i + (1 - P(f_j | C_k))(1 - D_i) \quad (14)$$

A feature f_i that does not occur on class C_k but is an indicator of the same class, suffers a heavier penalization. McCallum, Andrew et al (52), compares two instances of the Naïve Bayes classifier, the multi-variate Bernoulli model and the multinomial event model. The multinomial model takes in consideration word position and thus is an ordered sequence of word events. These event's occurrence position and context are independent of each other. The authors concluded that the multinomial event model should be a more accurate model if the given documents have larger variance in their length. Even though the results for both models were similar for smaller document lengths, the multinomial event model, overall, performed better when compared to the multi-variate Bernoulli counterpart.

These independence assumptions allow for a much more efficient classifier but hamper its performance in relation to other algorithms. To tackle this issue, (53) proposed several solutions to improve Naive Bayes downsides such as the inability to map good decision boundaries between imbalanced classes, and its “trademark” of feature independence. If a class has a stronger word dependency than a class with a weaker one, Naive Bayes attributes a lower weight to the former class. The authors propose that instead of calculating the likelihood of a feature occurring in a class, CNB calculates the likelihood that it occurs in other classes and seeks the minimum argument, given by:

$$\operatorname{argmin} p(C_k) \cdot \prod \frac{1}{P(f_j|C_k)^{f_{k,i}}} \quad (15)$$

In (54) Naïve Bayes (NB) classifier was adopted to classify an experimental dataset into four categories. The document was submitted to a preprocessing phase and feature selection before the classifier was trained. The model's ability to predict which document belonged to each label was addressed by evaluating performance metric's such as recall, precision and F-score. Furthermore, Naïve Bayes classifier was also compared to other classifiers such as Support Vector Machine (SVM), Nearest Neighbors (NN) and Decision Trees (DT). NB achieved 97% of correctly labeled instances and was superior to the other models (SVM: 96,9%; NN: 93,0%; DT: 91,1%) although the author states that a proper preprocessing and feature selection phase is of paramount importance to the performance of the mining process.

2.5.2. Support Vector Machine

Support Vector Machines (SVM) were first introduced by (55) to solve two-group classification problems.

Support Vector Machine algorithms separate data between hyper-planes. Thus, the SVM separates the data in two sets, negative and positive which are used to draw the decision surface that separates them, the hyperplane. The hyper-planes are defined by the target classes (56,57).

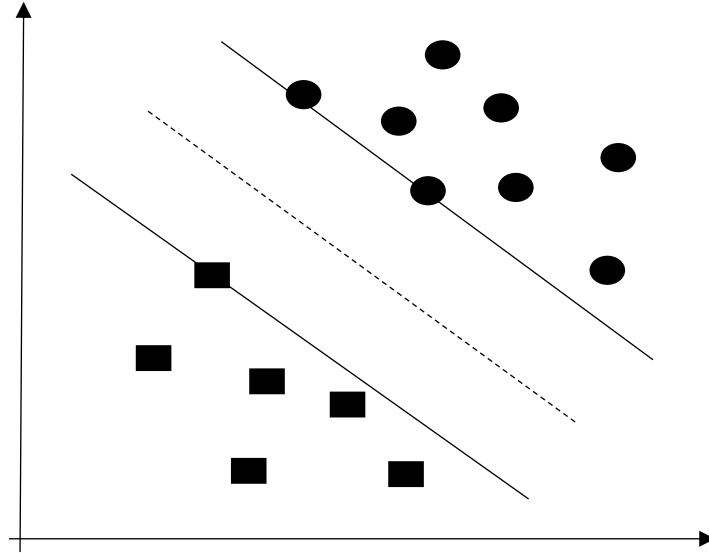


Figure 1– A hypothetical problem separated in a two-dimensional space (adapted).

Figure 1 (adapted from (55)) shows the SVM in action where the full lines define the largest separation between the two classes, the optimal margin while the optimal hyperplane is represented by the dashed line. Class C_i can either be positive or negative. The lines that represent the optimal margins are also called support vectors.

SVMs are adequately prepared to deal with a large number of features because they measure the hypotheses' complexity based on the margin that separates them in the hyperplane and not the number of features (58). Even though SVMs can deal with large feature vectors, having a large number of categories exponentially increases computational load (59).

The linear support vector machine is represented above on Figure 1. The documents are considered linearly separable if they fall on either side of the hyperplane. A sample is considered positive if

$$w \cdot u_i + b \geq 1 \quad (16)$$

where w is a vector and b a scalar. The unknown sample u_i is then considered positive if its dot product with w is greater or equal than a constant C or as represented above, $1 - b$. The inverse applies to a sample belonging to the negative class such that:

$$w \cdot u_i + b \leq -1 \quad (17)$$

If a set of constraints are applied to the inequalities (8; 9) then the optimal hyperplane is obtained by:

$$w_0 \cdot u_i + b_0 = 0 \quad (18)$$

The longer the magnitude of vector w_0 , the better our classifier is able to separate our data and thus the better it performs. The vector w_0 that influences the optimal hyperplane is given as a linear combinations of support vectors with enforced constraints:

$$w_0 = \sum_{i=1}^t \alpha_i^0 C_i u_i \quad (19)$$

α_i^0 is ≥ 0 where α_i is obtained by a quadratic equation problem. This means that a linear support vector machine is much more complex than for example a Naive Bayes algorithm and thus is a lot harder to compute. In addition, despite achieving fairly good results, if the data is not linearly separable then the algorithm will struggle to find the best hyperplane to split it and eventually fails to catch some information. In (58), the author reports that Support Vector Machines are promising algorithms for text classification due to four text characteristics. SVMs can deal with a high dimensional input space, characteristic of text classification problems, due to over-fitting protection. Text has also been reported to have few irrelevant features that if discarded can reduce classification performance. In addition, document vectors that represent the input text are sparse and SVMs are can deal with dense concepts and sparse instances. Due to the linear separability of text categorization and linear nature of SVMs, good performance can also be achieved.

A study by (60) established comparisons between five text categorization methods: Neural Network(NNet), SVM, Naïve Bayes (NB), k-Nearest Neighbor (kNN) and Linear Least-squares Fit(LLSF). In both macro-level and micro-level evaluation of categorization, SVM achieved a better performance when compared to the other methods. It was also shown that both at micro and macro levels that NB under-performed when compared to the other methods.

The SVM classifier was also reported to outperform the NB classifier by (61) on a multi-class classification task. The error margin was 10-20% lower than Naïve Bayes on one dataset and even larger when all the training dataset was included on the analysis. The authors concluded that the SVM was effective when used on an Error-Correcting Output Coding (ECOC) paradigm both in binary and multiclass categorization. ECOC learns several different binary classifiers and uses the output to determine a label for the inputted sample.

2.5.3. Nearest Neighbor

The Nearest Neighbor or k-nearest neighbor classifier takes a random input document and ranks its nearest neighbors against the training documents. Based on the defined number of k , the classifier uses the categories of the k top-ranking neighbors to predict the input document's label. Each neighbor's similarity score against the new document is computed and the result is used as the weight of each of its categories. The category ranking is the sum of these weights over the k nearest neighbors (62). Due to this, the time that the kNN classifier takes to find these similarities is exponentially increased by the k number and the size of the training set (63). In fact, a proper selection of k has a major impact as a k value

too small will reduce the amount of information, that is present on the training set, used by the algorithm while a value of k too high will skew the prediction towards bigger classes (64). In (64), the authors introduced an adaptative method of k -NN that does not use a fixed k value and instead uses an adaptative k value for different classes according to their sample size.

Tan, Songbo *et al* (65), proposed a method to deal with unbalanced datasets, the Neighbor-Weighted K-Nearest Neighbor (NWKNN). The neighbors that derive from a small sample of classes are given a bigger weight than those that derive from a bigger class frequency instead of balancing the dataset. The authors compared the results from the standard k NN algorithm versus the nearest weighted by evaluating the F1 measure. The NWKNN algorithm was shown to yield much better performance than the KNN counterpart when facing unbalanced datasets.

2.5.4. Decision Tree

Decision tree classifier (DT) follows a rule induction methodology where each category of text is set apart from another by a set of decision rules (66).

A decision tree consists of one root node with every other node having one parent. These nodes are labeled by word occurrences (attributes) in the case of text categorization with branches being their weight on the test document. A decision tree ends in leaves which represent the text's category (67).

A study by (67) on the Arabic text language compared different classifiers and their performance when tested against DT. It was reported that the decision trees outperformed the Naïve Bayes algorithm, the Maximum entropy static system and three different iterations of SVMs.

2.5.5. Ensemble Learning

On one hand, one of the most popular ensemble methods is bagging. Bagging ensemble methods train several individual learners in a parallel and independent way and by a randomly selected subset of the training data (bootstrapping).

On the other hand, methods based on adaptive reweighting of the training set have also been developed and this seem to outperform random ensemble method (68). These methods are called boosting methods and work on the premise that the hypotheses (learning algorithms that generate classifiers) should not be independent of one another and instead reweighted. The learning algorithm is trained several times over the training data and the resultant classifiers are combined in a composite one (69).

Random Forest. Random Forest is an ensemble bagging method comprised of a set of classification trees, a "forest" of decision trees (70). At each node split, a random independent vector is generated but identical to the past vectors. Thus, a tree is formed and each tree votes for the most popular class that represents the input vector. The instance is classified into the class by the majority of votes over all trees (68).

AdaBoost. AdaBoost is an adaptative boosting algorithm proposed by (71). The authors propose two versions of the algorithm (M1, M2) with the second (M2) being a more complex approach at tackling the difficulties of M1. AdaBoost.M1 repeatedly calls a learner over the training data, computing a hypothesis (prediction that should correctly classify the input vector). The learner's objective is to find a hypothesis that minimizes the training error related to the training data used. AdaBoost then combines all the hypothesis into a final composite one. Weak hypotheses that generate error values greater than $\frac{1}{2}$ are harder to handle by AdaBoost.M1. When the number of possible classes rises, so does this error. AdaBoost.M2 tries to tackle this problem by giving a set of plausible classes that represent the vector instead of only one. A bigger strain on computational processing is also demanded of the algorithm because it also uses a different error measurement, the pseudo-loss. This error is computed over all pairs of examples and incorrect labels.

2.5.6. Logistic Regression

Logistic regression algorithms, as the name implies, are often used to handle regression problems. In this case, logistic regression will be used in linear classification situation and thus tends to fall close to an SVM or AdaBoost algorithm. Usually, the condition probability event D_i is given a label C_k is calculated with the following logistic function:

$$f(C_k|D_i) = \frac{1}{1 + e^{(-C_k \alpha(D_i))}} \quad (20)$$

Due to the linearity of the algorithm, it has been reported to achieve similar results to SVMs in text categorization problems. LR is trained on a binary classification task and thus one classifier is fitted for each category for the document being labeled (72). In order to reduce overfitting by having a larger number of features over the number of observations, regularization is needed, thus L1 and L2 normalization are often employed. L1 is known as the LASSO penalization while L2 is the ridge penalization (73). The LASSO penalization – least absolute shrinkage and selection operator – is a regression method that can shrink and select features. It was proposed by (74) and allows the production of coefficients that are exactly 0 and according to the authors manages to compete rather well with ridge selection penalty.

2.6. Evaluation Metrics

In the field of machine learning evaluation metrics are required to access how well any given algorithm is performing. There are many possible methods to evaluate this and some of the most commonly used are accuracy, precision, recall and F-measure.

Accuracy is often the most employed of the four presented and is a measure of the proportion of correctly labeled data over the sum of all the data.

$$\text{accuracy} = \frac{\text{Correctly labeled data}}{\text{All results}} \quad (21)$$

The sum of correctly labeled data is given by the number of true positives plus the number of true negatives, thus the number of correctly labeled input data. The total number of results are given by this sum plus the sum of false negatives and false positives. We can sum these, given the class C_k :

- True positives (TP): Correctly labeled documents that belong to C_k ;
- True Negatives (TN): Correctly labeled documents as not belonging to C_k ;
- False positives (FP): Incorrectly labeled documents as belonging to C_k ;
- False negatives (FN): Incorrectly labeled documents classified as not C_k .

FP are also often referred as type I errors while type II errors refer to FN. The accuracy metric is thus given as:

$$\text{accuracy} = \frac{TP+TN}{TP+TN+FP+FN} \quad (22)$$

While accuracy is a straightforward metric, it does not allow the differentiation of the performance for different classes. A high accuracy value can be interpreted as a good algorithmic performance, but this can be misleading because it can be skewed and not representative of, for example, lesser observed classes. On one hand, precision measures the correctly labeled documents belonging to C_k over the total number of positive observations.

$$\text{Precision} = \frac{TP}{TP+FP} \quad (23)$$

Thus, the higher the precision value, the lower the number of FP obtained. On the other hand, recall presents the correctly predicted positives documents over the total number of documents belonging to that class. This is given by:

$$\text{Recall} = \frac{TP}{TP+FN} \quad (24)$$

In contrast to precision, a high recall value denotes a low FN number. In a brief sum up, precision relates to the number of relevant results while recall relates to the correctly labeled results.

The F1-measure is given by the harmonic mean of precision and recall which translates to:

$$F1 = \frac{2*(\text{Recall}*\text{Precision})}{(\text{Recall}+\text{Precision})} = \frac{2*(\frac{TP}{TP+FN}*\frac{TP}{TP+FP})}{(\frac{TP}{TP+FN}+\frac{TP}{TP+FP})} \quad (25)$$

Thus, recall takes into consideration both FN and FP and is more adequate towards uneven distributed datasets than accuracy.

These metrics are evaluated following a model selection method which typically also involves cross-validation. This method is explained below.

2.7. Model Validation – Cross-validation

In machine learning, the algorithms are traditionally trained on one set of the data called the training data and tested on an unseen set, the testing set. This procedure is also called the holdout method (75). This

approach stops the algorithm from predicting on data which it has already seen and thus avoids overfitting on the test set. Despite this, since the training data is fixed, the model can miss important features and even fail to generalize to the testing set well. One of the commonly used model validation methods to tackle this issue is cross-validation. The training is done on $k-1$ folds and testing done on an unseen separated test fold which allows for better data incorporation while training. This can be seen on Figure 2 (adapted from (76)).

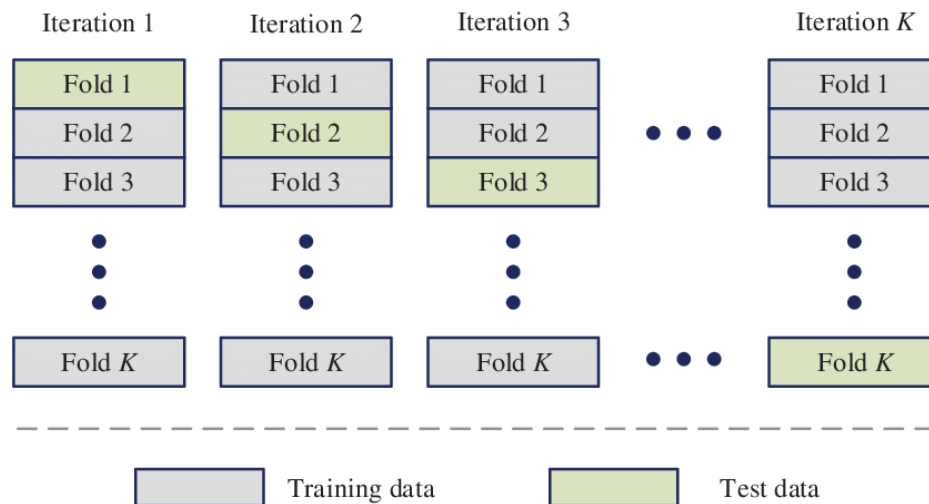


Figure 2 - K-fold cross validation (adapted).

The value of K is an important part of cross-validation. On one hand, a higher value of K can decrease the variance of the algorithm at the expense of a higher bias value. On the other hand, a lower K value leads to a decrease in bias but a higher variance in the dataset. This is demonstrated in (75) in which the authors also conclude that stratified cross-validation is preferred over the regular cross-validation model selection.

Stratified cross-validation follows the same premise as cross-validation but keeps the proportion of each label represented in the test set. In fact, with regular cross-validation the test fold might not be representative of the entire dataset as it might not contain samples related to every label. This is more evident in unbalance datasets. The figure below illustrates a hypothetical stratified cross-validation when compared to a regular cross-validation (adapted from (77)):

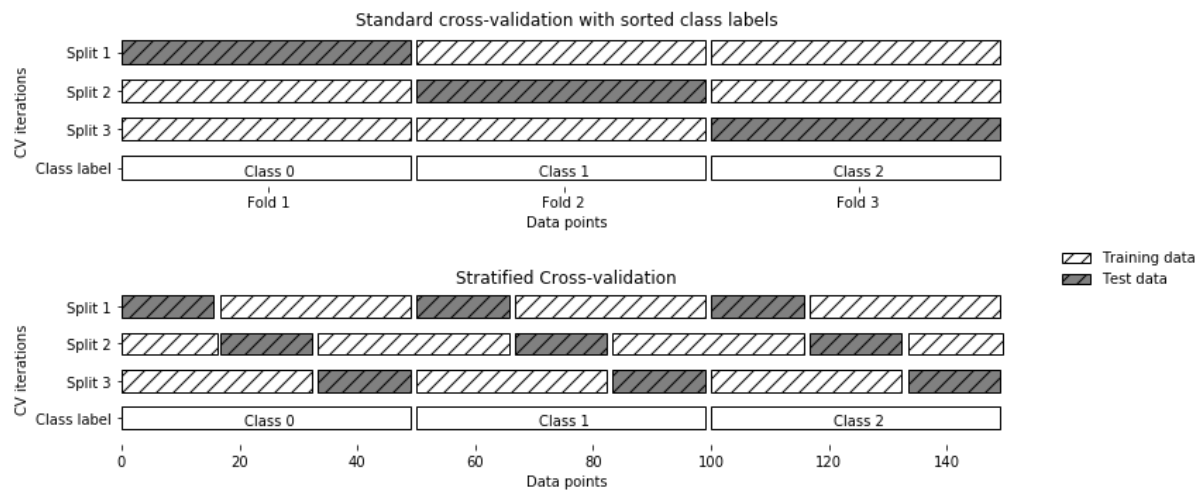


Figure 3 – Stratified K-Fold cross validation (adapted).

Indeed, by observing the above figure it is possible to denote that stratified cross-validation manages to keep every label represented on each split while cross-validation does not.

2.8. Related work

Text classification and subsequently, text categorization, is a trending topic as many of the information found on databases is unstructured and the ability to extract valuable information from these databases is incredibly powerful. In fact, there have been several instances of text categorization done on the complaints field such as hotel customer complains analysis using structural topic models (78) and customer complaints analysis for market-oriented product development (79), automated detection of text containing misrepresentations such as fake reviews (80), text classification based on character-level convolutional network (81), review of user feedback from app reviews with an econometric approach (82) as well as opinion related (83), vehicle consumer complaints which were derived from incidents (84), tools development for mobile network complaint analysis (85) and even classification of agricultural Arabic text complaints, which also involved a novel term weighting scheme proposed by the authors (86). As described, the applications of text mining to the complaints field can lead to endless possibilities in different fields and even in improving costumer protection and health safety related practices, as is our case.

Furthermore, there are numerous works related to different steps that are tackled on this dissertation. In the feature selection field, the authors of (87) conduct a review of feature selection methods providing an overview how different feature selection methods can be applied and related work that has been done in each case. In (44), the author conducts an extensive study on different feature selection methods for the text classification domain where he observed that the tested feature selection methods performed similarly in the accuracy department (slightly better results for binormal separation over 100 features) while recall and f-measure were shown to have better scores with binormal separation and precision with

information gain. The authors of (88) also conducted an evaluation of different feature selection methods but on two different datasets, a Portuguese and an English dataset. The feature selection methods evaluated were term frequency, mutual information and gain ratio with which the authors observed that both term frequency and mutual information performed better than gains ratio for both datasets.

In (89), the authors propose a novel method to obtain a semantic representation of short texts. The authors begin by using a clustering technique called fast clustering which is based on density peaks searching, thus obtaining and discovering embedding spaces between the words. This procedure is then followed by a convolutional neural network with one convolutional layer which produces optimized batches of word clusterings.

In the study of text classifiers and their performance, there are also numerous works related to new classifiers, fine-tuned existing classifiers and comparisons between already established ones. For instance, the authors of (90) propose a classifier called “fastText” to perform text categorization. The authors mention that it is often comparable to deep learning classifiers and several times faster for training and evaluation and evaluate their claims in tag prediction and sentiment analysis.

In (91), the authors compare a set of classifiers – Naïve Bayes, random forest, decision tree, support vector machines and logistic regression – regarding their respective classification accuracy based on sample size and the number of n-grams used. The authors used a dataset derived from amazon’s customer product-reviews totaling 2638274 samples that were labeled by the rating attributed by each customer (1 through 5 stars). The text content was lower-cased and tokenized, stop-words were removed, and stemming was applied, and the features extracted with a bag-of-words approach for unigrams, bigrams and trigrams with a hashing term-frequency vectorizer. It was observed that the logistic regression classifier achieved higher scores than the other algorithms with lower accuracy when using trigrams and the highest accuracy value when trained on unigrams, bigrams and trigrams vectors. The authors also observed that the support vector machine with linear kernel performed slightly worse than the naïve Bayes algorithm and that the results between the Linear Support Vector and the Random Forest were similar. It was also concluded that the usage of unigrams, bigrams and trigrams had the best classification performance although not significant from the usage of unigrams only and that an increase in sample size lead to insignificant growth of classification accuracy in the order of 1-2%.

A study conducted by (92) proposed the use of low-level features to classify web pages according to genres. The authors observed that the use of character n-grams as opposed to word n-grams provided better results for this task. The authors also report that the use of character n-grams allows to better capture nuances in the text while also not being constrict by the language on which it is employed. It was also reported to be more robust than word n-grams. However, this procedure leads to a much higher data dimensionality.

The authors of (93) compared several classification experiments on different stages of the text classification process. The authors computed three versions of the most frequent unigrams in a dataset in which the first version contained stop words, the second version and third version with incremental stop word removal. The documents were represented by a bag-of-words method with term frequency weighting and four supervised algorithms were used for the classification task: Bayes Networks, SimpleLogistic, Random Forest and Sequential minimal optimization. In this experiment, two filtering methods, information gain and correlation-based feature subset selection, were applied to the best model obtained which was then followed by parameter optimization and further evaluation.

3. Methodology

This chapter presents the reader with the procedures done in each step of the proposed KDD methodology. As mentioned above, Knowledge Discovery in Databases follows a set of five steps: selection, preprocessing, transformation, data mining and evaluation and interpretation.

The methodology embraced on this dissertation is based on a state-of-the-art research regarding text mining, its applications and different procedures. Although there are many documented cases of text mining procedures in the literature, ranging from e-commerce (94), stock price predictions (37), creating literature abstracts (38) and named entity recognition (19), those that perform text categorization mostly follow the same methodology.

In fact, the diversity shown in literature and all the different conclusions established by different authors regarding distinct procedures on each step of text mining, presents an opportunity to evaluate already established techniques and algorithms in the field of text mining and draw conclusions on our dataset.

In addition, the dataset used on this dissertation has real data regarding reports and complaints that are submitted every day to the Economic and Food Safety Authority and, as such, is a real use case combining text mining, food safety and economic crime prevention.

3.1. Software and Programming Languages for Knowledge Discovery

There is a wide array of software's used to perform data mining and general statistical tasks and henceforth text mining as well. Some of the well-known software's with graphical interfaces are WEKA (95), R (RStudio) (96), MATLAB (97), RapidMiner (98), among others. According to (99) proposes a 3-way separation of algorithms according to their maturity. Tools that require that to be fit to a memory of a single machine are grouped under the 1st generation of machine learning as is the case of WEKA, R and RapidMiner; Tools that enhance the capabilities of the first module and thus allow for the spread of tasks across Hadoop clusters (computational cluster that is design to hold and analyze huge amounts of data) are called the 2^o generation of machine learning models. The 3rd generation of machine learning tools such as Spark allow for a more efficient, and thus better performing Hadoop cluster usage.

Python (100) is an high-level interactive programming language that has a wide array of libraries that fulfil various data science needs. It is much like R in the way that both are freely available under open-source licenses and easy to use on any system being it any Linux ecosystem, Windows or OSX. Both can be expanded to fulfil the needs of the users by download or importing packages suited to the task at hand. The following figure (Figure 4) is extracted from a JetBrains 2018 survey (101) and illustrates what the most common languages in the data science field are.

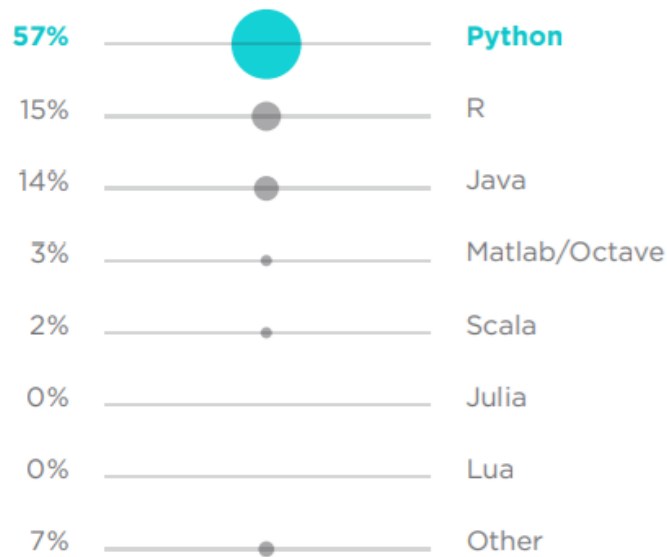


Figure 4 – Most commonly used programming languages in data science as of 2018 in a study by JetBrains.

Python and R are the two most commonly used languages by the respondents of the survey. R is an incredibly powerful statistical language and thus is able to perform tasks with incredible efficiency but being less lenient than python when it comes to data visualization and general use (102). It is also interesting that some of the respondents of the study mentioned above mentioned that they would be changing to the C++ language due to its efficiency speed-wise.

For this dissertation we have used the Python language as our “tool” of choice. One such case is the *Scikit-learn* library (103) which provides a huge array of packages such as, in this dissertation’s particular case, supervised learning classifiers, feature selection algorithms, methods for word representations, among others. However, being a high-level language sacrifices performance and thus other, less interpreted languages can provide faster results, such as Java or C++.

3.2. Experimental Structure

In order to extract knowledge from a database there needs to be proper selection of data that best represents our problem. This is the first step in the aforementioned methodology.

The dataset was supplied by ASAE and it is extracted from the respective storage database. The dataset ranges from the years 2014 to 2018 with a total of 104116 entries and 49 attributes. This data results of the merge of two separate but complementary files with corresponding IDs for each row. This unique ID is attributed to each complain / report when stored in the database.

The “e-mail content” field contains text that is submitted via e-mail to ASAE describing complaints or reports and its analysis allows for the completion of other attributes present on the database. In fact, it is

based on this content that the complaint is filtered to evaluate if it is of ASAE's jurisdiction to follow on it or if it should be redirected to other entities. The "jurisdiction" attribute is then one of the most important attributes and divided into 7 different categories: ASAE, Another entity, ASAE and another entity, Courts, ASAE and Courts, Another entity and Courts and undetermined. Along with the jurisdiction field, the infractions attribute also contains valuable information that helps establish the jurisdiction of each label and how each complain is handled. Since the infractions attribute is subdivided into an extremely large number of descriptors, a simpler representation of this attribute was created. It was decided that the different codes for each infraction would be grouped under 4 representative categories according to ASAE's information. Category I represents complains that are related to food safety, categories II and III regard offenses and category IV relates to other complains that are neither of the food nor economic fields. This, in turn, effectively turned the multi-label process into a multi-class task as was with the jurisdiction label.

3.2.1. Preprocessing and Word Normalization

In this work, the classification task (every analysis after the dataset merge and clean-up) begins by evaluating the classifiers on the capability to predict both labels (jurisdiction and infractions) on three different datasets with varying degrees of preprocessing. The first degree is a non-preprocessed e-mail content field, the second is a mere stop word removal applied to the contents of the first dataset and lastly, the third dataset contains the content which has been treated for stop word removal, punctuation, unreadable characters and non-alphabetic characters. All analysis were performed using a 10-fold stratified cross-validation process to better handle bias and variance. The process described below relates to the different steps taken for each dataset, Ds1, Ds2 and Ds3 as we labeled them.

In the first dataset (Ds1), text was converted to tokens which are mostly words that are separated by white-spaces, punctuation or line breaks and thus represent strings extracted from the text. This process is called tokenization and mostly follows simple heuristic rules. An example of tokenization is shown below (Figure 5) and is extracted from a complaint in the dataset (adapted). Each token is separated by a comma and is contained within single quotes.

Raw sample

"[...]venho por este meio demonstrar o meu desagrado e a minha indignação pelos serviços prestados pela conforma[...]"

Tokenized sample

[...] 'venho', 'por', 'este', 'meio', 'demonstrar', 'o', 'meu', 'desagrado', 'e', 'a', 'minha', 'indignação', 'pelos', 'serviços', 'prestados', 'pela', 'conforma' [...]

Figure 5 – Example of a tokenized and a non-tokenized sample.

Along with tokenization, punctuation was also removed from the samples. Punctuation often creates problems with text classification as it increases noise on the data and thus can have a negative impact on the classification task. The tokenization applied would sometimes remove accents or word related punctuation and thus change its context or even render it completely different of its intention.

In addition, the next step in the text processing is stop-words removal. There are some available Portuguese stop-words lists even though it pales in comparison with more commonly analyzed languages in this field. The retrieved tokens were compared to a list of Portuguese stop words and removed if present. Some of these stop words are described below (Figure 6).

[...]

"a", "acerca", "adeus", "agora", "ainda", "alem", "algmas", "algo", "algumas", "alguns", "ali", "além", "ambas", "ambos", "ano", "anos", "antes", "ao", "aonde", "aos", "apenas", "apoio", "apontar", "apos", "após", "aquela", "aquelas", "aquele", "aqueles", "aqui", "aquilo", "as", "assim", "através", "atrás", "até", "aí", "baixo", "bastante", "bem", "boa", "boas", "bom", "bons", "breve", "cada", "caminho", "catorze", "cedo", "cento", "certamente", "certeza", "cima", "cinco", "coisa", "com", "como", "comprido", "conhecido", "conselho", "contra", "contudo", "corrente", "cuja", "cujas", "cujo", "cujos", "custa", "cá", "da", "daquela", "daquelas", "daquele", "daqueles", "dar", "das", "de", "de"

[...]

Figure 6 – Examples of portuguese stop words (shortened).

Stemming is the process in which the word is reduced to its most basic form – the stem. It aims to remove suffixes, prefixes and the words' plural forms. Thus, if words share the same root it is also expected that it has the same meaning (20). Porter stemmer is a widely used stemming algorithm proposed by (30) with proven results but does not work every time and is widely volatile in its impact on the classification task. Using the same example as the tokenized text above, stemming results can be seen below (Figure 7).

Stemmed sample

*[...] 'venh', 'por', 'est', 'mei', 'demonstr', 'o', 'meu', 'desagr', 'e',
'a', 'minh', 'indign', 'pel', 'serviç', 'prest', 'pel', 'conform' [...]*

Figure 7 – Example of a stemmed sentence sample.

This process is much more destructive than lemmatizing and the loss of information is higher. It is also much harder to understand the lexical context in which words occur.

Lemmatization is a process that resembles stemming but instead of producing a stem it reduces verbs to their infinite tense and nouns to their singular form. In some cases, lemmatized or stemmed words can end up reduced to the same word (104). Despite this, lemmatization is, on the other hand, a much more expensive process to conduct as the algorithm is not as brute as the stemmer. It allows for much better context understanding and lexical analysis. An example, using the tokenized text above, is represented below (Figure 8).

Lemmatized sample

*[...] 'vir', 'por', 'este', 'meu', 'demonstrar', 'o', 'meu',
'desagradar', 'e', 'o', 'meu', 'indignação', 'pelos', 'serviço',
'prestar', 'pelar', 'conformar' [...]*

Figure 8 – Example of a lemmatized sentence sample.

An overview of the preprocessing methodology applied in this dissertation can be seen in Figure 9.

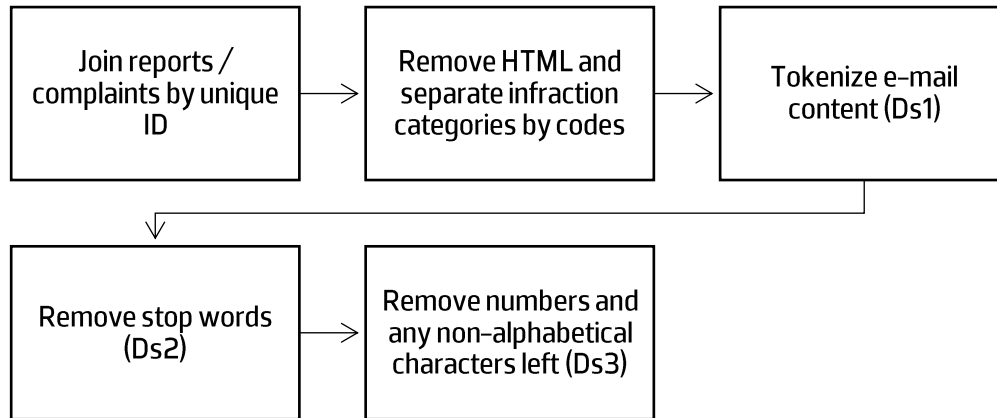


Figure 9 – Diagram representing the preprocessing steps performed on the dataset.

In a second instance of this dissertation we report the results from applying different well-known word normalization methodologies, stemming and lemmatization. Following the same procedures as the previous topic, we evaluate the average weighted accuracy for each classifier and the respective training times associated with each procedure. The results obtained are then used to perform the next analysis – feature extraction and feature selection.

3.2.2. Feature extraction, feature selection and evaluation

There are different feature selection methods as described in the state-of-the-art chapter and each has its own perks. As a first analysis, we establish comparisons between the simpler term frequency method (TF) and the term frequency-inverse document frequency (TF-IDF). These TF counts were not normalized unlike the TF implementation in the feature selection section. The Chi-square method of feature selection and ANOVA feature selection are also taken in consideration for the task of feature selection as these can help reduce dimensionality, find terms which are better representations of the labels, while maintaining performance.

The algorithms used during these tests were not optimized for the learning tasks as they were used with their default parameters. In some cases, an exhaustive search for the best set of parameters can lead to an increase in prediction capabilities and thus a hyper-parameter tuning section can help find more adequate parameters for the classifiers tested. In fact, this is the case of support vector machines where an optimization of the *Coefficient (C)* parameter leads to different outcomes of hyperplane's margins and fine tweaking can lead to performance increase (105). The SVM tries to find the higher margin that separates the data, thus when a higher margin is chosen, it incurs in a potentially higher misclassification rate. Balanced is needed between separation of data and misclassification rate. A larger C parameter value leads to a smaller margin and a lower C parameter value leads to higher margin.

For this analysis, we performed several split analyses regarding each parameter chosen to be optimized. This methodology was subjected to alterations according to the observed results related to performance and computational expense. We have not done an exhaustive grid search with parameter combinations on this step of the dissertation as it was a computational expensive process.

The selection of algorithms to use as a final analysis was derived from the results obtained throughout the dissertation and their performance was evaluated on all four performance metrics mentioned: accuracy, precision, recall and f1-score. We also report the findings shown on the confusion matrices of each classifier selected. A sum up of these procedures can be seen on Figure 10.

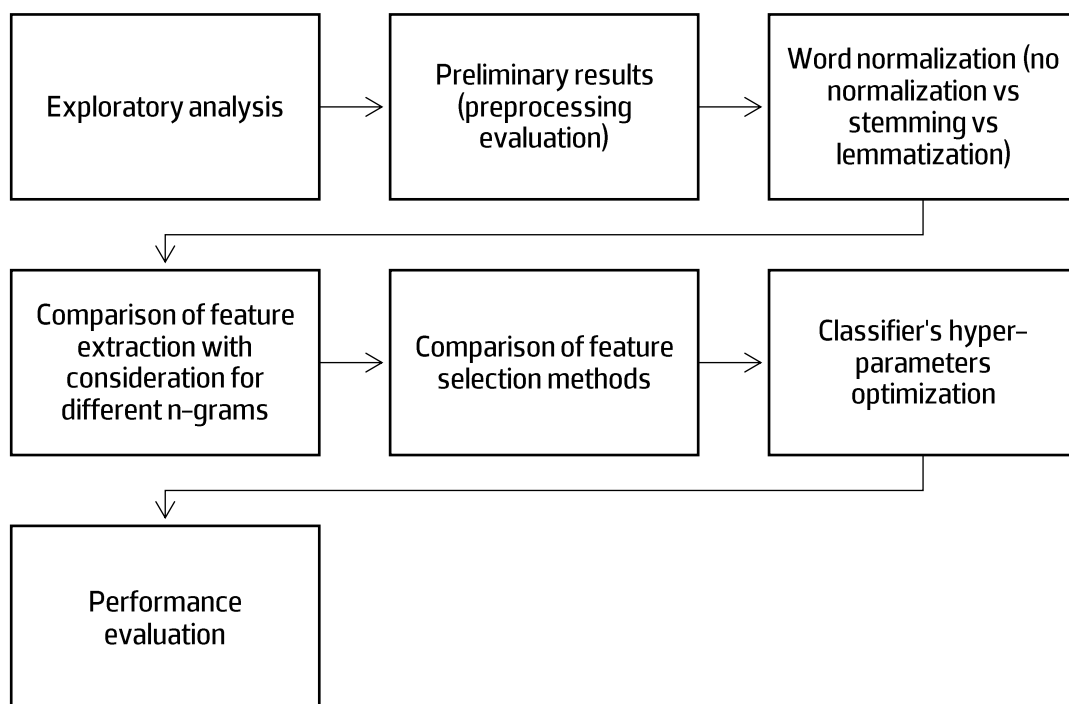


Figure 10 – Diagram representing the procedures involved in the classification task.

There are some drawbacks to analysis performed in this work. On one hand, the system on which the tests are run is not the most adequate for large scale analysis and thus some simplifications had to be made regarding some steps of the work. On the other hand, we have already mentioned the possibility to use other word representation methods such as the tensor model and the word2vec representation of words and thus this could possibly lead to better results. In addition, no attempts were made to understand semantical connection between words and their true meaning in the context on which they were written. As such, this could possibly lead to other interesting results as well.

The results outlined in this dissertation were performed on a computer with an AMD Ryzen 5 1600 Six-core processor @ 3.7Ghz and 16GB of RAM, running Fedora 29 / 30.

4. Results and Discussion

In the results chapter we address outcomes obtained from different analysis detailed on the methodology chapter. We explore the data in the first sub-section where we analyze how the data is distributed and draw conclusions from it. We also tackle the performance of the algorithms when using different preprocessing methods such as stemming and lemmatization in addition to the impact of stop words, feature scaling and feature reduction. The final sections present results regarding the best performing algorithms and their respective confusion matrices when used on the jurisdiction and infractions labels.

4.1. Exploratory Analysis

The retrieved dataset contained 104116 with most of the text entries having Hyper Text Markup Language or unreadable characters in its midst. Both add unneeded dimensionality and no valuable information and, as such, were removed from the text content. In addition, some of the entries did not contain any valuable information and were also removed along with any text labeled as “undetermined” since this class is not actually labeling anything, unlike the others. Out of the total entries, 24017 complains had empty strings and 224 jurisdiction labels were empty and 9417 were “undetermined” labels. Only 70458 samples remained after this process.

The sample distribution related to the number of entries per jurisdiction label can be seen on Table 3.

Table 3 – Sample distribution per label (jurisdiction). The sample distribution is measured in the number of documents each label has assigned.

Jurisdiction	Number of Documents
ASAE	33207
Ano Entity	16369
ASAE & Ano Entity	14306
Courts	4672
ASAE & Courts	1337
Ano Entity & Courts	567

From the table above (Table 3), an imbalance in the dataset can be observed regarding the number of complains per category. The ASAE group contains almost 50% of the distribution, more precisely, 47,13%. Furthermore, the labels that are directly or indirectly related to “Courts” are shown to have the lowest number of complains in the data with only 9,33 % representation.

To further understand how the data is distributed, the lengths of each complain per label were calculated and can be seen on the box-plot below (Figure 11).

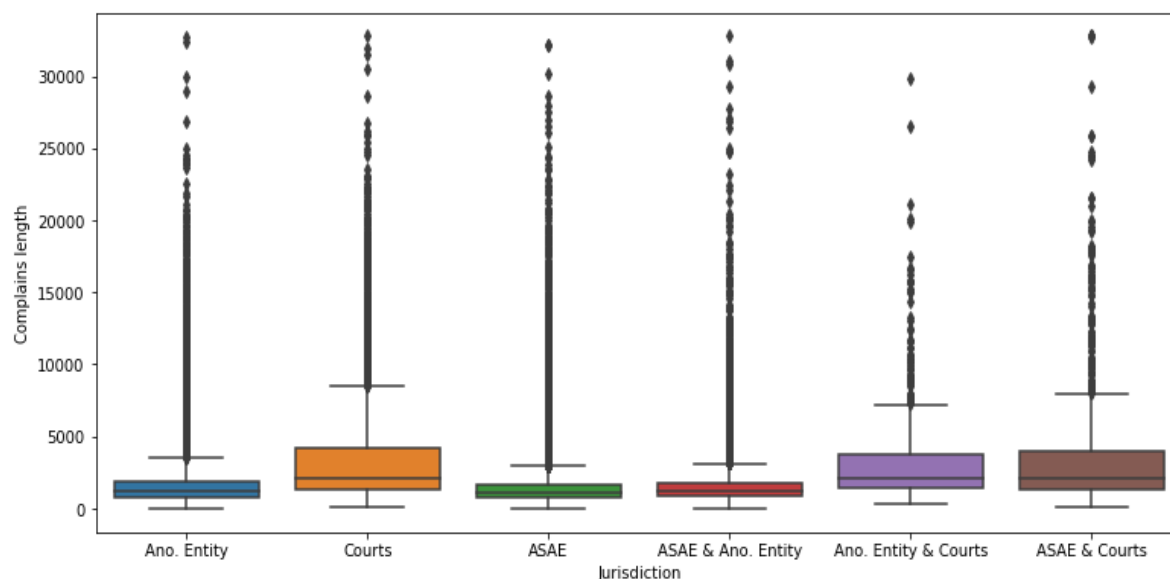


Figure 11 – Boxplot representing document's length for each jurisdiction label.

Even though there is a high presence of outliers towards the higher end of the spectrum related to the length of complains, the average does not fluctuate much between categories. The table below (Table 4) summarizes some of the important descriptive statistics regarding the complains length. This table shows the standard deviation between label lengths, the minimum and maximum length and the average lengths for each percentile (25, 50 and 75).

Table 4 – Descriptive metrics for documents length regarding the jurisdiction attribute.

Mean	1855.11
Std	2263.48
Min	0.00
Percentile 25	806.00
Percentile 50	1160.00
Percentile 75	1892.00
Max	32767.00

The mean value of 1855.11 can be influenced by the presence of outliers, which in this case have a maximum value of 32767. Since the discrepancy is quite high, looking at how 95% of the values are distributed can help verify if there is any clear variance in lengths. This is seen on Figure 12.

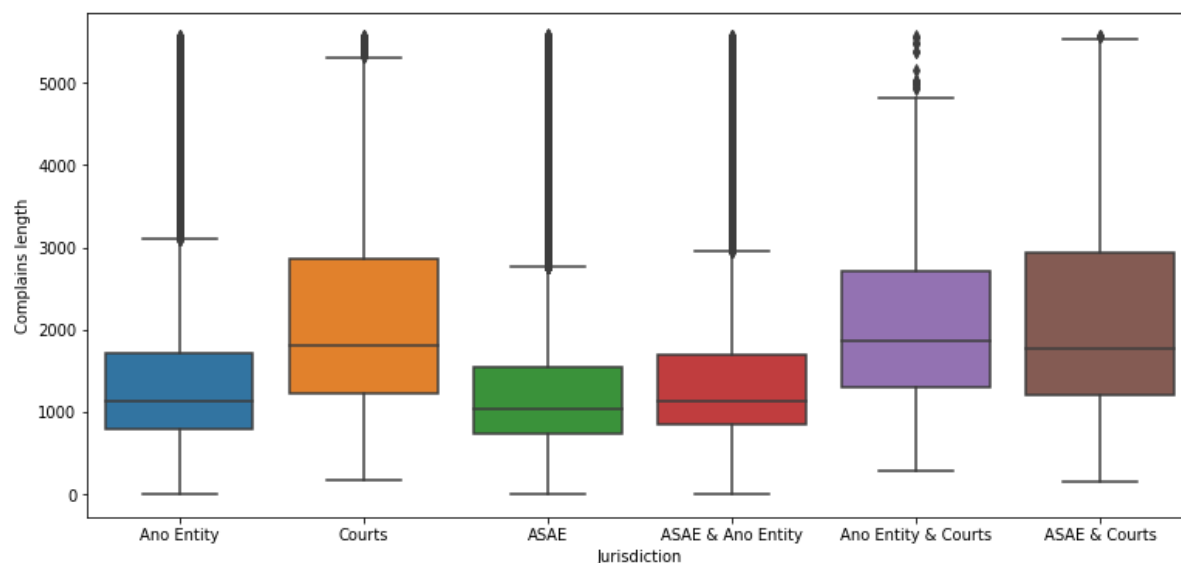


Figure 12 – Boxplot representing documents length for the 95 quantiles regarding each jurisdiction.

There is a higher length count for “Courts” related jurisdiction, but the difference does not seem to be substantial between the “another entity”, “ASAE” and “ASAE and another entity” labels. The labels “courts”, “another entity & courts” and “ASAE & courts” have the highest number of characters per complain but are still balanced between the three. Figure 13 indicates most of the documents length sits around 1000 characters and upwards for 95% of the documents.

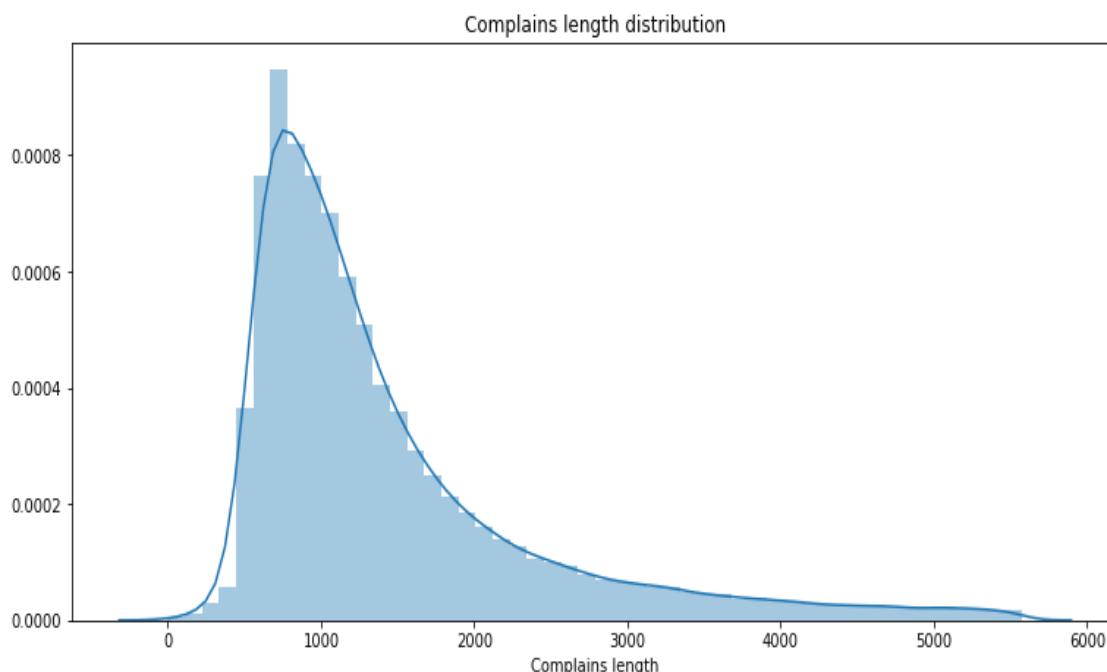


Figure 13 – Distribution of documents length for the entire document. The length is measured in characters and not words.

A representation of the most common words can be seen on the table below Table 5 which shows the topmost frequent 15 bigrams. It clearly shows that most of the frequent words are related to headers on the content such as “Zip-code”, “Economic Agent”, “E-mail”. In fact, the presence of these words can be a problem for the classifiers as these are substantially present on the dataset.

Table 5 – Most frequent bigrams for each jurisdiction label. Only the top 15 bigrams are represented (from 0 to 14).

	ASAE	Count	SAE & Ano. Entity	Count	Ano. Entity	Count	Courts	Count	ASAE & Courts	Count	Ano. Entity & Courts	Count
0	(E, mail)	47482	(E, mail)	23012	(E, mail)	23664	(que, o)	5957	(E, mail)	1667	(E, mail)	686
1	(Código, Postal)	46848	(Identificação, do)	22980	(Identificação, do)	23314	(que, não)	5464	(Identificação, do)	1652	(Identificação, do)	672
2	(Identificação, do)	46840	(Código, Postal)	22953	(Código, Postal)	23309	(E, mail)	5320	(Código, Postal)	1643	(Código, Postal)	672
3	(mail, Identificação)	33905	(mail, Identificação)	18311	(mail, Identificação)	16164	(Identificação, do)	5115	(que, o)	1486	(que, não)	650
4	(Telefone, E)	31617	(Telefone, E)	17124	(Telefone, E)	15081	(Código, Postal)	5084	(que, não)	1403	(que, o)	643
5	(do, Agente)	23410	(Agente, Económico)	11527	(Agente, Económico)	11695	(que, a)	5079	(que, a)	1303	(que, me)	571
6	(Agente, Económico)	23408	(do, Agente)	11519	(do, Agente)	11666	(que, me)	4490	(com, a)	1282	(e, que)	564
7	(Identificação, dos)	23395	(Económico, visado)	11492	(Identificação, dos)	11661	(com, o)	4412	(com, o)	1216	(com, a)	547
8	(Económico, visado)	23355	(Identificação, dos)	11485	(Económico, visado)	11647	(com, a)	4350	(que, me)	1184	(com, o)	538
9	(dos, Factos)	23350	(Queixoso, Denunciante)	11476	(dos, Factos)	11637	(gmail, com)	4161	(gmail, com)	1157	(que, a)	522
10	(Queixoso, Denunciante)	23338	(dos, Factos)	11473	(Queixoso, Denunciante)	11635	(e, que)	3700	(e, que)	1039	(gmail, com)	470
11	(visado, Nome)	23318	(do, Queixoso)	11464	(do, Queixoso)	11628	(o, que)	3352	(no, dia)	1027	(o, que)	429
12	(Denunciante, Nome)	23300	(visado, Nome)	11460	(das, Testemunhas)	11620	(para, a)	3205	(de, 2015)	972	(no, dia)	362
13	(do, Queixoso)	23298	(Factos, Data)	11459	(Identificação, das)	11618	(para, o)	3171	(o, que)	960	(que, se)	348
14	(Factos, Data)	23286	(Identificação, das)	11459	(visado, Nome)	11600	(no, dia)	3137	(para, a)	937	(Agente, Económico)	345

Most of the frequent bi-grams are either related to headers, like the results shown by the Word Cloud above, or are stop-words. Even though the headers add no value to the analysis, since they are so prevalent, the feature selection implementations should relegate them to a lower weight compared to more relevant terms, thus not impacting the classifiers in a negative fashion.

The infractions dataset totaled 42670 entries, a lower count than the jurisdiction dataset before. This difference is mainly due to the high amount of missing values present on the infractions attribute and were removed because they would not be used in the classification task. In addition, this attribute contained numerous amounts of labels describing the infraction that has taken place or is vided by the complaint. These infractions are code-labeled, and more than one can be attributed to the same complaint which turns it into a multi-label problem. In order to apply the classifiers we have tested so far, a multi-class approach was necessary and thus each category was grouped under an overruling category numbered 1 through 4, here represented as I, II, III and IV, as was mentioned in the methodology chapter.

This dataset’s sample distribution is shown on Table 6 . The majority of the samples regard infraction category I and II with 20954 and 19176 samples, respectively. Category IV is grossly under-represented with 103 samples, but category III also has a low population count at 2437 samples.

Table 6 – Sample distribution per label (infraction). The sample distribution is measured in the number of documents each label has assigned.

Infraction category	Number of samples
Category I	20954
Category II	19176
Category III	2437
Category IV	103

In a similar fashion as what was presented before, Figure 14 also represents the document's length for each infraction category in the dataset. It is also worth noting that the same behavior was observed with the jurisdiction dataset where most labels have similar densities of character lengths.

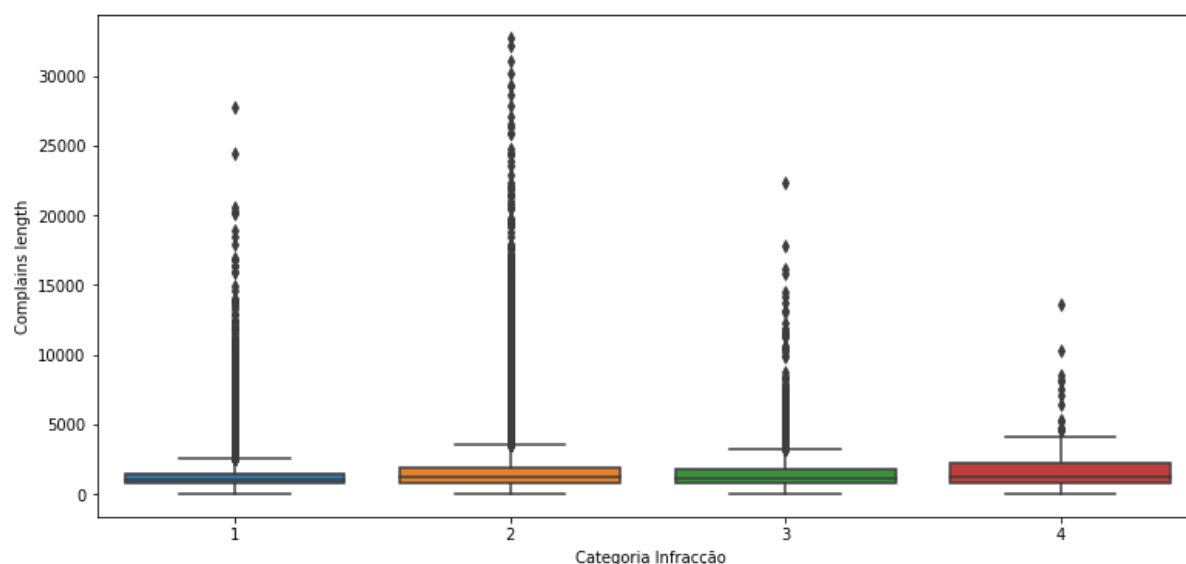


Figure 14 – Boxplot representing documents length for each infraction category.

In addition, a high number of outliers can be seen for infractions 1 and 2 while infractions 3 and 4 have fewer outlying values. The following table (Table 7) contains information regarding complaint's length and gives us some important descriptive metrics. This table shows the standard deviation between label lengths, the minimum and maximum length and the average lengths for each percentile (25, 50 and 75).

Table 7 – Descriptive metrics for documents length regarding the infractions attribute.

Mean	1575.04
Std	1723.30
Min	0.00
Percentile 25	776.00
Percentile 50	1083.00
Percentile 75	1659.00
Max	32767.00

Since the e-mail content is the same, the average values for document's length do not change much when compared to the ones obtained in the jurisdiction dataset. There is a slight loss in lengths that is most likely attributed to the lower number of samples. The standard deviation value remains in-line to what was obtained before despite being slightly lower. It was observed that 75% of the observations were up to 1659 characters long.

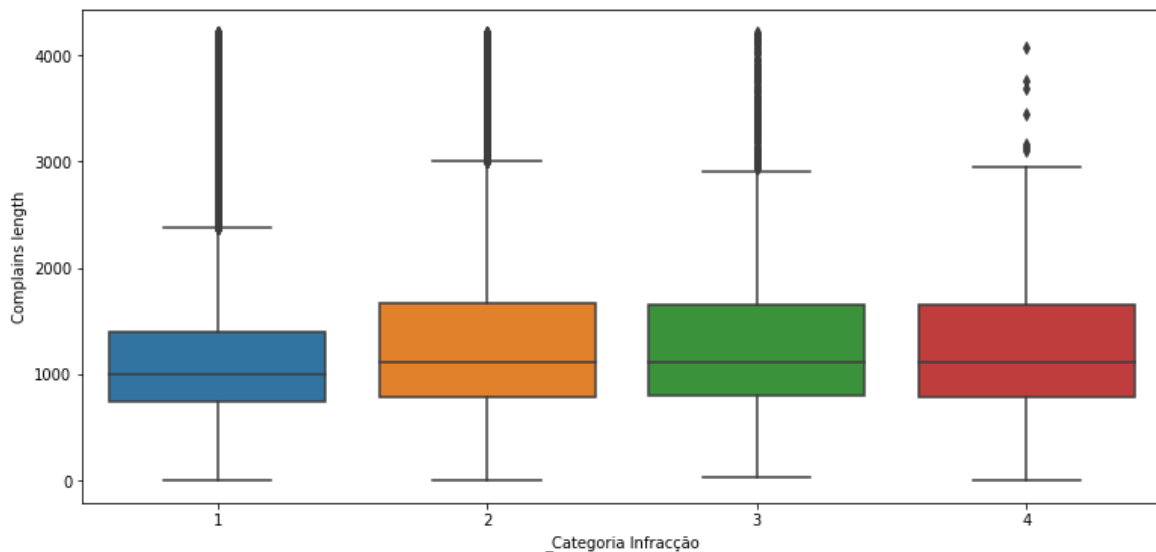


Figure 15 – Boxplot representing documents length for the 95 quantiles regarding each infraction category.

As seen on Figure 15, 95% of the distribution is relatively similar between the 4 categories. There is a slight increase in length between category and categories 2, 3 and 4 but it is not meaningful. Over the four categories, 50% of the distribution sits at roughly 1000 characters long and is only slightly lower than the previous dataset.

4.2. Preliminary Results about Jurisdiction and Infractions

In this sub-section of chapter 4 we present the preliminary results for both datasets, the jurisdiction and the infractions dataset. The results are divided in two different measures: mean weighted test set accuracy and training times. As was previously mentioned on the methodology chapter we use 9 different classifiers: MNB, CNB, BNB, SVM (linear), K-NN, DT, RF, AB and LR. The “e-mail content” label where all the textual information is stored was split into three different labels according to the processing methodology applied:

- Ds1: Original text obtained after HTML removal;
- Ds2: Original text but with stop words removed;
- Ds3: Fully processed text.

Ds2 contains the original text like Ds1 but without stop words while Ds3 has no stop words as well as no numbers or any odd entity (only words). The same coding applies to both datasets as the textual content is shared with the only variation being the number of entries in each dataset and the target labels for each document. The following results were obtained with 10-fold stratified cross-validation and each value is the average weighted accuracy obtained at each split. We also used term frequency without normalization as previously mentioned, with consideration for unigrams only.

Table 8 – Average weighted accuracy for each classifier with 10-fold stratified cross-validation for the jurisdiction dataset with no preprocessing (Ds1), dataset with only stop words removed (Ds2) and fully processed dataset(Ds3).

Classifier	Ds1	Ds2	Ds3
Multinomial NB	0.526	0.527	0.522
Bernoulli NB	0.541	0.553	0.556
Complement NB	0.568	0.571	0.574
Linear Support Vector	0.595	0.591	0.598
Logistic Regression	0.598	0.590	0.613
K-Neighbors	0.490	0.480	0.486
Decision Tree	0.506	0.517	0.553
Random Forest	0.529	0.546	0.566
AdaBoost	0.520	0.506	0.537

As seen on Table 8, the classifiers employed achieved better results in the Ds3 dataset with the exception of MNB and K-NN. MNB achieved the best result of 52.7% in the Ds2 dataset while being slightly less accurate on the Ds3 dataset while K-NN achieved 49% on the Ds1 dataset compared to 48.6% on the Ds3 dataset. It was also observed that the DT algorithm had the highest increase in accuracy with the lowest value of 50.6% in the Ds1 dataset and 55.3% in Ds3. RF achieved roughly 2% increase between each

dataset, totaling 56.6% in Ds3 (overall increase of approximately 4%). In addition, significant gains were also observed with the LR classifier between the two less preprocessed datasets (Ds1 and Ds2) and the full preprocessed dataset (Ds3).

The following table (Table 9) contains the average fit times in seconds for each classifier regarding each of the three different datasets.

Table 9 – Average training times in seconds for each classifier with 10-fold stratified cross-validation for the jurisdiction dataset with no preprocessing (Ds1), dataset with only stop words removed (Ds2) and fully processed dataset(Ds3).

Classifier	Ds1	Ds2	Ds3
Multinomial NB	15.556	10.222	7.679
Bernoulli NB	16.285	10.407	7.858
Complement NB	16.410	10.318	7.728
Linear Support Vector	373.750	254.856	194.344
Logistic Regression	302.881	279.433	172.134
K-Neighbors	16.2447	11.047	8.299
Decision Tree	383.952	352.851	249.823
Random Forest	125.861	127.682	101.344
AdaBoost	94.322	66.954	51.947

It was observed that the overall training times reduced with further processing of the text. Random forest is the exception to this observation as it had a higher training time with the second dataset (Ds2) although results for Ds3 are lower than the previous two (Ds1 and Ds2). It was also observed that the classifier that took longer to run was DT with 249.82 seconds to run on average at each split, followed by SVM (linear) with 194.34 seconds and LR with 172.13 seconds. The Naïve Bayes classifiers were much faster to train than the rest of the algorithms at approximately 7 seconds followed by K-NN at approximately 8 seconds. The table below (Table 10) shows the results with the procedures applied above but for the infractions dataset.

Table 10 – Average weighted accuracy for each classifier with 10-fold stratified cross-validation for the infractions dataset with no preprocessing (Ds1), dataset with only stop words removed (Ds2) and fully processed dataset (Ds3).

Classifier	Ds1	Ds2	Ds3
Multinomial NB	0.715	0.723	0.735
Bernoulli NB	0.680	0.686	0.696
Complement NB	0.715	0.723	0.732
Linear Support Vector	0.808	0.808	0.809
Logistic Regression	0.814	0.814	0.820
K-Neighbors	0.635	0.646	0.656
Decision Tree	0.757	0.763	0.763
Random Forest	0.761	0.781	0.789
AdaBoost	0.713	0.720	0.721

From Table 10 it was observed that, much like what was previously seen, the majority of the classifiers performed better with the Ds3 dataset. The exception to this was DT with 76.3% average weighted accuracy value with Ds2 and, once again, 76.3% with Ds3. Even though accuracy increases across the board, it is not as expressive in some cases as before.

An overall increase in accuracy was also observed when compared to the dataset analyzed before, in fact, LR reached as much as 82.0% accuracy, closely followed by the SVM (linear) and RF with 80.9% and 78.9%, respectively.

Table 11 shows the results regarding average training times for each classifier tested, in seconds.

Table 11 – Average training times in seconds for each classifier with 10-fold stratified cross-validation for the infractions dataset with no preprocessing (Ds1), dataset with only stop words removed (Ds2) and fully processed dataset(Ds3).

Classifier	Ds1	Ds2	Ds3
Multinomial NB	8.098	5.496	4.245
Bernoulli NB	8.132	5.519	4.271
Complement NB	8.265	5.459	4.144
Linear Support Vector	83.801	63.702	51.345
Logistic Regression	91.594	77.496	57.082
K-Neighbors	8.531	5.757	4.376
Decision Tree	116.919	98.523	72.971
Random Forest	38.059	36.730	30.143
AdaBoost	39.765	30.594	22.378

As seen on Table 11, average fit times for the classifiers tested are much lower than those observed on the previous iteration of these tests as this dataset is about half the size of the previous one. The same behavior was observed as was previously seen on Table 9 where there was a steep drop in training times with further text processing. In addition, DT took the longest to train (72.971 seconds) in Ds3, followed by LR which now took longer than the SVM (linear) with 57.083 seconds and 51.345 seconds, respectively.

4.2.1. Discussion on preliminary results

We observed that in both datasets, a fully preprocessed text content such as stop word removal, punctuation, special characters lead to, generally, an increase in accuracy for each classifier. It was also observed that for the most cases, stop words removal (Ds2) alone was also able to improve the models tested. The authors of (88) mention that a simple use of stop words removal leads to a positive impact on a classification task on a Portuguese dataset as was observed in our datasets. In addition, the datasets processed in full were also shown to improve the classifiers training time as the number of features is reduced by reducing the redundancy in the data as well as the noise such as single characters and punctuation.

These preliminary results also demonstrate that the more complex algorithms such as the SVM and LR (linear classifiers) achieved better results than the simpler classifiers such as the ones from the Naïve Bayes group (MNB, CNB and BNB). In the first dataset, CNB was observed to perform better than the MNB counterpart as it allowed for a better prediction. In fact, this behavior was mentioned earlier in the work of (53), although the same behavior was not observed for the infractions dataset. This difference is possibly due to the target label and the quality of the features associated as well as the lower number of samples in the infraction's dataset.

Furthermore, K-NN, DT and AB performed below the other classifiers tested in both datasets except for a single case in the jurisdiction dataset related to the lower performance of MNB.

In the following experimentations, we use the Ds3 dataset as our dataset of choice as it allows for faster train times than the non-preprocessed dataset while achieving slightly better results in both datasets and labels tested.

4.3. Word Normalization

In this sub-section we present the results regarding the use of stemming and lemmatization, two well-known word normalization methods. We establish comparisons between the previous Ds3 dataset and the following:

- Ds3-S: Text was stemmed over the previous Ds3 dataset;
- Ds3-L: Text was lemmatized over the previous Ds3 dataset.

The same 10-fold stratified cross-validation was employed with the feature selection method being term frequency for unigrams. We show the average weighted accuracy results as well as the difference in training times when using the different datasets.

Table 12 shows the results for the jurisdiction dataset without word normalization (Ds3), with stemming (Ds3-S) and with lemmatization (Ds3-L).

Table 12 – Average weighted accuracy for each classifier with 10-fold stratified cross-validation for the jurisdiction dataset with a fully preprocessed dataset (Ds3), stemmed dataset(Ds3-S) and lemmatized dataset (Ds3-L).

Classifier	Ds3	Ds3-S	Ds3-L
Multinomial NB	0.521	0.497	0.511
Bernoulli NB	0.555	0.538	0.550
Complement NB	0.573	0.561	0.569
Linear Support Vector	0.598	0.588	0.592
Logistic Regression	0.613	0.617	0.622
K-Neighbors	0.486	0.509	0.502
Decision Tree	0.552	0.545	0.550
Random Forest	0.565	0.554	0.558
AdaBoost	0.536	0.555	0.548

It was observed stemming the words (Ds3-S) only performed better over no normalization (Ds3) with LR (Ds3 – 61.3%; Ds3-S – 61.7%), AB (Ds3 – 53.6%; Ds3-S – 55.5%) and K-NN(Ds3 – 48.6%; Ds3-S – 50.9%). The classifiers belonging to the Naïve Bayes group performed better without any word normalization when compared to both stemming and lemmatization. LR and RF performed better with lemmatization over both stemming and no normalization. In addition, the SVM (linear) performed worse with stemming and lemmatization with the difference in accuracy between the lemmatized dataset (59.2%) and the non-normalized dataset (59.8%) being roughly 0.6%.

Table 13 – Average training times in seconds for each classifier with 10-fold stratified cross-validation for the jurisdiction dataset with a fully preprocessed dataset (Ds3), stemmed dataset(Ds3-S) and lemmatized dataset (Ds3-L).

Classifier	Ds3	Ds3-S	Ds3-L
Multinomial NB	7.679	6.667	7.044
Bernoulli NB	7.858	6.813	7.080
Complement NB	7.728	6.642	7.003
Linear Support Vector	194.344	198.018	191.123
Logistic Regression	172.134	116.703	138.414
K-Neighbors	8.299	7.028	7.392
Decision Tree	249.823	183.442	205.345
Random Forest	101.344	67.978	81.253
AdaBoost	51.947	38.409	42.123

Table 13 shows the results for the training time for each classifier tested on the three different normalized datasets. The dataset containing stemmed samples shows the lowest training times of the three except for the SVM which has a higher training time when compared to the other two datasets. The difference between Ds3, Ds3-S and Ds3-L for the Naïve Bayes algorithms is not significant as the difference is only of roughly 1% between Ds3 and Ds3-S and MNB, BNB and CNB already achieve incredibly fast training times. In addition, the same applies for the k-NN classifier which took 8.2 seconds to train for the Ds3 dataset and 7.0 for the Ds3-S, the lowest value for the classifier. The other algorithms except for the SVM were observed to be positively impacted by word normalization in a much higher degree than the ones mentioned before. LR decreased fit times in about 55 seconds from 172.1 seconds (Ds3) to 116.7 seconds (Ds3-S). The same was observed for the DT where train times drop significantly from 249.8 seconds to 183.4 seconds with stemming.

Table 14 – Average weighted accuracy for each classifier with 10-fold stratified cross-validation for the infractions dataset with a fully preprocessed dataset (Ds3), stemmed dataset(Ds3-S) and lemmatized dataset (Ds3-L).

Classifier	Ds3	Ds3-S	Ds3-L
Multinomial NB	0.735	0.733	0.737
Bernoulli NB	0.696	0.696	0.700
Complement NB	0.732	0.728	0.733
Linear Support Vector	0.809	0.801	0.804
Logistic Regression	0.820	0.824	0.823
K-Neighbors	0.656	0.689	0.678
Decision Tree	0.763	0.762	0.771
Random Forest	0.789	0.780	0.784
AdaBoost	0.721	0.731	0.733

The results for the infractions dataset with different normalization methods are shown on Table 14. A slight overall increase was observed when using lemmatization over the non-normalized dataset but for the SVM and RF. The use of stemmed samples increases accuracy for the K-NN, AB and LR classifiers with K-NN and LR being the only classifiers which achieved better performances with stemming when compared to the other two datasets. The SVM and RF achieved higher values in accuracy with the use of the Ds3 dataset reaching 80.9% and 78.9%, respectively.

The following table (Table 15) shows the results for the infractions dataset regarding average fit times of each classifier.

Table 15 – Average training time in seconds for each classifier with 10-fold stratified cross-validation for the jurisdiction dataset with a fully preprocessed dataset (Ds3), stemmed dataset(Ds3-S) and lemmatized dataset (Ds3-L).

Classifier	Ds3	Ds3-S	Ds3-L
Multinomial NB	4.245	3.888	4.297
Bernoulli NB	4.271	4.054	4.584
Complement NB	4.144	3.626	4.340
Linear Support Vector	51.345	50.916	55.284
Logistic Regression	57.082	42.879	55.627
K-Neighbors	4.376	3.779	4.515
Decision Tree	72.971	67.803	65.650
Random Forest	30.143	27.983	26.092
AdaBoost	22.378	23.868	19.966

It was observed that the results were in-line with those obtained before for the jurisdiction dataset. Stemming the features still proves to be the most efficient way to handle the data timewise. In some cases, such as the Naïve Bayes variants, K-NN and SVM, lemmatization proves to be less efficient than using the dataset which has not been normalized. The differences observed are much less evident than those obtained for the jurisdiction dataset, likely due to the smaller size of the dataset.

4.3.1. Discussion on word normalization

We observed that the accuracy values for the algorithms tested did not differ significantly whether stemming or lemmatization was applied or not. In fact, for the jurisdiction dataset we observed that stemming samples had a negative impact on accuracy for all but AB, K-NN and LR. Lemmatization also lowered accuracy values across the board but did allow the algorithms to perform better when compared to stemming. However, the second dataset (infractions related) shows better results for lemmatization than using no normalization at all. When compared to stemming, lemmatization also increases accuracy for all but for two classifiers, the K-NN and LR.

Stemming is a much more penalizing process when it comes to word normalization being much more destructive than lemmatization. While we have not looked at the semantics involved, this behavior can help understand the negative impact observed by applying stemming to the features considered. In addition, this behavior also allowed for a fast processing of the data which means that it possibly had a greater impact on word duplicates than lemmatization. Moreover, since it is much more destructive it can also destroy more information and thus reduce time to train.

The authors of (106) tested different normalization such as the Porter's stemmer, EWN lemmatization as well as Lovins's algorithm, among others while using a naive Bayes classifier. It was concluded that applying word normalizations such as lemmatization or stemming has a negative impact on both morphologically complex and morphologically simple languages as only some cases were observed to suffer a positive increase in macro f1-scores, but these were not statistically significant.

Although we did have worse results in the first dataset when using normalization with Naïve Bayes, the same did not happen for the infractions dataset and thus it is hard to generalize any meaningful conclusion. It is also worth noting that this dataset's language is Portuguese and thus strict comparisons are harder to establish.

4.4. Feature Extraction

So far, we've only tested the classifiers with TF weighting and the use a different weighting method such as TF-IDF can prove beneficial regarding accuracy while still allowing for a better search of the feature

space. In this sub-section we show the results obtained from comparing TF with TF-IDF on the lemmatized content with 10-fold stratified cross-validation. We also evaluate the use of n-gram ranges which so far have been only unigrams on the selected feature extraction method obtained from the best set of results from their comparison. We test the following ranges: unigrams (1,1), bigrams (2,2), trigrams (3,3), unigrams and bigrams (1,2) and unigram, bigrams and trigrams (1,3).

Table 16 – Average weighted accuracy for TF and TF-IDF feature extraction with unigrams for the jurisdiction dataset.

Classifier	TF	TF-IDF
Multinomial NB	0.511	0.490
Bernoulli NB	0.550	0.550
Complement NB	0.569	0.582
Linear Support Vector	0.593	0.654
Logistic Regression	0.622	0.671
K-Neighbors	0.502	0.449
Decision Tree	0.549	0.538
Random Forest	0.558	0.548
AdaBoost	0.548	0.552

Table 16 shows the results from comparing the use of TF-IDF and TF on the jurisdiction dataset. It was observed that the linear classifiers such as LR and SVM outperformed the TF implementation of feature extraction. SVM achieved a much higher accuracy value of 65.4% with TF-IDF while with TF it only reached 59.3%. LR also suffered a substantial increase from 62.2% with TF to 67.1% with TF-IDF. CNB and AB also suffered a slight increase in accuracy values but not as expressive as LR or SVM. In contrast to the much higher accuracy values for the SVM and LR, K-NN achieved a much lower accuracy value (44.9%) with TF-IDF while attaining 50.2% with TF. In addition, DT and RF also performed slightly better with TF than TF-IDF. Also, BNB did not show any difference from the two feature extraction methods as it merely uses a binary system to count word occurrences.

Table 17 – Average weighted accuracy for TF and TF-IDF feature extraction with unigrams for the infractions dataset.

Classifier	TF	TF-IDF
Multinomial NB	0.737	0.779
Bernoulli NB	0.700	0.700
Complement NB	0.733	0.786
Linear Support Vector	0.804	0.845
Logistic Regression	0.823	0.846
K-Neighbors	0.678	0.5419
Decision Tree	0.768	0.758
Random Forest	0.784	0.777
AdaBoost	0.733	0.720

As seen on Table 17, the same behavior for the infractions dataset was generally observed as with the jurisdiction dataset with the exception of AB and MNB which had their results reversed. Moreover, CNB and MNB had a much higher increase in accuracy from 73.3% to 78.6% with TF-IDF while the SVM and LR had a smaller increase but still significant. Outside of the results best obtained with TF-IDF, K-NN had a significant decrease in accuracy when paired with TF-IDF of roughly 13% while DT, RF and AB did show lower results for TF-IDF but less impactful.

Once again, BNB shows no difference from the two methodologies tested.

The next results relate to the observations for each unigram range (unigrams through trigrams and with combinations between them) with TF-IDF weighting. TF-IDF was chosen due to the results obtained before, particularly because it significantly improved results for LR, SVM and lastly CNB.

Table 18 – Average weighed accuracy for each classifier with TF-IDF feature extraction considering unigrams, bigrams and trigrams, the combination of unigrams and bigrams and the combination of unigrams, bigrams and trigrams, applied to the jurisdiction dataset.

Classifier	Unigrams	Bigrams	Trigrams	Unigrams & Bigrams	Unigrams, Bigrams & Trigrams
Multinomial NB	0.490	0.478	0.481	0.473	0.473
Bernoulli NB	0.550	0.484	0.474	0.494	0.477
Complement NB	0.582	0.525	0.474	0.487	0.483
Linear Support Vector	0.654	0.629	0.533	0.675	0.675
Logistic Regression	0.671	0.581	0.495	0.666	0.656
K-Neighbors	0.449	0.443	0.471	0.443	0.443
Decision Tree	0.539	0.470	0.428	0.531	0.443
Random Forest	0.546	0.511	0.477	0.531	0.516
AdaBoost	0.552	0.495	0.477	0.539	0.542

As seen on Table 18 most of the classifiers were observed to perform better with unigrams only, except for K-NN which performed better with trigrams and the SVM which performed better with unigrams and bigrams. Moreover, the SVM was observed to have an almost identical score with unigrams through trigrams (67.5%) when compared to unigrams through bigrams (67.5%). In addition, K-NN's accuracy of 47.1% with trigrams was roughly 3% above all the other ranges tested. Furthermore, despite LR achieving the best score with unigrams, unigrams through bigrams (66.6%) and unigrams through trigrams (65.6%) also show much higher scores than the bigram and trigram only implementation (58.1% and 49.5%, respectively).

In fact, for the majority of the classifiers, increasing the complexity by adding more n-gram ranges to the analysis decreases performance significantly for most cases.

Table 19 – Average train times for each classifier with TF-IDF feature extraction considering unigrams, bigrams and trigrams, the combination of unigrams and bigrams and the combination of unigrams, bigrams and trigrams, applied to the jurisdiction dataset.

Classifier	Unigrams	Bigrams	Trigrams	Unigrams & Bigrams	Unigrams, Bigrams & Trigrams
Multinomial NB	7.534	29.720	45.885	36.784	104.343
Bernoulli NB	7.417	29.970	46.262	37.328	119.498
Complement NB	7.369	29.997	46.262	36.821	121.549
Linear Support Vector	19.976	94.247	137.342	96.080	210.932
Logistic Regression	73.136	243.417	297.010	402.478	636.604
K-Neighbors	7.717	31.724	55.494	37.475	194.348
Decision Tree	196.357	3025.598	7563.827	2311.630	7327.509
Random Forest	62.773	3025.598	2340.128	705.633	1672.666
AdaBoost	62.289	212.132	316.473	293.655	627.436

Increasing the complexity of the analysis inevitably leads to higher training times. This is observed on Table 19 where it is shown that increasing the complexity leads to much higher training times, especially for those algorithms that are more complex. For the best n-gram range (unigrams through bigrams), the SVM took 96.0 seconds to train in average which translates to an increase of 76.1 seconds over unigrams. LR is, so far, the algorithm which has proven to be consistently performing better than the others while being quite slow compared to, for example, the Naïve Bayes variants. This is further expanded when higher feature complexities are added reaching as high as 636.6 seconds for unigrams through trigrams.

Table 20 – Average weighed accuracy for each classifier with TF-IDF feature extraction considering unigrams, bigrams and trigrams, the combination of unigrams and bigrams and the combination of unigrams, bigrams and trigrams, applied to the infractions dataset.

Classifier	Unigrams	Bigrams	Trigrams	Unigrams & Bigrams	Unigrams, Bigrams & Trigrams
Multinomial NB	0.779	0.758	0.676	0.775	0.764
Bernoulli NB	0.700	0.647	0.548	0.673	0.631
Complement NB	0.786	0.771	0.693	0.781	0.769
Linear Support Vector	0.845	0.8116	0.714	0.855	0.852
Logistic Regression	0.846	0.780	0.667	0.837	0.830
K-Neighbors	0.541	0.520	0.503	0.528	0.523
Decision Tree	0.757	0.649	0.567	0.754	0.735
Random Forest	0.774	0.696	0.604	0.750	0.734
AdaBoost	0.720	0.565	0.557	0.711	0.701

Table 20 shows the results from the same tests but applied to the infractions dataset. It is possible to observe that the same behavior explicated in the jurisdiction dataset is almost identical to the infractions' dataset. The SVM, once again, performs better with unigrams and bigrams combined while the rest of the classifiers perform better with unigrams where previously K-NN achieved better results with trigrams. Despite this, the difference between the usage of more complex n-gram selections for the SVM is not expressive and thus only increases slightly when compared to unigrams. Moreover, a general decrease in accuracy is observed with different feature groupings other than unigrams with the combination of unigrams with bigrams and unigrams with bigrams and trigrams performing overall better than bigrams and trigrams separated.

We also present the results for the training times and the impact which different n-grams had on it (Table 21).

Table 21 – Average train times for each classifier with TF-IDF feature extraction considering unigrams, bigrams and trigrams, the combination of unigrams and bigrams and the combination of unigrams, bigrams and trigrams, applied to the infraction dataset.

Classifier	Unigrams	Bigrams	Trigrams	Unigrams & Bigrams	Unigrams, Bigrams & Trigrams
Multinomial NB	7.534	29.720	45.885	36.784	104.343
Bernoulli NB	7.417	29.970	46.262	37.328	119.498
Complement NB	7.369	29.997	46.262	36.821	121.549
Linear Support Vector	19.976	94.247	137.342	96.080	210.932
Logistic Regression	73.136	243.417	297.010	402.478	636.604
K-Neighbors	7.717	31.724	55.494	37.475	194.348
Decision Tree	196.357	3025.598	7563.827	2311.630	7327.509
Random Forest	62.773	3025.598	2340.128	705.633	1672.666
AdaBoost	62.289	212.132	316.473	293.655	627.436

By observing the table above, much like with the previous dataset, it is possible to understand that further increase in complexity leads to a much higher train time for the classifiers, as expected. It was also observed that the training times for the tree-based algorithms (DT and RF) increase in a much more pronounced way than other algorithms, even those that are more complex such as LR and SVM.

4.4.1. Discussion on feature extraction methods

We have evaluated two different feature extraction methods commonly used in the bag-of-words representation, TF with term counts normalized, and TF-IDF. From the results obtained, TF performed better when combined with classifiers like MNB, K-NN, DT and RF. These classifiers are simpler in the way

they handle data separation than those that obtained better performance with TF-IDF, such as SVM, LR, AB. Furthermore, from the two datasets observed it was possible to understand that despite the changes in the target label and the number of samples associated with the dataset, LR, SVM and CNB were consistent in the higher accuracy obtained with TF-IDF. K-NN obtained consistently higher accuracy values with TF as a weighting method with clear gains over TF-IDF, a much more pronounced difference than the other algorithms and thus we conclude that it is beneficial to use TF over TF-IDF although further experiments with different datasets need to be conducted to generalize this information.

In addition to feature extraction methods, we have also evaluated different feature extraction ranges with the use of TF-IDF as our base feature extraction method. This choice was due to the consistently higher values obtained for the classifiers that achieved better performance so far, the LR and SVM. TF-IDF was also chosen because the difference between the classifiers CNB (except for the jurisdiction dataset which CNB performed much better with TF-IDF), RF, DT and AB was not significant when compared to TF. The inclusion of bigrams and trigrams whether in combination or separate did not produce better results for most of the algorithms but for the SVM and K-NN. While K-NN did achieve higher results with trigrams on the jurisdiction dataset, all other n-gram ranges show similar results and thus it is possible that the higher values for that dataset are related to the choice of the samples to train on. Despite the SVM obtained higher accuracy values for unigrams coupled with bigrams with consistency over both datasets, it increases training times by a considerable margin and thus the gains were neglected. Since this behavior is also consistent for all classifiers, the next results were obtained using unigrams only with TF-IDF for the reasons that have been stated before.

Moreover, we have also observed much higher training times with the increase in complexity. The usage of higher n-gram ranges has tremendous impact on the tree-based models such as DT and RF where the computational efficiency drops by a significant margin. This increase in time to run is due to the higher sparsity induced to the text representation matrix. Each document is represented by a vector of words and most documents only contain a limited set of words which effectively makes these matrices quite sparse. This sparsity is then increased when conjunction of words is taken in consideration as more zeroes get added to each vector, extending its length and complexity which in turn leads to much higher processing times.

Furthermore, different feature selection methods were tested on these datasets and are presented on the next section.

4.5. Feature Selection

In this section, we address different feature selection methods and how they impact each classifier's performance using the accuracy metric. We also evaluate different feature ranges from 10% up to 100% in increments of 10. The results are grouped by algorithms and their performance on both datasets.

We test the use of TF with normalization, TF-IDF, ANOVA and Chi-square feature selection methods on each increment.

Figure 16 and Figure 17 show the results for the MNB classifier.

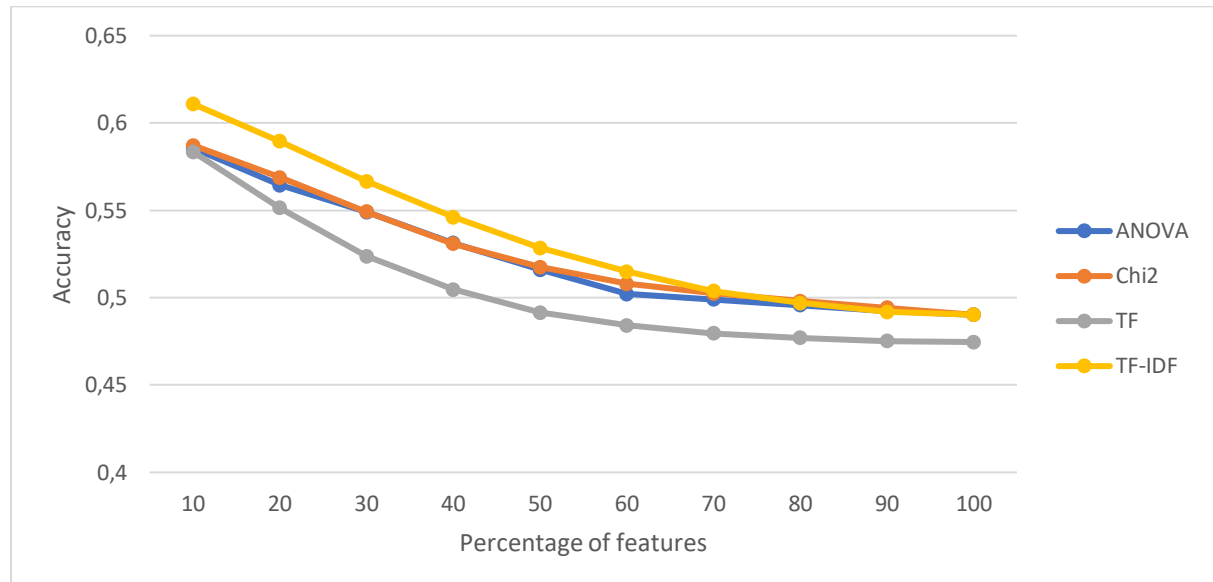


Figure 16 – Plotted accuracy values for MNB using ANOVA, Chi2, TF and TF-IDF. The results also show the difference between different feature ranges. Results obtained from the jurisdiction dataset.

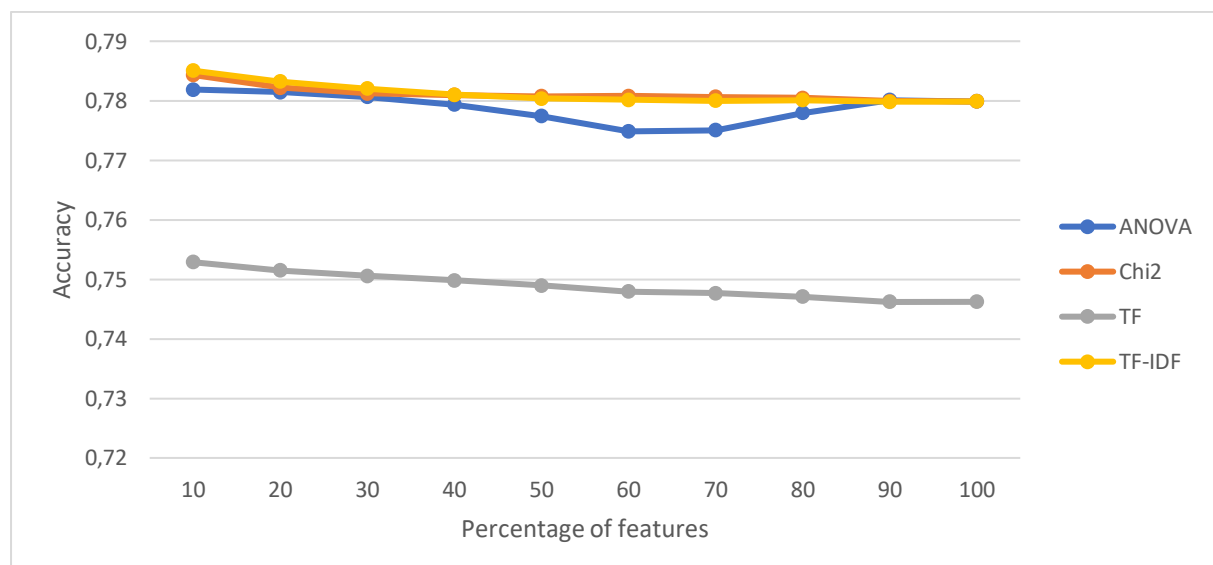


Figure 17 – Plotted accuracy values for MNB using ANOVA, Chi2, TF and TF-IDF. The results also show the difference between different feature ranges. Results obtained from the infraction dataset.

It was observed that MNB performs better with less features for both datasets. The decrease in accuracy is much less pronounced in the infractions dataset since it mostly occurs from 10% to 40% although not in a significant way. Accuracy for the jurisdiction dataset falls roughly 7 to 10%. It is possible to denote that MNB consistently achieves higher accuracy values with ANOVA, Chi-square and TF-IDF over TF.

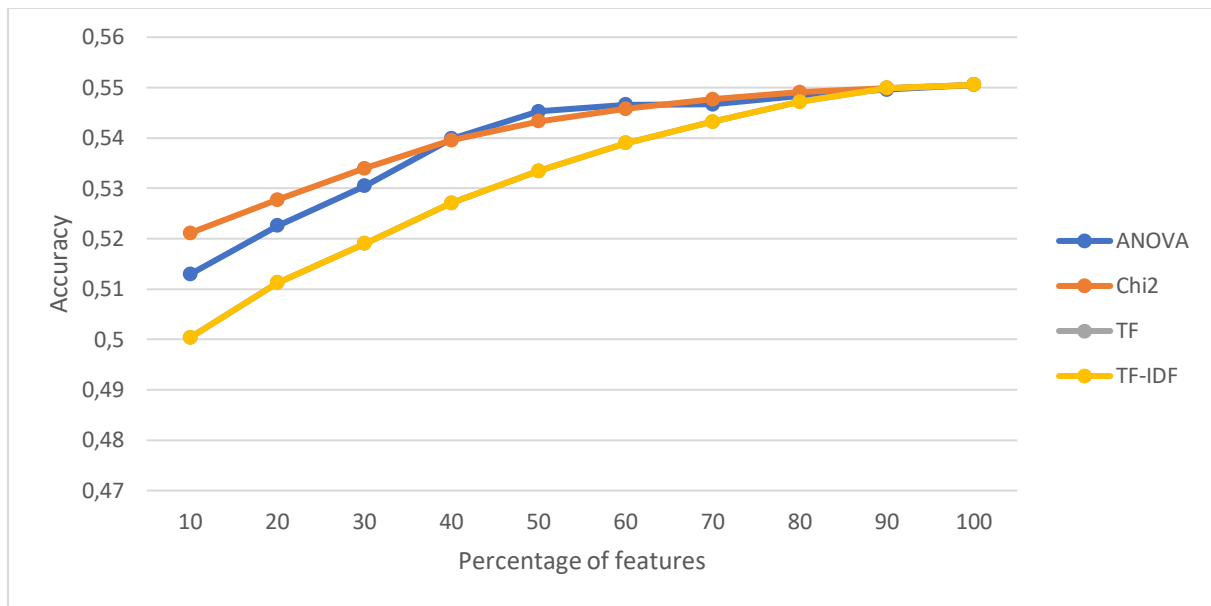


Figure 18 – Plotted accuracy values for BNB using ANOVA, Chi2, TF and TF-IDF. The results also show the difference between different feature ranges. Results obtained from the jurisdiction dataset.

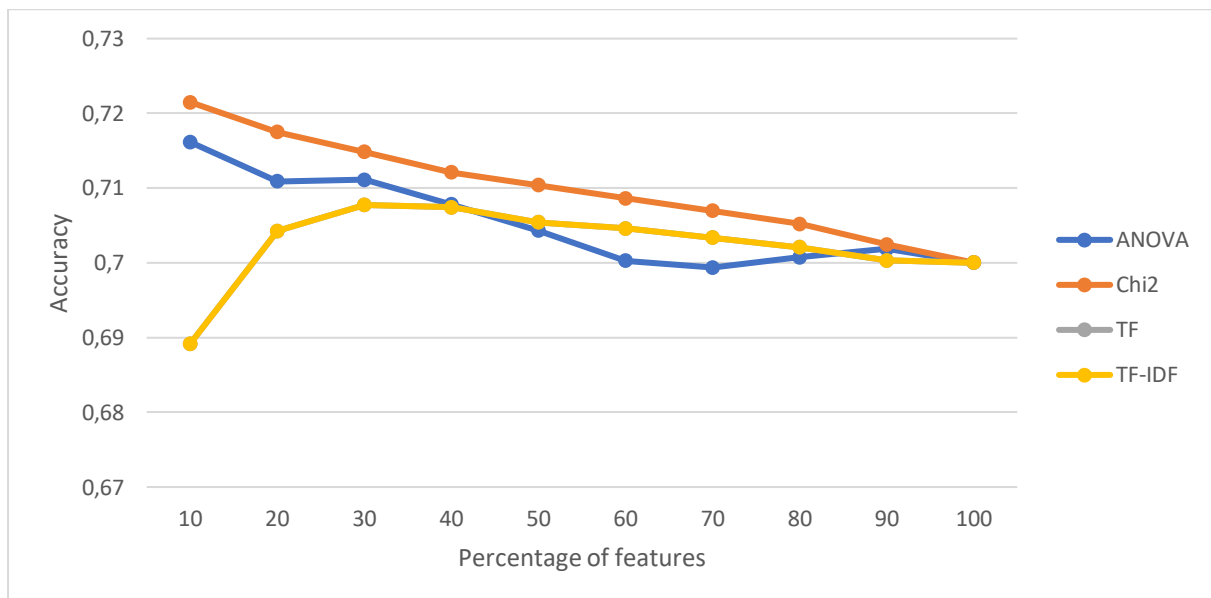


Figure 19 – Plotted accuracy values for BNB using ANOVA, Chi2, TF and TF-IDF. The results also show the difference between different feature ranges. Results obtained from the infractions dataset.

Figure 18 and Figure 19 show the results for BNB classifier.

BNB was observed to perform better with Chi-square and ANOVA feature selection rather than TF or TF-IDF. The values of TF-IDF overlap in similarity to Table 16 because BNB uses binary word occurrences rather than weights. Features ranging from 10% to 30% show similar results for TF-IDF / TF but the reverse behavior was observed for the majority of the feature ranges. In fact, BNB performed better as

more features were added for the first dataset and worse for the second dataset. Despite this, both behaviors were observed to converge towards the full feature size.

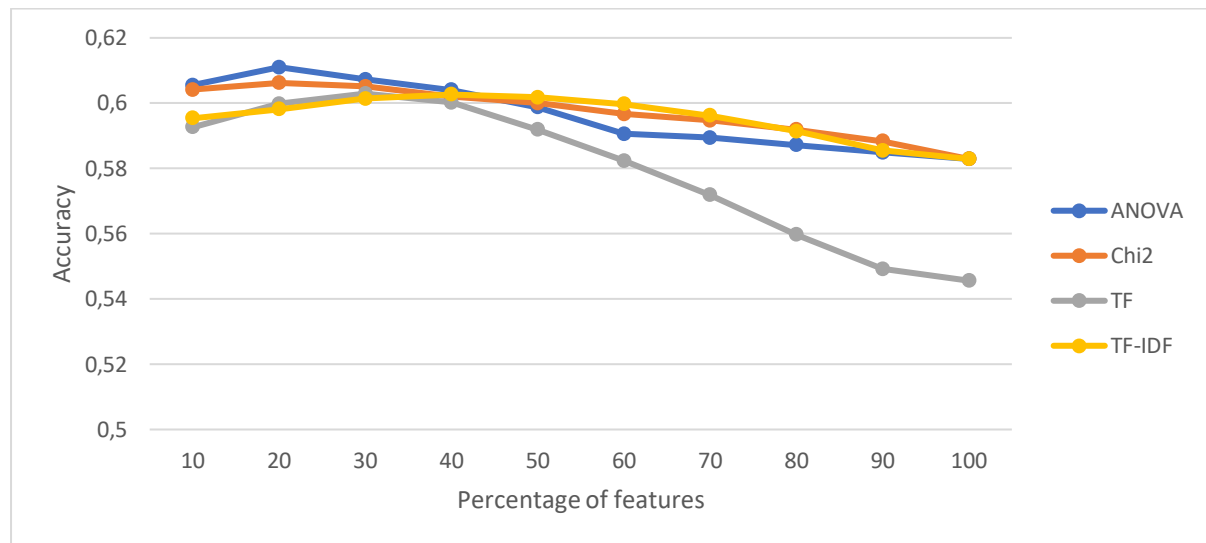


Figure 20 – Plotted accuracy values for CNB using ANOVA, Chi2, TF and TF-IDF. The results also show the difference between different feature ranges. Results obtained from the jurisdiction dataset.

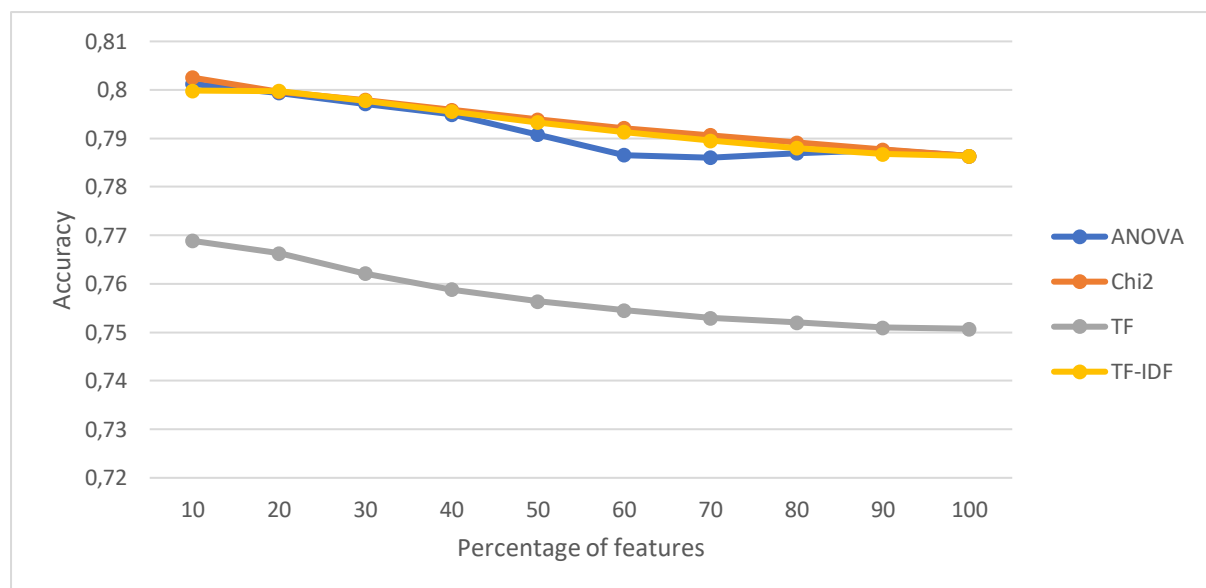


Figure 21 – Plotted accuracy values for CNB using ANOVA, Chi2, TF and TF-IDF. The results also show the difference between different feature ranges. Results obtained from the infractions dataset.

From Figure 20 and Figure 21, TF was observed to negatively impact accuracy for the CNB classifier in both datasets. Both TF-IDF and Chi-square show consistent and similar results across the feature ranges. In addition, all of the feature selection methods achieve lower results when the full number of features is taken into consideration with a slight increase in the 10–40% range for the jurisdiction dataset with TF and TF-IDF.

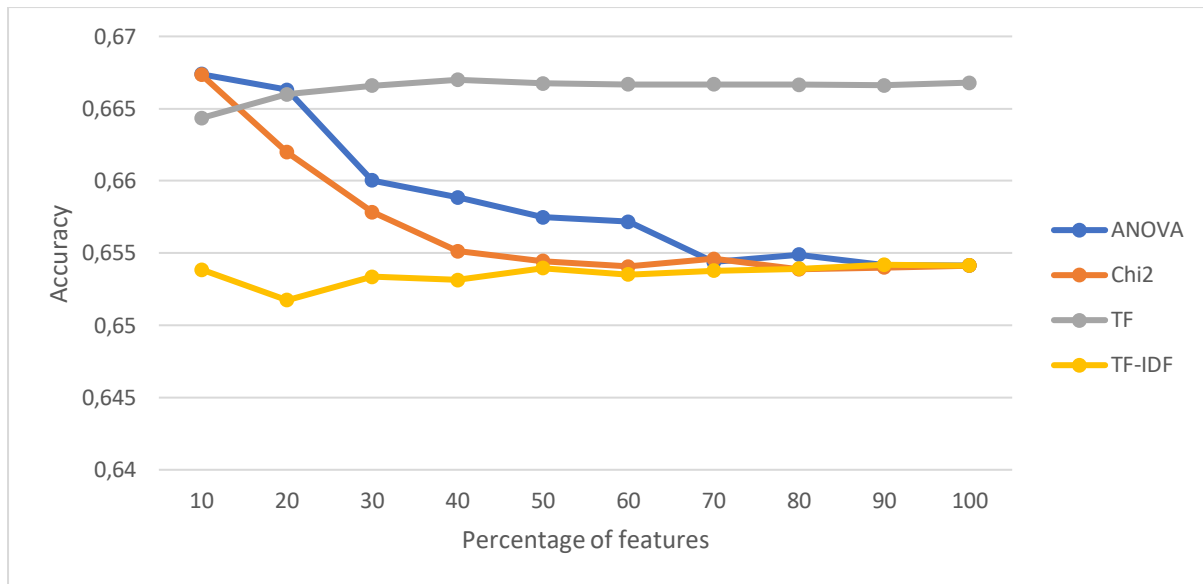


Figure 22 – Plotted accuracy values for SVM using ANOVA, Chi2, TF and TF-IDF. The results also show the difference between different feature ranges. Results obtained from the jurisdiction dataset.

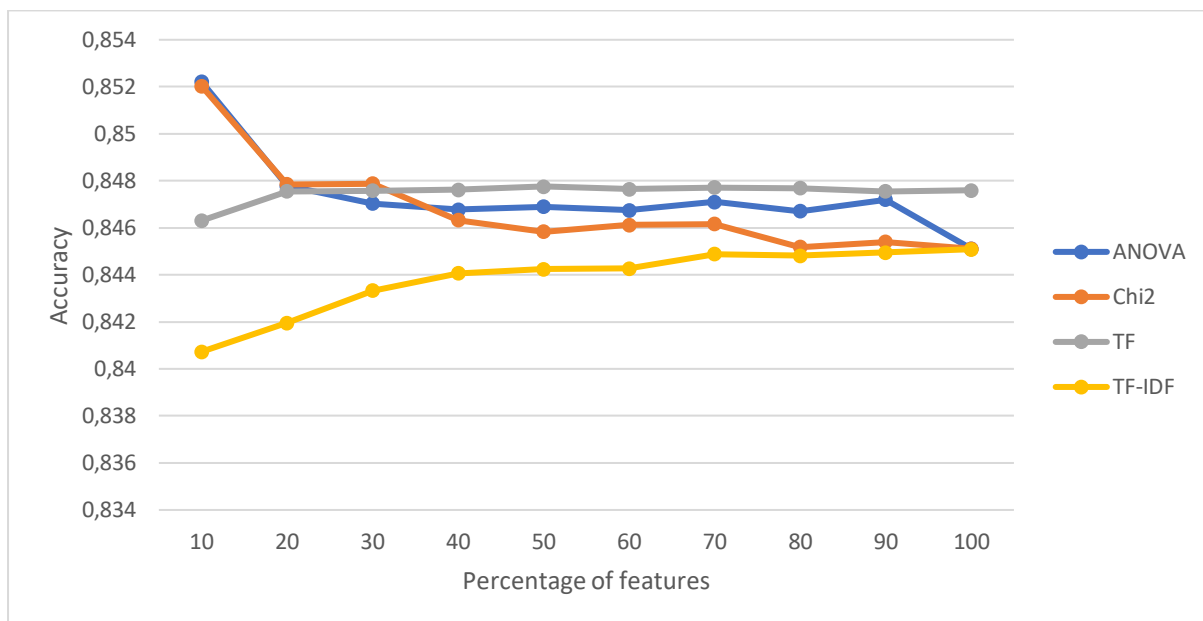


Figure 23 – Plotted accuracy values for SVM using ANOVA, Chi2, TF and TF-IDF. The results also show the difference between different feature ranges. Results obtained from the infractions dataset.

The results for the SVM are shown on Figure 22 and Figure 23. In similarity to what was found for the other classifiers, SVM performs better in both datasets with less features. The best results were observed at the 10% feature mark with either ANOVA or Chi-square feature selection. TF increases the SVM accuracy with more features as this behaviour was observed in both datasets. A decrease in performance was also found regarding Chi-square and ANOVA with more features added. In contrast, TF-IDF increases the SVM's predictive capability for the infractions dataset while achieving mixed results in the jurisdiction dataset.

The figures presented below show the results for LR with the different feature selection methods.

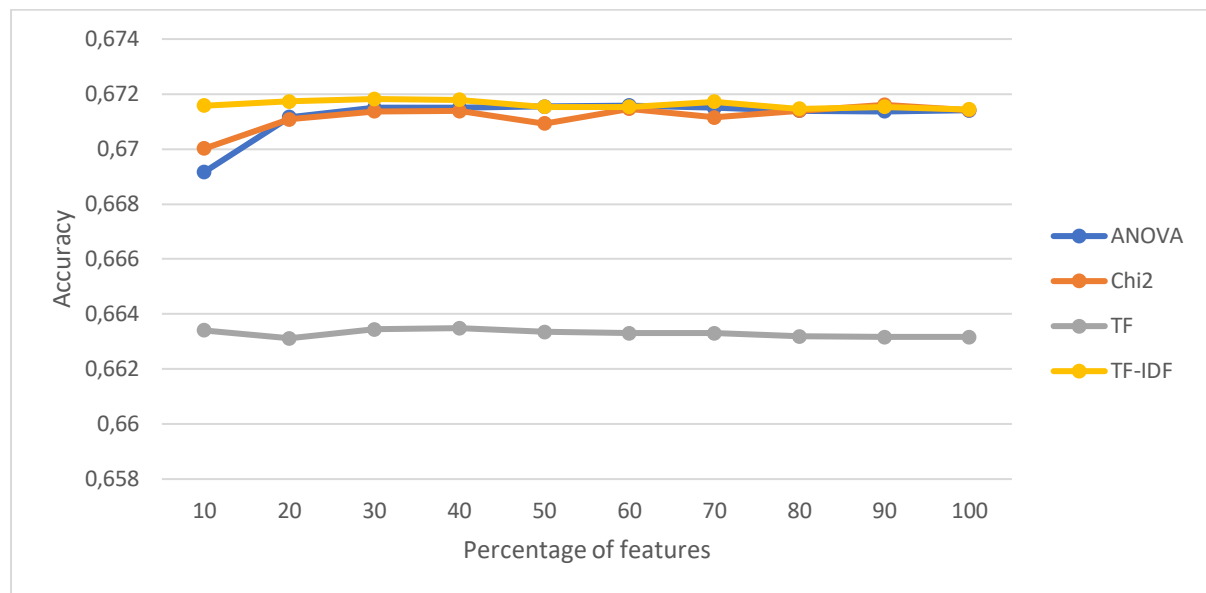


Figure 24 – Plotted accuracy values for LR using ANOVA, Chi2, TF and TF-IDF. The results also show the difference between different feature ranges. Results obtained from the jurisdiction dataset.

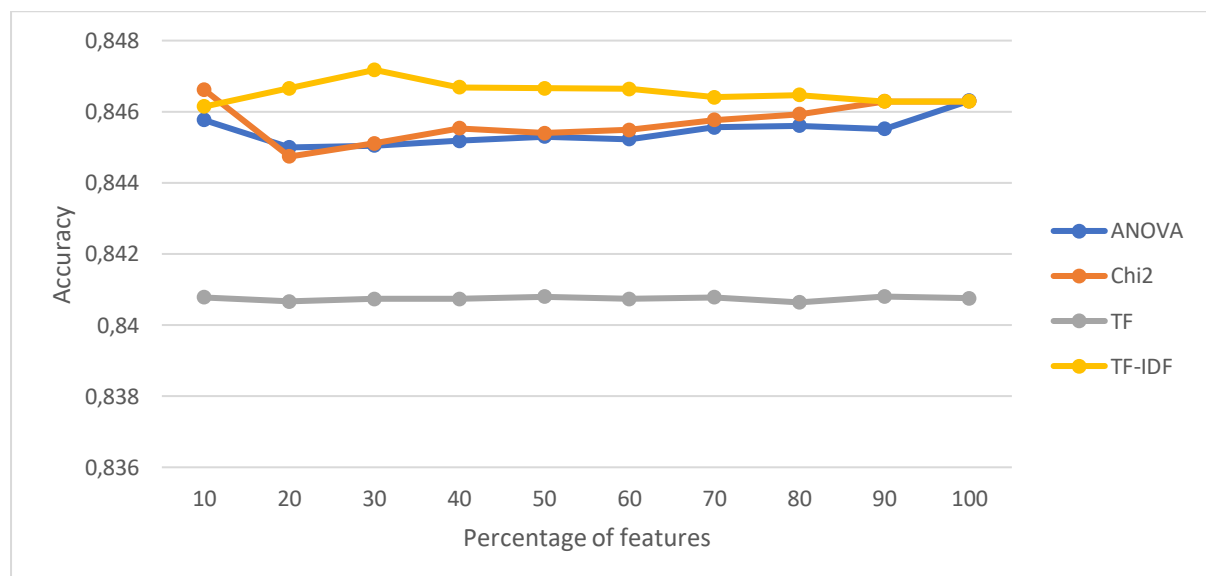


Figure 25 – Plotted accuracy values for LR using ANOVA, Chi2, TF and TF-IDF. The results also show the difference between different feature ranges. Results obtained from the infractions dataset.

The results for LR are similar between both datasets. TF-IDF consistently achieves higher values across most feature ranges with sporadic cases of lower accuracy values. It is also noticeable that the addition of further features does not incur in a significantly higher performance. In addition, it was observed that the ANOVA, Chi-square and TF-IDF methods rank much better than TF which is always below the others.

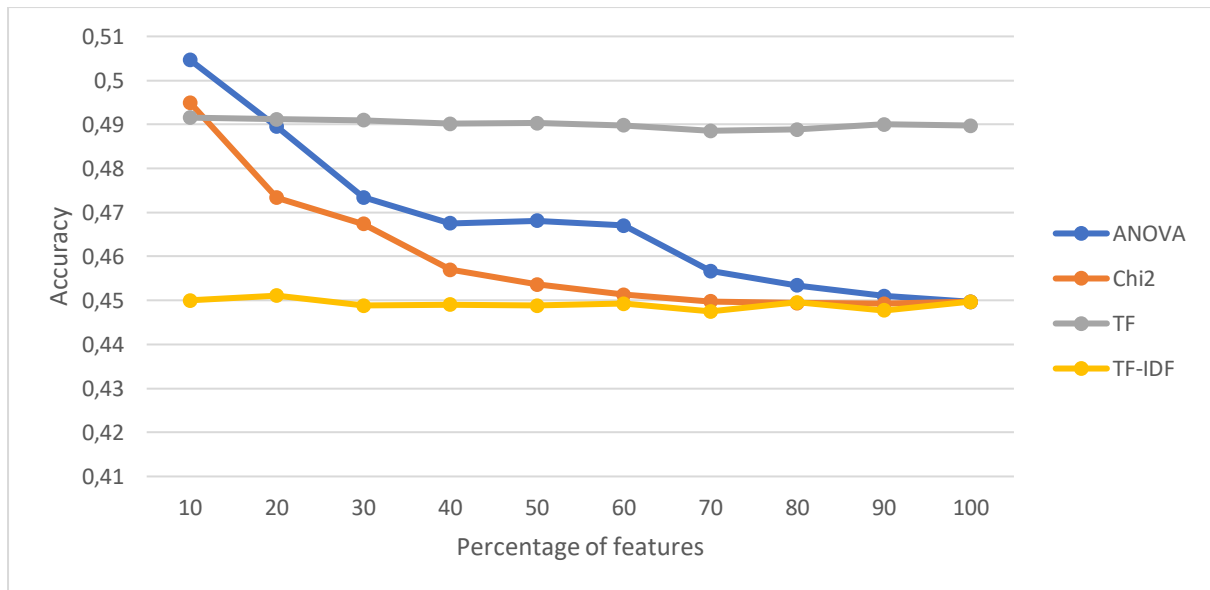


Figure 26 – Plotted accuracy values for K-NN using ANOVA, Chi2, TF and TF-IDF. The results also show the difference between different feature ranges. Results obtained from the jurisdiction dataset.

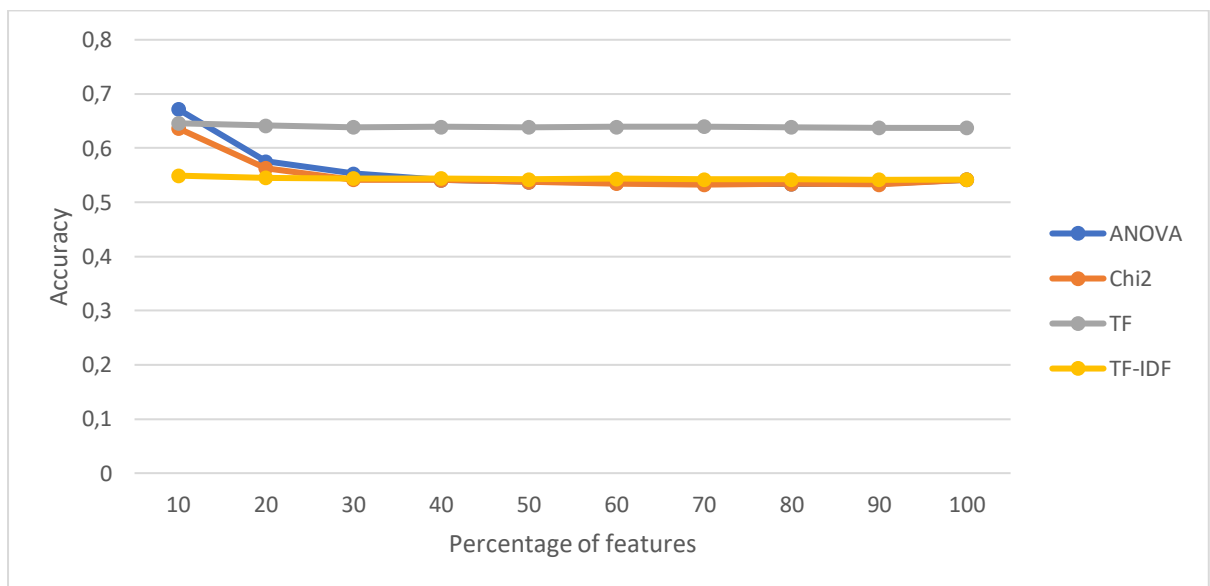


Figure 27 – Plotted accuracy values for K-NN using ANOVA, Chi2, TF and TF-IDF. The results also show the difference between different feature ranges. Results obtained from the infractions dataset.

As seen on Figure 26 and Figure 27, K-NN reports mostly the same behavior across both datasets. TF feature ranking is consistent across the different feature sizes and with it, K-NN achieves higher results when more features are included. Despite this, with smaller feature sizes, ANOVA and Chi-square achieve better results than TF or TF-IDF although converging towards higher feature sizes towards a lower accuracy value. Moreover, TF-IDF was also observed to be stable across different feature ranges.

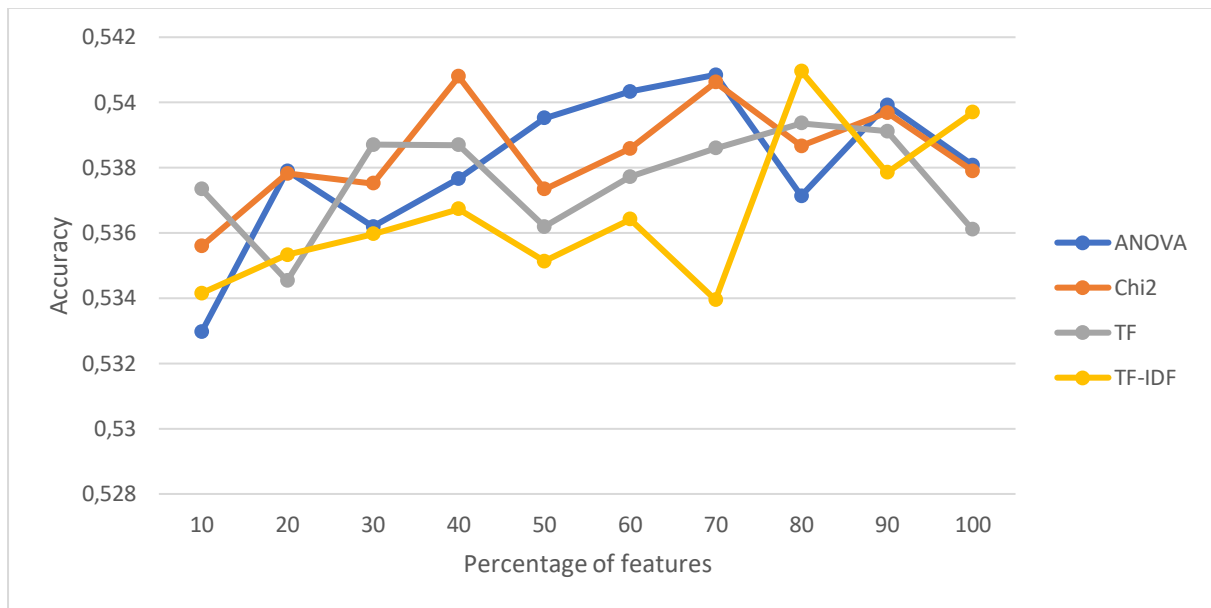


Figure 28 – Plotted accuracy values for DT using ANOVA, Chi2, TF and TF-IDF. The results also show the difference between different feature ranges. Results obtained from the jurisdiction dataset.

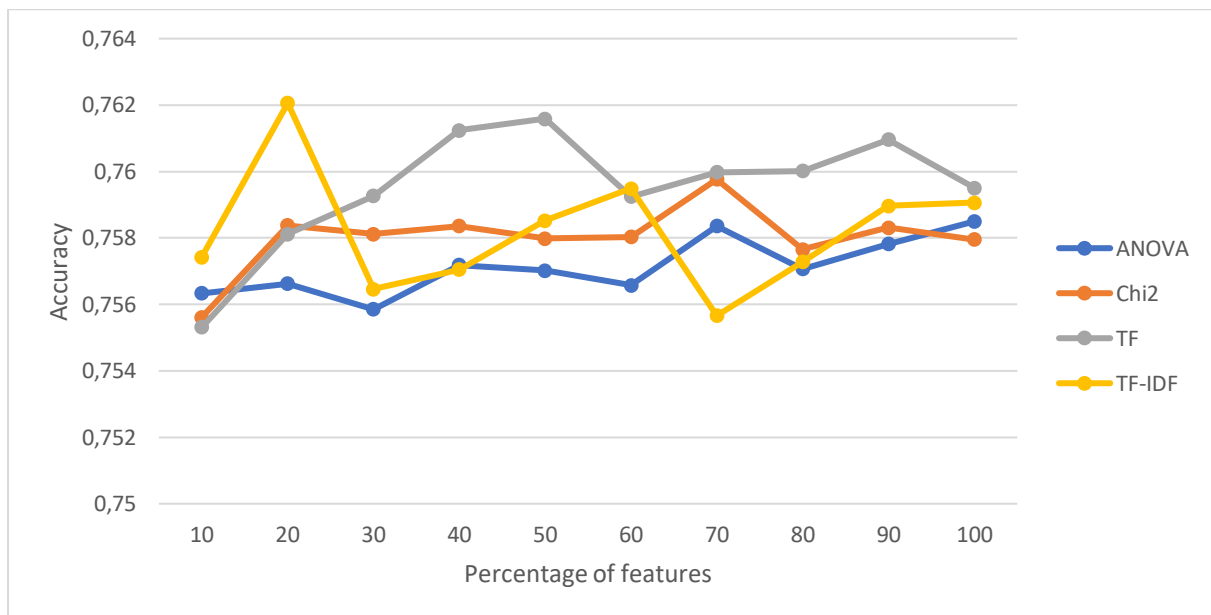


Figure 29 – Plotted accuracy values for DT using ANOVA, Chi2, TF and TF-IDF. The results also show the difference between different feature ranges. Results obtained from the infractions dataset.

It was observed that the results for DT are wildly different from those obtained so far. No clear pattern can be seen on the plots as there are constant “jumps” in accuracy values. Despite this, there is a slightly better performance with TF in the infraction’s dataset.

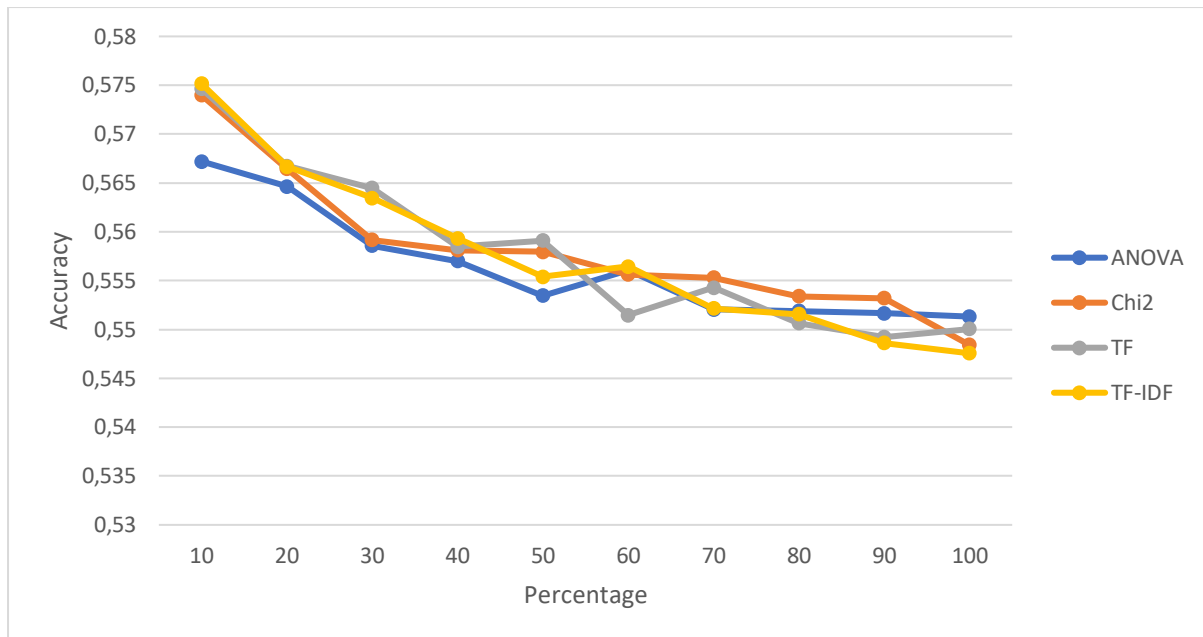


Figure 30 – Plotted accuracy values for RF using ANOVA, Chi2, TF and TF-IDF. The results also show the difference between different feature ranges. Results obtained from the jurisdiction dataset.

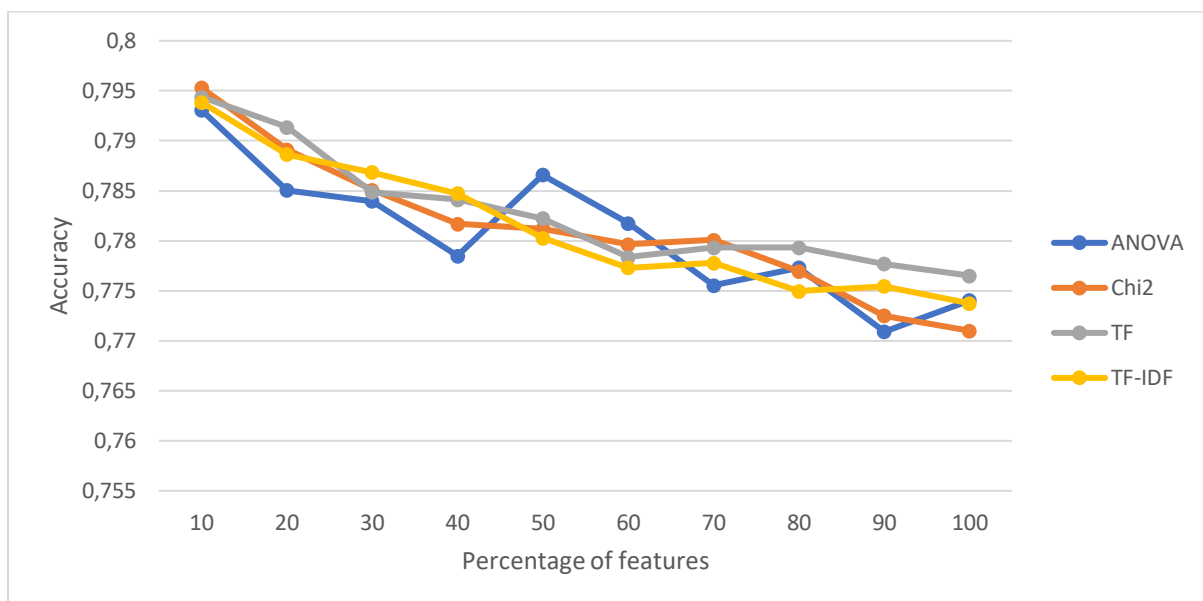


Figure 31 – Plotted accuracy values for RF using ANOVA, Chi2, TF and TF-IDF. The results also show the difference between different feature ranges. Results obtained from the infractions dataset.

Figure 30 and Figure 31 show the results for RF classifier. In both datasets, RF has a similar performance with different feature selection algorithms and feature sizes. In fact, as more features get added, performance lowers with most of the different feature selection methods. Indeed considering 10% of the feature sizes with either TF, TF-IDF or chi-square yields better results.

The general decrease in accuracy was denoted across all four methods.

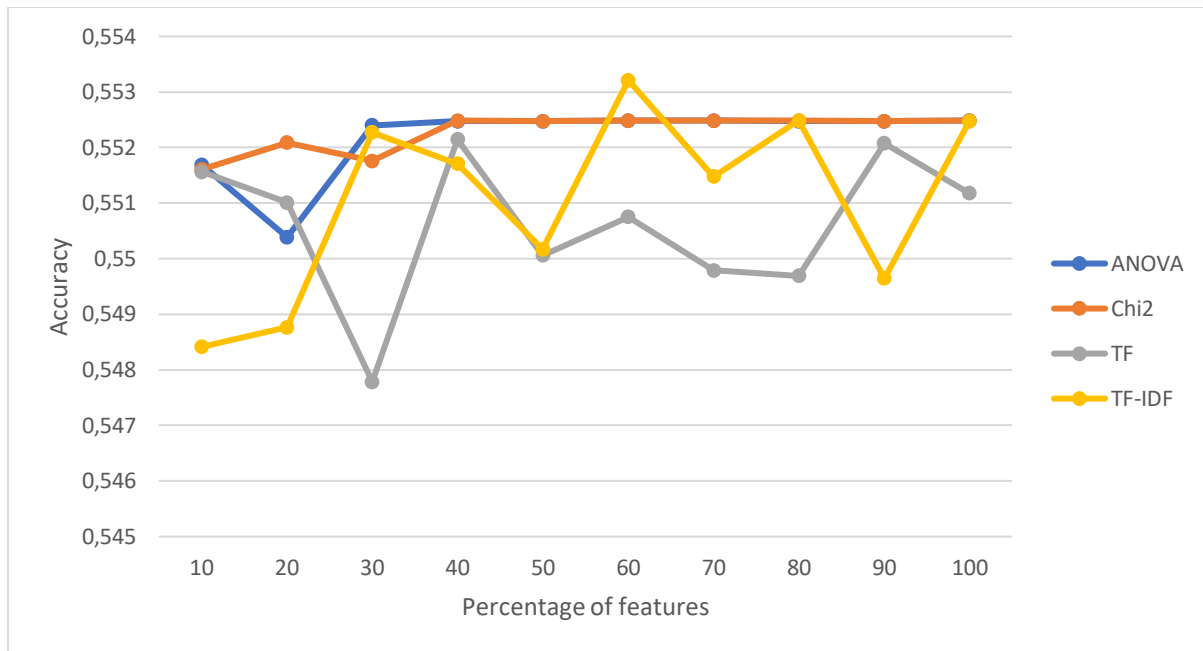


Figure 32 – Plotted accuracy values for AB using ANOVA, Chi2, TF and TF-IDF. The results also show the difference between different feature ranges. Results obtained from the jurisdiction dataset.

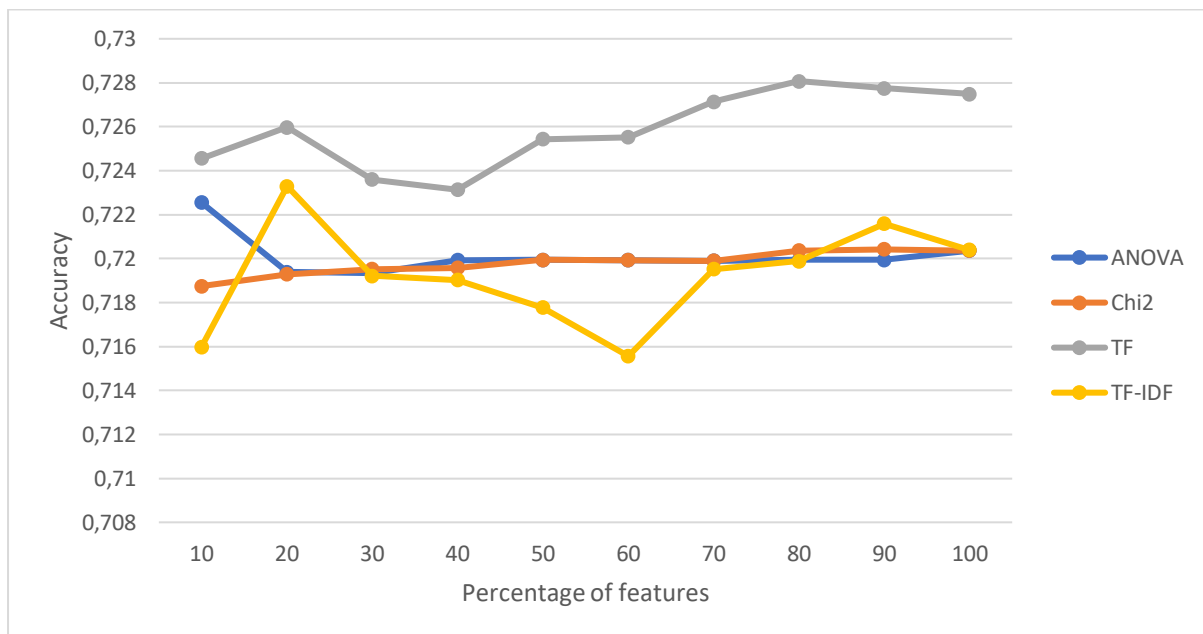


Figure 33 – Plotted accuracy values for AB using ANOVA, Chi2, TF and TF-IDF. The results also show the difference between different feature ranges. Results obtained from the infractions dataset.

The results for AB were observed to be extremely inconsistent for all feature selection algorithms but Chi-square and ANOVA. In Figure 32, the usage of TF-IDF or TF achieved mixed results with increases and decreases across the feature ranges with TF-IDF peaking at 60% feature size above all other feature selection methods. Furthermore, chi-square starts was observed to have a higher accuracy at 20% over ANOVA but the difference was not significant for either higher or lower feature ranges.

Similarly, in the infractions dataset, both chi-square and ANOVA were observed to lead to similar results with ANOVA having a better average accuracy for the 10% mark. In addition, in this dataset, TF outperformed every other feature selection algorithm in every interval of the feature size considered.

4.5.1. Discussion on feature selection and feature sizes

Different feature selection methods were evaluated in this section: TF, TF-IDF, Chi-square and ANOVA. We've also evaluated the impact of different feature percentiles from 10% to 100% of the feature size and plotted the learning curves for each classifier.

It was reported that the classifiers belonging to the Naïve Bayes group (MNB, BNB and CNB) performed similarly with different feature selection methods. In fact, considering more features for MNB and CNB did not yield better results in both datasets without the exception of 20–30% of features in CNB for the jurisdiction dataset. It was also denoted that using TF as a feature selection method yielded worse results than the rest of the methods tested. ANOVA, chi-square and TF-IDF were observed to perform in a similar manner across the feature ranges. On the other hand, BNB shows the opposite behavior when compared between the datasets as accuracy increases in the jurisdiction dataset with more labels added but decreases in the infraction's dataset. It was also concluded that there is no difference between TF-IDF and TF as BNB only considers binary word occurrences and thus no difference is observed between both methods. The author of (107) evaluates different feature selection methods and their impact on a Naïve Bayes classifier in datasets with different languages. The use of chi-square was observed to either maintain or increase accuracy over the test space as well as in the Portuguese dataset tested which was also observed in this experiment.

Unlike the Naïve Bayes classifiers, TF was observed to perform better when used as a feature selection method for the SVM the more features are taken in consideration. It was also observed that TF-IDF underperformed when compared to the other methods in both datasets but converging towards the higher number of features considered. Moreover, the other linear classifier, LR, was observed to perform worse with TF while achieving better accuracy values with TF-IDF for many of the feature ranges. Similar performance between ANOVA and chi-square was also reported for the LR classifier.

The K-NN classifier was observed to perform better with TF across most feature sizes with the exception being between 10% to 20% features. It was possible to conclude that the various feature selection methods did not positively impact accuracy as more features were added. In fact, chi-square and ANOVA incurred in a performance decrease for the jurisdiction dataset while TF and TF-IDF were shown to maintain the same accuracy values across the feature ranges.

The accuracy values reported for the DT, RF and AB did not allow us to extract any meaningful conclusions as they were shown to be quite inconsistent across several feature sizes. The exception in these

classifiers was RF when applied to the infractions dataset as a general decrease in accuracy was observed.

For the next section we make use of chi-square as our feature selection method with 10% of the most representative features included. This choice was due to the fairly consistent results for most of the classifiers and due to the better results observed with less features. We considered LR, SVM, K-NN and RF for parameter optimization as these classifiers allow for further tweaking and experimentation than classifiers such as CNB, BNB and MNB.

4.6. Hyper-parameter optimization

In this section we report the results obtained from optimizing some of the LR, SVM, K-NN and RF's parameters. There are many possible ways to do hyper-parameter optimization such as random search or exhaustive grid search. The first implementation allows the search of the parameter space provided to each algorithm by doing random selections of combinations and returning the one that achieves the lowest error while the latter implies that an exhaustive search is performed over every parameter combination. Randomly searching the parameter space allows for faster results tracking which can then be used to search the space around the values reported with grid search, performing a more in-depth analysis. Additionally, there have been uses of other parameter optimization algorithms such as Bayesian optimization which, in most cases, achieve the same results as the other methods but in a much less computationally expensive manner (108). The results we report on this section do not make use of the methods mentioned before but instead focus on a manual intervention. We considered using random and exhaustive grid search, but it was computationally expensive to run and thus these were not used.

The results obtained are separated by classifier and each results table contains information related to both datasets for easier comparison and readability. In addition, the results tables also contain information related to the training as well as the test sets average weighted accuracy values. This allows us to establish comparisons between different values and assess how the algorithm is performing in relation to overfitting or underfitting.

So far, the value of K for the K-NN classifier has been 5. Different values of K lead to different outcomes as mentioned in the background chapter. To further understand the impact of this parameter we evaluated the classifier with different K values as shown on the table below (Table 22) while the results for each iteration with 10-fold stratified cross-validation are reported on Table 23.

Table 22 – Hyper-parameters' searching space for K-NN.

Parameter	Value range
K	1, 3, 5, 7, 10, 20

Table 23 – Average weighted accuracy for train and test sets with different values of K (K-NN).

K	Jurisdiction		Infractions	
	Train scores	Test scores	Train scores	Test scores
1	0.999	0.476	0.998	0.640
3	0.738	0.493	0.842	0.651
5	0.650	0.494	0.764	0.636
7	0.614	0.498	0.729	0.630
10	0.575	0.500	0.685	0.614
20	0.528	0.4951	0.666	0.617

It was possible to observe that for the jurisdiction dataset, increasing the value of K generally lead to better results for the test set as denoted by the increase in accuracy from K=1 (0.47684) to K=10 (0.50011). In contrast, the same classifier exhibits lower accuracy the more the nearest neighbor's value was increased with the exception when K=3 which was better than K = 1. Despite this, both datasets exhibit much higher training accuracy values when K takes a lower value, being indicative that the estimator is overfitting. Increasing the value of the nearest neighbors to consider leads to less overfitting and an approximation of training and testing values.

The following results reported regard the LR classifier where we've tested L1 and L2 penalties, one-vs-rest and multinomial multi-class schemes and a set of inverse regularization strength values. We used the 'saga' solver for algorithmic optimization so far as it is better suited for larger datasets as well as allowing for more combinations of parameters to be tested. In similarity to the SVM, a lower value of C specifies a stronger regularization strength which leads to a higher data separation while a higher C value leads to a lower data plane separation.

The values tested for the three LR parameters are shown on the following table (Table 24) while the results are shown on Table 25.

Table 24 – Hyper-parameters' searching space for LR.

Parameters	Value range
Penalty	L1, L2
Multi-class	OVR, Multinomial
C	0.001, 0.01, 0.1, 1, 5

Table 25 – Average weighted accuracy for train and test sets with different values of C (LR).

C	Jurisdiction		Infractions	
	Train scores	Test scores	Train scores	Test scores
0.001	0.471	0.471	0.584	0.585
0.01	0.477	0.477	0.753	0.744
0.1	0.637	0.614	0.824	0.813
1	0.727	0.670	0.870	0.846
5	0.772	0.672	0.898	0.852

We observed that increasing the inverse regularization strength yields better results both on the test set and the training set. A low inverse regularization strength proves to be much more penalizing for the algorithm despite achieving practically the same average accuracy between train and test sets. The best scores were obtained with C=5 where the accuracy was observed to be as high as 57.2% for the jurisdiction dataset and 85.2% for the infraction's dataset.

Table 26 – Average weighted accuracy for train and test sets with different multi-class schemes (LR).

Multi-class	Jurisdiction		Infractions	
	Train scores	Test scores	Train scores	Test scores
OVR	0.72	0.670	0.870	0.846
Multinomial	0.746	0.677	0.881	0.849

Table 26 shows the accuracy values for two multi-class handling schemes, one-vs-rest and multinomial. One-vs-rest fits a binary classification problem for each label and chooses the one that performs the best while multinomial minimizes loss across the entire probability distribution. The classifier defaults to one-vs-rest when it is not specified.

We observed that the values are slightly higher for multinomial multi-class handling for the test set but not significant. Higher values were also observed for the training sets which in conjunction to small increases in the test set lead to more overfitting.

Table 27 – Average weighted accuracy for train and test sets with different penalties (LR).

Penalties	Jurisdiction		Infractions	
	Train scores	Test scores	Train scores	Test scores
L1	0.726	0.674	0.870	0.851
L2	0.727	0.670	0.870	0.846

As seen on Table 27, using L1 regularization leads to higher scoring on both datasets with an increase from 67.0% with L2 to 67.4% with L1 for the first dataset and from 84.6% to 85.1% for the infractions related dataset.

The following results reported are for the RF classifier. We've conducted tests regarding the number of estimators, the minimum samples split, minimum samples leaf and the maximum depth of the trees (Table 28)

Table 28 – Hyper-parameters' searching space for RF.

Parameters	Value range
Nº Estimators	10, 100, 300, 500
Min. Samples split	2, 5, 10, 25, 50, 100
Min. Samples leaf	2, 5, 10, 25
Max. depth	'Auto', 5, 10, 20, 30, 50, 100, 150

Table 29 – Average weighted accuracy for train and test sets with different number of estimators (RF).

Nº estimators	Jurisdiction		Infractions	
	Train scores	Test scores	Train scores	Test scores
10	0.986	0.572	0.988	0.794
100	0.999	0.622	0.999	0.820
300	0.999	0.629	0.999	0.824
500	0.999	0.629	0.999	0.824

The number of estimators is the number of trees in a forest. The default value is 10 while the authors maintaining the classifier's package will increase it to 100 in a new version. The results we've obtained so far were considering only 10 trees and thus a different number of trees was tested.

It was observed that the increase in the number of trees generally leads to higher test set accuracy values as well as slightly higher train scores even though it is already quite high when a low number of is taken in consideration. The biggest jump in accuracy was observed from 10 to 100 trees while increasing it further did not substantially impact accuracy.

Table 30 – Average weighted accuracy for train and test sets with different minimum samples split values (RF).

Min. Samples Split	Jurisdiction		Infractions	
	Train scores	Test scores	Train scores	Test scores
2	0.986	0.569	0.989	0.793
5	0.975	0.581	0.981	0.798
10	0.958	0.584	0.972	0.802
25	0.920	0.588	0.956	0.801
50	0.875	0.592	0.939	0.802
100	0.819	0.594	0.9155	0.805

The minimum samples split value explains the number of samples that are required to split an internal node. Increasing this value leads to more samples being required to create a node for a given attribute.

We observed (Table 30) that increasing the default value of 2 leads to better prediction capability denoted by the increased accuracy values for the test set. Moreover, increasing the minimum value also leads to a reduction in the training score accuracy, which was quite high at the default value, indicating an overfitting problem. The best metrics value was obtained at 100 minimum samples split.

Table 31 – Average weighted accuracy for train and test sets with different minimum samples leaf values (RF).

Min. Samples Leaf	Jurisdiction		Infractions	
	Train scores	Test scores	Train scores	Test scores
2	0.879	0.587	0.921	0.802
5	0.741	0.584	0.859	0.796
10	0.665	0.577	0.829	0.795
25	0.588	0.547	0.803	0.785

The minimum samples leaf represents the number of samples required to be at a leaf node. It was observed that increasing this value generally leads to a poorer classifier's performance as the accuracy values decrease. Despite this, considering higher values for the minimum samples leaf does yield lower accuracy values for the training set which equates to higher training errors and thus less overfitting since the test score values do not alter much.

Table 32 – Average weighted accuracy for train and test sets with different maximum depth values (RF).

Max. Depth	Jurisdiction		Infractions	
	Train scores	Test scores	Train scores	Test scores
Auto	0.986	0.573	0.989	0.795
5	0.474	0.472	0.662	0.651
10	0.501	0.483	0.731	0.713
20	0.620	0.525	0.812	0.762
30	0.746	0.551	0.870	0.782
50	0.880	0.569	0.932	0.790
100	0.967	0.5715	0.980	0.798
150	0.982	0.572	0.987	0.796

The maximum depth parameter of the random forest equates to the trees' maximum length. If the value is set to 'auto', as it defaults to, all the nodes are expanded until all the leaves are either pure or contain less than the number of samples of the minimum samples split value.

From Table 32 it is possible to observe that the default value reports a test and train score similar to those obtained at values of 100 and 150. In these value ranges, overfitting is observed as the training score is much higher than the test score and thus the model does not generalize well. At the 10 to 30 maximum samples for the depth value, results seem to be more balanced as the test score is similar to the train score or much closer while not losing too much information.

As Table 33 shows, the optimization for the SVM was done on the C and penalty parameters.

Table 33 – Hyper-parameter's searching space for the SVM.

Parameters	Value range
C	0.01, 0.1, 0.5, 2
Penalty	L1, L2

Table 34 – Average weighted accuracy for train and test sets with different values of C (SVM).

C	Jurisdiction		Infractions	
	Train scores	Test scores	Train scores	Test scores
0.01	0.633	0.609	0.824	0.813
0.1	0.727	0.668	0.872	0.846
0.5	0.774	0.670	0.900	0.853
1	0.791	0.667	0.910	0.851
2	0.805	0.662	0.919	0.849

In similarity to what was observed before with LR, a higher value of the C parameter leads to higher results both on the test set and on the training set (

Table 34). An extremely low C value (0.01) makes the SVM lose information when trying to classify the labels as the accuracy is significantly lower than higher C values. However, train scores are similar to the test scores observed and thus there is significantly lower overfitting tendency with a smaller C value on both datasets. A value of 0.1 shows good results with a positive and significant increase in the test set accuracy.

Table 35 – Average weighted accuracy for train and test sets with different penalties (SVM).

Penalty	Jurisdiction		Infractions	
	Train scores	Test scores	Train scores	Test scores
L1	0.778	0.667	0.902	0.854
L2	0.791	0.667	0.910	0.851

It was observed that the L1 and L2 penalty methods incur in the same behavior observed for LR. On one hand, L1 allows the algorithm to capture more information while not increasing overfitting. On the other hand, L2 was observed to show slightly lower results while increasing the training accuracy which is undesirable.

4.6.1. Discussion on hyper-parameters optimization

We've observed that the use of a higher value of K for the nearest neighbor classifier has a positive impact on accuracy, albeit small, for the jurisdiction classification task but shows a negative impact on the infraction's classification task. Despite this opposite behavior, we've noticed that the training accuracy is reduced with higher values for the nearest neighbors considered which has a positive impact in decreasing

overfitting. In fact, with $K=1$ we observe extremely disproportional results for the train and test set accuracy with the train set accuracy reaching almost 100% in both datasets. We conclude that a value of K between the default value of 5 to 10 is beneficial for both datasets as it reduces overfitting while slightly increasing performance for the jurisdiction classification, reducing the performance by a small margin on the infraction classification task.

As far as the RF classifier goes, several tests were conducted regarding different parameters from tree depth to the number of samples required to split a node. Although RF did not achieve results as high as LR or SVM, the increase in accuracy is meaningful when the parameters are optimized. We've observed that using the default parameters as of the time of writing is leading to overfitting due to the high discrepancy between train and test set scores. While not impacting training scores, increasing the number of estimators to 100 in a random forest yields much better results than the default value of 10. In addition, increasing the values of the minimum samples to split an internal node leads to better results as accuracy increases for the test set and decreases for the train set. This increase in the test set is not observed with increasing the minimum samples required at each leaf but does decrease overfitting substantially. Moreover, the default value that sets the maximum depth of each tree also benefits from manual intervention as similar accuracy values can be observed while decreasing the difference between the train and test set.

We've observed that a similar behavior was obtained from the optimization of the penalty and C parameters for the linear classifiers LR and SVM. A lower C value creates a bigger margin separating the data which leads to misclassification of some of the data. This is explained by the lower accuracy values reported for extremely small C values (0.001 and 0.01). The default value of 1 for the C parameter achieves a good balance between the train and test sets accuracy values and further increasing this does not seem to yield better results. In addition, it was also concluded that the L1 penalty yields better performance for both classifiers.

In the next section we evaluate the classifiers that have so far achieved better performance values, LR, SVM and RF. We test the classifiers on the four metrics mentioned before: accuracy, precision, recall and f1-score and we also analyze the respective confusion matrices. The parameters used for each classifier were selected according to the results obtained in this section.

4.7. Final Evaluation

In this section we evaluate the best performing classifiers so far: LR, SVM and RF. We report the accuracy, precision, recall and f1-measure on macro level and extrapolate the results regarding their performance across all labels. We also show the results for the confusion matrices which help further understand the

classifier's performance and which labels are not being well predicted. The parameters used for each classifier are shown before each result and each result contains information obtained from both datasets. LR was trained with a L1 penalty and inverse regularization strength of 1. These values were observed to be a good balance between accuracy and misclassification rates as further increasing or decreasing this value lead to worse results across the smaller represented labels on the datasets. The different metrics results for LR are shown on Table 36.

Table 36 – Evaluation metrics for LR: Accuracy, precision, recall and f1-measure (macro).

Metrics	Jurisdiction	Infractions
Accuracy	0.674	0.851
Precision	0.577	0.728
Recall	0.425	0.551
F1-measure	0.446	0.582

A high average accuracy value was observed for both datasets, 67.4% and 85.1%. We also observed that the precision values for the jurisdiction and infractions (57.7% and 72.8%, respectively) dataset are higher than those obtained for the recall metric (42.5% and 55.1%, respectively). The harmonic mean of these two metrics (f1-measure) was observed to be much lower than the values obtained in the accuracy metric, at 44.6% for the jurisdiction dataset and 58.2% for the infraction's dataset. This difference is indicative of misclassification probably due to class imbalance. To test this assumption, we computed the confusion matrices related to the LR classifier as seen on Figure 34 and Figure 35.

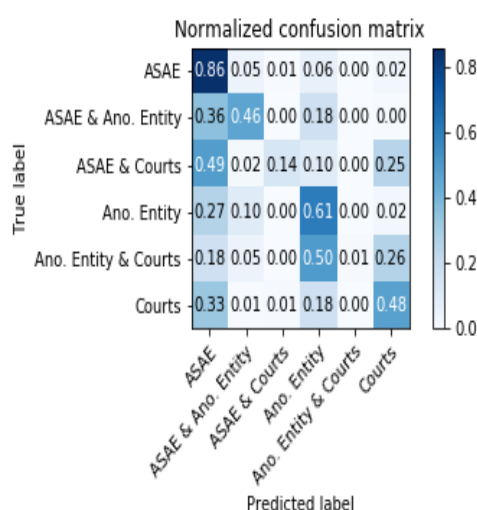


Figure 34 – Normalized confusion matrix for LR (jurisdiction dataset).

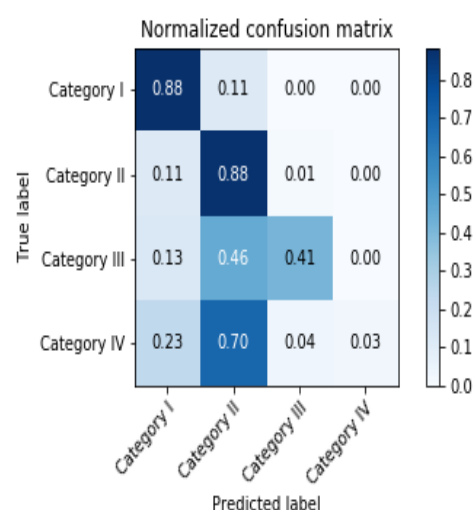


Figure 35 – Normalized confusion matrix for LR (infractions dataset).

The figures above shown the normalized confusion matrices for each dataset with the use of the LR classifier. It was observed that the labels which are better represented on the dataset tend to have better

results. For the infraction's dataset, the "ASAE" label was shown to have 86% of correct predictions followed by "Another Entity" with 61% and "ASAE and another entity" with 46%. In fact, "ASAE" is the most prevalent label in this dataset, followed by "ASAE and another entity" and "Another entity". We also observed that labels which share jurisdictions, such as "another entity and courts" tend to be misclassified towards one or both of the jurisdictions as 50% of this label was predicted as being "another entity" and 26% as "courts". Furthermore, it was also observed that LR managed to predict category I and II for the infractions dataset with almost 90% correctly labeled documents. 11% of the documents for category I were misclassified as being of category II while the inverse happened for category II towards category I. In addition, category III was only predicted correctly in 41% of the cases and 46% of the predictions were labeled as being of category II. Category IV was severely misclassified. These results will be further discussed ahead.

SVM was trained with the same parameters as LR, C=1 and L1 penalty. The metric results for this classifier are shown on Table 37.

Table 37 – Evaluation metrics for SVM: Accuracy, precision, recall and f1-measure (macro).

Metrics	Jurisdiction	Infractions
Accuracy	0.668	0.854
Precision	0.536	0.684
Recall	0.429	0.570
F1-measure	0.447	0.598

As seen in Table 37, results for the SVM are similar to the LR classifier. A lower precision but a slightly higher recall value was observed for both datasets compared to the previous linear classifier. SVM also achieves high accuracy values for both datasets, 66.8% for the jurisdiction dataset and 85.4% for the infractions dataset. F1-measure are also slightly higher than LR with values below 50% for the jurisdiction dataset and above 50% for the infractions dataset as was observed before. Figure 36 and Figure 37 show the respective confusion matrices for the SVM classifier.

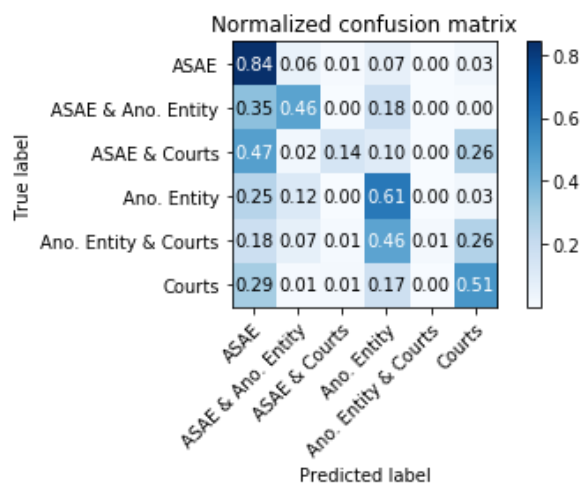


Figure 36 – Normalized confusion matrix for SVM (jurisdiction dataset).

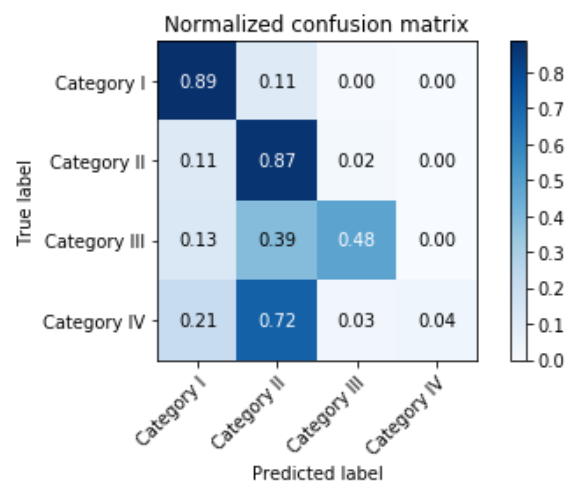


Figure 37 – Normalized confusion matrix for SVM (infractions dataset).

As seen on the figures above, SVM achieves mostly the same results as the previous classifier. It does outperform LR by 1% on the “courts” label but generally misclassifies labels in the same manner. SVM also exhibits the same pattern in relation to the lower represented labels such as any of the “courts” related labels while also skewing its prediction towards one or both classes that constitute a mixed label. We observed that for the “ASAE and courts” label, SVM classifies 47% as being of “ASAE”’s jurisdiction and 26% as being for the “courts”. Moreover, the classifier also predicts the infractions label much in the same way as LR with only a 1% increase in prediction accuracy for category I but achieving 7% better results for category III. It also performs 1% better in category IV even though it is grossly misclassified.

Below are the results for the RF classifier. RF was trained with only the number of estimator’s parameter at 100 as the other combinations were not performing as well considering all the labels.

Table 38 – Evaluation metrics for RF: Accuracy, precision, recall and f1-measure (macro).

Metrics	Jurisdiction	Infractions
Accuracy	0.621	0.821
Precision	0.545	0.620
Recall	0.331	0.456
F1-measure	0.352	0.463

Table 38 shows the values for accuracy, precision, recall and f1-measure for the RF classifier. It shows that despite showing only a roughly 4% decrease in accuracy across both datasets, RF does not manage to achieve good performance when it comes to recall and f1-measure. Given that precision is not as low as recall and that recall does not surpass the 50% mark in any of the datasets, RF is thus not able to capture most of the labels and classify them correctly. This is further illustrated below on Figure 38 and Figure 39.

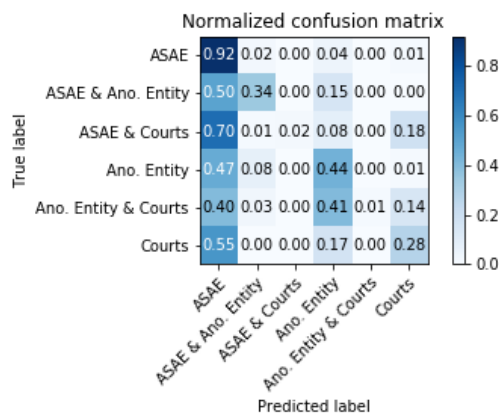


Figure 38 – Normalized confusion matrix for RF (jurisdiction dataset).

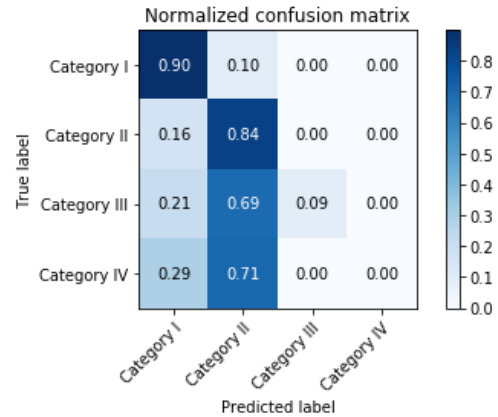


Figure 39 – Normalized confusion matrix for SVM (infractions dataset).

RF was observed to perform better in the label that is mostly present on both datasets, ASAE and category I with 92% and 90% correctly predicted documents respectively. The results for these labels are higher than those obtained with SVM and LR. Despite this performance, it loses ground when the other labels are taken in consideration with a 12% decrease for the “ASAE and another entity” label and a 3% decrease for category II in the second dataset. It is almost not able to catch any information related to “ASAE and courts” and “another entity and courts” unlike the previous classifiers. In addition, category III of the infractions dataset was observed to have a correctly predict ratio of only 9% compared to the previous 48% while category IV did not have a single correctly classified document.

4.7.1. Discussion on the final results

From the results shown above, we concluded that the high accuracy values obtained are representative of the prediction power of the different classifiers towards the more prevalent classes. We observed that precision was always higher than recall for the three classifiers which denotes that these classifiers manage to correctly label documents that belonged to the target groups although not being able to correctly recall all of them. Furthermore, we observed that in the jurisdiction dataset, labels that have more than one jurisdiction associated to it are shown to be misclassified towards one or the other or both constituent jurisdictions.

It was observed that the classifiers were not able to distinguish between these single and grouped labels as is seen by the ratio of predictions made towards one or the other constituent labels. In addition, it is interesting to note that these predictions were not “off the tracks” as a label that is both for “courts” and “ASAE” is not, for the most part, classified as being of “another entity” or other non-related label. This is

important because it means the classifiers are capable of distinguishing between the different domains when they are grouped but unable to group them under a single domain.

Moreover, the prediction accuracy for the infraction dataset were also quite high for category I and category II. By observing results for category III, we conclude that most of the documents belonging to this collection are being classified as being of category II which is of the same domain as category III (economic related complaints). This behavior is, most likely, due to the lower number of documents on that group and its similarity which in turn makes the feature learning process harder for the classifiers. The extremely high misclassification rate for category IV might be due to the low number of samples when compared to the other categories or the quality of the data in the documents. Furthermore, it is worth noting that across all three classifiers, predictions for category IV were skewed towards category II, hence a similarity between the two labels could also be an explanation.

It was also possible to conclude that the usage of accuracy as a metric is limited. In this case, we've obtained fairly high results for the labels that present the bulk of complains arriving at ASAE as well as the models not being able to predict labels which denote more than one class with the same performance as individual labels. This leads to high accuracy values which are not representative of the entirety of the predictions made by the classifiers and thus, even though we've achieved good results on the most prevalent labels, misclassification can also lead to complains and reports processing errors and thus negatively impact the speed at which they can be followed and analyzed.

5. Conclusions and Future Work

In this dissertation we have focused on a text classification task involving several algorithms and different methods. Text classification is ever more prevalent on today's work scopes, particularly prevalent in the automatization of tasks which might or might have a business goal in mind. We have tested different preprocessing methods, feature selection and reduction approaches and classifiers on a Portuguese dataset with the goal of predicting two labels present in it. This approach was also aimed at comparing the different procedures done as well as observing if the obtained results can help improve food safety and economic law enforcing by expediting the complains and reports flow at The Economic and Food Safety Authority.

5.1. Main Contributions and Conclusions

We have conducted a comparison study of the performance of different text classifiers prevalent on the text categorization field. These comparisons were established at different levels according to the average accuracy values obtained as well as precision, recall and f1-measure scores and confusion matrices in the last section. We compared preprocessing differences, word normalization methods, feature selection and feature extraction methods as well as hyper-parameter optimization. The dataset used to establish these comparisons was obtained from ASAE and was comprised of roughly 120000 documents which were later reduced due to missing values and split according to the label targeted.

From the results obtained we concluded that the use of a dataset which has been processed for stop words, punctuation, odd characters and numbers shows slightly better results and that only removing stop words has a positive impact on accuracy, mainly in the dataset which contained less labels and overall achieved better results. In addition, we also conclude that the use of stemming carried a negative impact on accuracy for most of the classifiers across both datasets, but the difference was smaller in the dataset which carried higher accuracy values and a smaller number of labels (infractions). In contrast, it was proven to be the most impactful by lessening the time required to train for each classifier. Lemmatization was shown to obtain better results on our dataset while carrying an increase in training times over stemming which was higher the more samples were involved in the process.

We have also concluded that the use of weighting feature selection methods that take in consideration term frequency over documents and the respective labels generally perform better than binary or simple count methods. From the results obtained in the feature selection section we have also observed that the inclusion of more features did not improve the model's performance and that selecting a smaller percentage of the features that best explained the respective labels wielded better results.

Additionally, optimizing the classifiers for their hyper-parameters also improved our model's prediction capabilities. Even though we evaluated different parameters by doing a manual search over the parameter

space for each algorithm, it was possible to observe quite different performances with different parameters and that by doing further testing the results could prove to be even better. In fact, the random forest classifier was observed to have significant shifts in accuracy values with different parameters.

By observing every result and the possible reasoning behind them, the following was concluded:

- It was observed that across all our experiments, the Support Vector machine with linear kernel, Logistic Regression classifier and Random Forest classifier were observed to outperform the other classifiers tested. Naïve Bayes classifiers achieved fairly good performance for much smaller training times which might be helpful in larger datasets. Thus, we have concluded that the use of the Linear Support Vector, Logistic Regression and Random Forest are the most adequate to our datasets.
- Classifiers such as Multinomial Naïve Bayes, Bernoulli Naïve Bayes, Complement Naïve Bayes and K-Nearest Neighbors perform much better with lower feature sizes and the impact is much higher when the same comparison is established in relation to the more complex classifiers such as the linear support vector machine and logistic regression.
- Despite high accuracy obtained by the classifiers denoted above, there are some severe misclassifications in labels that are either sub-represented or that contain information that is shared with other labels such as labels that have more than one class or that complement another label. However, the linear classifiers (LR and SVM) were observed to predict documents with more than one labels associated as being of either the class represented (for the most part).

There have also been some limitations in this work which will be further explained below in conjunction with the future work that to tackle these issues.

5.2. Limitations and Future work

Although we have established comparisons and evaluated the results regarding different steps on a text categorization methodology, there are some drawbacks in the work that has been done. These are as follows:

- In the preliminary results and word normalization the use of the same dataset although targeting different classes makes it harder to extract meaningful conclusions from the use of the different procedures.
- The bag-of-words method is commonly employed in the text classification field but there are more recent and potentially better performing methods for this task. We did not consider the use of semantics or any deeper word meaning as we used a purely statistical approach to text classification.
- In the feature selection section, we considered the use of term frequency, term frequency-inverse document frequency, chi-square and ANOVA and different n-gram ranges to establish

comparison. The use of other well-known feature selection methods could potentially yield different results and extract further conclusions.

- Despite providing results regarding the impact of different values for several hyper-parameters associated with the linear classifiers, k-neighbors and random forest classifier, these could be better correctly selected with the use of semi-automated optimization algorithms such as grid search, random grid search and even better, Bayesian optimization. These restrictions to this work were based on the processing power of the computer used as it crashed often on these tasks.
- One of the most important issues on this work is the data selection, preprocessing and inherently feature selection. We considered this to be of utmost importance from the results we have obtained throughout the dissertation since class imbalance affects the classifiers ability to differentiate classes and create good decision boundaries for these smaller classes as well as the features that are associated with each label. In fact, with less labels it is also likely that the algorithms can extract meaningful information from them which in turn leads to overfitting on the bigger classes. With this in mind, over sampling the smaller classes or under sampling the more prevalent classes could also be considered to tackle this issue. Better feature-label correlation could also improve classification.
- Lastly, the use of other classifiers can also potentially achieve better results. In addition, the use of neural networks, despite warranting an increase in learning times, could potentially allow for better classification or even feature selection.

With these limitations in mind, we propose the following the use of different word representations such as Word2vec as well as further processing of the data to find better correlation between features and labels. In addition, we also propose testing over sampling and under sampling methods as well as applying a multi-label approach to this classification combining the infractions categories field and the jurisdiction which could potentially narrow down the results and improve classifications for the classes which have less correctly predicted samples. Remaining with a multi-class approach, taking in consideration which labels were predicted by their probability could also help improve those classes that have more than one label associated. In addition, the correlation between infractions and jurisdiction could also be used to further improve classification by combining the classifiers prediction on the different labels and categories.

5.3. Final Notes

All the findings and contributions present on this work reveal that the possibility of applying statistical, knowledge discovery and machine learning workflows to tasks that require extensive manual intervention can help facilitate and improve these tasks by either serving as a support or even assisting in

decision making. Despite this, our results indicate that further work needs to be done to improve the task at hand but that the use of text classification can expedite and help in complaints processing and thus indirectly and by means of faster processing and better infraction information extraction, facilitate law enforcing which inevitably leads to better health related problems anticipation, prevention and resolution as well as better economic crime prevention.

References

1. European Parliament and Council. Regulation (EC) N° 178/2002 of 28 January 2002. Off J Eur Communities [Internet]. 2002;31:1–24. Available from: <http://eur-lex.europa.eu/LexUriServ/LexUriServ.do?uri=OJ:L:2002:031:0001:0024:EN:PDF>
2. European Commission. RASFF 2017 Annual Report [Internet]. 2017. Available from: https://ec.europa.eu/food/sites/food/files/safety/docs/rasff_annual_report_2017.pdf
3. EC, EP. Directive 2001/95/EC of the European Parliament and of the Council of 3 December 2001 on general product safety. Off J Eur Communities. 2002;(7):14.
4. Frawley WJ, Piatetsky-Shapiro G, Matheus CJ. Knowledge Discovery in Databases: An Overview. AI magazine. 1992;
5. Han J, Cai Y, Cerconet N. Knowledge Discovery in Databases: An Attribute-Oriented. Proc 18th VLDB Conf Vancouver, Br Columbia, Canada. 1992;
6. Fayyad U, Piatetsky-Shapiro G, Smyth P. Knowledge Discovery and Data Mining: Towards a Unifying Framework. Int Conf Knowl Discov Data Min. 1996;
7. Matheus CJ, Piatetsky Shapiro G, Chan PK. Systems for Knowledge Discovery in Databases. IEEE Trans Knowl Data Eng. 1993;
8. Ristoski P, Paulheim H. Semantic Web in data mining and knowledge discovery: A comprehensive survey. Journal of Web Semantics. 2016.
9. Soibelman L, Kim H. Data preparation process for construction knowledge generation through knowledge discovery in databases. J Comput Civ Eng. 2002;
10. Abidi SRSR, Han CY, Abidi SRSR, Aggarwal CC, Hinneburg A, Keim DA, et al. Cyber security for your organisation starts here. Haemophilia [Internet]. 2018;11(4):487–97. Available from: https://en.wikipedia.org/wiki/Integrated_development_environment%0Ahttps://en.wikipedia.org/wiki/Waterfall_model%0Ahttps://en.wikipedia.org/wiki/Minkowski_distance%0Ahttps://en.wikipedia.org/wiki/List_of_web_browsers%0Ahttp://adultmeducation.com/downloads
11. Fayyad U, Piatetsky-Shapiro G, Smyth P. The KDD Process for Extracting Useful Knowledge from Volumes of Data. Commun ACM. 1996;
12. Simon C, Davidsen K, Hansen C, Seymour E, Barnkob MB, Olsen LR. BioReader: A text mining tool for performing classification of biomedical literature. BMC Bioinformatics. 2019;19(Suppl 13).

13. Hammar K, Jaradat S, Dokoochaki N, Matskin M. Deep Text Mining of Instagram Data without Strong Supervision. In: Proceedings – 2018 IEEE/WIC/ACM International Conference on Web Intelligence, WI 2018. 2019.
14. Manning CD, Schütze H. Foundations of Natural Language Processing. Reading. 2000;
15. Vijayarani S, Ilamathi J, Nithya M. Preprocessing Techniques for Text Mining–An Overview. Int J Comput Sci Commun Networks. 2015;5(1):7–16.
16. Abiteboul S, Antoine E, Stoyanovich J. Viewing the web as a distributed knowledge base. In: Proceedings – International Conference on Data Engineering. 2012.
17. Hearst MA. Untangling text data mining. Proc 37th Annu Meet Assoc Comput Linguist Comput Linguist – [Internet]. 1999;3–10. Available from: <http://portal.acm.org/citation.cfm?doid=1034678.1034679>
18. Mooney, R.J.; Bunescu R. Mining Knowledge from Text Using Information Extraction. Vol. 7, ACM SIGKDD Explorations Newsletter – Natural language processing and text mining. 2005. p. 3–10.
19. Ritter A, Sam C, Mausam, Etzioni O. Named entity recognition in tweets: An experimental study. In: EMNLP 2011 – Conference on Empirical Methods in Natural Language Processing, Proceedings of the Conference. 2011.
20. Hotho A, Nürnberger A, Paaß G. A Brief Survey of Text Mining. LDV Forum – Gld J Comput Linguist Lang Technol. 2005;
21. Gupta V, Lehal GS. A survey of text mining techniques and applications. Journal of Emerging Technologies in Web Intelligence. 2009.
22. Sebastiani F. Machine Learning in Automated Text Categorization. ACM Computing Surveys. 2002.
23. Aggarwal CC, Zhai CX. Mining text data. Mining Text Data. 2013.
24. Tsuruoka Y. Deep Learning and Natural Language Processing. Brain and nerve = Shinkei kenkyu no shinpo. 2019.
25. Kao A, Poteet SR. Natural language processing and text mining. Natural Language Processing and Text Mining. 2007.
26. Srividhya V, Anitha R. Evaluating Preprocessing Techniques in Text Categorization. Int J Comput Sci Appl. 2010;49–51.

27. Powers DMW. Applications and explanations of Zipf's law. In 1998.
28. Prokosch E, Zipf GK. Selected Studies of the Principle of Relative Frequency in Language. Language (Baltim). 2006;
29. Comportamento bibliométrico da língua portuguesa, como veículo de representação da informação. Ciência da Informação. 1973;
30. Porter MF. An algorithm for suffix stripping. Program. 1980.
31. Bird S, Klein E, Loper E. Natural Language Processing with Python: Analyzing Text with the Natural Language Toolkit. Text. 2009.
32. Honnibal M, Montani I. Spacy 2: Natural language understanding with Bloom embeddings, convolutional neural networks and incremental parsing. 2017.
33. Miller GA. WordNet: A Lexical Database for English. Commun ACM. 1995;
34. Rodrigues R, Oliveira HG, Gomes P. Lemport: A high-accuracy cross-platform lemmatizer for portuguese. In: OpenAccess Series in Informatics. 2014.
35. Ning L, Benyu Z, Jun Y, Zheng C, Wenyin L, Fengshan B, et al. Text representation: From vector to tensor. In: Proceedings – IEEE International Conference on Data Mining, ICDM. 2005.
36. Mikolov T, Chen K, Corrado G, Dean J. Efficient Estimation of Word Representations in Vector Space. 2013;1–12. Available from: <http://arxiv.org/abs/1301.3781>
37. Mittermayer MA. Forecasting intraday stock price trends with text mining techniques. In: Proceedings of the Hawaii International Conference on System Sciences. 2004.
38. Lunh HP. The Automatic Creation of Literature Abstracts. IBM J Res Dev [Internet]. 1958;2(2):159–65. Available from: [http://www.di.ubi.pt/~jpaulo/competence/general/\(1958\)Luhn.pdf](http://www.di.ubi.pt/~jpaulo/competence/general/(1958)Luhn.pdf)
39. Jones KS. Index term weighting. Inf Storage Retr. 1973;
40. Pedersen JO, Yang Y. A Comparative Study on Feature Selection in Text Categorization. Proceeding ICML '97 Proc Fourteenth Int Conf Mach Learn. 1997;
41. MESLEH AMA. Chi Square Feature Extraction Based Svms Arabic Language Text Categorization System. J Comput Sci [Internet]. 2007;3(6):430–5. Available from: <http://www.thescipub.com/abstract/?doi=jcssp.2007.430.435>
42. Lee C, Lee GG. Information gain and divergence-based feature selection for machine learning-based text categorization. Inf Process Manag. 2006;

43. Manning CD, Raghavan P, Schütze H. Chapter 8: Evaluation in information retrieval. *Introduct to Inf Retr.* 2009;
44. Forman G. An extensive empirical study of feature selection metrics for text classification. *J Mach Learn Res.* 2003;
45. Elssied NOF, Ibrahim O, Osman AH. A novel feature selection based on one-way ANOVA F-test for e-mail spam classification. *Res J Appl Sci Eng Technol.* 2014;7(3):625–38.
46. Weiss SM, Indurkha N, Zhang T, Damerau FJ. Text mining: Predictive methods for analyzing unstructured information. *Text Mining: Predictive Methods for Analyzing Unstructured Information.* 2005.
47. Dumais S, Platt J, Heckerman D, Sahami M. Inductive learning algorithms and representations for text categorization. In 1998.
48. Lewis DD. Naive(Bayes)at forty: The independence assumption in information retrieval. In: *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics).* 1998.
49. Myaeng SH, Han KS, Rim HC. Some effective techniques for naive bayes text classification. *IEEE Trans Knowl Data Eng.* 2006;
50. Friedman N, Geiger D, Goldszmidt M. Bayesian Network Classifiers. *Mach Learn.* 1997;
51. Larkey LS, Croft WB. Combining classifiers in text categorization. In: *SIGIR Forum (ACM Special Interest Group on Information Retrieval).* 1996.
52. McCallum A, Nigam K. A Comparison of Event Models for Naive Bayes Text Classification. *AAAI/ICML-98 Work Learn Text Categ.* 1998;
53. Rennie JDM, Shih L, Teevan J, Karger D. Tackling the Poor Assumptions of Naive Bayes Text Classifiers. In: *Proceedings, Twentieth International Conference on Machine Learning.* 2003.
54. Ting SL, Ip WH, Tsang AHC. Is Naïve bayes a good classifier for document classification? *Int J Softw Eng its Appl.* 2011;
55. Cortes C, Vapnik V. Support-Vector Networks. *Mach Learn.* 1995;
56. Basu A, Walters C, Shepherd M. Support vector machines for text categorization. In: *Proceedings of the 36th Annual Hawaii International Conference on System Sciences, HICSS 2003.* 2003.
57. Mandowara J, Jain A. Text Classification by Combining Text Classifiers to Improve the Efficiency of

- Classification. *Int J Comput Appl* [Internet]. 2016;6(2):126–9. Available from: <http://rspublication.com/ijca/2016/april16/15.pdf>
58. Joachims T. Text categorization with support vector machines: Learning with many relevant features. In: *Lecture Notes in Computer Science* (including subseries *Lecture Notes in Artificial Intelligence* and *Lecture Notes in Bioinformatics*). 1998.
 59. Liu T-Y, Yang Y, Wan H, Zeng H-J, Chen Z, Ma W-Y. Support vector machines classification with a very large-scale taxonomy. *ACM SIGKDD Explor Newsl*. 2005;
 60. Yang Y, Liu X. A re-examination of text categorization methods. In: *Proceedings of the 22nd Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, SIGIR 1999*. 1999.
 61. Rennie JDM, Rifkin R. Improving Multiclass Text Classification with the Support Vector Machine. *Massachusetts Inst Technol AI Memo* 2001-026. 2001;
 62. Yang Y. An evaluation of statistical approaches to text categorization. *Inf Retr Boston*. 1999;
 63. Wang Y, Wang ZO. A fast KNN algorithm for text categorization. In: *Proceedings of the Sixth International Conference on Machine Learning and Cybernetics, ICMLC 2007*. 2007.
 64. Baoli L, Qin L, Shiwen Y. An adaptive k-nearest neighbor text categorization strategy. *ACM Trans Asian Lang Inf Process*. 2004;3(4):215–26.
 65. Tan S. Neighbor-weighted K-nearest neighbor for unbalanced text corpus. *Expert Syst Appl*. 2005;
 66. Apté C, Damerau F, Weiss SM. Automated Learning of Decision Rules for Text Categorization. *ACM Trans Inf Syst*. 1994;
 67. Harrag F, El-Qawasmeh E, Pichappan P. Improving Arabic text categorization using decision trees. In: *2009 1st International Conference on Networked Digital Technologies, NDT 2009*. 2009.
 68. Cutler A, Cutler DR, Stevens J. Ensemble Machine Learning: Random Forests. *Ensemble Machine Learning: Methods and Applications*. 2012.
 69. Sebastiani F, Sperduti A, Valdambrini N. An improved boosting algorithm and its application to text categorization. In 2000.
 70. Xu B, Guo X, Ye Y, Cheng J. An improved random forest classifier for text categorization. *J Comput*. 2012;

71. Freund Y, Schapire RE. Experiments with a New Boosting Algorithm. Proc 13th Int Conf Mach Learn. 1996;
72. Yang Y, Zhang J, Kisiel B. A Scalability Analysis of Classifiers in Text Categorization. In: SIGIR Forum (ACM Special Interest Group on Information Retrieval). 2003.
73. Aseervatham S, Antoniadis A, Gaussier E, Burlet M, Denneulin Y. A sparse version of the ridge logistic regression for large-scale text categorization. Pattern Recognit Lett. 2011;
74. Tibshirani R. Regression Shrinkage and Selection Via the Lasso. J R Stat Soc Ser B. 1996;58(1):267–88.
75. Kohavi R. A Study of Cross-Validation and Bootstrap for Accuracy Estimation and Model Selection. Int Jt Conf Artif Intell. 1995;
76. Ren Q, Li M, Han S. Tectonic discrimination of olivine in basalt using data mining techniques based on major elements: a comparative study from multiple perspectives. Big Earth Data [Internet]. 2019;3(1):8–25. Available from: <https://doi.org/10.1080/20964471.2019.1572452>
77. Andreas C. Mueller and Sarah Guido. Introduction to Machine Learning with Python: A Guide for Data Scientists. O'Reilly Media, Inc. 2016.
78. Hu N, Zhang T, Gao B, Bose I. What do hotel customers complain about? Text analysis using structural topic model. Tour Manag. 2019;
79. Joung J, Jung K, Ko S, Kim K. Customer complaints analysis using text mining and outcome-driven innovation method for market-oriented product development. Sustain. 2018;
80. Pisarevskaya D, Galitsky B, Ozerov A, Taylor J. An anatomy of a lie: Discourse patterns in customer complaints deception dataset. In: The Web Conference 2019 – Companion of the World Wide Web Conference, WWW 2019. 2019.
81. Tong X, Wu B, Wang S, Lv J. A Complaint Text Classification Model Based on Character-Level Convolutional Network. In: Proceedings of the IEEE International Conference on Software Engineering and Service Sciences, ICSESS. 2019.
82. Tong G, Guo B, Yi O, Zhiwen Y. Mining and analyzing user feedback from app reviews: An econometric approach. In: Proceedings – 2018 IEEE SmartWorld, Ubiquitous Intelligence and Computing, Advanced and Trusted Computing, Scalable Computing and Communications, Cloud and Big Data Computing, Internet of People and Smart City Innovations, SmartWorld/UIC/ATC/ScalCom/CBDCo. 2018.

83. Genc-Nayebi N, Abran A. A systematic literature review: Opinion mining studies from mobile app store user reviews. *J Syst Softw.* 2017;
84. Das S, Mudgal A, Dutta A, Geedipally SR. Vehicle Consumer Complaint Reports Involving Severe Incidents: Mining Large Contingency Tables. *Transp Res Rec.* 2018;
85. Kalyoncu F, Zeydan E, Yigit IO, Yildirim A. A customer complaint analysis tool for mobile network operators. In: *Proceedings of the 2018 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining, ASONAM 2018.* 2018.
86. Guru DS, Ali M, Suhil M. A Novel Term Weighting Scheme and an Approach for Classification of Agricultural Arabic Text Complaints. In: *2nd IEEE International Workshop on Arabic and Derived Script Analysis and Recognition, ASAR 2018.* 2018.
87. Jović A, Brkić K, Bogunović N. A review of feature selection methods with applications. In: *2015 38th International Convention on Information and Communication Technology, Electronics and Microelectronics, MIPRO 2015 – Proceedings.* 2015.
88. Gonçalves T, Quaresma P. Evaluating preprocessing techniques in a text classification problem. *Unisinos.* 2005;
89. Wang P, Xu B, Xu J, Tian G, Liu CL, Hao H. Semantic expansion using word embedding clustering and convolutional neural network for improving short text classification. *Neurocomputing.* 2016;
90. Joulin A, Grave E, Bojanowski P, Mikolov T. Bag of tricks for efficient text classification. In: *15th Conference of the European Chapter of the Association for Computational Linguistics, EACL 2017 – Proceedings of Conference.* 2017.
91. Pranckevičius T, Marcinkevičius V. Comparison of Naive Bayes, Random Forest, Decision Tree, Support Vector Machines, and Logistic Regression Classifiers for Text Reviews Classification. *Balt J Mod Comput.* 2017;
92. Kanaris I, Stamatatos E. Learning to recognize webpage genres. *Inf Process Manag.* 2009;
93. HaCohen-Kerner Y, Dilmon R, Hone M, Ben-Basan MA. Automatic classification of complaint letters according to service provider categories. *Inf Process Manag.* 2019;
94. Deng B, Shao P, Zhao D. E-commerce reviews management system based on online customer reviews mining. In: *CICC-ITOE 2010 – 2010 International Conference on Innovative Computing and Communication, 2010 Asia-Pacific Conference on Information Technology and Ocean Engineering.* 2010.

95. Frank E, Hall MA, Witten IH. The WEKA Workbench Data Mining: Practical Machine Learning Tools and Techniques. Morgan Kaufmann, Fourth Ed. 2016;
96. Rstudio Team. RStudio: Integrated development for R. RStudio, Inc., Boston MA. RStudio. 2016.
97. The MathWorks Inc. MATLAB and statistics toolbox, Release 2010b [computer software]. The MathWorks Inc. 2010.
98. Kotu V, Deshpande B. Predictive Analytics and Data Mining: Concepts and Practice with RapidMiner. Predictive Analytics and Data Mining: Concepts and Practice with RapidMiner. 2014.
99. Agneeswaran VS. Why Look Beyond Hadoop Map-Reduce? In: Big Data Analytics Beyond Hadoop. 2014.
100. Python Software Foundation. Python Language Reference, version 2.7. Python Software Foundation. 2013.
101. JetBrains. Data Science Survey [Internet]. 2018. Available from: https://www.jetbrains.com/research/data-science-2018/?_ga=2.208787601.2035995664.1567563552-665554834.1567563552
102. Ozgur C, Colliat T, Rogers G, Hughes Z, Myer-Tyson EB. MatLab vs. Python vs. R. J Data Sci. 2017;
103. Pedregosa F, Varoquaux G, Gramfort A, Michel V, Thirion B, Grisel O, et al. Scikit-learn: Machine learning in Python. J Mach Learn Res. 2011;
104. Plisson J, Lavrac N, Mladenić DD. A rule based approach to word lemmatization. Proc 7th Int Multiconference Inf Soc. 2004;
105. Diehl CP, Cauwenberghs G. SVM Incremental Learning, Adaptation and Optimization. In: Proceedings of the International Joint Conference on Neural Networks. 2003.
106. Toman M, Tesar R, Jezek K. Influence of word normalization on text classification. Proc InSciT. 2006;
107. Novakovic J. The Impact of Feature Selection on the Accuracy of Naive Bayes Classifier. 18th Telecommun forum TELFOR. 2010;
108. Xia Y, Liu C, Li YY, Liu N. A boosted decision tree approach using Bayesian hyper-parameter optimization for credit scoring. Expert Syst Appl. 2017;
109. Kurdi MZ. Natural Language Processing and Computational Linguistics. 2016.
110. Thanaki J. Python Natural Language Processing.

111. Jo T. Text Mining Concepts, Implementation, and Big Data Challenge. In: Springer. 2019. p. 3–6.
112. Missão, Visão e Valores [Internet]. [cited 2019 Apr 29]. Available from: <https://www.asae.gov.pt/asae20/missao-visao-e-valores.aspx>
113. Feinerer I, Hornik K, Meyer D. Text Mining Infrastructure in R. *J Stat Softw.* 25(5).
114. Faria BM, Pimenta R, Moreira J. Projects of Fisioterapia and Terapia Ocupacional: A Classification Approach using Text Mining in R. In: Rocha Á, Reis LP, Costa MP, Marco Painho, Neto MC, editors. *Sistemas e Tecnologias de Informação. APPACDM. Póvoa de Varzim, Portugal: 4ª Conferência Ibérica de Sistemas e Tecnologias de Informação, Sistemas e Tecnologias de Informação; 2009.* p. 367–72.
115. Chaovalit P, Zhou L. Movie Review Mining: a Comparison between Supervised and Unsupervised Classification Approaches. 2005;00(C):112c–112c.
116. Joachims T. Text Categorization with SVM: Learning with Many Relevant Features. :2–7.
117. Arusada MDN, Putri NAS, Alamsyah A. Training data optimization strategy for multiclass text classification. 2017 5th Int Conf Inf Commun Technol ICoIC7 2017. 2017;0(c).
118. Fayyad U, Piatetsky-Shapiro G, Smyth P. From Data Mining to Knowledge Discovery in Databases. *AI Mag.* 1996;
119. Feldman R, Dagan I. Knowledge Discovery in Textual Databases (KDT). First Int Conf Knowl Discov Data Min [Internet]. 1995;112–7. Available from: <http://www.aaai.org/Papers/KDD/1995/KDD95-012.pdf>
120. Murugan A, Hill C, Nolan T. Practical Text Analytics: Maximizing the Value of Text Data. 46 p.
121. IEEE. Standard for Information Technology – Portable Operating System Interface (POSIX ®) Technical. IEEE Xplore. 2016.
122. nltk.tokenize.treebank – NLTK 3.4.1 documentation [Internet]. [cited 2019 Apr 24]. Available from: https://www.nltk.org/_modules/nltk/tokenize/treebank.html#TreebankWordTokenizer
123. Viera AFG, Virgil J. Uma revisão dos algoritmos de radicalização em língua portuguesa. *Inf Res.* 2007;

TEXT MINING APPLICATIONS TO FACILITATE ECONOMIC AND FOOD SAFETY LAW ENFORCEMENT

Gustavo Magalhães¹, Brigida Monica Faria^{1,2}, Luis Paulo Reis^{2,3}, Henrique Lopes Cardoso^{2,3}

¹*Escola Superior de Saúde – Instituto Politécnico do Porto (ESS-P.Porto), Porto, Portugal*

²*Laboratory of Artificial Intelligence and Computer Science (LIACC), University of Porto, Porto, Portugal*

³*Faculty of Engineering, University of Porto (FEUP), Porto, Portugal*

ABSTRACT

Economic and Food Safety Authority receives on a daily basis reports and complaints regarding infractions, delicts and possible food and economic crimes. These reports and complaints can be in different forms, such as e-mails, online forms, letters, phone calls and complain books present in every establishment. This paper aims to apply text mining and classification algorithms to textual data extracted from these reports and complains in order to help identify if the responsible entity to analyze the content it is, in fact, the Economic and Food Safety Authority. It describes text preprocessing procedures applied to Portuguese text data and its feature extraction. This paper also applies supervised multi-class classification methods such as Naïve Bayes and Support Vector Machine Classifiers to predict a category for each sample. It shows that a non-semantical text mining approach can achieve good results, around 70% of accuracy with the opportunity to be even optimized.

KEYWORDS

Text Mining; Economic and Food Safety; Natural Language Processing; Text Classification, Multi-class Classification.

1. INTRODUCTION

Natural Language Processing (NLP) is a discipline between Computer Science and Linguistics (109). The extraction of features and patterns present on text can be of great value to data analysis and prediction. Natural language understanding systems allow the conversion of human text samples into a representation that a computer can read and understand (Thanaki, 2017). Therefore, text mining is of paramount importance to the NLP research as it is defined as the process of extracting explicit knowledge from unstructured data, the text (111). Text mining encompasses process such as information retrieval, text classification and clustering, in turn, allowing for better natural language perceiving and knowledge extraction (25). Economic and Food Safety Authority is Portugal's national criminal police body responsible for supervising and preventing criminal delicts in the food and other economic sectors (*Missão, Visão e Valores*, 2019). Several reports and complaints are directed to Economic and Food Safety Authority regarding infractions, delicts and possible food and economic crimes. These reports and complaints can be submitted by several ways such as: e-mails, online forms, letters, phone calls and complain books present in every establishment. The main objective of this paper was to classify entries of text related to reports and complaints, present on Economic and Food Safety Authority's data bases, in relation to their respective competence. This is accomplished using data mining techniques (specifically text mining) such as text processing and supervised machine learning. Using text mining and machine learning techniques can help immensely Economic and Food Safety Authority's supervision and prevention mission by speeding up the reports and complaints follow up processes. In turn, this allows for faster response mechanisms and better law enforcing. This classification process is, at the time of writing, completely human based and with no artificial intelligence aid.

2. RELATED WORK

Text mining is an area with several applications and generally refers to the process of extracting information from text. This process involves several phases such as structuring the input text like addition of linguistic features, removal of words, stemming and conversion to lower case letters, deriving patterns within the structured data and

finally the evaluation and interpretation of the results (Feinerer, Hornik and Meyer, 2008). Text classification, text clustering, ontology and taxonomy creation, document summarization and latent corpus analysis are standard techniques in this area. Classical applications in text mining come from the data mining such as document clustering and document classification (114).

Chaovalit, Pimwadee establishes comparisons between machine learning approaches (corpus based) to semantic orientation. Machine learning approaches are largely dependent on preprocessing and feature selection and yield more accurate results. Semantic analysis, while more resource friendly and efficient, isn't as accurate and can be skewed due to how human speech is constructed and interpreted (115). Joachims, Thorsten (Joachims, 1998) uses Support Vector Machines (SVM) for text categorization. Stemming words is suggested, as well as reducing the number of features from the text by computing the number of times these words occur in any given document and excluding those that are considered stop-words. Inversed document frequency (IDF) is also suggested to improve performance. The paper also establishes comparisons between SVMs and other classifiers such as Naïve Bayes and K-Nearest Neighbors (k-NN) and, concludes that SVMs are well suited for text categorization. A study conducted at the School of Economic and Business, Telkom University (117), investigates how training data optimization can improve a multiclass text classification problem. The case study uses human-defined set of categories (supervised learning) and tries to fit sentences to their respective category. The preprocessing steps applied to the text are tokenization (separating words), stop-word removal, stemming and inverse document frequency weighting. The training data is separated into three conditions: each complaint is directly labeled in its category; a label is given for each sentence manually (built upon its keyword) and a sentence with more than one complaint will be separated into suitable categories for each. Both Naïve Bayes and the SVM classifier were observed to show similar results across the three categories although a bigger discrepancy is shown when using the first condition (each complaint is directly labeled).

3. METHODOLOGY

This chapter presents the main phases following the Knowledge Discovery in Databases (KDD) methodology. KDD methodology divides in 9 phases from business understanding (application domain), data set selection, data cleaning and preprocessing, feature selection, model creation, model selection, data mining (pattern recognition), results interpretation and applying the knowledge acquired to the business model (118). Knowledge discovery should be efficient and able to extract valuable information from the data. It should also be efficient in space and in time to cope with large data sets (119). Each of the KDD phases are described below along with their respective procedures.

3.1 Data Preparation and Text Preprocessing

The dataset used totaled 120193 entries ranging from the years 2014 to 2018 and with a total of 49 attributes. While there are many attributes in the dataset, most are directly or indirectly related to the content of the complaint or report. The targeted attribute of this article is the "competence" as it is the first step in automating the complaints and reports process. This attribute is divided into 7 categories: ASAE, ASAE and courts, ASAE and another Entity, another Entity, another Entity and Courts, and Courts. From these entries, 5000 samples were selected for further analysis and classification due to computational requirements. Of these, train and test samples were selected with the test size being 30% of the train size.

Preprocessing encompasses several steps such as tokenization, standardization, cleansing of the text data, removal of stop words and stemming or lemmatization (Murugan, Hill and Nolan, 2019). The objective of text processing is returning cleansed tokens from raw text. Tokens are single words or groups of words that can be fed to machine learning algorithms to be able to understand how the data is structured and how it can help predict attributes related to it (Murugan, Hill and Nolan, 2019). As a first step, the text data was tokenized using the NLTK (Natural Language Toolkit) package. NLTK allows the building of Python programs to work with human language data (Bird, Klein and Loper, 2009). This package uses regular expressions (context-independent syntax that represent a variety of character sets that define a search pattern) to tokenize the text (121). This package also splits standard contractions, treats most punctuation characters as separate tokens, splits of commas and single quotes when these are followed by whitespaces and separates periods that appear at the end of line (122). The tokens retrieved were evaluated against a set of Portuguese stop words and removed if present. The same was done regarding punctuation and special characters as well as any single character. Furthermore, stemming was applied on the tokens. Stemming is the process in which a word is reduced to its stem, base or root. This stem is shared between words from the same etymological family (123). It works by cutting off the end or the beginning of the word, sometimes making the word unreadable or lose its context. Snowball stemmer was chosen due to its

support for the Portuguese language, unlike its previous iteration Porter stemmer (30). The Snowball stemmer is faster and is an improvement over the original Porter stemmer (Porter, 2019).

One popular method of feature weighting is term frequency-inverse document frequency (Tf-idf). “Tf-idf” values terms proportionally to their frequency but reduces their weight based on the frequency they appear on the corpus (inverse document frequency). This is illustrated on equation 1 where “n” stands for the total number of documents in the document set and $df(t)$ the document frequency of t (number of documents in the document collection that contain the term t).

$$idf(t) = \log \frac{1 + n}{1 + df(t)} + 1 \quad (\text{Eq.1})$$

Term frequency-inverse document frequency weighting was chosen due to its ability to inherently give less importance to terms that appear frequently in most documents and a higher weight to those that appear often in a small collection of documents (39). The vocabulary extracted was based on the length of the features extracted.

3.2 Multinomial Class Prediction

Several multi-class prediction algorithms were employed: Multinomial Naïve Bayes (NB), Support Vector Machines (SVM), Decision Trees (DT), Random Forest (RF), K-Nearest Neighbors (k-NN) and Logistic Regression (LR). The results were then compared. A search grid was created which featured options varying from 1-gram (unigrams) to 2-grams (bigrams) and 3-grams (trigrams) with N-gram being a sequence of “N” words. This was done to select the best n-gram range for each classifier. Table 1 shows these results.

Table 1. Best N-grams found for each classifier

Classifier	N-grams
Linear Support Vector	Unigrams and Bigrams
Random Forest	Unigrams
Decision Tree	Unigrams
Logistic Regression	Unigrams and Bigrams
Multinomial Naïve Bayes	Trigrams
K-Neighbors Classifier	Unigrams

The selected algorithms were all implemented from the Scikit-learn packages and their default values were not altered to provide a baseline evaluation. To evaluate the models, several scoring measures were calculated: Accuracy, precision, recall and F1. Accuracy is a ratio of correctly predicted observations to the total observations. It is a simple scoring measure and usable in cases where there is similar ratio of false positives and false negatives. Precision measures how precise the model is according to how many of the positive samples are correctly labeled as positive. Recall represents the ratio of correctly predicted positive observations (True Positives) to the total positive observations. It should be used when there is a high False Negative cost. F1 Score is the weighted average of precision and recall. It takes in consideration False Negatives and False Positives and should be used in cases where there is uneven class distribution. Precision, recall and F1 average were all calculated with the class weight “macro” setting which takes label imbalance in consideration, as is our case. Central Processing Unit time (CPU time) stands for the amount of time that a central processing unit (CPU) was used since instructions were given. It is measured in clock ticks and is more accurate than the measurement of real time elapsed (Gnu, 2019).

4. RESULTS

Out of the six classifiers evaluated, the most accurate was the support vector machine with linear kernel (70.2%), followed by Logistic Regression Classifier (66.3%) and the Random Forest Classifier with 60.5% accuracy. Regarding precision, the most precise model was the K-Neighbors Classifier with results of approximately 70% (69.9%), followed by the SVM with 61.9%, and closely followed by the RF with 58.2%.

The Recall scoring measure was observed to have similar results at the lower end of the spectrum between Multinomial NB and K-Neighbors Classifier (23.8% and 20.3% respectively). The SVM reached predictions of 50% (49.9) with the second best being the Decision Tree (39.2%) which had a slightly higher percentage than the LR with 38.2%. The weighted average between precision and recall (F1 score) calculated for the SVM was 53.1%, the best result of all the classifiers, followed by the RF classifier (40.5%) and DT (40.2%).

All the results outlined above can be seen on table 2.

Table 2. Scoring for each model fitted.

Model	Accuracy	Precision	Recall	F1
Linear Support Vector	0.702	0.619787	0.499354	0.531114
Random Forest	0.604667	0.582377	0.362279	0.405056
Decision Tree	0.561333	0.415173	0.391984	0.401897
Logistic Regression	0.662667	0.44028	0.38222	0.396264
Multinomial Naive Bayes	0.538667	0.565695	0.237704	0.233666
K-Neighbors Classifier	0.484	0.699205	0.202965	0.175576

It was observed that some of the predicted labels had individual F1 scores of 0, which meant that a prediction was not established. This, in turn lowers the weighted F1 score. On one hand, the Support Vector Machine model was able to achieve high F1 scoring, approximately 53%, and high accuracy (70.2%). It was also observed that it had the highest scores for each measure but precision, achieving the second best result with approximately 62%. On the other hand, the SVM model was the second slowest model to train with 2.2 Central Processing Unit time while the fastest model trained was the Random Forest (0.82). The central processing unit time (CPU time) for each model trained was also calculated, as seen on Figure 1.

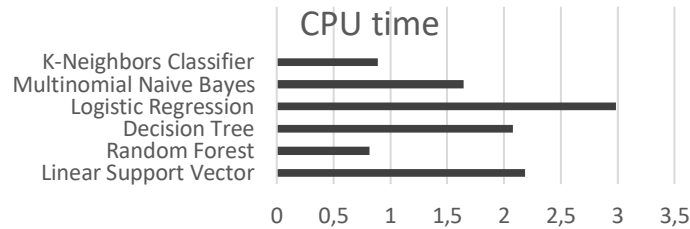


Figure 1. Central Processing Units Time (CPU Time) for each model fitted.

Multinomial Naïve Bayes, although expected to be faster than the Random Forest Classifier, had mixed results regarding CPU time (1.64). Inconsistent results were also observed regarding prediction scoring with low recall and F1 score which can be explained by labels with no predictions. Although baseline results demonstrated high accuracy (~70%) regarding the Linear Support Vector model, F1 scoring was observed to be average (~53%). The LSVC was the second slowest model to train despite achieving the best results. However, CPU time is a measure that can be influenced by several factors external to the models and was only calculated to establish comparisons between the models and how each model fared regarding their computational performance.

5. CONCLUSION AND FUTURE WORK

This paper provides a comparison between established methods of text classification using supervised machine learning regarding their ability to predict categories in a multi-class example and their computational performance. There are many different algorithms that can be used to classify textual data, with varying degrees of success and speed. The algorithms chosen here were selected due to their established capabilities and lower processing requirements. It was concluded that the support vector machine classifier outperformed the others by a significant margin regarding every measure. Since the macro setting was considered when calculating the F1 score, the metric was biased towards the least populated classes. In turn, this can mean that the smaller labels were misclassified. In addition, the SVM classifier was shown to have higher precision than recall which means that the classifier predicts many actual true positive labels as true positives (high precision) but fails to predict a lot of the relevant items from the total amount of relevant items (low recall). Even though the number of tools available to conduct text mining procedures in the Portuguese language is expanding, these are still lacking when comparing to the English language. This, in turn, makes it harder to conduct text mining procedures on Portuguese unstructured data and even more so when conducting lexical analysis due to the lack of Portuguese lemmatizers. Furthermore, grammatical errors are prone to increase the number of features of the data and, because of this, spelling correction applied on the input data can help reduce dimensionality and increase the classifiers' performance but at the cost of preprocessing run times. The process described in this article can be helpful to automate classification tasks. One such case is the attribution of "competence" to each complaint and report directed to Economic and Food Safety Authority. This application has a positive impact on the speed which complaints and reports are analyzed and processed, meaning faster responses, swifter inspections and law enforcing.

ACKNOWLEDGEMENT

The authors would like to thank the IA.SAE – “Inteligência Artificial na Segurança Alimentar e Económica” project, funded by the FCT/MCTES through national funds (PIDDAC), as part of the “Programa Iniciativa Nacional Competências Digitais” e.2030 – INCoDe.2030, enrolled in the National Reform Plan.

REFERENCES

- Arusada, M. D. N., Putri, N. A. S. and Alamsyah, A. (2017) ‘Training data optimization strategy for multiclass text classification’, *2017 5th International Conference on Information and Communication Technology, ICoICT 2017*, 0(c). doi: 10.1109/ICoICT.2017.8074652.
- Bird, S., Klein, E. and Loper, E. (no date) ‘Natural Language Processing with Python - Analyzing Text with the Natural Language Toolkit’.
- Chaovalit, P. and Zhou, L. (2005) ‘Movie Review Mining: a Comparison between Supervised and Unsupervised Classification Approaches’, 00(C), pp. 112c-112c. doi: 10.1109/hicss.2005.445.
- Faria, B. M., Pimenta, R. and Moreira, J. (2009) ‘Projects of Fisioterapia and Terapia Ocupacional: A Classification Approach using Text Mining in R’, in Rocha, Á. et al. (eds) *Sistemas e Tecnologias de Informação*. APPACDM. Póvoa de Varzim, Portugal: 4ª Conf. Ibérica de Sistemas e T. de Informação, Sistemas e Tecnologias de Informação, pp. 367–372.
- Fayyad, U., Piatetsky-Shapiro, G. and Smyth, P. (1996) ‘From Data Mining to Knowledge Discovery in Databases’, *AI Magazine*.
- Feinerer, I., Hornik, K. and Meyer, D. ‘Text Mining Infrastructure in R’, *Journal of Statistical Software*, 25(5).
- Feldman, R. and Dagan, I. (1995) ‘Knowledge Discovery in Textual Databases (KDT)’, *The First International Conference on Knowledge Discovery and Data Mining (KDD)*, pp. 112–117. doi: 10.1.1.47.7462.
- IEEE (2016) *Standard for Inf. Technology — Portable Operating System Interface (POSIX ®) Technical*, IEEE Xplore.
- Jo, T. (2019) ‘Text Mining Concepts, Imp., Big Data Challenge’, in *Springer*, pp. 3–6. doi: 10.1007/978-3-319-91815-0.
- Joachims, T. (no date) ‘Text Categorization with SVM: Learning with Many Relevant Features’, pp. 2–7.
- Jones, K. S. (1973) ‘Index term weighting’, *Inf. Storage and Retrieval*, 9(11), pp. 619–633. doi: 10.1016/0020-0271(73)90043-0.
- Kao, A. and Poteet, S. R. (2007) *Natural language processing and text mining*, *Natural Language Processing and Text Mining*. doi: 10.1007/978-1-84628-754-1.
- Kurdi, M. Z. (2016) *Natural Language Processing and Computational Linguistics*.
- Missão, Visão e Valores (no date). Available at: <https://www.asae.gov.pt/asae20/missao-visao-e-valores.aspx> (Accessed: 29 April 2019).
- Murugan, A., Hill, C. and Nolan, T. (no date) *Practical Text Analytics: Maximizing the Value of Text Data*.
- `nlTK.tokenize.treebank` — *NLTK 3.4.1 documentation* (no date). Available at: https://www.nltk.org/_modules/nltk/tokenize/treebank.html#TreebankWordTokenizer (Accessed: 24 April 2019).
- Porter, M. F. (1980) ‘An algorithm for suffix stripping’, *Program*. doi: 10.1108/eb046814.
- Thanaki, J. *Python Natural Language Processing*.
- Viera, A. F. G. and Virgil, J. (2007) ‘Uma revisão dos algoritmos de radicalização em língua portuguesa’, *Inf. Research*.