University of Surrey

Faculty of Engineering & Physical Sciences

Final Year Project Report (COM 3001)

Applying machine learning techniques to recognize hate speech from offensive language

Name: **Ximena Castro**

URN: **6515820**

Supervisor: **Helen Treharne**

Acknowledgments

I would like to say thank you to my supervisor, all my friends, partner, family, and flatmates who supported me throughout the this project.

Abstract

This project has as objective the recognition of hate speech on Twitter and the differentiation between itself and offensive language. To achieve this objective the research proposes different machine learning models that can be used and have been used in past research papers. Due to this, background research is conducted in the literature review in order to explain different models and how different machine learning approaches work. This paper also discusses how the data set was obtained and its ethical consequences. Furthermore, it provides a detailed description of the pre-processing that said data went through and the implementation of the different models. This helps get results and compare the efficiency of supervised machine learning techniques for text recognition as well as make a comparison between different models. Finally, a conclusion based on the previous research is made.

Contents

Table of figures

1. Introduction

1.1. Problem Background

Nowadays social media is used constantly, this has grown exponentially during the last years reaching a peak during the pandemic and despite this being a useful virtual tool to socialise and communicate it also has downsides such as hate speech. Hate speech can be defined as *"language that is used to expresses hatred towards a targeted group or is intended to be derogatory, to humiliate, or to insult the members of the group"* (Davidson, Warmsley, Macy and Weber, n.d.) *". It is a type of speech that shows a strong intent to cause harm, provoke violence, or encourage hate"* (Mohiyaddeen and Siddiqui, 2021). With time this hateful comments and words have gone out of the screen and now most cases of hateful speech on social media can result in physical attacks: *"A mounting number of attacks on immigrants and other minorities has raised new concerns about the connection between inflammatory speech online and violent acts"* (Lub, 2019). For instance, in Germany there was a correlation between the anti-refugee hate posts in Facebook by the far-right alternative for the German party and the attacks against the refugees.

Twitter has been one of the main social media applications used for discussion and known for users being able to express their opinions about different topics and beliefs, however, it is also known to be one of the apps that features the most amount of hateful speech following Facebook. Twitter defines hate speech as "Promoting violence against or directly attacking or threatening other people on the basis of race, ethnicity, national origin, caste, sexual orientation, gender, gender identity, religious affiliation, age, disability or serious disease (Hateful conduct policy, n.d.). This platform has enforced some strict rules when hateful content is detected, i.e., tweets or images. This goes from downranking tweets and making tweets ineligible for amplification, to permanent suspension being their "most severe enforcement action" (Twitter), where the account of the perpetrator would be removed globally. Despite this, in 2006 Twitter did not have a filtering algorithm yet: "Twitter itself does not have a filtering system regarding whether tweets are positive or negative" [Implementation of Naïve Bayes Classifiers Algorithm on Social Media (Twitter) to the teaching of Indonesian Hate Speech] it wasn't until later on, that a "realtime monitor of online conversations at scale" was developed becoming their "Online Hate Speech Dashboard" earning them the 1st place in the SIGLEX and Microsoft sponsored SemEval competition in 2019. (Twitter, 2019). This dashboard uses the latest machine learning techniques to classify toxic speech which will then bring insights to charities, governments and other entities interested in creating positive content that would diffuse the toxic content. This has an aim of reducing toxicity and improving a positive and healthy conversation online. However, this still is not quite clear and precise in the importance of differentiating hate speech with offensive language.

Following on this, the importance on highlighting this difference, is due to offensive language being quite frequently used in social media with no harmful intention, whereas hate speech is mainly shown against gender, race, sexual orientation, religion, and ethnicity. It is violence and speech to harm a certain group of people. Most of the times the difference will rely on the context certain words are being used on. Some research papers divide the dataset, so they can target hateful speech in a more effective way. This research will be following that example and focus on the efficiency of different models to classify hateful speech and offensive language.

1.2. Project Description

This project will focus on analysing the effectiveness of three different machine learning approaches in identifying offensive language and hate speech in a Twitter dataset. The models will be trained in a supervised manner and comparisons will be made between the factors affecting performance in these models.

This project will begin by defining hate speech and how this has become a problem specifically in social media platforms, followed by the details on the acquisition of the dataset, how it was used throughout the project, the processes followed to work with it, i.e., the pre-processing of the data, a construction of a TFIDF matrix, a description of which models were used and how they were implemented along with a description on how they were tested; finally an analysis and comparison on the result and on the data.

There have been other research papers and classifiers done with this purpose in the past. Some of them did not present a good level of performance on differentiating hateful speech from offensive language (Djuric et al. 2015) this is due to the approach used for doing the classification. The classifier used by Davidson, Warmsley, Macy and Weber, n.d. from which the dataset is being used has a better level of accuracy, however it fails with certain words depending on the context.

The three models use the same dataset. Gaussian Naïve Bayes and logistic regression were implemented first, following the same pre-processing methods and using the scikit-learn documentation on both models. After this a CNN model was implemented from scratch using the same database and following similar but not equal pre-processing techniques.

It is important to highlight that **I did not take part of the Artificial Intelligence (AI) or Natural Language Processing (NLP) modules for this year**, which made it more challenging to implement these models as I had to teach myself the different types of machine learning techniques, which module would fit best the aim of the project and how to best implement these classifiers.

1.3. Aim of the project (overall goal):

Be able to identify and classify hate speech tweets with different models, focusing on the difference between hate speech and offensive language

1.4. Project objectives:

1. Understand the difference between hate speech and offensive language by reading about it in different research papers
2. Carry out a Literature Review of the different approaches for text classification.
3. Identify and learn to use at least one of the approaches for text classification
4. Identify a dataset with social media information that provides both offensive language and hate speech examples.
5. Compare the accuracy of different classifiers that were implemented following different machine learning approaches to identify which approach and model could be more suitable for differentiating hate speech from offensive language.
6. Have a classifier with a good level of accuracy for identifying hate speech

1.5. Report structure

This report is formed by 8 main chapters (excluding appendixes and references).

The first chapter will focus on an introduction to the problem background, i.e., the definition of hate speech, how it has affected social media and therefore our society and an introduction to twitter and its different features and rules to deal with "negative tweets". As well as present a brief description of the project and the main aim and objectives of this dissertation.

Chapter 2 will contain the literature review, where all the different machine learning models are presented, and different papers with different approaches are discussed in order to find the model that could best fit the main objective of this project. This chapter also describes the four main papers used as guidance and resource for this research, as well as a more detailed description of the three models being used in this project, i.e., Naive Bayes, Logistic regression, CNN,

Chapter 3 includes a problem analysis, a design discussion, a description of the choices for the dataset, design choices for the models as well as for the evaluation and a pre-processing design description. In the first section, the problem analysis is discussed, discussing the impact of online hate speech on society and how it could be solved with machine learning techniques. Following this, a design discussion is presented showing the dependencies between tasks, stating which topics should be researched first in order to progress with the rest of the tasks and be able to achieve the aim of this report, as well as the different models used throughout this report are mentioned and a diagram of an overall structure of the project implementation. The design choices for the dataset, include a description of the dataset that was found for this project and the fact that it had already been pre-processed when found, this is because it has been used by another research paper, as well as the reason for choosing this dataset. The pre-processing design, this section discusses the pre-processing that had already been done when the data set was found as well as more pre-processing techniques that were added for purposes of this research. Following this, there will be a discussion of the different models to be used in this project. Finally, the design choices for evaluation, contains the different approaches used to assess if the results of the project were successful, i.e., the description of F1 score and, the reason and way it was used for this case

Chapter 4 focuses on the description and explanation of the implementation of the three different models, i.e., Gaussian Naïve Bayes model, the logistic regression model, and the CNN model. As well as, showing a comparison of the results between these three.

Chapter 5, where a comparison between the project objectives, i.e., recognizing tweets that contain hateful speech, and actual results are discussed.

Chapter 6 shows the testing approach that the code went through to assess if the models were working and were giving out the desired output.

Chapter 7 discusses the ethics of the project, focusing on the confidentiality of the data, the level of harm that the algorithms of this project could cause if being used for the wrong purposes, the social responsibility that this project could carry and the relationship of this project with the British Computer Society Code of Conduct.

Finally, chapter 8 shows a conclusion of the research and some recommendations for future work, including some literature that could have been interesting to work with.

2. Literature Review

2.1. Introduction:

This chapter will focus on explaining the different machine learning techniques such as bag of words, neural networks, naïve bayes classifier algorithms, and the way different research papers use them, as shown in table 2.1, each technique, model and the processes that each dataset went through will be furthered discussed in the potential model to use section, as well as an overview and background information on machine learning in the following section. This chapter will further describe the models that have been used for this research and the reason they were chosen, as well as a detailed description of which papers were used as guidance and how where they used.

2.2. Machine Learning Techniques:

Machine Learning (ML) "is a branch of Artificial Intelligence (AI) and computer science which focuses on the use of data and algorithms to imitate the way that humans learn, gradually improving its accuracy" (IBM cloud education, 2020). Nowadays, machine learning is an important component for business, social media, etc. as it can be used to analyse and predict customers' behaviours. This will be attainable by training algorithms to make classifications or predictions, using "data mining to identify historic trends and inform future models" according to the university of Berkley. Machine Learning has four primary categories or approaches: Supervised Learning, unsupervised learning, semi-supervised learning and reinforcement learning. All of which will be briefly described below as well as some techniques of each approach. The techniques used in the models of this project will further discussed in the following sections.

Supervised learning:

It uses labelled training datasets, where the input and desired output is given to the model in order for it to learn from said dataset and make future predictions

- To avoid overfitting and underfitting, a cross validation process is done, where "as the input data is fed into the model it adjusts its weights until the model has been fitted properly". (IBM cloud education, 2020)
- This approach is heavy reliant on manual labelling which can make it expensive and can lead to functional limits when labelling large datasets.

  Supervised Learning has two different types of algorithms: The first one being classification models (logistic regression, etc), which are used to predict possible outcomes and have a discrete value as output. Whereas the second type is regression models (linear regression, etc) which is used for predicting numerical outcomes which would also be continuous values.

There are different supervised learning techniques that are commonly used, some of which will be explained next:

Neural Networks: The name is given due to this algorithm trying to replicate a human's brain, "mimicking the way biological neurons signal to one another" (IBM Cloud Education, 2020)

This model has three or more node layers, where the first one would be the input layer, then it would have one or more hidden layers and finally an output layer.

All nodes are interconnected and each of them has a threshold and weight allocated, if this value is exceeded by the output of a node, then this node will be activated, and the data will pass onto the next layer. Otherwise, no data will move to the next layer.

Naïve Bayes: A set of supervised learning algorithms, that apply the Bayes' theorem with the assumption of conditional independence between every pair of features. (1.9. Naive Bayes, n.d.). There are three different models under the scikit-learn library[1] that can be used with Naive Bayes: Bernoulli, Gaussian, and a Multinomial type of model. This will be furthered discussed in the next sections.

Linear Regression: It is a linear model, that assumes a linear relationship between the input variables (x) and the output (y) (Brownlee, 2016). It is called linear regression due to the complexity of the regression model; this refers to the number of coefficients that will be used in said model. The output is continuous.

Logistic Regression: Used to predict the probability of a binary outcome. A logistic or sigmoid function is used to calculate the probability in this regression." This function is an S-shaped curve that can take any real-valued number and map it into a value between 0 and 1 but never exactly at those limits" (Brownlee, 2016). The outcome is discrete.

There are three types of logistic regression: Binary logistic regression, Multinomial logistic regression, and ordinal logistic regression. This will be furthered discussed in the next sections.

Random Forest: Creates decision trees on different samples. The classification, i.e., the final output, is based on an average of the result from each decision tree or a majority raking. It doesn't use any formulas.

Support Vector Machine (SVM): It uses classification algorithms for two-group classification problems (Stecanella, 2017). As input a labelled dataset needs to be fed into each category, after this the model should be able to classify the new data.

The way SVM works is it defines a decision boundary to decide which elements should go in each category. When the data is nonlinear a not linear decision boundary is required, which can be achieved by inspecting the data not only in a two-dimensional way but in a third dimension.

However, this might not be the limit for the number of dimensions, and it can get complicated to calculate every vector in the dataset, hence the "Kernel trick" is applied. This maps the space between different interdimensional points by calculating the inner product of them instead of the data points itself.

NLP techniques:

Natural Language Processing techniques, which analyse the structure and meaning of the human language, by processing semantics, syntax, morphology and pragmatics. It can help machines understand big quantities of unstructured text data. These techniques are highly recommended since the differentiation between hate speech and offensive language mainly relies on syntactic features to better identify the targets and intensity

[1] Scikit-learn library: a free software machine learning library for the Python programming language

Unsupervised Learning:

Unlike the first approach, unsupervised learning models do not need an input and output data to learn or human supervision but rather the algorithms become aware of hidden patterns. It discovers the similarities and differences in the input information thus analysing and clustering unlabelled data.

However, it can provide less accurate results.

Unsupervised Learning can also be classified into two categories: Parametric Unsupervised learning, which assumes a "parametric distribution of data", meaning that it assumes that the input data follows a probability distribution on a fixed set of parameters. The second category is non-parametric unsupervised learning, which groups the input data into clusters.

There are different unsupervised learning techniques that are commonly used, some of which will be explained next:

Neural Networks: These nets can be both supervised and unsupervised. For an unsupervised approach they have no desired output which enables the network to design its own pattern for classification. This is furthered explained in the following sections.

K-means clustering: "Clustering attempts to identify a relationship between n observations (data points) without being trained by the response variable" (Tyagi, 2020). In case of the K-means algorithm it explores the number of possible pre-planned clusters in an unlabelled dataset. These clusters are assumed by interpretating the way that an optimized cluster can be expressed in.

For the input an unlabelled dataset should be fed into the model, the algorithm then splits this data into k-number of clusters, then it iterates the process several times in order to find the right clusters. The value of K should be predetermined, this depends on the number of centroids needed for the dataset. "This centroid being an imaginary or real location representing the middle of a cluster." (J. Garbade, 2018)

Probabilistic clustering methods: It is a type of clustering algorithm, which is able to output a prediction and a probability distribution for a set of classes, given an input.

NLP techniques:
These techniques can be used either in supervised or unsupervised learning

Semi-supervised learning:

Is an approach that combines supervised with unsupervised learning. The model is fed a smaller labelled dataset for training and instead will perform feature extractions from a larger unlabelled dataset. The model can develop an understanding of the data on its own.

There is a set of semi-supervised learning techniques that are commonly used, some of which will be explained next:

Deep Learning neural networks like LSTM (long short-term memory): It is a type of recurrent neural network, meaning that it has feedback connections. "It is capable of learning order dependence in sequence prediction problems" (Brownlee, 2021).

An LSTM model contains a set of memory blocks (recurrently connected blocks). Each of them is formed by one or more memory cells, and three other multiplicative units: an input gate, output gate and a forget gate. The cells remember values over arbitrary time intervals, whereas the other units provide equivalents from writing, reading, and resetting operations for the cells, therefore regulating the flow of information that goes in and out of the cell. "The network can only interact with said cells through the gates." (Brownlee, 2021)

SALNet text classifier:

This was developed by Ju-Hyoung Lee et al. where they propose a "semi-supervised bootstrap learning framework" (Lee, Ko and Han, n.d.). Using the attention mechanism, this framework generates a lexicon which is then used for pseudo labelling. "The lexicon can select the correct label among the predictions of the classifier and correctly predict the unlabelled data that a model incorrectly predicts" (Lee, Ko and Han, n.d.).

This is mainly effective for tasks like sentiment analysis

Reinforcement Learning:

Similar to supervised learning, however it does not use a training data set but rather trains the classifier with a trial-and-error method. The successful outputs will be reinforced.

Mainly used "to teach a machine to complete a multi-step process for which there are clearly defined rules" (Burns, 2021)

A comparison between approaches is shown in figure 2.1.

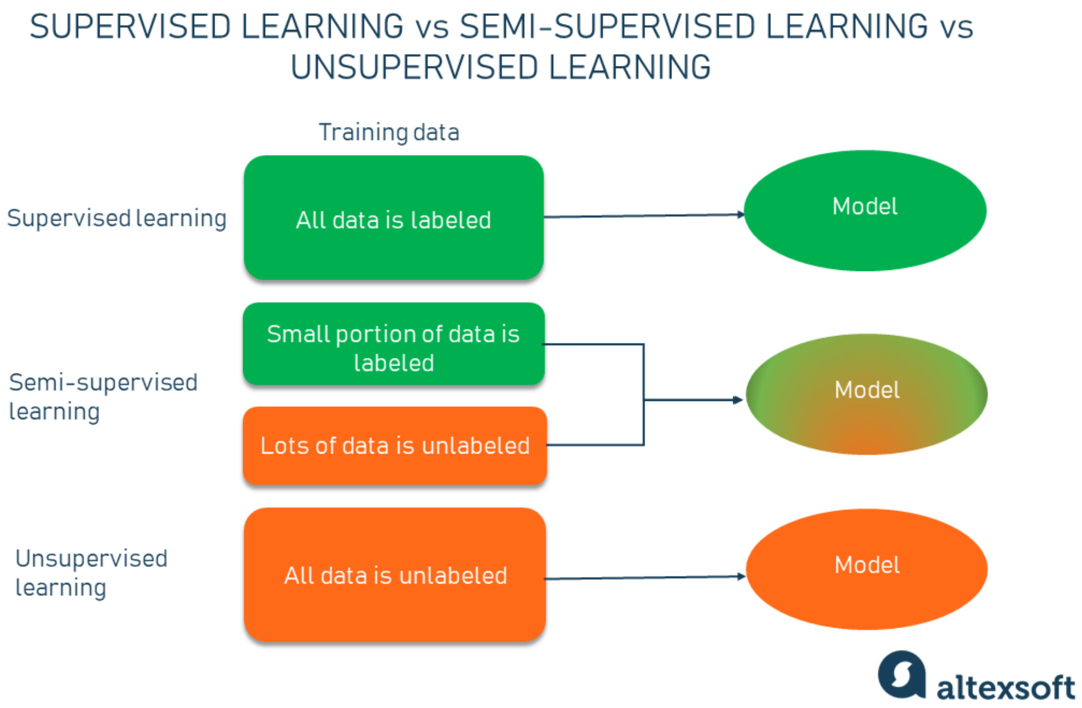


Figure 2-1 Supervised vs unsupervised vs semi-supervised learning

(Adapted from Semi-Supervised Learning, Explained with Examples, 2022)

2.3. Potential Models to Use

This section will focus on some of the potential models that had been used in previous hate speech recognition experiments and in some cases used for the papers previously discussed in the literature review. There are five different machine learning approaches that can be used to identify hate speech: rule-based Linguistic, supervised learning, unsupervised learning, deep learning, and a hybrid approach. The following table shows an overview of each paper which will later be described in detail.

Table 2.1 Machine learning approaches

Approach	Technique	Research Paper/ Author	Main Objective of the research	Results
Rule-based Linguistic approach	VADER sentiment	Used in 2014 by C.J.Hutto et al	For identifying sentiment on Twitter	96% of accuracy
	Rule-based method. Hate speech categorized in religion, nationality and race	Used in 2015 Dennis Glariet al	For identifying sentiment on social media texts	precision of 71.55%
Supervised Learning approach	Naive Bayes Classifier Algorithm with a k-nearest neighbour	Used in 2017 Fatahillah et al.	Identification of hate speech on Instagram	accuracy of 93%.
	Five different classification algorithms, inclusive of K-nearest neighbour, Random Forest, Naive Bayes, Support Vector Machine and Maximum Entropy	Used in 2018 M. Ali Fauzi et al.	Identification of hate speech on Twitter	78.3% accuracy
	TFIDF technique supported with n-gram	Used in 2020 Oluwafemi Oriola et al.	Identification of hate speech from free speech	accuracy of 89.4%
	K-nearest classifier algorithm	Used in 2020, Annisa Brilliani et al	Identification of hate speech on Instagram	accuracy of 98.13%
Unsupervised Learning approach	Semantic-Enhanced Marginalized Denoising Auto-Encoder	Used in 2015 by Rui Zhao et al.	Detection of cyberbulling in Twitter and MySpace	- 84.9% accuracy for Twitter - 89.7% accuracy for MySpace
	k-means clustering algorithm	Used in 2019 by Axel Rodriguez et al.	Sentiment analysis in Facebook data	
	NLP techniques	Used in 2019 by Sylvia Jaki et al	Sentiment analysis of Twitter	F1 score: 84.21% score

Deep Learning	Neural Network models (Bi-GRU-LSTM-CNN, Bi-GRU)	Used in 2019 Tin Van Huynh et al.		70.57% of F1 score
	word2vec based CNN model	Used in 2019 by Gambäck et al.	Detection of hateful speech on Twitter	78.3% of F-score
Hybrid Based	A three-class classification with CNN and BERT	Used in 2020 by Safa Alsafari et al	Hate speech recognition for Arabic social media	75.51% of the F1-score

The first one, a **Rule-Based Linguistic approach**, for this case Hate speech detection uses a linguistic engine that understands the grammar, morphology, and semantics of a specific language. Furthermore, the program adds rules that check for unique core semantic terms in the sentence in order to determine their potential meanings. *"For instance, if we input the keyword "bad." The linguistic engine will automatically search for the terms "terrible/awful/unsatisfactory" as well"* (Mohiyaddeen and Siddiqui, 2021). This approach was used in 2014 by C.J.Hutto et al. by analysing sentiment using VADER (Valence Aware Dictionary and Sentiment Reasoner), this tool is mainly used to identify sentiment on social media. For this a list of lexical features was first created that were highly sensitive to sentiment on social media, then this list was combined with other lexical features that followed five general syntactical and grammatical rules for identifying sentiment intensity. This approach had a 96% of accuracy on Twitter sentiments. Following this in 2015 Dennis Glariet al. used a rule-based method to identify sentiment on social media texts. For this, hate speech was first categorized in three fields: religion, nationality and race. This model was able to "classify and rank the polarity sentiment phrase" (Mohiyaddeen and Siddiqui, 2021). With this approach a precision of 71.55% was achieved.

The second type of approach that can be used is **supervised learning approach**, in 2017 Fatahillah et al. used a Naive Bayes Classifier Algorithm with a k-nearest neighbour classifier in order to identify hate speech on Instagram. The data was collected using the Twitter API and it was labelled manually. The Naïve Bayes Classifier was applied after the pre-processing phase and it achieved an accuracy of 93% (Fatahillah, Suryati and Haryawan, 2017). In 2018 M. Ali Fauzi et al. ensembled five different classification algorithms, inclusive of K-nearest neighbour, Random Forest, Naive Bayes, Support Vector Machine and Maximum Entropy. This data, was like in the previous case, collected using a Twitter API and labelled manually. In the pre-processing phase tokenization, filtering and term-weighing methods were used and bag of words features with TFIDF techniques. The Naïve Bayes classifier algorithm showed a 78.3% accuracy in this case. Following this, in 2020 Oluwafemi Oriola et al. used a TFIDF technique supported with n-gram the dataset was obtained using the Twitter API and then labelled into two categories: Hate Speech "HT" and Free Speech "FS". During the pre-processing stage special characters were removed, i.e., removal of stop words, emojis, etc. TFIDF techniques were implemented for transforming text into vectors and then a support vector machine with n-gram was used for optimization. This classifier had an accuracy of 89.4%. In 2020, Annisa Brilliani et al. used a K-nearest classifier algorithm to identify hate speech on Instagram, collecting the data by using the Instagram API and labelling the data manually. This dataset was divided in two sections, labelled zero or one. First the data was cleaned and then a TF-IDIF technique was applied, following this the K-nearest neighbour algorithm was applied. This had an accuracy of 98.13%.

For **unsupervised learning approaches**, Axel Rodriguez et al. in 2019 used sentiment analysis in a dataset extracted from Facebook, The Graph API was used for extracting the post and comments from Facebook, for removing non pertinent content VADER and JAMMIN were used, stop words and symbols were also removed in the pre-processing stage. Then TF-IDF was employed to convert the documents into vectors. This resulting matrix would go through the k-means clustering algorithm as an input matrix. Sylvia Jaki et al. also used another unsupervised learning approach in 2019, in this case on Twitter, which data was extracted by using the Twitter API. NLP techniques were employed to group the words into similar clusters. After this, spherical k-means clustering, and skip-grams were used to program three clusters of the top 250 most biased terms (Mohiyaddeen and Siddiqui, 2021). In this case F1-score was used as success measurement, achieving a 84.21% score.

On the other hand, a **deep learning approach** was used in 2019 by Gambäck et al. to detect hate speech on Twitter, which data was extracted using the Twitter API. This data was categorized in four: sexism, racism, racism and sexism combined and non-hate speech. Four CNN models were used, this were trained with character n-gram and word2vec random vectors combined (Mohiyaddeen and Siddiqui, 2021). The word2vec based CNN model was the best one to perform with a 78.3% of F-score, a 10-fold technique was used for improving the accuracy of the models.

Finally, a **Hybrid based approach**, was used in 2020 by Safa Alsafari et al. For this study the main objective was to identify hate speech on Arabic social media. The data was extracted using Twitter Search API, this data set was divided in four classes: Religious, Nationality, Gender and Ethnicity. Unnecessary words, i.e., stop words, URLs, etc, were removed during the pre-processing stage like in the previous papers and was followed by the implementation of a three-class classification with CNN and Bert. This paper achieved a 75.51% of the F1-score.

2.4. Related Work:

This project used several research papers as source of information but four of these were used as main guide theoretically and practically.

The paper "Automatic Hate Speech Detection: A literature review" (Mohiyaddeen and Siddiqui, 2021) that focuses on reviewing different machine learning approaches to identify hate speech on social media. As well as, on analysing and comparing of all the different models used on previous research papers. This paper was used mainly for theoretical purposes and to learn some possible models or approaches that had been used and proved to work in the past under different circumstances and with different objectives, after some more research on how the different models worked, it was possible to find the classifiers that would be a better fit for the objectives of this dissertation: Gaussian Naïve Bayes model with a k-nearest algorithm, and TF-IDF techniques with an n-gram.

The second research paper "Automated Hate Speech Detection and the Problem of Offensive Language" (Davidson, Warmsley, Macy and Weber, n.d.), defines a clear difference between hate speech and offensive language, as well as confirms that manual labelling is not the best approach. It also shows the implementation of a model that can identify hate speech from offensive language and not put them both in the same category, supporting the importance of being able to distinguish between the two given the moral and legal implications. This paper was used for the practical purposes of this project and to understand the difference between hate speech and offensive language. The database used for this dissertation was obtained from this paper. It was pre-processed and labelled so that this paper could start directly with the experimentation of different models on said tweets. It was useful as a guide as to which stages should be used when implementing a classifier model, examples of hate speech words and tweets, and what output should be the expected as well as which features should be implemented in models depending on what output is the desired one.

The third research paper "Using Convolutional Neural Networks to Classify Hate-Speech" (), shows the urge for different governments to reduce the amount of cyber bulling and hate speech in social media, pressuring companies like Meta (formerly Facebook) and Google to increase their efforts and resources in stopping this. This paper introduces a Convolutional Neural Network (CNN) approach to classify hate speech as well as a detailed description of the implementation of said model. It also includes a section for experiments with different models, where the data is classified in 4 categories: non-hate-speech, racism, sexism and both. This paper is used mainly for guidance in the practical side of the project to have an idea on the implementation of a neural network model for recognising hateful words in the dataset and how to classify hate speech into different categories such as sexism or racism.

The fourth paper used is "Implementation Of Naive Bayes Classifier Algorithm On Social Media (Twitter) To The Teaching Of Indonesian Hate Speech" (). This paper focuses on Twitter and its lack of measurements to reduce hate speech, it also gives an insight on how to collect, pre-process and classify data with a naïve bayes model. This paper was used for the theoretical understanding of the Naïve Bayes classifiers and practically to check which pre-processing features are best to use for this type of classifier as well as how to use the k-nearest neighbour algorithm with this model.

2.5. Approaches chosen:

There are several approaches to analyse hate speech, as mentioned before, which go from ruled based linguistic approaches, machine learning approaches, deep learning approaches to hybrid approaches.

For this project, three supervised models will be implemented and a comparison on results will be done at the end in order to help understand which approach could be more suitable for text classification or recognition. The first model implemented will be a supervised Gaussian Naïve Bayes model with k-nearest neighbour algorithm, using a labelled dataset.

Following this another supervised learning method will be implemented, creating a Logistic regression model. This will be implemented by using a scikit-learn class and adjusting the different parameters to fit it appropriately to the dataset.

The third model used will be following a supervised learning approach, implementing a CNN model. There are not many research papers where a neural network is being used to differentiate hate speech from offensive language, most of them classify images and if they classify text, it is either neutral or hateful. This is the reason why developing this model might be challenging but also might help furthered the knowledge in this topic. The neural network approach would consist of feeding the classifier labelled examples, following this it would recognise the features that are shared by these examples and after training the classifier with more examples it should be capable of classifying unlabelled data.

All the background information of the algorithms being used by the different implemented models will be discussed in this section:

Naive Bayes approach and k-nearest algorithm

Naïve Bayes background:

Naïve Bayes is a group of supervised learning probabilistic algorithms based on applying the "bayes theorem" and on the "naïve" assumption which concludes that the "effect of an attribute of value in a given class is independent from other attributes" (Fatahillah, Suryati and Haryawan, 2017).

$$P\,(H|X) = \frac{P(X|H)P(H)}{P(X)}$$

Taking this theorem, let X be a data tuple, or in other words the "evidence", H would be the hypothesis and P(H|X) would be the probability of the hypothesis H being correct given the evidence "x", i.e. we would be looking for "the probability that the tuple X belongs to class C, given that we know the attribute description of X" (Fatahillah, Suryati and Haryawan, 2017) .

As said previously, Naïve Bayes is a probabilistic type of algorithm which relies on the Bayes Theorem to predict the tag of a text. In order to do so, first the probability of each tag for a given text will be calculated. Then it will output the tag with the highest one (the highest probability). These probabilities are based on the Bayes' Theorem, i.e., the probability of a feature is based on the prior knowledge of "the conditions that might be related to that feature" (Stecanella, B., 2017).

When comparing different classifiers, the Naïve Bayesian classifier is known since despite its simplicity it has a performance that can be comparable to other more complex or sophisticated classifiers like decision tree and selected neural network classifiers. This is because this theorem supposes that attributes are independent from each other and that therefore there is no conditional feature distributions, each distribution will be independently estimated as a one-dimensional distribution. Therefore, alleviating stemming problems due to the lack of dimensionality.

The Naïve Bayes classifier can show a high accuracy and speed with extensive databases, due to demanding small amounts of training data to estimate the necessary parameters. This classifier has been used in cases such as document classification, spam filtering, and said to work particularly well with Natural Language Processing (NLP) problems, which would suggest the idea that it would also be suitable for filtering through tweets for the purpose of this research paper.

There are three different models under the scikit-learn library[2] that can be used with Naive Bayes: Bernoulli, a binary distribution mainly used for cases when a feature can be either present or absent; Gaussian, used for classification when dealing with continues type of data and it follows a normal distribution; and finally, a Multinomial type of model, used for discrete counts and mainly used for representing frequencies of certain events. In case of this paper the focus will be in the Gaussian Naïve Bayes since that is one of the models that will be employed.

[2] Scikit-learn library: a free software machine learning library for the Python programming language

K-nearest neighbour algorithm:

This algorithm is used for a supervised learning approach, it works “by finding the distances between a query and all the examples in the data, selecting the specified number examples (K) closest to the query, then votes for the most frequent label (in the case of classification) or averages the labels (in the case of regression)” (Harrison, 2018). This algorithm stores all the input data during the training phase and performs an action on the dataset during the classification process, which is a disadvantage when it comes to extensive datasets. When new data or testing data is fed into the model it will find other data that had been previously fed and that has the most similar features and classify it into that category. To implement a KNN model the steps from the following diagram were followed in this research. In order to find the most adequate value for K, the model will follow a trail process where different values will be tested. As the value of K decreases to 1, the predictions can become less stable whereas when the value for K increases, predictions become more stable due to averaging.

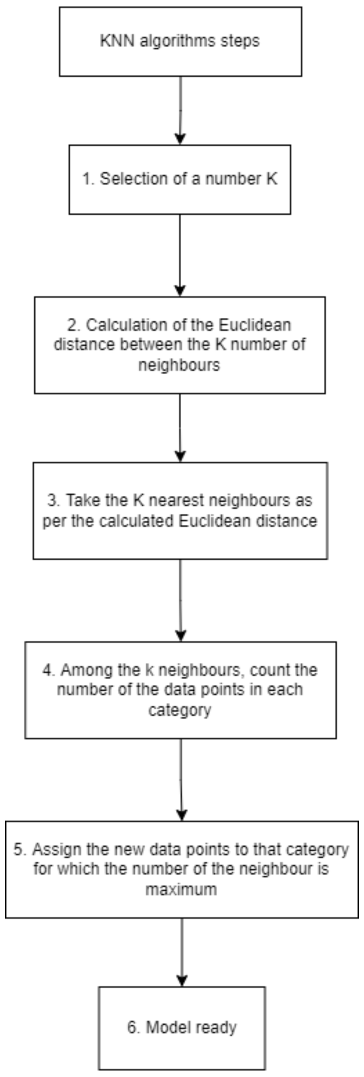

KNN algorithms steps
1. Selection of a number K
2. Calculation of the Euclidean distance between the K number of neighbours
3. Take the K nearest neighbours as per the calculated Euclidean distance
4. Among the k neighbours, count the number of the data points in each category
5. Assign the new data points to that category for which the number of the neighbour is maximum
6. Model ready

Figure 2-2 K-nearest Algorithm

Logistic Regression:

A logistic regression algorithm is a supervised learning technique of classification type. It has the main aim of predicting the probability of a binary and discrete outcome. This algorithm is based in the logistic or sigmoid function, which is an S-shaped curve that can map any value between 0 and 1 except for those. Below is the logistic function, where L is the curve's maximum value, e is the base of natural logarithms, x_0 and x are the values of the Sigmoid's midpoint and k is the logistic growth rate or the steepness of the curve ( ).

$$\frac{L}{1+\ e^{-k(x-x_0)}}$$

When it comes to the input data that is fed into the models, the input values are combined using weights or coefficient values, to predict the output. Each of the values from the input data has a coefficient assigned to them which should be learnt from the training dataset by doing a maximum-likelihood estimation that can be implemented with a gradient descent algorithm (Brownlee, 2016). If the model is able to predict 1 for the default categories and 0 for the other one that means that the data coefficients are at the best estimation, this can be achieved with a minimization algorithm implemented in practice by a numerical optimization algorithm.

A prediction in a logistic regression model is done following the logistic function, hence making the output binary values, meaning that the prediction can no longer be understood as a linear combination of the inputs.

There are three types of logistic regression: Binary logistic regression, where the output is binary i.e., there is only two possible outcomes; a multinomial logistic regression, where there could be multiple outcomes; and an ordinal logistic regression, where the outcome is ordered (Chandrasekaran, 2021). In this research paper, the Binary logistic regression will be the one used.

Neural Networks:

As said previously, this algorithm has as objective to replicate the behaviour of a human's brain. This approach has several layers, including an input, some other layers in between and an output layer, as well as several nodes in each layer, which are all connected. These contain a threshold and weight that in case it is surpassed, it will activate the node and will pass the data to the next layer. This algorithm is "feed-forward", meaning that the data will only move through the layers in one direction

The way neural networks work is as if each node was its own linear regression model, weight, threshold, and output. (IBM Cloud Education, 2020). A simple artificial type of neuron is called "perceptron". This perceptron is the node that takes in several binary inputs (x_1, x_2, etc) and produces a binary output which can be 1 or 0 determined by the whether the weigh sum of the perceptron is higher or lower than the threshold as shown in figure 2.2. The weigh sum being in algebraic terms: $\sum_j w_j x_j$ (Nielsen, 2019).

$$\text{output} = \begin{cases} 0 & \text{if } \sum_j w_j x_j \leq \text{ threshold} \\ 1 & \text{if } \sum_j w_j x_j > \text{ threshold} \end{cases}$$

Figure 2-3 Output from a perceptron

(adapted from Nielsen, 2019)

A perceptron can be used to compute simple logical examples, thus with a network of perceptrons any logical function could theoretically be computed. However, the problem with perceptrons is that if a small change was desired to be done in one of them in order to tweak the overall output, it would mean a change in the behaviour of the rest of the network as well, making the output possibly less efficient than what it originally was. This makes it harder to achieve the desired behaviour for the network by gradually modifying weights and biases. This is the reason to use Sigmoid neurons instead. These are similar to the perceptron; however, they were modified so if a small change on their weight or bias was made it would also result only in a small shift in the output. As the perceptron the receive an input, have a weight and produce an output, nevertheless the input cannot only be 1 and 0 but also any value in between, same with the weight. Regarding the output it is not 0 or 1 but rather σ(w. x + b), σ being the sigmoid function. Thus, the output of a sigmoid neuron would be obtained with the following formula:

$$\frac{1}{1 + exp\,(-\sum_j w_j x_j - b)}$$

(Adapted from Nielsen, 2019)

Where the inputs are x_1, x_2 and their respective weights are $w_j x_j$ and b is the bias. As said before the output will no longer be 1 or 0 but rather it can be any value between those numbers, which will be found by the formula above, this can mean a higher precision of the model.

Neural networks can be used in both supervised and unsupervised learning. During a supervised learning approach, a training data set and desired output is fed into the model, the input patterns are given to the input layer, this is propagated through the net into the output layer. This layer generates an output pattern which is then compared to the desired output, depending on the difference between these two an error value is computed, the greater this is, the more the weight of the values will change (Supervised and unsupervised learning - Neural Networks with Java, n.d.). On the other hand, for an unsupervised approach the nets would have no desired output. Thus, during the training process, the weight values are organised inside a certain range, depending on the values of the given input data. The purpose of this is to group similar weights close together in certain areas of the value range. For this project a supervised CNN will be implemented.

Convolutional Neural Network:

It is a type of neural network; however, this network operates over a volume of inputs. This is the reason this network uses a convolution method, since mathematically speaking, it "combines two relationships to produce a third one. Joining two sets of information" (Choubey, 2020). This convolution slides over the input to extract features by applying a filter i.e., kernel. It is important to take into consideration the parameters of said filter, such as the amount of input that is capable to take at once, and the amount of extent that the input should be overlapped by this is solved by the CNN by using a stride which represents the number of steps the filter is moving for every instance, as shown in figure 2.3, by default, is one.

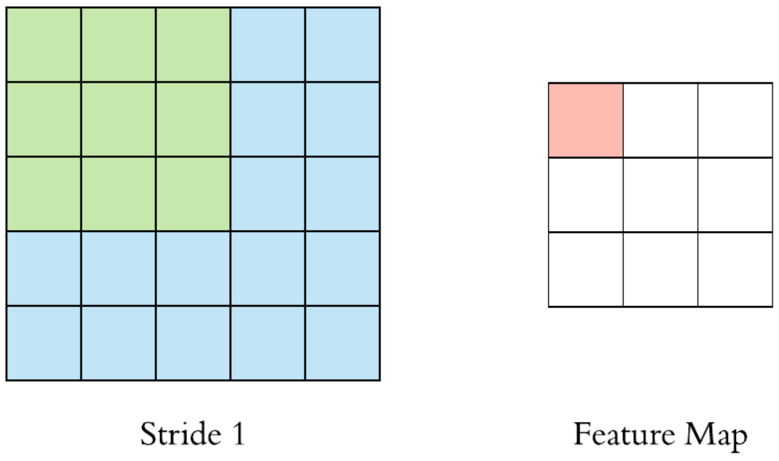


Figure 2-4 Convolution with stride 1

(Adapted from dshahid, 2019)

After this, multiple feature maps will be generated and then an activation function will go over the output and "provide a non-linear relationship for the final output" (Choubey, 2020), such as ReLu. CNN can have five different layers as shown in figure 2.4

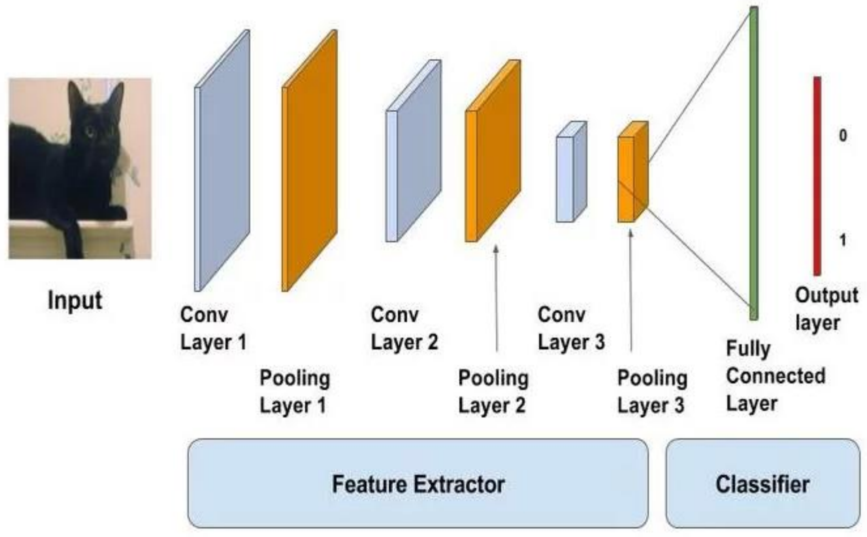


Figure 2-5 Model Visualization

(Adapted from dshahid, 2019)

The input layer, where the data is input, in case of it being images a reshaping of a three-dimensional matrix into a single column should be done, however this process will not be discussed in depth in this report as for this research the network will be working with words rather than images.

The second one is a Convo layer (Convo and ReLu) or feature extractor layer, this is where the features of the input data are extracted. In case of images, i.e., in most cases of CNN, a dot product between the receptive field and the filter will be calculated, which results in a single integer for the output volume. However, as said before this research will be dealing with words and not images, so for this case the words need to be transformed into vectors and then a dot product of said vectors would be done. Nevertheless, this can be tough to achieve between certain vectors as the product might come out as 0 even if those words belong to the same class, this is why the dot product should be done on embedded word vectors which will be more efficient for finding the interrelation of words for a particular class. After this the filter is slid into the next receptive field of the same input by a Stride i.e., the kernel is slid over the embeddings which are then further dimensionally reduced, and the process is repeated until reducing the complexity and computation in the Max Pooling Layer.

The next layer is the pooling layer, this layer's main purpose is to reduce the spatial volume and keeps the relevant information from convolutions, to avoid the model being computationally too expensive. This layer is used between convolutional layers and has as hyperparameters filters (F) (kernel) and stride (S). The fact that it reduces the spatial volume thus reducing the dimensional complexity also stops the CNN model from overfitting data hence stops it from memorizing the training data and rather learn from it.

The fourth layer is the fully connected (FC) layer is used to connect neurons from different layers. The Softmax/ logistic layer it is at the end of the network, the logistic layer is used for binary classification whereas the softmax layer is used for multi-classification. Finally, the output layer which outputs the label that the input has been classified with.

3. Problem Analysis and Design

3.1. Introduction

The purpose of this chapter is to scope out the problem, to discuss the identification of the datasets and a discussion about the models that are to be created as a basis for experimentation. The design of the experiments is based around using the same data set over a number of experiments where each experiment will be based on using a pre-processing technique and a classification model. A summary of the programming environment and libraries chosen to support the design of the experiments will also be outlined.

3.2. Problem analysis discussion

This project discusses the importance of recognizing hateful tweets and differentiating it from offensive language as well as mentions the lack of strong measurements against hateful comments by Twitter and finally proposes different models that could be used in order to tackle this issue.

As discussed previously in the problem background in the introduction chapter, social media is quickly growing and despite its main aim being to connect people, most of the applications also comes with some cons such as users publishing hateful content or speech, specially towards minorities. According to a report published by UNESCO, Twitter removed 1,628,281 pieces of content only between July and December 2020 that were deemed to violate their hate speech policy (UNESCO, n.d.).

Hate speech is treated differently in different platforms like Facebook, Twitter, etc. and each of them has different rules and approaches to try and reduce the number of hateful comments or tweets. For instance, Twitter doesn't allow any hateful speech or images that may portray it. They define it as "promotion of violence against or directly attack or threaten other people on the basis of race, ethnicity, national origin, caste, sexual orientation, gender, gender identity, religious affiliation, age, disability, or serious disease." (Hateful conduct policy, n.d.), and the consequences go from the app removing the content from the platform to suspending accounts. Twitter is one of the applications that is mostly criticised by people since it does not seem to do enough to reduce the amount of hate in the platform.

Likewise, different countries have different approaches on dealing with hate speech where in some of them is illegal and in others it isn't. United States categorises hate speech as free speech, which is protected under the first amendment, even though it has been in several cases "debated in the legal sphere (Davidson, Warmsley, Macy and Weber, n.d.), whereas in Canada, the United Kingdom, Germany and other European countries, hate speech is prohibited by the law, which can result in expensive fines or even imprisonment.

The European Commission set a "code of conduct" in 2016 to enforce a higher commitment from part of social media companies to "remove racist and xenophobic content from their platforms" (Schulze, 2019). This was due to the fact that companies like Facebook and Twitter were only capable of reviewing 40% of the content flagged as hateful in 24 hours, however after 48 hours they had reviewed more than 80% (Kharpal, 2016), which showed that it was possible to review the content faster it would just need stronger efforts from the tech companies. Since then, the monitoring for hateful content has increased, however it might not be enough since most of these companies still rely on manual double checking.

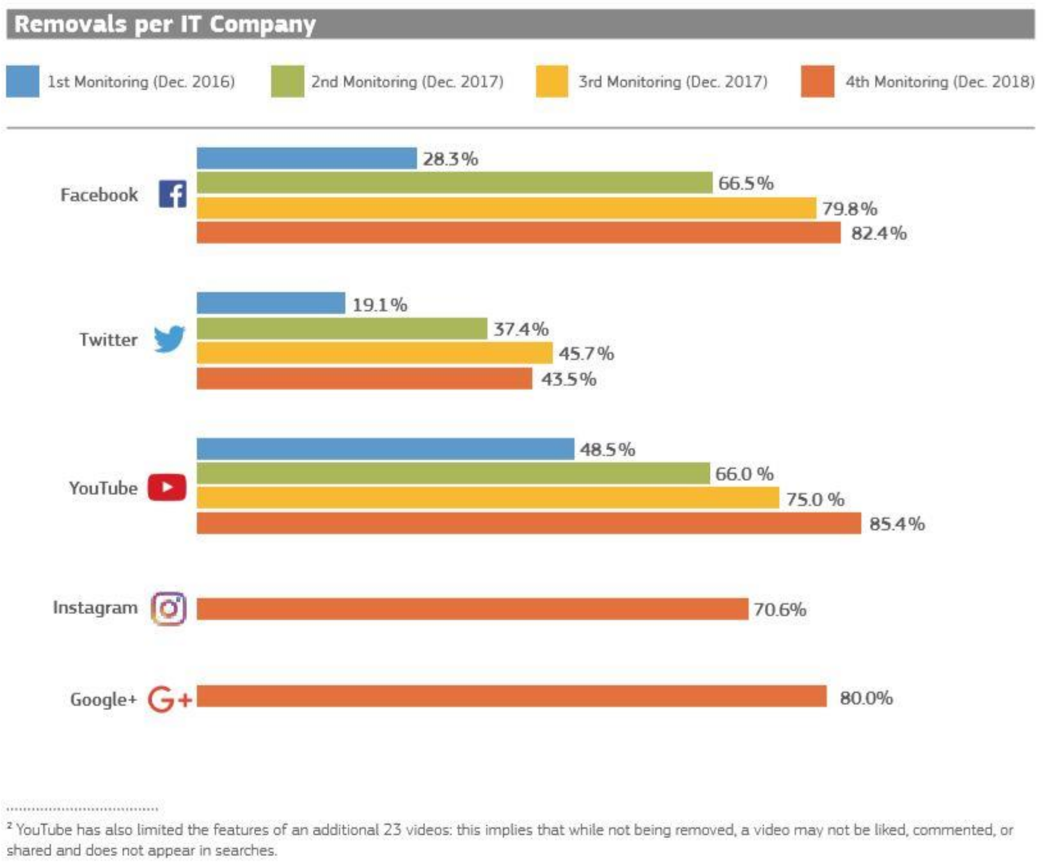


Figure 3-1 Content Removal per IT company

(Adapted from Schulze, 2019)

How ever each platform and each country choose to deal with hate speech, the truth is that it can be dangerous and has been proven to in some cases even turn into physical violence. In 2018, a German firefighter trainee attacked a refugee group house and tried to set it on fire, as most people won't attribute his reasons directly to a social media platform but it has proven that he had isolated himself and was part of several groups in Facebook that expressed negative sentiments towards refugees: "Mr. Denkhaus had isolated himself in an online world of fear and anger that helped lead him to violence" (Taub and Fisher, 2018). This was later furthered studied and proved in a research paper, published by The University of Warwick, "Fanning the Flames of Hate: Social Media and Hate Crime", which concludes that social media may play a role in the propagation of "short-lived bursts in sentiment within a location" and that politicians often don't take in consideration online hate crime in social media, as well as confirming the idea that hateful speech in social media can "motivate real-life action" (Müller and Schwarz, n.d.)

Due to the serious implications and consequences for the use of hate speech that some countries have, the importance of differentiating between hate and offensive language becomes clear. However, UNESO explains this can be hard to identify as there is no internationally accepted term for hate speech (UNESCO, n.d.). In this research the definition of the "Automated Hate Speech Detection and the Problem of Offensive Language" paper will be used in order to keep a consistency throughout the report and with the dataset, since this was extracted from that project. It defines hate speech as "*language that is used to expresses hatred towards a targeted group or is intended to be derogatory, to humiliate, or to insult the members of the group*" (Davidson, Warmsley, Macy and Weber, n.d.). Whereas offensive language is more casually used and depends on the syntactical context of the word, i.e., does not have a hurtful or hatred objective.

Despite other papers testing different machine learning techniques in hateful speech, not many others have researched models that can efficiently differentiate hateful tweets from just offensive language. This shows a gap in the market since, as mentioned previously, it is crucial to make a clear difference between those two, specially nowadays when offensive language can be use casually and an "offensive" word cannot always be intended maliciously or in a hateful manner. Thus, the importance of considering syntactic context and training a classifier with different labels for hateful speech and for offensive language.

This project explores whether this classification process would be more efficient with supervised or unsupervised learning, and by comparing different evaluation methods such as precision, accuracy, and F1-score. It will be using three supervised learning models, these being: Gaussian Naïve Bayes, Logistic regression, and CNN. These models will be all using the same dataset as it is the best suitable for doing a recognition from hate speech and offensive language, this will be furthered explained in the design choices for the dataset.

3.3. Design discussion:

To achieve the set of goals discussed in the introduction, a waterfall development methodology was used for the development of different phases of this report. This is a methodology that follows a sequential development process flowing like a waterfall, where each phase is finished before beginning with the other one. It is important to clarify that no requirements (functional and non-functional) are mentioned in this project as it is research based, however there is a dependency between certain sections and tasks. Diagram 3.1 shows the overall structure of the project

Following that, the introduction of this project had to be done first in order to set the project objective and work towards that goal.
The literature review had to be done before any of the implementation to understand the topic and the different models and the way to implement them, i.e., Learn what constitutes as hate speech and its differences with offensive language to be able to train the classifier with the dataset as well as understand the different machine learning techniques to analyse the dataset.
Finding a suitable dataset that was labelled so it could be used to train and test the supervised models and the pre-processing of data itself, was the next task. As well as this, a flow and process design that the different models would follow also had to be done prior to the implementation to have a better idea of the different implementation stages.
In case of the implementation three supervised models (Gaussian Naïve Bayes and Logistic regression) were implemented first and then a Convolutional Neural Network classifier (CNN), as the first two were simpler to do and could give an idea of the amount of time that it could take to implement different classifiers and be able to plan time accordingly.
These stages had to be done before a comparison between results of different models could be made and thus an evaluation on which model is the best suited for this task. The classifier must be ready and working to be able to go into the testing phase

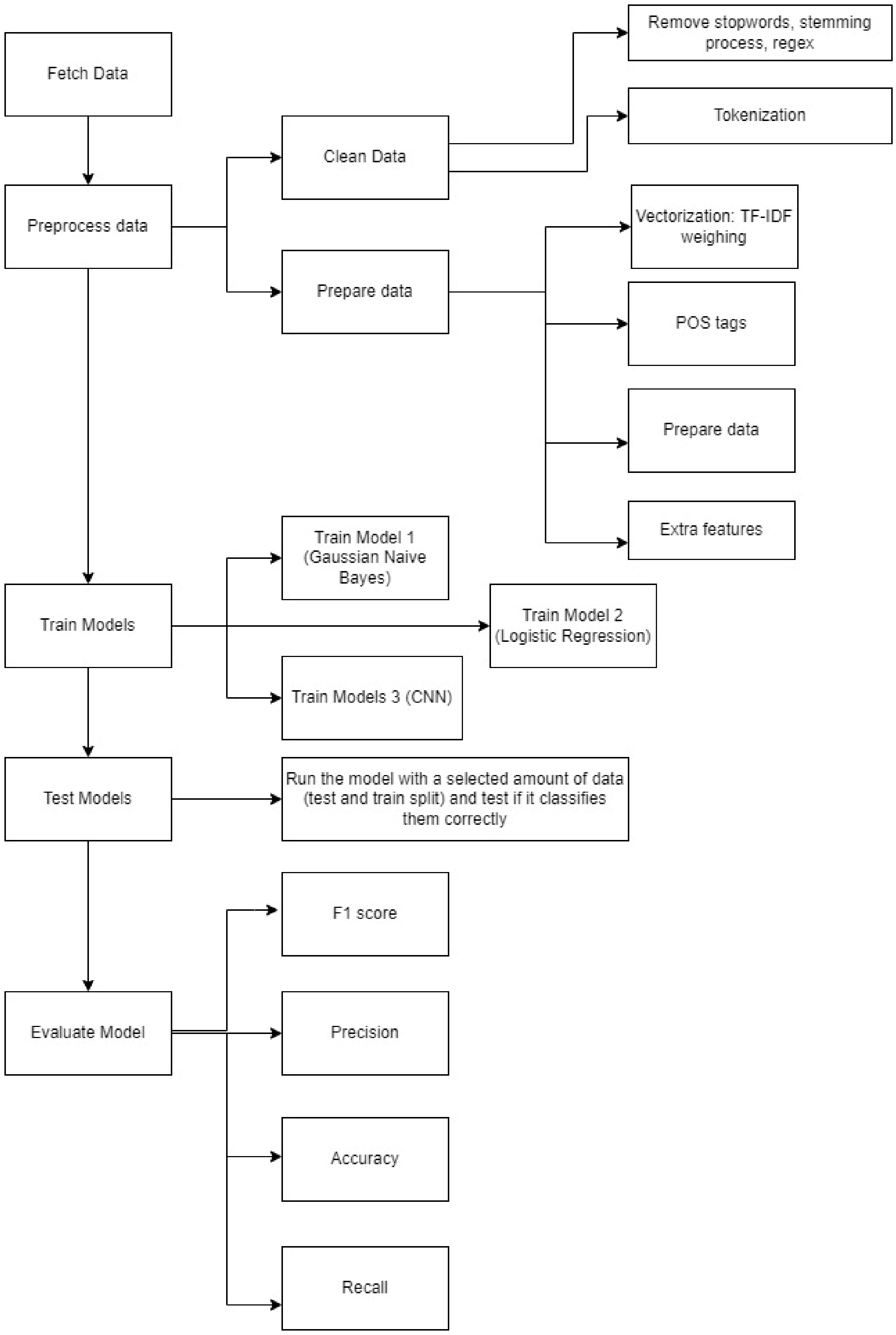

Fetch Data
Preprocess data
Clean Data
Remove stopwords, stemming process, regex
Tokenization
Prepare data
Vectorization: TF-IDF weighing
POS tags
Prepare data
Extra features
Train Models
Train Model 1 (Gaussian Naive Bayes)
Train Model 2 (Logistic Regression)
Train Models 3 (CNN)
Test Models
Run the model with a selected amount of data (test and train split) and test if it classifies them correctly
Evaluate Model
F1 score
Precision
Accuracy
Recall

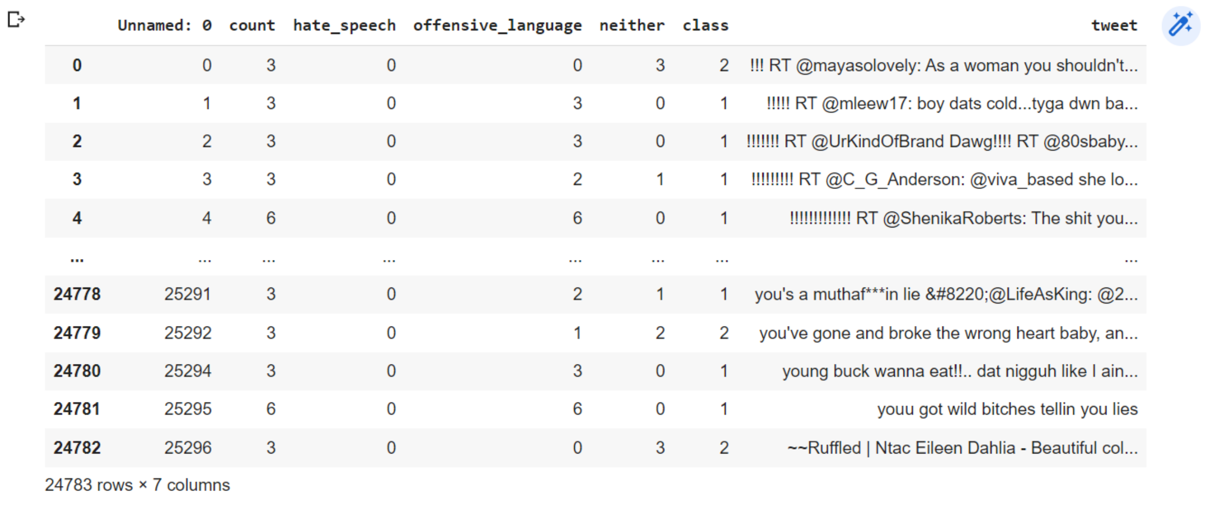
Figure 3-2 Overall Structure of the project implementation

The next sections will discuss the reasons to choose this dataset, pre-processing features, models, and evaluation choice as well as the way the models were implemented.

3.4. Design choices for dataset

As mentioned before the "Automated Hate Speech Detection and the Problem of Offensive Language" paper was used for guidance regarding implementation as it is one of the only research papers that divides the labelling of the dataset between hate speech, offensive language or neither. For this project the dataset of said research paper was used. This data set was already cleaned and labelled. The authors of the research paper paid workers of CrowdFlower (CF) to label each tweet into three different categories: offensive language, hate speech or neither of them. In this case we focus more on the context in which the "hateful" words were used: "Users were asked to think not just about the words appearing in each tweet but about the context in which they were used. They were instructed that the presence of a particular word, however offensive, did not necessarily indicate that a tweet is hate speech" (Davidson, Warmsley, Macy and Weber, n.d.). For this they were given a hate speech lexicon, compiled by Hatebase.org, that contained a set of words that had been identified by online users as hateful. Following this, the Twitter API was used for the extraction of data (tweets) containing terms from said lexicon. The next step was manual labelling of a random sample from the data collected, i.e., the labelling of 25K tweets. This were categorized in three groups by CF: Hate speech, Offensive Language, and neither offensive nor hateful. For precision purposes, this labelling was done by three CrowdFlowers. This dataset results in 24,802 labelled tweets, in which only 5% of these were considered hateful speech. These tweets categorised as hateful speech tend to be most of the times multiple racial and homophobic slurs. The features learned from this results and manual labelling were then taken in consideration when building the model. After this the data was used as training and testing data for their model, it first followed some pre-processing features such as removing unnecessary text (tweeter mentions, URLs, spaces), tokenization, stopwords, stemming. Following this they vectorised words with a TF-IDF matrix and added a POS (part-of-speech) tag to the vectorizer. Finally, they added a sentiment analyser VS. After these steps they would implement their models, train them, and test them using that data.

The dataset has 8 columns as shown in figure 3.2, that show the id for each tweet, the count was the amount of people that labelled the tweet. Then it has three columns for hate_speech, offensive_language or neither, this is for labelling the dataset, where the correct category for a particular tweet is marked and the number of people that labelled it that way will appear in this column. Finally, there is a column to specify the label of the tweet, where **0 is hate speech, 1 is offensive language, and 2 is neither**.



	Unnamed: 0	count	hate_speech	offensive_language	neither	class	tweet
0	0	3	0	0	3	2	!!! RT @mayasolovely: As a woman you shouldn't...
1	1	3	0	3	0	1	!!!!! RT @mleew17: boy dats cold...tyga dwn ba...
2	2	3	0	3	0	1	!!!!!!! RT @UrKindOfBrand Dawg!!!! RT @80sbaby...
3	3	3	0	2	1	1	!!!!!!!!! RT @C_G_Anderson: @viva_based she lo...
4	4	6	0	6	0	1	!!!!!!!!!!!!! RT @ShenikaRoberts: The shit you...
...	...	...	...	...	...	...	...
24778	25291	3	0	2	1	1	you's a muthaf***in lie &#8220;@LifeAsKing: @2...
24779	25292	3	0	1	2	2	you've gone and broke the wrong heart baby, an...
24780	25294	3	0	3	0	1	young buck wanna eat!!.. dat nigguh like I ain...
24781	25295	6	0	6	0	1	youu got wild bitches tellin you lies
24782	25296	3	0	0	3	2	~~Ruffled | Ntac Eileen Dahlia - Beautiful col...

24783 rows × 7 columns

Figure 3-3 Dataset

For this research the dataset will be used for training and testing the model doing a random split using the scikit-learn method "train_test_split", this function splits and optionally subsamples the data. The same dataset will be used for the three models.

Other datasets were also found but were unable to be used because they didn't provide the labels needed to train the model, i.e., didn't have two different labels between offensive language and hate speech, most of them would do a binary classification between hate speech (HS) or free speech (FS).

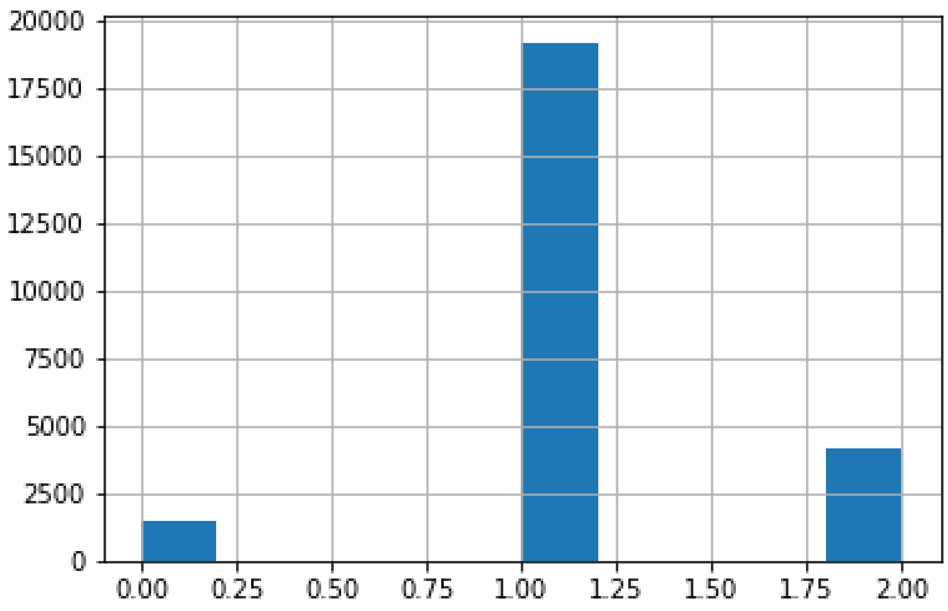


Figure 3-4 Distribution of the data

By looking at Figure 3.3. it is also important to take in consideration that this dataset has a clear class imbalance, where it provides more data labelled as offensive language. This should be stated as it can then impact the evaluation results depending on the metric used.

Like mentioned previously, the data set had already some pre-processing when found, but in the next chapter additional pre-processing added throughout this project will be discussed.

3.5. Pre-processing Design

This section summarises the processing already done in that paper and how it was replicated for this project as well as other pre-processing techniques added throughout this project in order for it to be a better fit for the different models.

3.5.1. Pre-processing for Naïve Bayes and Logistic Regression:

Features*:*

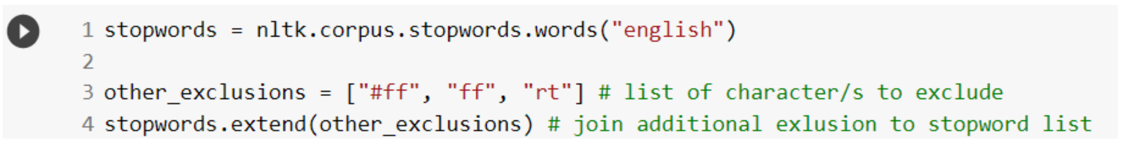
Before implementation the dataset will be pre-processed. This process was heavily based and guided by the paper Automated Hate Speech Detection and the Problem of Offensive Language (Davidson, Warmsley, Macy and Weber, n.d). Throughout the pre-processing the data will undergo stemming processes using PorterStemmer, removing punctuation signs, setting all words to lowercase. In the next stage, a biagram, unigram and trigram features will be created and weighed by its TF-IDF, developing a TF-IDF matrix.
In relevance to the syntactic structure, a Penn Part-of-Speech (POS) tag unigram, bigrams, and trigrams were created with help of NLTK. A sentiment lexicon based on VADER (Valance Aware Dictionary and sEntiment Reasoner) was also used for capturing the syntactic structure. This would assign polarity scores to each tweet.
Some extra features were added in order to capture the quality of each tweet, modified Flesch-Kincaid Grade Level and Flesch Reading ease scores were employed. Binary and count indicators for hashtags, mentions, retweets and URLs were also used, as well as features for obtaining the number of characters, words and syllables.

Stopwords and Stemming*:*

Stopwords such as “the”, “a”, “and”, etc. can be defined as the words that are not relevant for the classification process and should be filtered out when preparing data for implementation, i.e., during the pre-processing stage. Even though the removal of stopwords is not always a necessary step for the aim of this research it is essential as it is an analysis on each word and removing low-level information like stopwords can let the classifier to focus on more important words and moreover reduce the size of the dataset thus reducing the training and testing time.

In the implementation of this projects the package stopwords from the nltk platform will be used. This contains a list of stopwords in 16 different languages, in this case the code will use them in English.

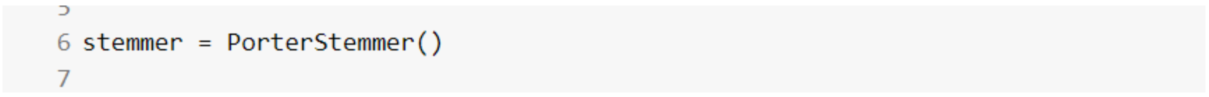
A list of other exclusions, which will be declared afterwards, are also added as stopwords.



```python
stopwords = nltk.corpus.stopwords.words("english")

other_exclusions = ["#ff", "ff", "rt"] # list of character/s to exclude
stopwords.extend(other_exclusions) # join additional exlusion to stopword list
```

Figure 3-5 Stopwords code

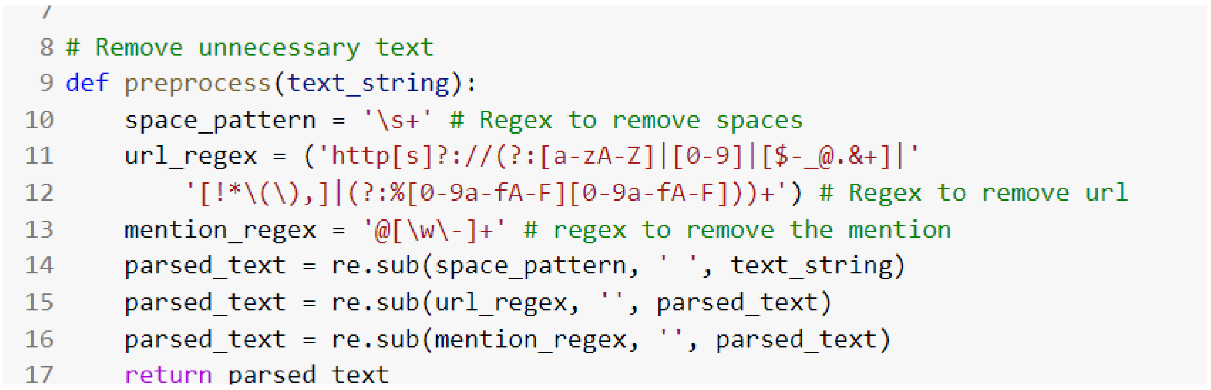
Stemming was also added to the data for reducing certain words to their morphological root or base. This was done by calling the Porter’s Stemmer algorithm, which removes suffixes of words in English. This was the chosen algorithm to stem the words due to it having the less error rate compared to other algorithms of the same type. However, sometimes the reduce morphological based might not be a real word.



```
stemmer = PorterStemmer()
```

Figure 3-6 Stemming Code

On top of removing stopwords, other unnecessary words or elements, like mentions, spaces, URLs were removed by adding regex i.e., regular expression, which helps locate and match a piece of text in this case so it can be removed:



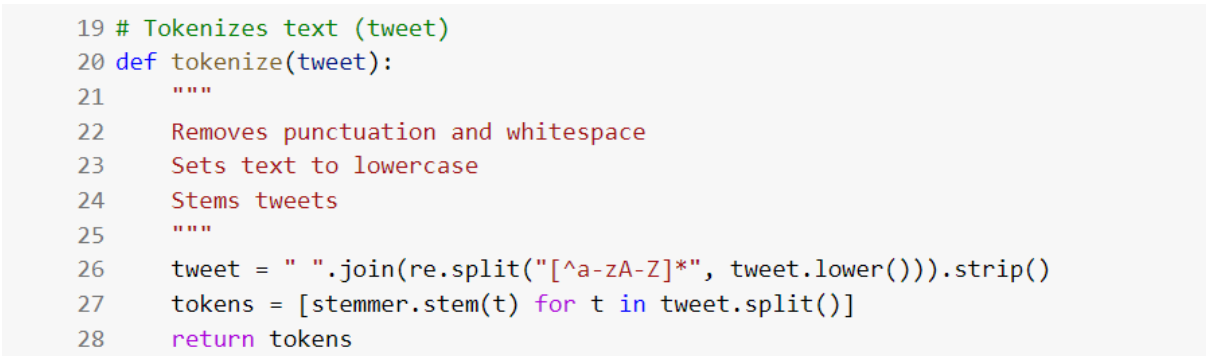
```
# Remove unnecessary text
def preprocess(text_string):
    space_pattern = '\s+' # Regex to remove spaces
    url_regex = ('http[s]?://(?:[a-zA-Z]|[0-9]|[$-_@.&+]|'
        '[!*\(\),]|(?:%[0-9a-fA-F][0-9a-fA-F]))+') # Regex to remove url
    mention_regex = '@[\w\-]+' # regex to remove the mention
    parsed_text = re.sub(space_pattern, ' ', text_string)
    parsed_text = re.sub(url_regex, '', parsed_text)
    parsed_text = re.sub(mention_regex, '', parsed_text)
    return parsed_text
```

Figure 3-7 Regex code

Tokenization:

Through a tokenization process sequences of strings are broken down into pieces, called tokens, this can be words, sentences, or phrases. To separate tokens, it follows a set of heuristics, which states that whitespaces or punctuation marks don't need to be included if its not relevant for the task and that a token includes all characters within contiguous strings. (What is Tokenization? - Definition from Techopedia, 2021)

In this case two separators will be passed onto the tweets through the function split, which allows multiple separators to be passed at once. It will pass on a regex and then modify all tweets to lower case by using "tweet.lower()". Finally, it uses the split function, i.e., "tweet.split()" to split the tweets into sentences and will use a for loop for passing a pattern into it, i.e., for stemming each tweet.



```
# Tokenizes text (tweet)
def tokenize(tweet):
    """
    Removes punctuation and whitespace
    Sets text to lowercase
    Stems tweets
    """
    tweet = " ".join(re.split("[^a-zA-Z]*", tweet.lower())).strip()
    tokens = [stemmer.stem(t) for t in tweet.split()]
    return tokens
```

Figure 3-8 Tokenization

Another tokenization is also done without stemming so it can be later on used for the part-of-speech (POS) tag method.

```python
# Tokenizes text (tweet) without stemming
def basic_tokenize(tweet):
    """
    Removes punctuation and whitespace
    Sets text to lowercase
    """
    tweet = " ".join(re.split("[^a-zA-Z.,!?]*", tweet.lower())).strip()
    return tweet.split()
```

Figure 3-9 Tokenization without stemming

Vectorizations (TF-IDF)

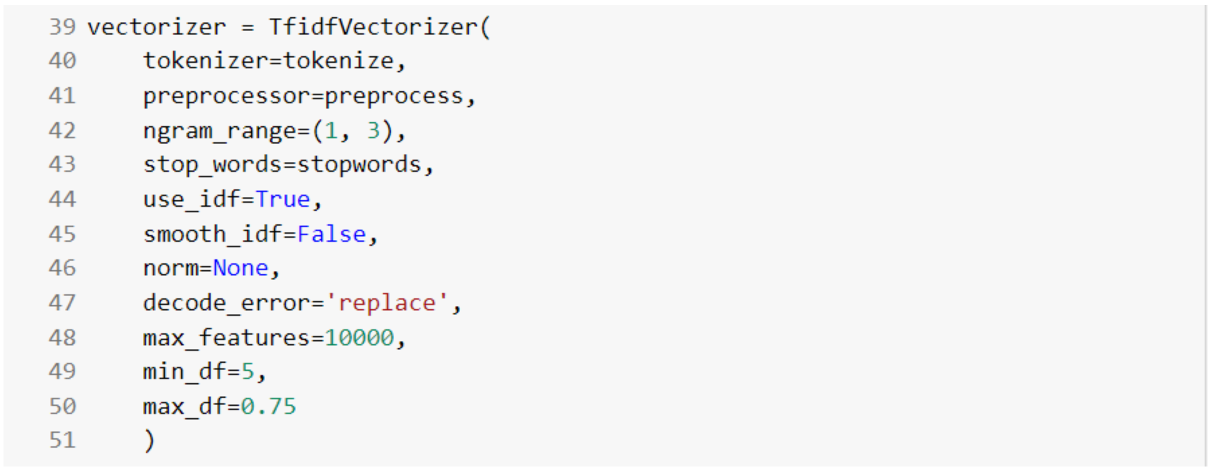
TF-IDF (Term Frequency – Inverse Document Frequency) is based on bag of words (BOW), however in difference to BOW, TF-IDF is better at considering the importance of words in the text. In this approach term frequency and Inverse document frequency are analysed first separately and then together for the model to retrieve the most relevant words/information. The Term Frequency (TF) measures the number of words (frequency) that there is in a document, whereas the Inverse Document Frequency (IDF) measures the importance of a word by providing a weight to each of them. These two measures are used together for complementing each other, since TF does not consider the relevance of each word. However, the disadvantage of TF-IDF is that it is uncapable of capture the semantics, i.e., it is uncapable of recognising synonyms; and it can also be expensive to implement when having a large dataset.

For the implementation of TF-IDF in this project, the "TfidifVectorizer" function from scikit-learn was used, which "converts a collection of raw documents/ words to a matrix of TF-IDF features" (sklearn.feature_extraction.text.TfidfVectorizer, n.d.). This has a list of parameters, some of which were used for this code and can be seen in figure 3.10

The first, second and fourth parameter are set to functions previously discussed: the tokenizer parameter is set to tokenize function, the pre-processing function being set to the regex, and the stopwords parameter is set to the stopwords function.

The n-gram range parameter, establishes the lower and upper boundary for the range of n-values for different n-grams, meaning that all values of n will be min_n<=n <=max_n. In this case, producing and weighing a biagram, unigram and trigram features.

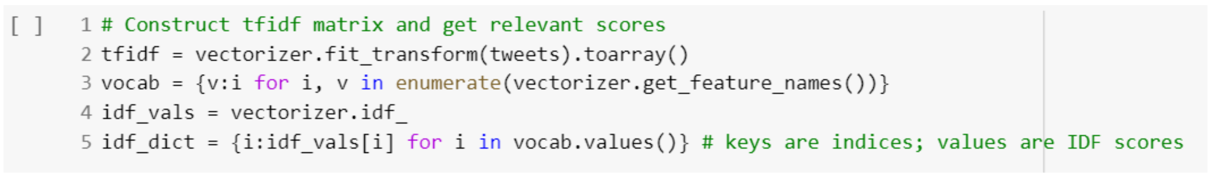
The last three parameters are used for the matrix to build a vocabulary following certain boundaries. "max_features" builds a vocabulary that only considers the amount that it is set to. The parameter max_df is for setting a threshold, so that when building the vocabulary, the matrix ignores any term that has a document frequency higher than said threshold, to avoid stopwords. Finally, the parameter min_df, helps setting a lower limit, i.e., for the matrix to ignore any term that has a lower frequency than the set limit when building the vocabulary. This value is called a "cut-off" according to documentation.



```python
vectorizer = TfidfVectorizer(
    tokenizer=tokenize,
    preprocessor=preprocess,
    ngram_range=(1, 3),
    stop_words=stopwords,
    use_idf=True,
    smooth_idf=False,
    norm=None,
    decode_error='replace',
    max_features=10000,
    min_df=5,
    max_df=0.75
    )
```

Figure 3-10 TF-IDF code

After that, the matrix is transformed to an array format and assigned to the variable tfidf. For the vocabulary it loops through the tweets to add them to the vocabulary and vectorize them. “get_feature_names” is an attribute for array mapping from feature integer indices to feature names. It also calls the idf values by calling the “idf_” attribute of the previous function, which is only possible if the use_idf parameter is said to true like in the previous figure (3.8)



```python
# Construct tfidf matrix and get relevant scores
tfidf = vectorizer.fit_transform(tweets).toarray()
vocab = {v:i for i, v in enumerate(vectorizer.get_feature_names())}
idf_vals = vectorizer.idf_
idf_dict = {i:idf_vals[i] for i in vocab.values()} # keys are indices; values are IDF scores
```

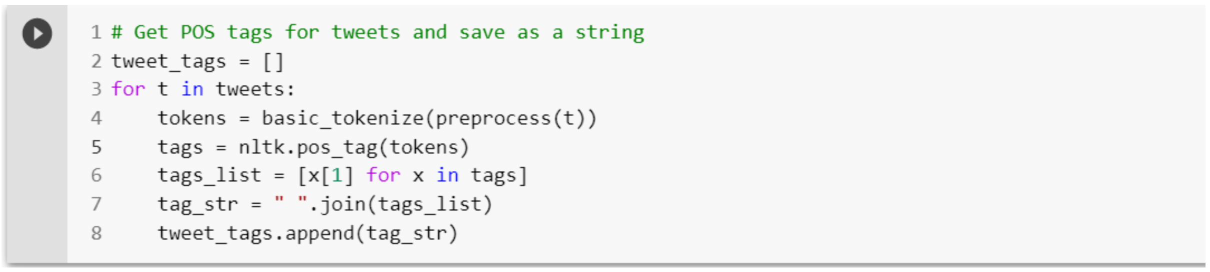
Figure 3-11 TF-IDF matrix code

Penn POS (Part-Of-Speech):

It is a process used for categorizing words in a text, in this case each word of the tweets, in correspondence to a particular part of speech. This is useful when it comes to analysing the semantics of the text, since it describes the characteristic structure of lexical terms within a sentence, thus enabling to make assumptions of the semantics of the text (Pykes, 2020). POS tags are commonly used in corpus searches and text analysis algorithms, in this case the use of POS with syntactic structure could be applied: “*<intensity > <user intent > <hate target >, e.g. “I f*cking hate white people”* (Silva et al. 2016).*”* [Automatic Hate Speech Detection: A Literature Review].

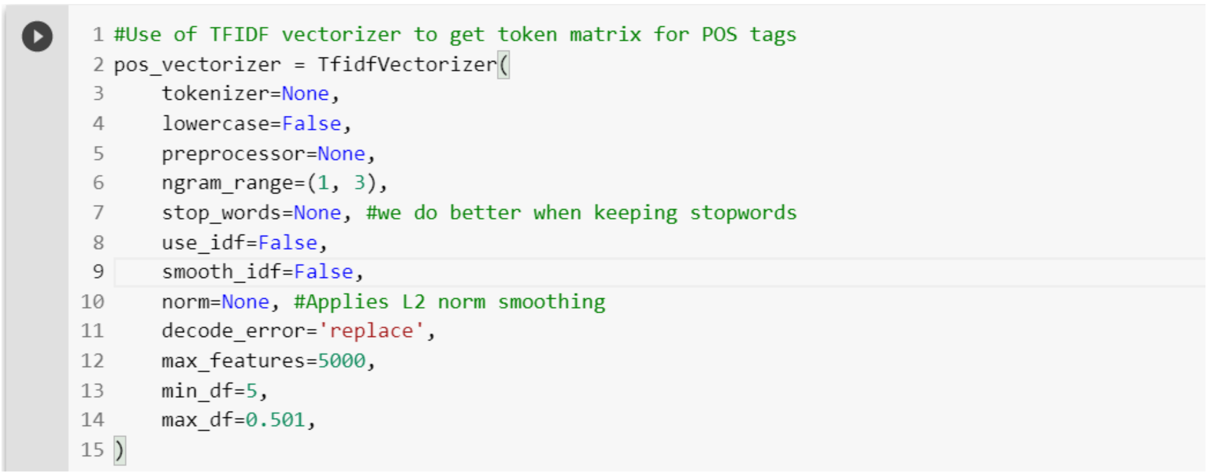
For this case the package pos_tag from the nltk platform (platform used for building python programs and connecting them with human language data) will be used. According to NLTK documentation, “this processes the sequence of words and attaches a part of speech tag to each word” (5. Categorizing and Tagging Words, n.d.).

In the code, it creates an empty array to input the tags that the different tweets will have. It loops through all the tweets to basic_tokenize them (tokenization process without the stemming) and adds the regex method “preprocess” to them. It then sets the output of the nltk.pos_tag function as tags and creates a list of them and outputs the POS tags in one string “tag_str”, adding a whitespace in between each tag to be able to recognize them. Finally, it adds said list to the empty array “tweet_tags” that was previously created.



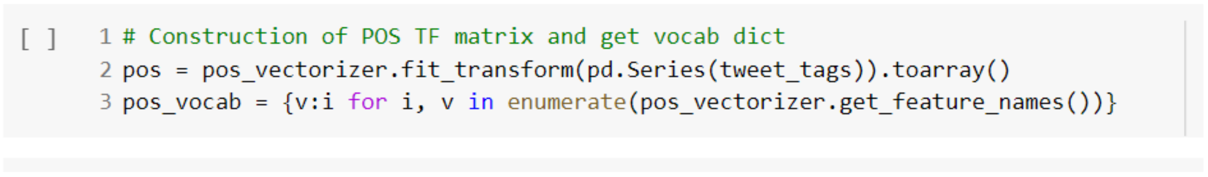
```python
# Get POS tags for tweets and save as a string
tweet_tags = []
for t in tweets:
    tokens = basic_tokenize(preprocess(t))
    tags = nltk.pos_tag(tokens)
    tags_list = [x[1] for x in tags]
    tag_str = " ".join(tags_list)
    tweet_tags.append(tag_str)
```

Figure 3-12 POS tags



```python
#Use of TFIDF vectorizer to get token matrix for POS tags
pos_vectorizer = TfidfVectorizer(
    tokenizer=None,
    lowercase=False,
    preprocessor=None,
    ngram_range=(1, 3),
    stop_words=None, #we do better when keeping stopwords
    use_idf=False,
    smooth_idf=False,
    norm=None, #Applies L2 norm smoothing
    decode_error='replace',
    max_features=5000,
    min_df=5,
    max_df=0.501,
)
```

Figure 3-13 POS tags with TFIDF vectorizer

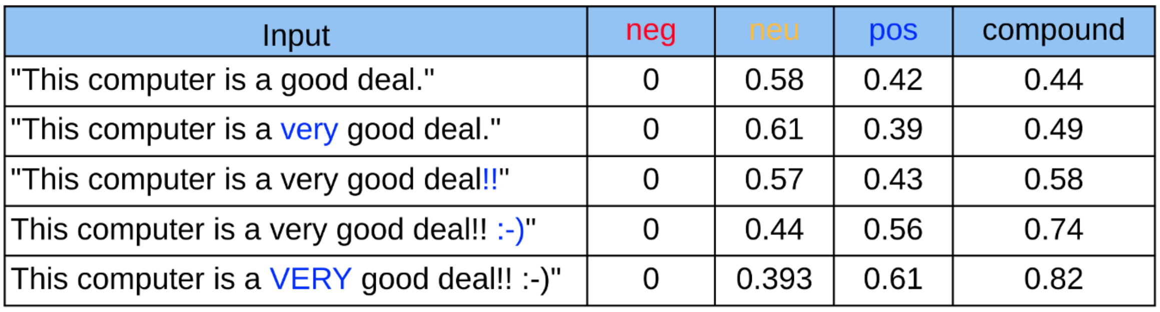


```python
[ ]
# Construction of POS TF matrix and get vocab dict
pos = pos_vectorizer.fit_transform(pd.Series(tweet_tags)).toarray()
pos_vocab = {v:i for i, v in enumerate(pos_vectorizer.get_feature_names())}
```

Figure 3-14 POS TF matrix

Sentiment Analyzer (VS):

Sentiment Analyzer is a tool for analysing the tone, intent and emotions behind text, tweets. One of the most used is VADER (Valence Aware Dictionary for sEntiment Reasoning). This is a natural processing (NLP) algorithm that expresses sentiment polarity and intensity. It achieves this by combining the sentiment lexicon approach with grammatical rules and syntactical conventions. Following heuristic rules of capitalization, punctuations, contrastive conjunctions and adverbs, the VADER algorithm is capable of calculating polarity scores: positive, negative, neutral and compound as shown in the figure below.

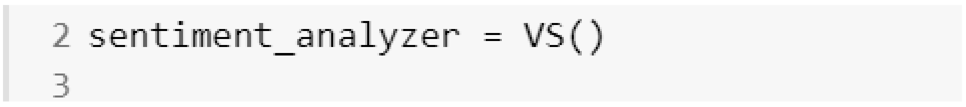


Input	neg	neu	pos	compound
"This computer is a good deal."	0	0.58	0.42	0.44
"This computer is a very good deal."	0	0.61	0.39	0.49
"This computer is a very good deal!!"	0	0.57	0.43	0.58
This computer is a very good deal!! :-)"	0	0.44	0.56	0.74
This computer is a VERY good deal!! :-)"	0	0.393	0.61	0.82

Figure 3-15 Polarity Scores example

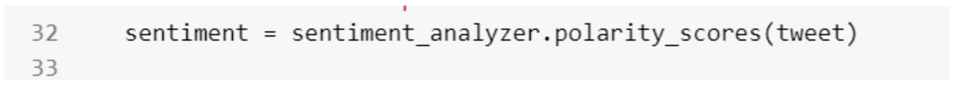
(adapted from Ma, 2020)

For this project, an open-source package called "vader" from NLTK (Natural Language Toolkit) will be used. More precisely, the code will be using the package "SentimentIntensityAnalyzer" from nltk.sentiment.vader, defining it as VS() and then assigning it to the variable "sentiment_analyzer", which will be later on used to show the negatice, positive, neutral and compound scores of each tweet.



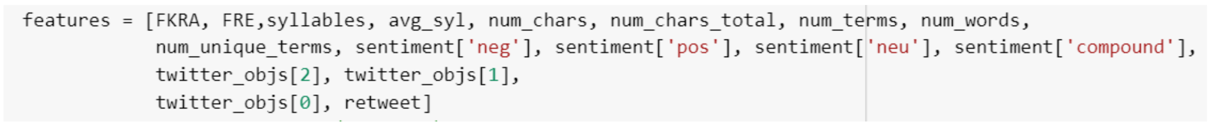
```
sentiment_analyzer = VS()
```

Figure 3-16 Sentiment Analyzer implementation



```
    sentiment = sentiment_analyzer.polarity_scores(tweet)
```

Figure 3-17 Creating the polarity scores

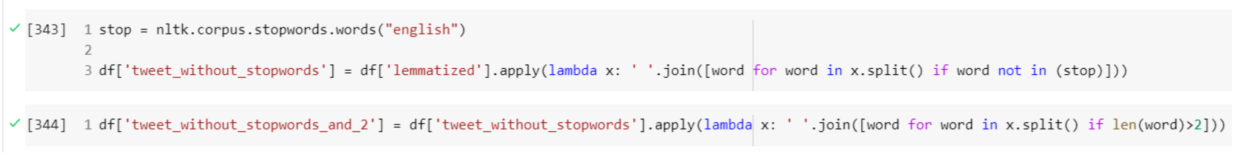


```
features = [FKRA, FRE,syllables, avg_syl, num_chars, num_chars_total, num_terms, num_words,
            num_unique_terms, sentiment['neg'], sentiment['pos'], sentiment['neu'], sentiment['compound'],
            twitter_objs[2], twitter_objs[1],
            twitter_objs[0], retweet]
```

Figure 3-18Adding the polarity scores feature

3.5.2. CNN special pre-processing:

An extra pre-processing step was added when implementing the CNN model, a more specific type of vectorization to transform the words into numbers expected by this model such as the ones received when working with images. This vectorization process had to turn words into embedded vectors by



```
✓ [343]  stop = nltk.corpus.stopwords.words("english")

         df['tweet_without_stopwords'] = df['lemmatized'].apply(lambda x: ' '.join([word for word in x.split() if word not in (stop)]))

✓ [344]  df['tweet_without_stopwords_and_2'] = df['tweet_without_stopwords'].apply(lambda x: ' '.join([word for word in x.split() if len(word)>2]))
```

Figure 3-19 CNN stopwords

According to a Medium article "How to build a (quasi) real-time hate speech classifier for Twitter" (Tortella, 2020), it has been proven that the sentiment of a tweet can also be measured by its length, meaning that the longest the tweet is the more probability it has of having a positive or neutral message as shown in the figure below.

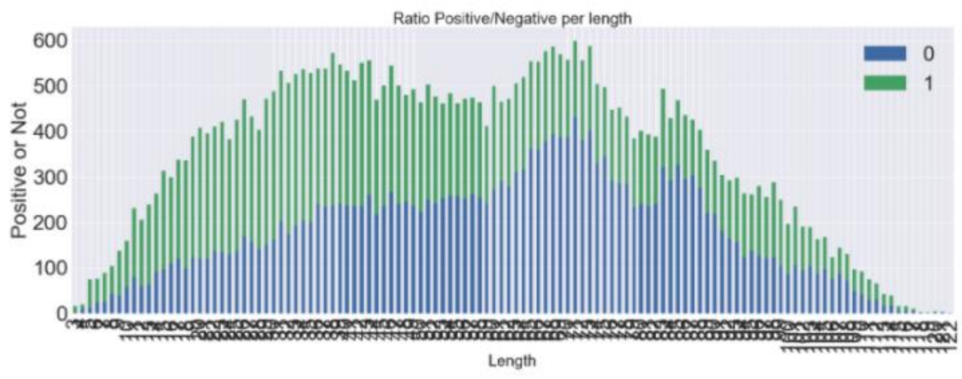


Figure 3-20 Tweet ratio positive to length

(Adapted from Tortella, 2020)

It also shows that the greater number of handles that a tweet has the more prone it is to have hateful speech.

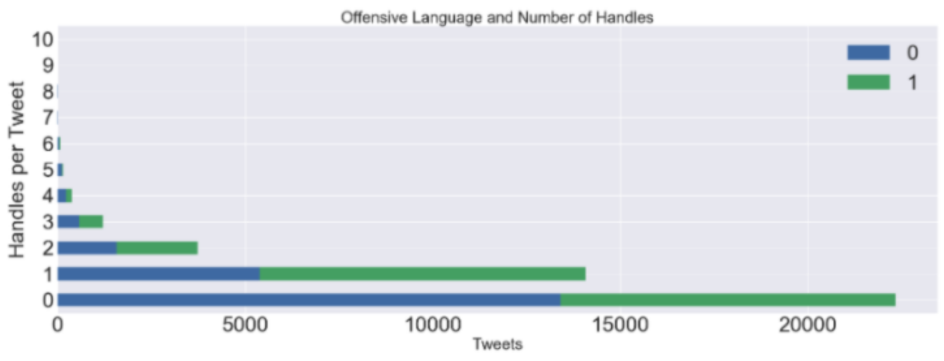
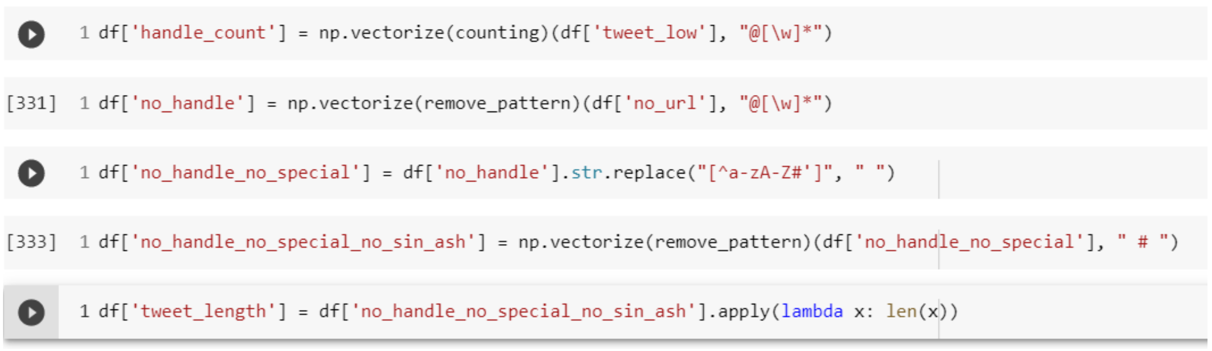


Figure 3-21 Number of handles and hate speech

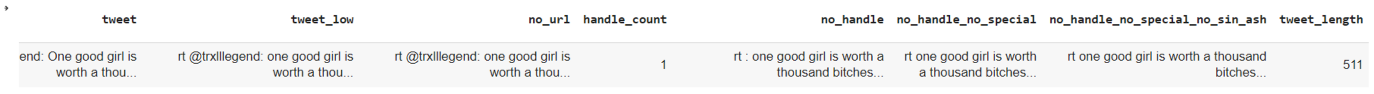
(Adapted from Tortella, 2020)

Therefore, during the implementation of the model, these aspects will be taken into consideration and count the number of handlers in each tweet as well as the length of each tweet. The code for this is in the following figure as well as the data after those processes.



```
df['handle_count'] = np.vectorize(counting)(df['tweet_low'], "@[\w]*")

[331] df['no_handle'] = np.vectorize(remove_pattern)(df['no_url'], "@[\w]*")

df['no_handle_no_special'] = df['no_handle'].str.replace("[^a-zA-Z#']", " ")

[333] df['no_handle_no_special_no_sin_ash'] = np.vectorize(remove_pattern)(df['no_handle_no_special'], " # ")

df['tweet_length'] = df['no_handle_no_special_no_sin_ash'].apply(lambda x: len(x))
```

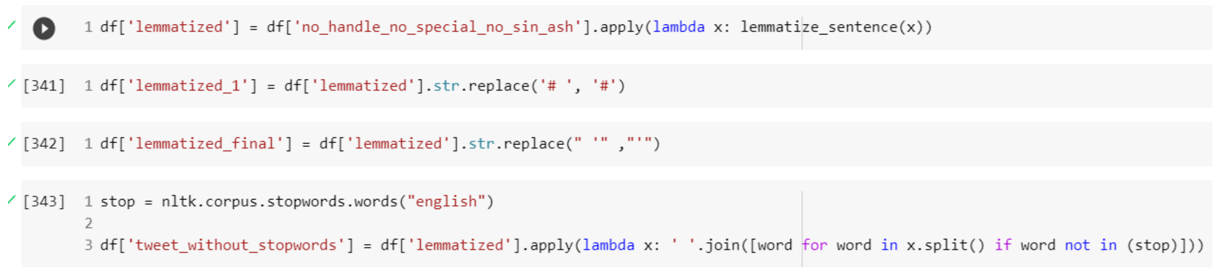
Figure 3-22 Handle count code



tweet	tweet_low	no_url	handle_count	no_handle	no_handle_no_special	no_handle_no_special_no_sin_ash	tweet_length
end: One good girl is worth a thou...	rt @trxlllegend: one good girl is worth a thou...	rt @trxlllegend: one good girl is worth a thou...	1	rt : one good girl is worth a thousand bitches...	rt one good girl is worth a thousand bitches...	rt one good girl is worth a thousand bitches...	511

Figure 3-23

The tweets were also lemmatized, for analysing the morphological and syntactical side of each word of the tweet.



```
df['lemmatized'] = df['no_handle_no_special_no_sin_ash'].apply(lambda x: lemmatize_sentence(x))
```

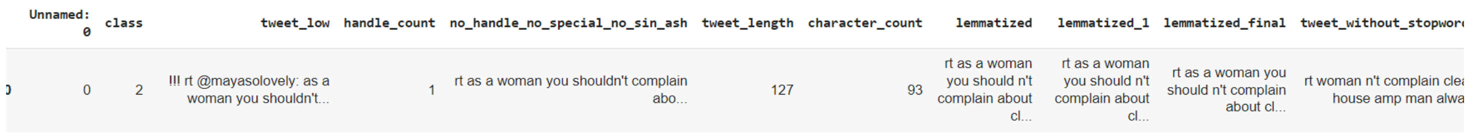
```
[341]
df['lemmatized_1'] = df['lemmatized'].str.replace('# ', '#')
```

```
[342]
df['lemmatized_final'] = df['lemmatized'].str.replace(" '" ,"'")
```

```
[343]
stop = nltk.corpus.stopwords.words("english")

df['tweet_without_stopwords'] = df['lemmatized'].apply(lambda x: ' '.join([word for word in x.split() if word not in (stop)]))
```

Figure 3-24 Lemmatization of tweets

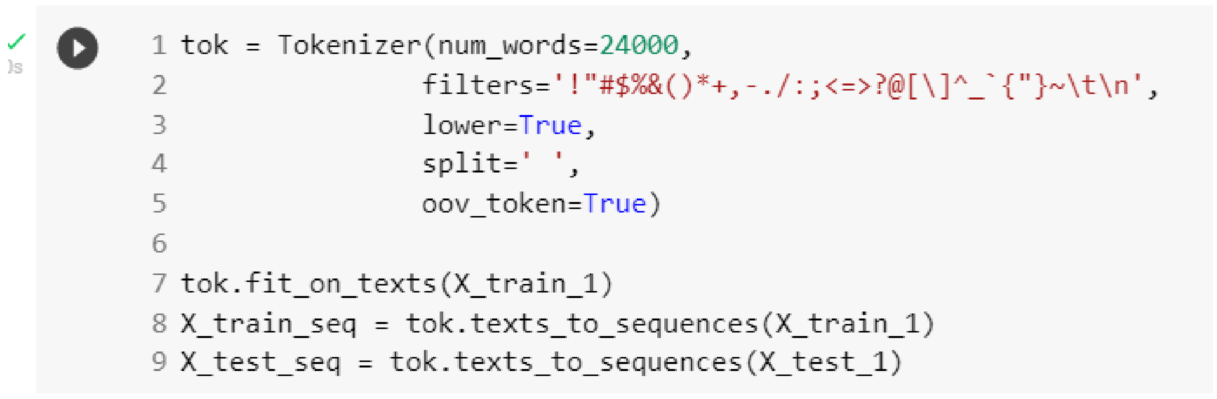
The figure bellow shows the difference between before the text was lemmatized and after



	Unnamed: 0	class	tweet_low	handle_count	no_handle_no_special_no_sin_ash	tweet_length	character_count	lemmatized	lemmatized_1	lemmatized_final	tweet_without_stopwor
0	0	2	!!! rt @mayasolovely: as a woman you shouldn't...	1	rt as a woman you shouldn't complain abo...	127	93	rt as a woman you should n't complain about cl...	rt as a woman you should n't complain about cl...	rt as a woman you should n't complain about cl...	rt woman n't complain clea house amp man alwa

Figure 3-25

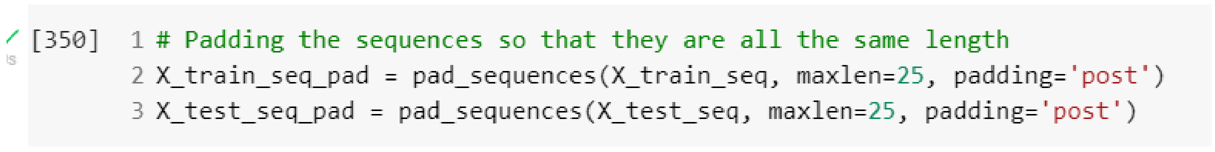
The tweets are also tokenized previous to the model implementation, in order to divide the tweets into tokens and avoid non relevant words or characters during the classification process.



```
tok = Tokenizer(num_words=24000,
                filters='!"#$%&()*+,-./:;<=>?@[\]^_`{"}~\t\n',
                lower=True,
                split=' ',
                oov_token=True)

tok.fit_on_texts(X_train_1)
X_train_seq = tok.texts_to_sequences(X_train_1)
X_test_seq = tok.texts_to_sequences(X_test_1)
```

Figure 3-26 CNN tokenization

Padding is used to fill out with zeros so all the tokenized tweets data to the same length, as shown in the figure below



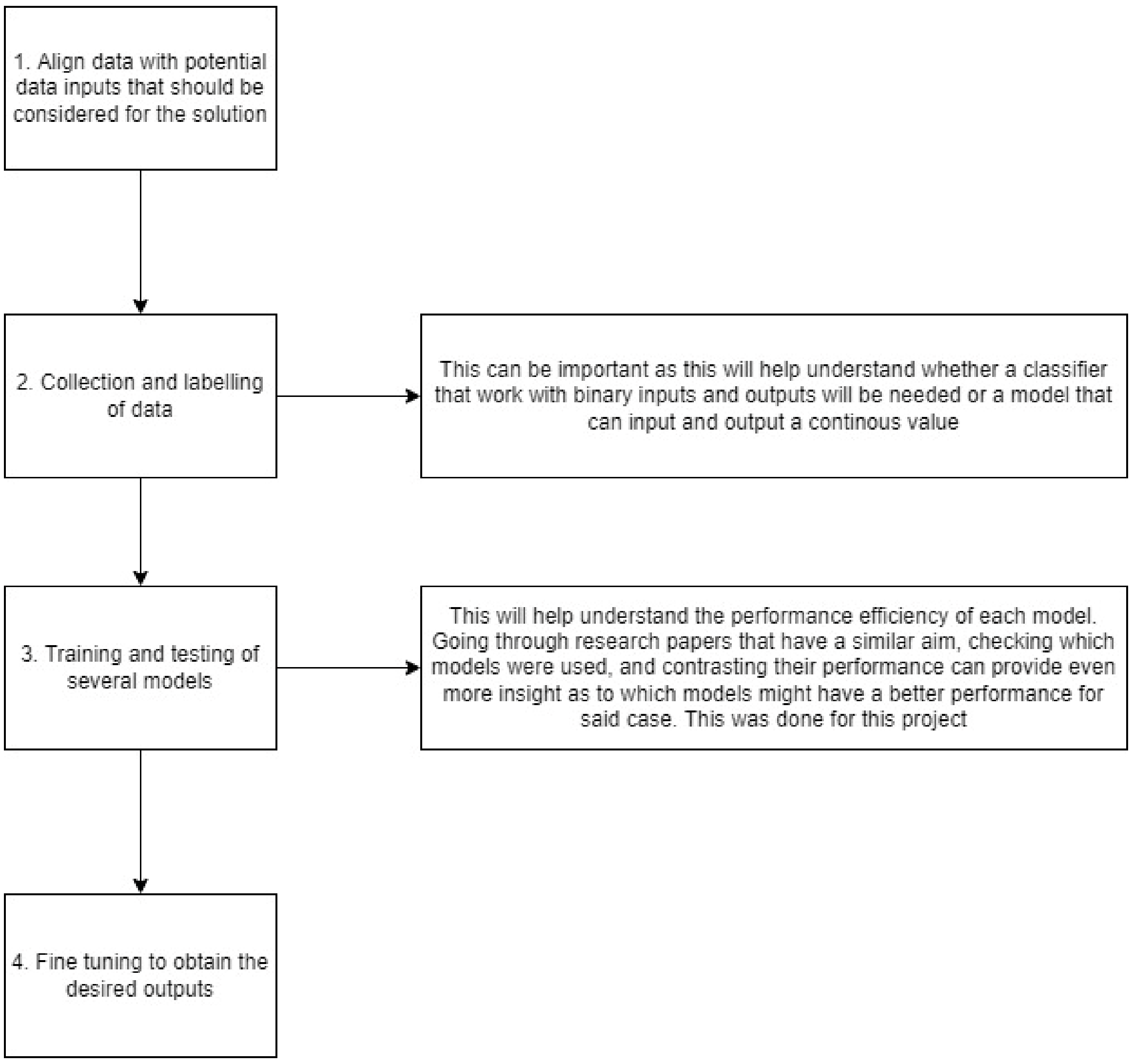
```
[350]
# Padding the sequences so that they are all the same length
X_train_seq_pad = pad_sequences(X_train_seq, maxlen=25, padding='post')
X_test_seq_pad = pad_sequences(X_test_seq, maxlen=25, padding='post')
```

Figure 3-27 CNN padding

3.6. Design Choices for the models:

For this project three models were used: Gaussian Naïve Bayes, Logistic regression, and CNN. The three of them were developed following a supervised learning approach. Having different models from different machine learning techniques allows for a better understanding of which one can be more efficient for text classification.

According to an article by TechTarget "the process of choosing the right machine learning model to solve a problem can be time consuming if not approached strategically" (Burns, 2021). The following steps were done to decide strategically which specific model of each category to use:



1. Align data with potential data inputs that should be considered for the solution

2. Collection and labelling of data

This can be important as this will help understand whether a classifier that work with binary inputs and outputs will be needed or a model that can input and output a continous value

3. Training and testing of several models

This will help understand the performance efficiency of each model. Going through research papers that have a similar aim, checking which models were used, and contrasting their performance can provide even more insight as to which models might have a better performance for said case. This was done for this project

4. Fine tuning to obtain the desired outputs

Figure 3-28 Deciding the model

Following on this, according to an article, published by Medium there are seven parameters to take in consideration when picking a model: the performance of the model, explainability of results, the model's complexity, the size of the dataset, the dimensionality of the data, the training time and cost, and the inference time (Valdarrama, 2021). After comparing the models used in research papers that had similar objectives to this project (table in section 2.1 "Potential Models to Use") and reading an article "Multi-class Text Classification Model Comparison and Selection" (Li, 2018), which describes the best machine learning algorithms for text classification depending on the type of data and type of output. It can be concluded that the following models would achieve the best performance for the aim of recognising hate speech: SVM (particularly Linear Support Vector Machine), CNN (or a modified version of it, i.e., trained with character n-gram, word2vec, or used together with BERT), logistic regression, Naïve Bayes, Bag of Words with Keras (A python library based on deep learning).

However, the level of complexity of those models should also be considered, thus in this project three models will be implemented: one straightforward model i.e., logistic regression, one of medium complexity i.e., Gaussian Naïve Bayes model, and finally a neural network (CNN) which becomes more challenging when trying to classify text rather than images.

Table 3.1 Past performance of three selected models

Performance of the three selected models in past research and their complexity			
Machine learning technique used	Year and Author	Performance of the model	Complexity level of the machine learning technique
Naïve Bayes	In 2017, in the paper by Fatahillah et al.	Accuracy of 93%.	Runtime complexity of Naïve Bayes: O(d*dc) d = Size of the vector c= Total classes
	In 2018 by M. Ali Fauzi et al	**78.3% accuracy**, this paper tested five different machine learning algorithms: K-Nearest Neighbours, Random Forest, Naive Bayes, Support Vector Machine, and Maximum Entropy. Naïve Bayes performed the best	
Logistic Regression	Article "Build Your First Text Classifier in Python with Logistic Regression" (Ganesan, n.d.	Logistic regression with tf-idf weighting, achieving an accuracy of 0.87	Complexity level of Logistic regression: O(d) d = size of the vector
CNN	In 2019 a paper by Gambäck et al.	Used four CNN models, trained with character n-gram, word2vec, random vectors combined (character n-gram and word2vec). It achieved a 78.3% of F-score	Complexity level of CNN: O(n) It is a linear complexity n= number of pixels
	In 2020 paper by Safa Alsafari et al.	Three-class classification with CNN and Bert. It	

		achieved 75.51% of the F1-score.	

3.7. Design choices for evaluation

When it comes to measuring the performance of a model, there is different ways of doing it depending on the task we are trying to achieve. When performing classification predictions, the output can be of four different types: True positives, when the prediction matches the output, i.e., the prediction of a data value belonging to a class, and it is matching the output. True negatives, again the prediction matches the output but in this case the prediction is that the data value does not belong to a class, and the output shows the same. False positives, when the prediction and the output don't match, i.e., when the prediction says that a data value belongs to a class, but in the output it doesn't. False Negative, again the prediction doesn't match the output but in this case the prediction says that a data value does not belong to a certain class whereas the output says the opposite. These different outputs can be plotted in a confusion matrix.

For this research 4 different evaluation metrics were used, which will be discussed below:

Accuracy:

It is the percentage of predicted values that match with the actual values of the test data. It is one of the most used metrics, however it does not work well when it comes to imbalanced classes, thus it should not be the only evaluation metric used to measure the performance of the models. To calculate accuracy, it divides the correct precisions between all the predictions.

$$\text{Accuracy} = \frac{correct\ predictions}{all\ predictions}$$

(Jordan, 2017)

Precision:

Is the percentage of positive instances over the **predicted** positive instances: "is defined as the fraction of relevant examples (true positives) among all of the examples which were predicted to belong in a certain class." (Jordan, 2017). To calculate precision the next formula can be used:

$$\text{Precision} = \frac{True\ positives}{True\ positives + false\ positives}$$

(Jordan, 2017)

Recall/ sensitivity:

Is the percentage of positive instances over all the **actual** positive instances. It is calculated as follows:

$$\text{Recall} = \frac{True\ positives}{True\ positives + false\ negatives}$$

(Jordan, 2017)

To understand better the difference between precision and recall, the denominator should be the point of focus since for precision it is the model's prediction of a value belonging to a class (i.e., positive) including false positives (FP). Whereas, in case of the recall the denominator is the "actual" number of positives instances therefore including False negative values (FN), refer to figure 3.29

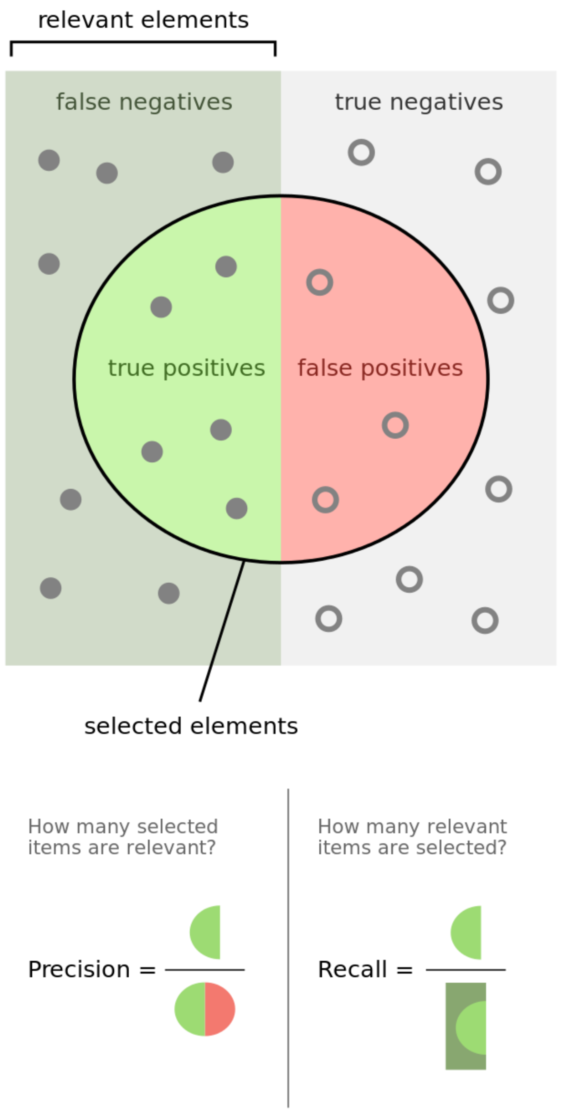


Figure 3-29 Difference between precision and recall

(Jordan, 2017)

Recall and precision are often used together, when combining them we get the f-score of a classifier.

$$F_{\beta} = (1 + \beta^2 \frac{precision\ x\ recall}{(\beta^2\ x\ precision) + recall})$$

Where the **β** parameter enables to control whether precision or recall receives more importance, i.e., if **β** < 1 then the F-score will focus more on precision while if **β** > 1 then it focuses more on recall. F1-score becomes then a type of F-score which is also known as the basic F-score, were precision and recall are equally important.

F1-score:

In terms of measuring the performance of the model it can be measured by the accuracy achieved. However, this option might not always be the best one to prove that a model has been trained correctly. Like discussed previously, accuracy can be a problem when applying it to data that has a severe class imbalance. Therefore F1-score becomes an appealing option.

F1 score is the "overall measure of a model's accuracy that combines precision and recall" [https://wiki.com/accuracy-precision-recall-f1], it considers a model being perfect when the score is 1 whereas when it is 0 the model has failed. In this project the F1 score will be the one applied to measure how well trained the model was. This is because our data might be of large class imbalance, i.e., there might be more offensive language rather than hate speech lexicon so the model could predict the value of the majority class for all predictions and still achieve a high accuracy despite it not being precise in recognising the difference between hate speech and offensive language; whereas when using F1 score it can be more detailed on analysing the performance of the model, checking that in fact the model has less low false positives and low false negatives, identifying phrases that do contain hate speech language. This class imbalance, however, should be taken in consideration and accounted for error analysis.

F1-score is measured following the F-score formula but replacing the parameter **β** by 1, thus giving the following formula for F1-score:

$$F_1 = (1 + \ 1^2 \frac{precision\ x\ recall}{(\beta^2\ x\ precision) + recall})$$

$$F_1 = (2 \frac{precision\ x\ recall}{(\beta^2\ x\ precision) + recall})$$

4. Design Documentation: Machine Learning Classification Implementation and Results

4.1. Introduction:

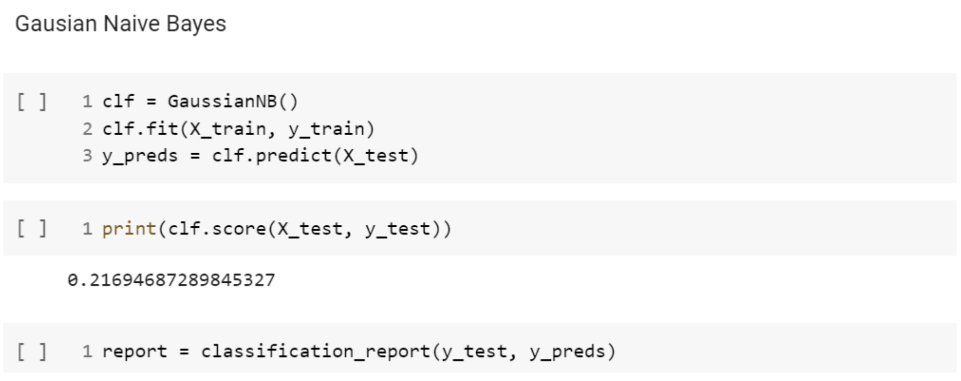
This chapter will discuss the different models implemented as well as the challenges faced during said phase. It will also show snippets of the code developed for each model and the performance result for each of them.

4.2. Set-Up

All this code was implemented in Google Collab only by me, the reason Google Collab was used is to avoid losing the code. It is important to have in consideration that for running this project the **dataset and code should be in the same directory and folder**. All of the code was implemented in python.

4.3. Naive bayes implementation

This model is implemented using the scikit-learn platform, a pre-design algorithm is employed, where the parameters will be adjusted for fitting the dataset for training and testing.



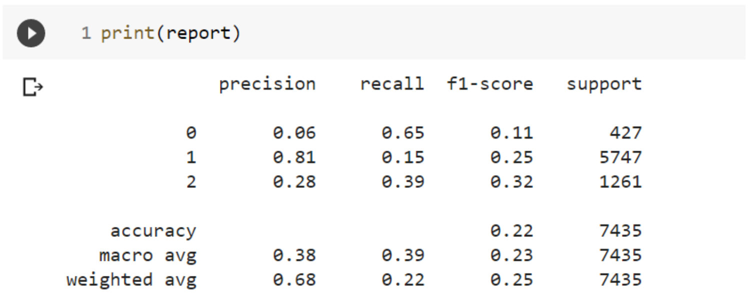
Gausian Naive Bayes

```
[ ]  clf = GaussianNB()
     clf.fit(X_train, y_train)
     y_preds = clf.predict(X_test)
```

```
[ ]  print(clf.score(X_test, y_test))

   0.21694687289845327
```

```
[ ]  report = classification_report(y_test, y_preds)
```

Figure 4-1 Gaussian Naive Bayes Code



```
print(report)

              precision    recall  f1-score   support

           0       0.06      0.65      0.11       427
           1       0.81      0.15      0.25      5747
           2       0.28      0.39      0.32      1261

    accuracy                           0.22      7435
   macro avg       0.38      0.39      0.23      7435
weighted avg       0.68      0.22      0.25      7435
```

Figure 4-2 Gaussian Naive Bayes results

The figure above shows the results of the Naïve Bayes algorithm with the dataset, which I 22% for accuracy, 0.68 for precision, 0.39 for precision and 0.23 for f1-score. Originally it was planned to add a K-nearest neighbour algorithm to optimise the performance of the model, however due to time constraint I was unable to finish the implementation of it.

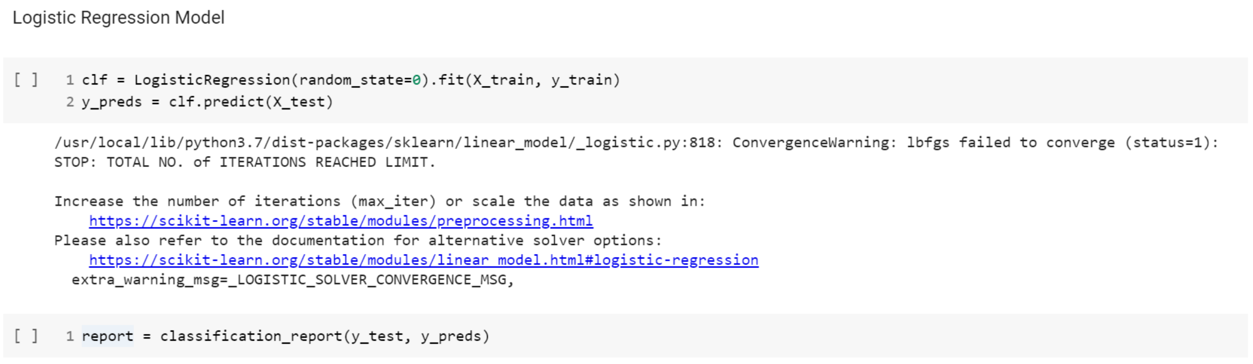
4.4. Logistic regression

For the logistic regression model there were some important steps to take into consideration for the pre-processing of the data such as: removing noise from the input dataset since the logistic regression model assumes no error it could possibly wrongly classify the data. It is suggested that highly correlated inputs should also be removed from the dataset since it can overfit if having several similar inputs. However, since the same database is being used for different models a possibility of removing certain words might be unlikely, but it can justify the overfitting of this model.

This model is implemented using the scikit-learn logistic regression class, documentation, and several libraries with the purpose of creating a multiclass classifier, train it and test it. To implement the model, it requires of the "liblinear" library and its capable of handling dense and sparse input data. The attribute "random_state" is set to 0 for shuffling the data in this project.

"X_train" and "y_train" are the data and labels respectively, used for training the model.

It uses the scikit-learn inbuilt function "classification report" to print the accuracy, precision, recall and F1-score

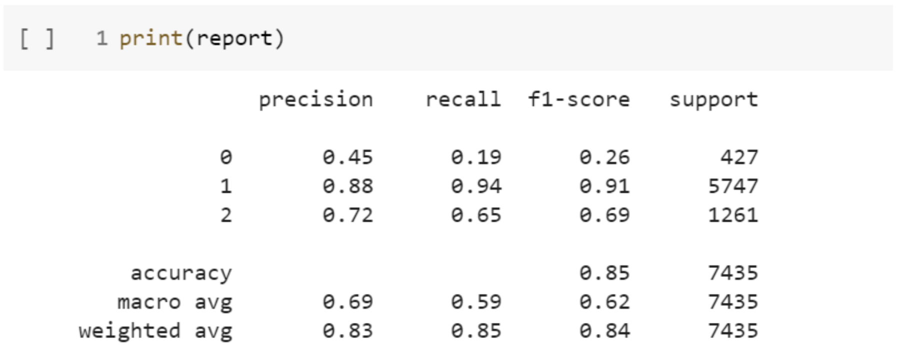


Logistic Regression Model

```
[ ]  clf = LogisticRegression(random_state=0).fit(X_train, y_train)
     y_preds = clf.predict(X_test)

/usr/local/lib/python3.7/dist-packages/sklearn/linear_model/_logistic.py:818: ConvergenceWarning: lbfgs failed to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression
  extra_warning_msg=_LOGISTIC_SOLVER_CONVERGENCE_MSG,
```

```
[ ]  report = classification_report(y_test, y_preds)
```

Figure 4-3 Logistic Regression Code

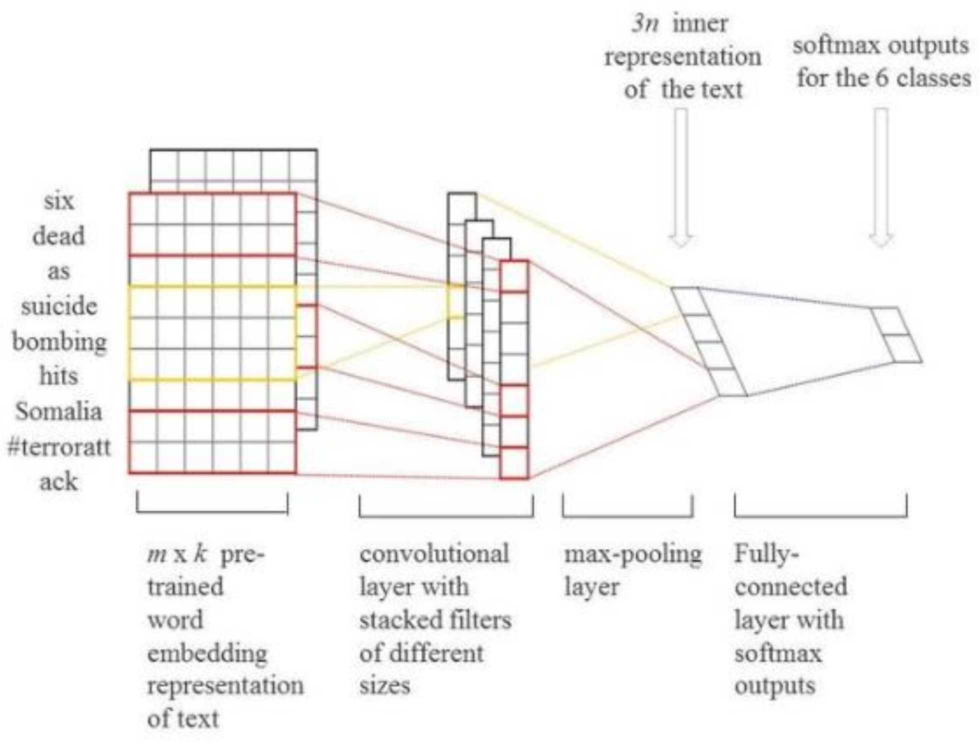


```
[ ]  print(report)

              precision    recall  f1-score   support

           0       0.45      0.19      0.26       427
           1       0.88      0.94      0.91      5747
           2       0.72      0.65      0.69      1261

    accuracy                           0.85      7435
   macro avg       0.69      0.59      0.62      7435
weighted avg       0.83      0.85      0.84      7435
```

Figure 4-4 Logistic Regression Results

4.5. Neural Networks:

A CNN model is chosen from the neural networks for this research. This model is a supervised learning approach and uses the figure below as a guidance for the implementation



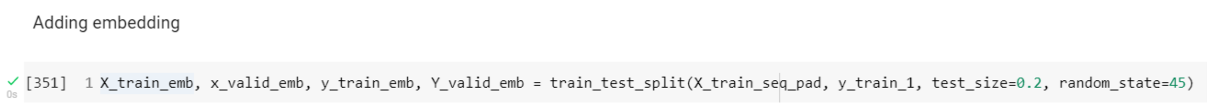
3n inner representation of the text
softmax outputs for the 6 classes
six dead as suicide bombing hits Somalia #terroratt ack
m x k pre-trained word embedding representation of text
convolutional layer with stacked filters of different sizes
max-pooling layer
Fully-connected layer with softmax outputs

Figure 4-5 Implementation of a CNN model for text classification

(Choubey, 2020)

Using CNN is mainly used with images but in order to be able to use it with words these would have to be converted into vectors. As discussed in chapter 2, getting the dot product of two vectors can lead to getting a 0 as an output which is why the dot product should be done on embedded word vectors. A special pre-processing method is done for this model, where all tweets are transformed to lowercase, stopwords are applied, the duplicates are identified and removed, the retweet symbol is also removed as well as URLs, the tweets are also lemmatized to analyse only relevant words. After this, the data gets shaped and prepared to be input in the classifier by tokenizing it and adding padding to it.

The next step is the embedding of the data which will translate the high dimensional vectors of the input data into low-dimensional vectors. This is helpful due to the size of the dataset, making the classifier more efficient.



Adding embedding

```
[351] X_train_emb, x_valid_emb, y_train_emb, Y_valid_emb = train_test_split(X_train_seq_pad, y_train_1, test_size=0.2, random_state=45)
```

Figure 4-6 Embedding

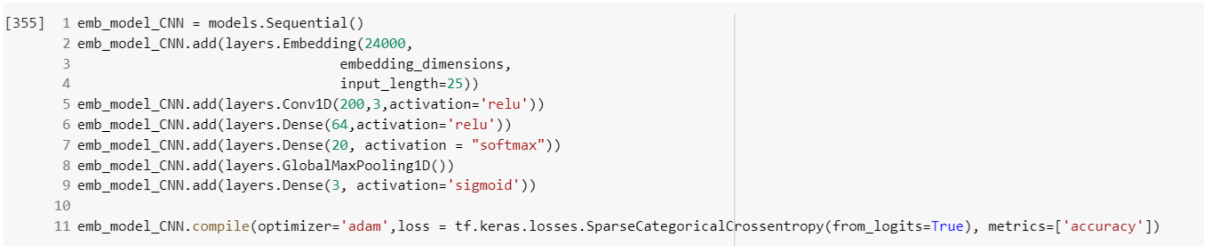
A CNN has different layers: the model has six node layers, where the first one would be the input layer, then the hidden layers i.e., Conv1D, Dense, and GlobalMaxPooling, and finally an output layer.

The Conv1D layer is a one-dimensional layer used for the creation of the Kernel which will then be convolved with the input layer over a single spatial dimension, producing a tensor of outputs.

The Dense layer is used between layers, and it has as a purpose to classify the data depending on their output from the convolutional layer.

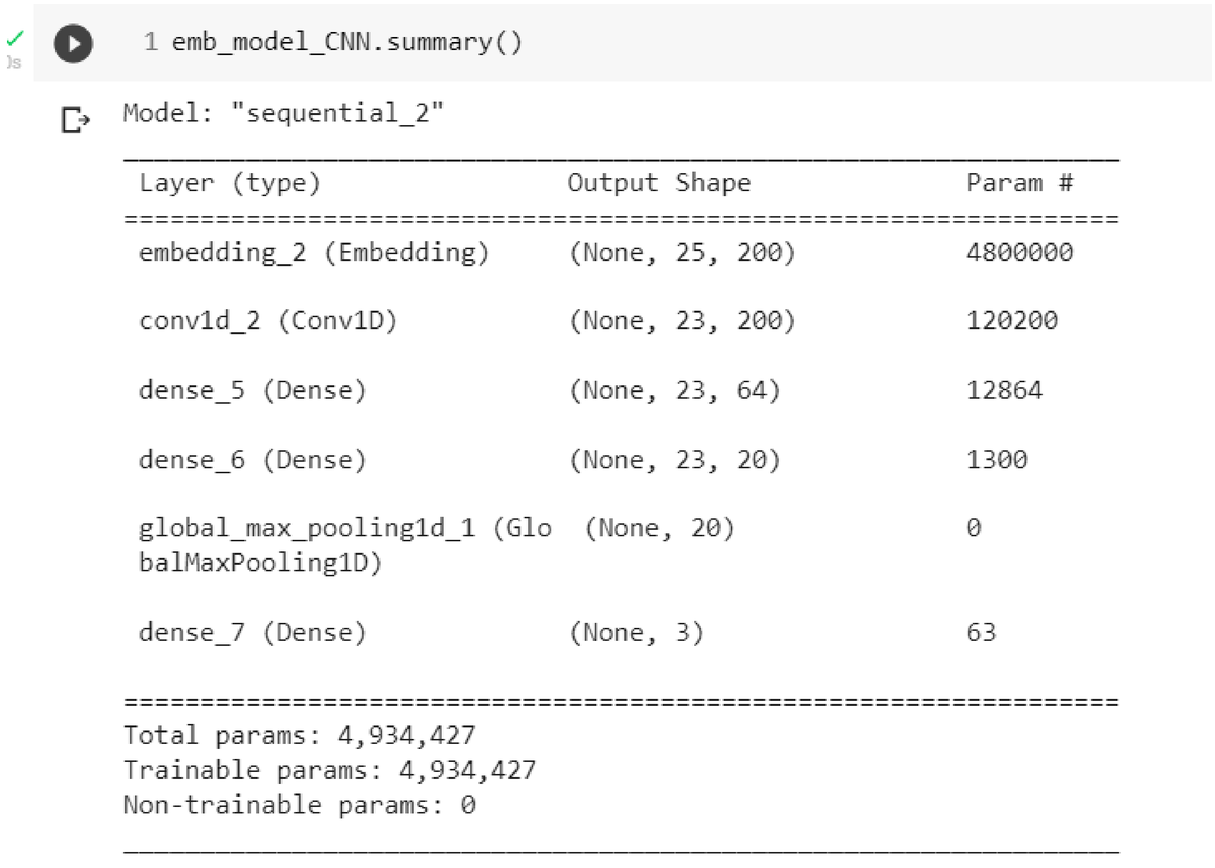
A polling layer is also added to reduce dimensional complexity, computation and avoid overfitting of data. In this case a MaxPooling layer is used for calculating the value in each feature map.

SparseCategoricalCrossentropy is also used in the following code. It allows for multiclassification, which is required in this case with the three labels.



```
[355]
emb_model_CNN = models.Sequential()
emb_model_CNN.add(layers.Embedding(24000,
                                embedding_dimensions,
                                input_length=25))
emb_model_CNN.add(layers.Conv1D(200,3,activation='relu'))
emb_model_CNN.add(layers.Dense(64,activation='relu'))
emb_model_CNN.add(layers.Dense(20, activation = "softmax"))
emb_model_CNN.add(layers.GlobalMaxPooling1D())
emb_model_CNN.add(layers.Dense(3, activation='sigmoid'))

emb_model_CNN.compile(optimizer='adam',loss = tf.keras.losses.SparseCategoricalCrossentropy(from_logits=True), metrics=['accuracy'])
```

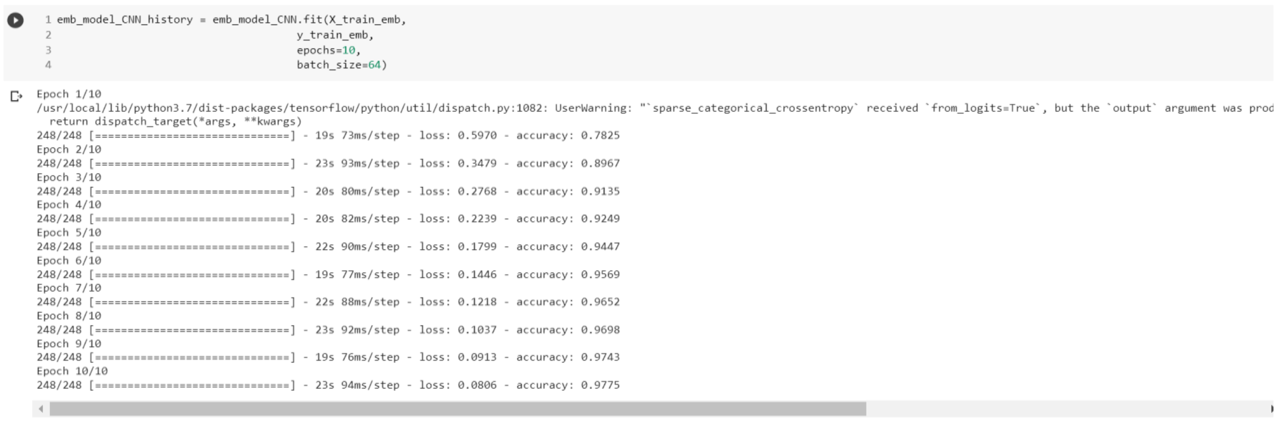
Figure 4-7 CNN layers



```
1 emb_model_CNN.summary()

Model: "sequential_2"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 embedding_2 (Embedding)     (None, 25, 200)           4800000

 conv1d_2 (Conv1D)           (None, 23, 200)           120200

 dense_5 (Dense)             (None, 23, 64)            12864

 dense_6 (Dense)             (None, 23, 20)            1300

 global_max_pooling1d_1 (Glo  (None, 20)               0
 balMaxPooling1D)

 dense_7 (Dense)             (None, 3)                 63

=================================================================
Total params: 4,934,427
Trainable params: 4,934,427
Non-trainable params: 0
_________________________________________________________________
```

Figure 4-8 CNN model summary

This next figure shows all the epochs that the model did during training. The epochs are the processing of all the dataset one time individually, whether it is a forward or backward network.

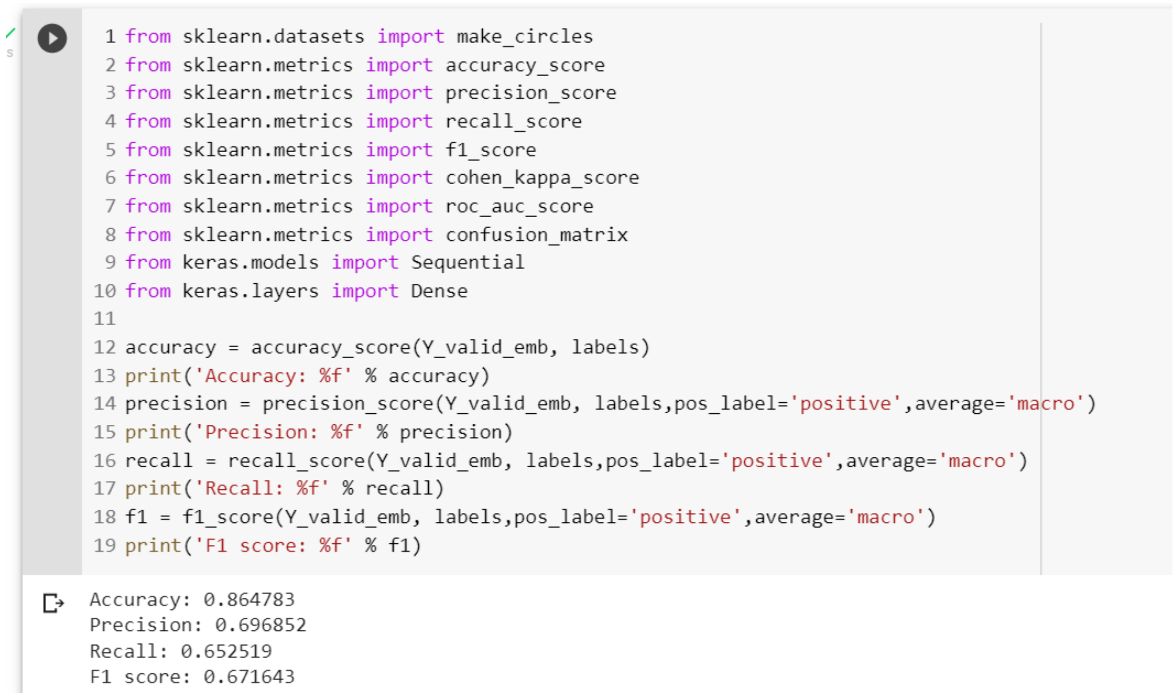


```
emb_model_CNN_history = emb_model_CNN.fit(X_train_emb,
                                          y_train_emb,
                                          epochs=10,
                                          batch_size=64)
```

```
Epoch 1/10
/usr/local/lib/python3.7/dist-packages/tensorflow/python/util/dispatch.py:1082: UserWarning: "`sparse_categorical_crossentropy` received `from_logits=True`, but the `output` argument was prod
  return dispatch_target(*args, **kwargs)
248/248 [==============================] - 19s 73ms/step - loss: 0.5970 - accuracy: 0.7825
Epoch 2/10
248/248 [==============================] - 23s 93ms/step - loss: 0.3479 - accuracy: 0.8967
Epoch 3/10
248/248 [==============================] - 20s 80ms/step - loss: 0.2768 - accuracy: 0.9135
Epoch 4/10
248/248 [==============================] - 20s 82ms/step - loss: 0.2239 - accuracy: 0.9249
Epoch 5/10
248/248 [==============================] - 22s 90ms/step - loss: 0.1799 - accuracy: 0.9447
Epoch 6/10
248/248 [==============================] - 19s 77ms/step - loss: 0.1446 - accuracy: 0.9569
Epoch 7/10
248/248 [==============================] - 22s 88ms/step - loss: 0.1218 - accuracy: 0.9652
Epoch 8/10
248/248 [==============================] - 23s 92ms/step - loss: 0.1037 - accuracy: 0.9698
Epoch 9/10
248/248 [==============================] - 19s 76ms/step - loss: 0.0913 - accuracy: 0.9743
Epoch 10/10
248/248 [==============================] - 23s 94ms/step - loss: 0.0806 - accuracy: 0.9775
```

Figure 4-9 CNN Epochs

As results, the model gets an accuracy of 86%, a precision of 0.70, a recall of 0.65 and an f1-score of 0.67. These values can vary for some decimal points due to the randomisation of data but should not cause a significant change. Overall, the CNN model had an optimal performance. For a better performance an option could have been lowering the threshold but that would mean it could be less precise in identifying the difference between hateful and offensive speech.



```
from sklearn.datasets import make_circles
from sklearn.metrics import accuracy_score
from sklearn.metrics import precision_score
from sklearn.metrics import recall_score
from sklearn.metrics import f1_score
from sklearn.metrics import cohen_kappa_score
from sklearn.metrics import roc_auc_score
from sklearn.metrics import confusion_matrix
from keras.models import Sequential
from keras.layers import Dense

accuracy = accuracy_score(Y_valid_emb, labels)
print('Accuracy: %f' % accuracy)
precision = precision_score(Y_valid_emb, labels,pos_label='positive',average='macro')
print('Precision: %f' % precision)
recall = recall_score(Y_valid_emb, labels,pos_label='positive',average='macro')
print('Recall: %f' % recall)
f1 = f1_score(Y_valid_emb, labels,pos_label='positive',average='macro')
print('F1 score: %f' % f1)
```

```
Accuracy: 0.864783
Precision: 0.696852
Recall: 0.652519
F1 score: 0.671643
```

Figure 4-10 CNN results

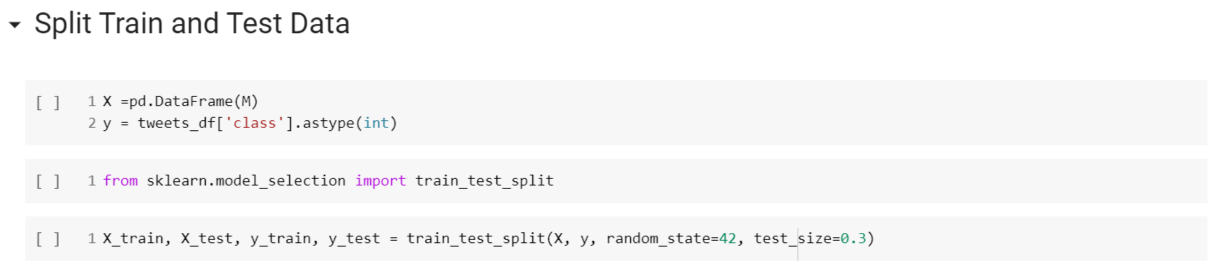
5. Evaluation of Outputs/Testing

5.1. Introduction:

This chapter will focus on the comparison of the performance of the different models used and the approach to test their performance. Like discussed preciously, accuracy is the most used measuring metric however it might not be the best way to measure the performance of the models in this case, since the data could be biased. As established previously in the description of the dataset, the classes are not evenly distributed since it provides more data labelled as offensive language, which can make precision and accuracy better measurement techniques, since they are known to be useful in cases where the data isn't evenly distributed, this is why they are often used together, the "common approach for combining these metrics is f1-score" (Jordan, 2017). Thus, in order to measure the performance of the models used along this project 4 evaluation metrics will be employed: Accuracy, precision, recall, and f-score. A comparison between models will be achieve through these 4 metrics.

5.2. Comparison evaluation

The data was split using a random number to initialise the division between training and testing data. In all of the models the data split has an integer value assigned to the random state, meaning that the same value of random numbers will be created each time.

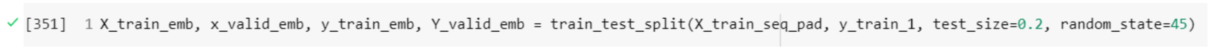


Split Train and Test Data

```
[ ]  X =pd.DataFrame(M)
     y = tweets_df['class'].astype(int)

[ ]  from sklearn.model_selection import train_test_split

[ ]  X_train, X_test, y_train, y_test = train_test_split(X, y, random_state=42, test_size=0.3)
```

Figure 5-1 Split Train and Test data Naive Bayes and Logistic Regression

Split train and data set for CNN



```
✓ [351]  X_train_emb, x_valid_emb, y_train_emb, Y_valid_emb = train_test_split(X_train_seq_pad, y_train_1, test_size=0.2, random_state=45)
```

Figure 5-2 Split Train and Test data CNN

For these results the macro averages of these measurements were used. The macro average was chosen over the micro average due to the first one measuring the metrics independent for each class and then averaging on each model whereas the latter one will add the contributions of all classes and then output an average.

Table 5.1 Comparison Between models

Comparison between models (macro average)				
Model	Accuracy	Precision	Recall	F1-score
Gaussian Naïve Bayes	22%	0.38	0.39	0.23
Logistic Regression	85%	0.69	0.59	0.62

Convolutional Neural Network (CNN)	86.96%	0.71	0.67	0.69

When comparing the results, it is clear that CNN had the best model performance overall, this might be due to a more specific and detailed pre-processing or also to the inclusion of more parameters to fit better the dataset. Logistic regression also had an optimal performance since it is above 50% in any of the metrics. However, Gaussian Naïve Bayes had poor performance only achieving scores below 50 in any metric. There can be different reasons as to why they have different performance scores, such as the pre-processing methods, for example in CNN a lemmatization process was used whereas for Logistic regression and Gaussian Naïve Bayes had a stemming process. It is well known that lemmatization is superior to stemming, since the first one is capable of analysing a word syntactically and morphologically whereas the latter can only reduce the word to its base without much analysis. Another example can be the use of embeddings for vectors in CNN, these are known to be useful specially in cases where there are similar values or in this case words. It also helps the classifier to perform more efficiently with a large dataset like in this case. Whereas for the Gaussian Naïve Bayes and Logistic regression only a TF-IDF matrix was implemented, and the classifiers would base on the weight of the vectors. Other examples can be a better understanding of the way CNN works, more resources and information on CNN implementation, creating the CNN model from scratch. This results also showed that CNN and logistic regression are effective models for text recognition

6. Project Evaluation: review of project outcomes

6.1. Introduction:

This chapter will show a comparison of the project objectives and the project results moreover it will describe the way they were achieved.

6.2. Evaluation of goals:

In chapter 1 a set of objectives were established for this project, each of the following are the results and actual accomplishments throughout this research.

7. Objective 1 focuses on understanding the difference between hate speech and offensive language through research papers. Different research papers Automatic Hate Speech Detection: A Literature Review () and Automated Hate Speech Detection and the Problem of Offensive Language (Davidson, Warmsley, Macy and Weber, n.d) were studied and compared to understand the difference between hate speech and offensive language, however it was concluded that there was not only one international definition for hate speech and that its difference with offensive language relies mainly in syntactical context.
8. In respective to objective 2, several research papers and articles were used throughout this project, mainly for literature review and for implementation, all of them are listed as in-text citation and in the references list. Following on that on the literature review there is a specific section dedicated to discussing four of those research papers that were used as main source for this project and how were they used.
9. For objective 3, the section "Approaches chosen" in the literature review has a deep research on the different classifiers implemented in this project, learning not only one but three machine learning techniques for text classification
10. To achieve the requirement of objective 4, a dataset that had data with different labels for offensive language and for hate speech was found in the research paper "Automated Hate Speech Detection and the Problem of Offensive Language" (Davidson, Warmsley, Macy and Weber, n.d) and used to train and test the models of this project.
11. Objective 5 states a comparison between the different accuracies achieved by the different models, however after conducting research in evaluation metrics it was concluded that it would be more optimal to use four different evaluation metrics (accuracy, precision, recall and f1-score) due to the dataset having a clear class imbalance. The use of different metric gives us a better idea of the model's performance.
12. Corresponding to objective 6, the classifier presents a good performance level when differentiating hateful speech from offensive language. In case of the supervised classifiers, the model can process the rest of the data following the labelled sample. Furthermore, all models are able to go through the dataset and classify the values. Logistic Regression and CNN achieved an optimal performance; however Naïve Bayes had a poor performance, it was still able to identify some hateful speech but not enough.

Moreover, the main aim of the project was achieved by having three models implemented with the task of recognising hate speech and differentiating it from offensive language.

Potential Risks and Contingency plans:

In the planning stage of this project after defining the aim and objectives, a set of potential risks was also drafted and the contingency plans for each of them.

Throughout this project there were several challenges and some of them were dealt following the contingency plans.

Table 6.1 Potential risks and contingency plans

Potential Risk	Contingency Plan
1. The precision of the classifier can be minimal	Try adjusting the classifier to a lower threshold of which words could be consider hateful.
2. The classifier can get hateful speech and offensive language confused, which is very common since the terms used by most people in social media can be insults (even if it is not hate directed towards someone/something).	Try a different data set
3. The classifier cannot be able to identify certain words on certain contexts as hateful speech	Try a different classifier
4. A model has a poor performance, thus outputting too many false positives or not distinguishing classifying the data values in the right classes.	Try a different machine learning approach

The first potential risk happened in the Naïve Bayes classifier, where the contingency plan was applied thus lowering the threshold, but its performance was still not optimal.

For the second potential risk, this was not an issue, however a second data set was researched to allow a comparison between different models and between different datasets. This was not possible as no data set that had two different labels for hate speech and offensive language was found, besides the one that has already been used.

The third potential risk was also applied, originally a BERT transformer was planned as one of the models to be implemented, however after implementing it did not work. A different approach which is also known for having a good performance in text recognition was applied, i.e., CNN was implemented instead.

The fourth potential risk is not applicable for this case, since it was for the case of only implementing one model in this project. However, for this project only one machine learning approach, i.e., supervised learning was used to implement different models. The contingency plan would then be in this case that if that approach did not work an unsupervised approach could have been implemented and trained.

7. Statement of Ethics

7.1. Introduction:

This chapter shows the different sections of ethics that were taken into consideration when doing this research, including the repercussions that this type of dataset and algorithms could have as well as the impact that it can cause for society.

7.2. Legal: Informed Consent

There were no human nor animal participants or genetic material used in this research paper. However, if there had been any active participants or human subjects, an extra ethical form would be filled and in case of genetic material used a "Human Tissue Governance Form" plus the project would have followed extra caution measurements and a base of guidelines, such as data confidentiality would be followed. This project does not involve any of the participants or authors into any hazardous materials nor importing or exporting any samples. It is also understood that if that had been the case a relevant health and safety from would have had to be completed and import/export licenses would have been requires.

Despite using a dataset extracted from a social media application, in this case it does not count as a third-party collaboration since the dataset was public and did not require permission to access potential participants during this research. This dataset of tweets was compiled by "Hatebase.org", and internet users had already deemed some of the tweests as hateful speech. Hatebase.org is an open platform with the aim of helping organizations and online communities to "detect, monitor, and analyze hate speech" (About-Hatebase, 2022). The dataset used throughout this project is public and has also been used previously by another research paper, which did the pre-processing and labelling of the data and have it in a public repository. Furthermore, all the papers, websites and books used have been documented following the university guidelines, i.e., using Harvard's citation style, moreover all of these references are public. The results obtained from any of those research papers are not private and can be cited.

This research is not security-sensitive as said before the dataset used throughout this project is public and so are the algorithms. Thus, there is no material in this project used for research projects commissioned to the military nor any extremist material.

A Self-Assessment for Governance and Ethics (SAGE) has been completed and submitted. After filling this form, it should be clear whether any additional ethical review by the University Ethics Committee should be necessary. For this research, the SAGE form shows that there is no need for extra reviews. A copy of the completed SAGE form can be found in the appendixes.

7.3. Legal: Confidentiality of Data

This project does not process personal data, despite the data set including the username of the users that published each tweet it is not considered as personal data as said before this dataset was compiled from hatebase.org, which is an open source and collects public data. Moreover, during the pre-processing of data usernames are removed and not stored. This project has not been uploaded to any cloud services in a public way, only uploaded to a google drive only accessible by the author and has only been used in my personal machine and not by any other third-parties, therefore not violating the Computer Misuse Act. No other resources than tweets are used as part of this research.

7.4. Ethical: Do no Harm

This project is done with research purposes only, thus the data and algorithms will not be used in reality. The data stored in this project is from a public domain so this project should not be responsible for any harm inflected using that dataset, besides no modifications apart from pre-processing were done to the data therefore not violating the Computer Misuse Act. None of the algorithms or manipulation of data is used irresponsibly nor developed with any harmful goal. Even though, these algorithms are well implemented and could be used in real practice, they would just be a second interpreter of a labelled dataset which in case used with any other dataset, the data should be confirmed to be extracted from a public domain or follow or the ethical guidelines, therefore the algorithms not being responsible for any harm inflicted.

In regards, to victimisation in case these algorithms were to be used in malicious way, it could be possible to trace the account holder of a specific tweet and possibly blackmail them or cause harm, however this dataset, being public, has already gone through the appropriate security filters and permissions to publish this data.

7.5. Social: Social Responsibility

This project does not require any piece of hardware or password in order to be run or understood.

This paper's goal is to recognise hate speech therefore it could be useful and insightful for social media applications for improving the number of tweets that can be removed from the platform or the quantity of users inflicting harm in others and sanction them. This in a broader aspect could then impact people's mental health as it could be used to reduce cyberbullying. It could also be used by data analysts to evaluate in which context there is more hate speech and if implemented further it could also analyse the social groups most targeted with hate speech.

Another main focus point in this paper is the differentiation between hate speech and offensive language, this can also have a big impact in society, since in different countries hate speech can be severely punished and in other cases users can get social media accounts banned or suspended, thus these models enabling to reduce the amount of content incorrectly flagged as hateful speech, when in reality is offensive language

This project also contributes to an expansion of research on the subject, since it has not been explored yet by many people and can be difficult to find sources that state an effective way of differentiating hate speech from offensive language and how it can be implemented in Twitter.

It also discusses how Twitter has no strict policy when it comes to users "tweeting" hateful speech and how this can impact society not only virtually but also physically.

7.6. British Computer Society Code of Conduct

The University of Surrey is associated with the British Computer Society (BCS), thus this section will discuss how this project followed the BCS code of conduct practices which is comprised in four principles:

7.6.1. Public Interest

In regards to the security and well-being of others and the environment. The prime focus of this project, as stated before, is to recognize hate speech and differentiate it from offensive language. This research implements and compares different models following different machine learning approaches (supervised and unsupervised learning). This can

have a big positive impact in society, as discussed previously, which can include but is not limited to a faster recognition of hate speech tweets by the platform leading to deleting harmful or toxic content.
In regards to the legitimate rights of third parties, all of the sources have been cited accordingly and following the Harvard's style of citation as well as it has been checked that all sources allow to be referenced in other people's work. Furthermore, no paper or source was discriminated due to the author's race, gender, beliefs, opinions, ideas, or any other reason. All the research has been conducted professionally and without any type of discrimination.
All the technology i.e., algorithms, pre-processing, etc. has been made as understandable and accessible as possible. It has also been considered to make this report and the implementation code public to be available to anyone interested on the topic.

7.6.2. Professional Competence and Integrity

Throughout this research I was able to enrich my knowledge regarding machine learning and develop further skills on models' implementation and the different pre-processing techniques. I was also given the opportunity to explore more a subject of much interest to myself.

It also made me aware of the newest technological developments, procedures, and standards in this field. Technological developments such as the new machine learning models and python packages to develop them as well as how different research papers combine them to get a better performance. Procedures, such as doing a background literature review to understand better the implementation and designing a project before coding the project. Finally, the standards by comparing different performance measurement metrics and the steps different research papers and articles follow to develop machine learning models.

This project was developed using the best practices, it followed code versioning, requirement captures, etc. and in case of realising it to the public it could obtain a BSD license which would allow for the code to be an open source.

7.6.3. Duty to Relevant Authority

This project has been done with the professional responsibilities to care for the relevant authorities' requirements, it has only been developed by me with no disclosure to third-parties in order to get personal gain nor to benefit said third-party or for any reason. It also does not withhold any information of systems or take advantage of the inexperience of others.

7.6.4. Duty to the profession

For this project I made a great effort in always maintaining professional and ensuring that there were no damaging actions during the production of this project. When seeking to improve the professional standards of this paper, I tried to use my critical thinking and my skills to produce a high standard project in this field. I also outperformed the literature by comparing the research done of past papers and how different tasks performed better with different models, as well as discussed and established a clear difference between the different machine learning approaches. I consider that this project can be insightful and positively assist in further developments in this field.

Throughout this research I was able to maintain a professional relationship with my supervisor, take into account her input, suggestions and constructive critics to make a better report.

8. Final Conclusion and future work

After learning more about the different machine learning techniques and comparing different models that were trained and tested using the same dataset and evaluation measures, it can be concluded that there is different ways to do text recognition. The best performing model in this case was CNN.

It can also be recommended that in future work it could be insightful to analyse which social group is targeted more by hate speech and implement more algorithms to train the model to recognise it. In this project this was not possible as the dataset labels did not allow for a further split of the data labelled as hate speech. Thus, not making it possible to train a dataset to learn which data can be targeted to each group nor have any labels to compare it against when testing the data.

References

[3] N. R. Fatahillah, P. Suryati, & C. Haryawan. (2018). Implementation of naive bayes classifier algorithm on social media (Twitter) to the teaching of Indonesian hate speech. In: Proc. - 2017 Int.Conf. Sustain. Inf. Eng. Technol. SIET 2017, pp. 128–131.
DOI: 10.1109/SIET.2017.8304122

AltexSoft. 2022. *Semi-Supervised Learning, Explained with Examples*. [online] Available at: <https://www.altexsoft.com/blog/semi-supervised-learning/> [Accessed 12 May 2022].

Amidi, A., 2022. *CS 230 - Convolutional Neural Networks Cheatsheet*. [online] Stanford.edu. Available at: <https://stanford.edu/~shervine/teaching/cs-230/cheatsheet-convolutional-neural-networks> [Accessed 24 May 2022].

Banerjee, W., 2022. *Train/Test Complexity and Space Complexity of Logistic Regression*. [online] Medium. Available at: <https://levelup.gitconnected.com/train-test-complexity-and-space-complexity-of-logistic-regression-2cb3de762054> [Accessed 24 May 2022].

Brownlee, J., 2016. *Linear Regression for Machine Learning*. [online] Machine Learning Mastery. Available at: <https://machinelearningmastery.com/linear-regression-for-machine-learning/> [Accessed 14 May 2022].

Brownlee, J., 2016. *Logistic Regression for Machine Learning*. [online] Machine Learning Mastery. Available at: <https://machinelearningmastery.com/logistic-regression-for-machine-learning/> [Accessed 14 May 2022].

Brownlee, J., 2021. *A Gentle Introduction to Long Short-Term Memory Networks by the Experts*. [online] Machine Learning Mastery. Available at: <https://machinelearningmastery.com/gentle-introduction-long-short-term-memory-networks-experts/> [Accessed 15 May 2022].

Burns, E., 2021. *What Is Machine Learning and Why Is It Important?*. [online] SearchEnterpriseAI. Available at: <https://www.techtarget.com/searchenterpriseai/definition/machine-learning-ML#:~:text=Machine%20learning%20(ML)%20is%20a,to%20predict%20new%20output%20values.> [Accessed 12 May 2022].

Chandrasekaran, M., 2021. *Logistic Regression for Machine Learning | Capital One*. [online] Capital One. Available at: <https://www.capitalone.com/tech/machine-learning/what-is-logistic-regression/> [Accessed 14 May 2022].

Choubey, V., 2020. *Text classification using CNN*. [online] Medium. Available at: <https://medium.com/voice-tech-podcast/text-classification-using-cnn-9ade8155dfb9> [Accessed 15 May 2022].

Davidson, T. and Warmsley, D., 2022. Automated Hate Speech Detection and the Problem of Offensive Language. *ICWSM*, 2017.

dshahid, 2019. *Convolutional Neural Network*. [online] Medium. Available at: <https://towardsdatascience.com/covolutional-neural-network-cb0883dd6529> [Accessed 15 May 2022].

El-Alami, F., Ouatik El Alaoui, S. and En Nahnahi, N., 2021. A multilingual offensive language detection method based on transfer learning from transformer fine-tuning model. *Journal of King Saud University - Computer and Information Sciences*,.

E R, S., 2021. *Random Forest | Introduction to Random Forest Algorithm*. [online] Analytics Vidhya. Available at: <https://www.analyticsvidhya.com/blog/2021/06/understanding-random-forest/> [Accessed 15 May 2022].

Fatahillah, N., Suryati, P. and Haryawan, C., 2017. Implementation of Naive Bayes classifier algorithm on social media (Twitter) to the teaching of Indonesian hate speech. *2017 International Conference on Sustainable Information Engineering and Technology (SIET)*,.

Fauzi, M. and Yuniarti, A., 2018. Ensemble Method for Indonesian Twitter Hate Speech Detection. *Indonesian Journal of Electrical Engineering and Computer Science*, 11(1), p.294.

Ganesan, K., n.d. *Build Your First Text Classifier in Python with Logistic Regression - Kavita Ganesan, PhD*. [online] Kavita Ganesan, PhD. Available at: <https://kavita-ganesan.com/news-classifier-with-logistic-regression-in-python/#:~:text=More%20importantly%2C%20in%20the%20NLP,algorithm%20for%20text%20related%20classification.> [Accessed 19 May 2022].

Harrison, O., 2018. *Machine Learning Basics with the K-Nearest Neighbors Algorithm*. [online] Medium. Available at: <https://towardsdatascience.com/machine-learning-basics-with-the-k-nearest-neighbors-algorithm-6a6e71d01761#:~:text=Summary-,The%20k%2Dnearest%20neighbors%20(KNN)%20algorithm%20is%20a%20simple,that%20data%20in%20use%20grows.> [Accessed 15 May 2022].

Hatebase.org. n.d. *Hatebase*. [online] Available at: <https://hatebase.org/> [Accessed 14 May 2022].

Hatebase.org. 2022. *About-Hatebase*. [online] Available at: <https://hatebase.org/about> [Accessed 9 May 2022].

IBM cloud education, 2020. *What is Machine Learning?*. [online] Ibm.com. Available at: <https://www.ibm.com/uk-en/cloud/learn/machine-learning> [Accessed 13 May 2022].

IBM Cloud Education, 2020. *What are Neural Networks?*. [online] Ibm.com. Available at: <https://www.ibm.com/cloud/learn/neural-networks> [Accessed 14 May 2022].

Jaki, S. and De Smedt, T., 2019. Right-wing German Hate Speech on Twitter: Analysis and Automatic Detection. *Arxiv*,.

Jayaswal, V., 2022. *Understanding Naïve Bayes algorithm*. [online] Medium. Available at: <https://towardsdatascience.com/understanding-na%C3%AFve-bayes-algorithm-f9816f6f74c0#:~:text=Na%C3%AFve%20Bayes%20algorithm%20is%20efficient,c%20is%20the%20total%20classes.> [Accessed 24 May 2022].

J. Garbade, D., 2018. *Understanding K-means Clustering in Machine Learning*. [online] Medium. Available at: <https://towardsdatascience.com/understanding-k-means-clustering-in-machine-learning-6a6e67336aa1> [Accessed 14 May 2022].

Jordan, J., 2017. *Evaluating a machine learning model.*. [online] Jeremy Jordan. Available at: <https://www.jeremyjordan.me/evaluating-a-machine-learning-model/#:~:text=The%20three%20main%20metrics%20used,the%20number%20of%20total%20predictions.> [Accessed 19 May 2022].

Kharpal, A., 2016. *US tech giants like Facebook could face new EU laws forcing them to tackle hate speech*. [online] CNBC. Available at: <https://www.cnbc.com/2016/12/05/us-tech-giants-facebook-illegal-hate-speech-face-new-eu-laws.html> [Accessed 3 May 2022].

Lee, J., Ko, S. and Han, Y., n.d. *SALNet: Semi-Supervised Few-Shot Text Classification with Attention-based Lexicon Construction*. Kangwon National University, Kangwon, Republic of Korea.

Li, S., 2018. *Multi-Class Text Classification Model Comparison and Selection*. [online] Medium. Available at: <https://towardsdatascience.com/multi-class-text-classification-model-comparison-and-selection-5eb066197568#:~:text=Linear%20Support%20Vector%20Machine%20is,the%20best%20text%20classification%20algorithms.&text=We%20achieve%20a%20higher%20accuracy,5%25%20improvement%20over%20Naive%20Bayes.> [Accessed 17 May 2022].

Lub, Z., 2019. *Hate Speech on Social Media: Global Comparisons*. [online] Council on Foreign Relations. Available at: <https://www.cfr.org/backgrounder/hate-speech-social-media-global-comparisons> [Accessed 1 May 2022].

Ma, Y., 2020. *NLP: How does NLTK.Vader Calculate Sentiment?*. [online] Medium. Available at: <https://medium.com/ro-codes/nlp-how-does-nltk-vader-calculate-sentiment-6c32d0f5046b> [Accessed 19 May 2022].

Mohiyaddeen, M. and Siddiqui, S., 2021. Automatic Hate Speech Detection: A Literature Review. *SSRN Electronic Journal*,.

Müller, K. and Schwarz, C., n.d. *Fanning the Flames of Hate: Social Media and Hate Crime*. The University of Warwick.

Nielsen, M., 2019. *Neural Networks and Deep Learning*. Online, pp.Chapter 1 and 2.

scikit-learn. n.d. *1.9. Naive Bayes*. [online] Available at: <https://scikit-learn.org/stable/modules/naive_bayes.html> [Accessed 5 May 2022].

Nltk.org. n.d. *5. Categorizing and Tagging Words*. [online] Available at: <https://www.nltk.org/book/ch05.html> [Accessed 7 May 2022].

Nnwj.de. n.d. *Supervised and unsupervised learning - Neural Networks with Java*. [online] Available at: <https://www.nnwj.de/supervised-unsupervised.html#:~:text=The%20learning%20algorithm%20of%20a,either%20be%20supervised%20or%20unsupervised.> [Accessed 15 May 2022].

Pykes, K., 2020. *Part Of Speech Tagging for Beginners*. [online] Medium. Available at: <https://towardsdatascience.com/part-of-speech-tagging-for-beginners-3a0754b2ebba> [Accessed 16 May 2022].

scikit-learn. n.d. *sklearn.feature_extraction.text.TfidfVectorizer*. [online] Available at: <https://scikit-learn.org/stable/modules/generated/sklearn.feature_extraction.text.TfidfVectorizer.html> [Accessed 17 May 2022].

Schulze, E., 2019. *EU says Facebook, Google and Twitter are getting faster at removing hate speech online*. [online] CNBC. Available at: <https://www.cnbc.com/2019/02/04/facebook-google-and-twitter-are-getting-faster-at-removing-hate-speech-online-eu-finds--.html> [Accessed 2 May 2022].

Stecanella, B., 2017. *Support Vector Machines (SVM) Algorithm Explained*. [online] MonkeLearn. Available at: <https://monkeylearn.com/blog/introduction-to-support-vector-machines-svm/> [Accessed 15 May 2022].

Taub, A. and Fisher, M., 2018. *Facebook Fueled Anti-Refugee Attacks in Germany, New Research Suggests (Published 2018)*. [online] Nytimes.com. Available at: <https://www.nytimes.com/2018/08/21/world/europe/facebook-refugee-attacks-germany.html> [Accessed 13 May 2022].

Techopedia.com. 2021. *What is Tokenization? - Definition from Techopedia*. [online] Available at: <https://www.techopedia.com/definition/13698/tokenization> [Accessed 19 May 2022].

Tortella, M., 2020. *How to build a (quasi) real-time hate speech classifier for Twitter*. [online] Medium. Available at: <https://towardsdatascience.com/how-to-build-a-quasi-real-time-hate-speech-classifier-for-twitter-286d7cc440c2> [Accessed 20 May 2022].

Twitter. n.d. *Hateful conduct policy*. [online] Available at: <https://help.twitter.com/en/rules-and-policies/hateful-conduct> [Accessed 1 April 2022].

Tyagi, N., 2020. *What is K-means Clustering in Machine Learning? | Analytics Steps*. [online] Analyticssteps.com. Available at: <https://www.analyticssteps.com/blogs/what-k-means-clustering-machine-learning> [Accessed 14 May 2022].

UNESCO, n.d. *Addressing hate speech on social media: Contemporary challenges*. [online] Unesdoc.unesco.org. Available at: <https://unesdoc.unesco.org/ark:/48223/pf0000379177/PDF/379177eng.pdf.multi> [Accessed 17 May 2022].

Valdarrama, S., 2021. *Considerations when choosing a machine learning model*. [online] Medium. Available at: <https://medium.com/p/aa31f52c27f3> [Accessed 19 May 2022].

Verhulst, Pierre-François (1838). "Notice sur la loi que la population poursuit dans son accroissement" (PDF). Correspondance Mathématique et Physique. 10: 113–121. Retrieved 3 December 2014

Appendix

Timeline and workplan

First Semester:

Up to the project overview:

i. Decide on a topic

ii. Investigation on hate speech report and the best techniques to analyse databases for finding hate speech (neural networks, bag of words, etc).
iii. Setting up a meeting with lecturer and PHD student to talk about ideas on how to analyse hate speech.
iv. Find more literature studies on hate speech and offensive language on social media
v. Find more about Twitter's measurements against hate speech
vi. Write aims and objectives for my project

After project overview

vii. Start analysing the database
viii. Learning how to use one specific (previously decided) approach to do the previous point
ix. Start pre-processing implementation
x. Begin report
1. Write the overview (with feedback given from the interim discussion), the introduction and background to this project. Following on that, discuss the possible analysis approaches that I could use and state which one I will be using in this project

Second Semester:

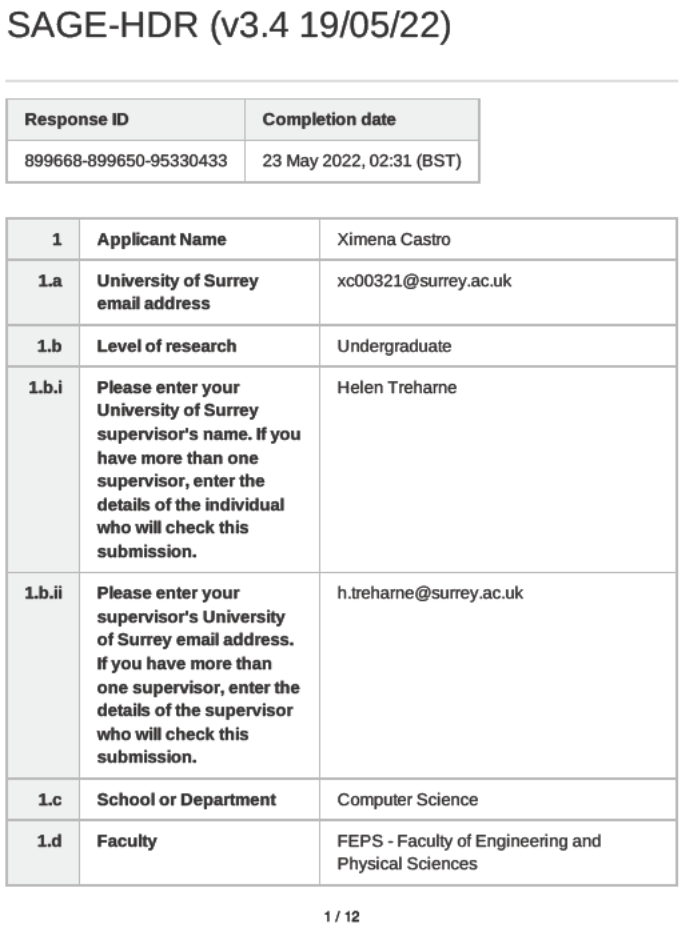
Beginning of January - May

xi. Build different classifiers
xii. Start feeding data to the classifier for text analysis recognition
xiii. Write technical part of the report (analysis)
xiv. Identify and compare the accuracy between the models implemented.

After May:

xv. Start Testing
xvi. Finish Writing the technical side of my report (documentation, comparison, ethics)
xvii. Add screenshots and images of the classifiers into the report
xviii. Check all the literature and references (bibliography) of the report is done correctly as well as the organisation of the report.

SAGE Form:



SAGE-HDR (v3.4 19/05/22)

Response ID	Completion date
899668-899650-95330433	23 May 2022, 02:31 (BST)

1	**Applicant Name**	Ximena Castro
1.a	**University of Surrey email address**	xc00321@surrey.ac.uk
1.b	**Level of research**	Undergraduate
1.b.i	**Please enter your University of Surrey supervisor's name. If you have more than one supervisor, enter the details of the individual who will check this submission.**	Helen Treharne
1.b.ii	**Please enter your supervisor's University of Surrey email address. If you have more than one supervisor, enter the details of the supervisor who will check this submission.**	h.treharne@surrey.ac.uk
1.c	**School or Department**	Computer Science
1.d	**Faculty**	FEPS - Faculty of Engineering and Physical Sciences

1 / 12

The rest of the SAGE form could not be added into the word document and can be found in the zip folder of the documents submitted.