Predicting review ratings with the help of sentiment analysis

Comparing sentiment analysis pre-processing techniques with machine learning models

Marc Vogtländer

THESIS SUBMITTED IN PARTIAL FULFILLMENT
OF THE REQUIREMENTS FOR THE DEGREE OF
MASTER OF SCIENCE IN DATA SCIENCE & SOCIETY
DEPARTMENT OF COGNITIVE SCIENCE & ARTIFICIAL INTELLIGENCE
SCHOOL OF HUMANITIES AND DIGITAL SCIENCES
TILBURG UNIVERSITY

STUDENT NUMBER
2080254

E-MAIL
m.c.vogtlander@tilburguniversity.edu

THESIS COMITTEE
Dr. P. Hendrix
Dr. A. Alishahi

LOCATION
Tilburg University
School of Humanities and Digital Sciences
Department of Cognitive Sciences & Artificial Intelligence
Tilburg, The Netherlands

DATE
February 13, 2023

WORD-COUNT
7756

Acknowledgements

In this thesis, I would like to thank Mr. Hendrix for organising the weekly meetings and providing detailed feedback on the questions that arose. I am also delighted to have been assigned to a motivated research group which, due to the questions that arose, raised the level of this thesis tremendously. I also want to thank Tilburg University for making research papers and other useful resources available. Lastly, I would like to thank Kaggle for making datasets publicly available. These people and organisations had a considerable contribution to the thesis.

Comparing pre-processing techniques with machine learning models

Abstract

People love to share their opinions about the products and services they've bought. The internet nowadays is an easy medium on which to do so. The opinions contain information about whether they liked the product or service, or not. In Natural Language Processing (NLP) there are several models that can classify such reviews based on the context of sentences, or the usage of words. Different pre-processing techniques such as the removal of stop words, stemming and lemmatization have been invented in the world of NLP to feed text into machines. In this thesis some of the most common pre-processing techniques in the NLP landscape are compared with various machine learning models to classify text in a glance overview. These techniques and models are compared based on the F1 and Matthews Correlation Coefficient (MCC) score. In this thesis the following research question will be answered: "To what extent can Amazon product reviews be predicted with the help of sentiment analysis?" In order to answer this, a dataset consisting of Amazon Reviews will be implemented. Overall, it shows that pre-processing techniques, with the help of sentiment classification, do not have any significant impact on the prediction power of machine learning models. Naïve Bayes (NB) shows to be the most robust and best performing model for classifying consumer reviews with an F1 and MCC score of 0.6800 and 0.3701, followed by Decision Trees (DT) with scores of 0.6100 and 0.2253 for spaCy stop words respectively. K-Nearest Neighbors (KNN) seems with an F1 and MCC score of 0.3800 and 0.1036 to predict inadequately.

1 DATA SOURCE/CODE/ETHICS STATEMENT

This thesis did not involve any work on non-public published data collected from human participants or animals, nor for the coding, which was used to define predictive models. The ownership of the data used in this thesis remains with Datafiniti, both during, and after the completion of this thesis. The author of this thesis acknowledges that he does not have a legal claim to the data obtained from Datafiniti nor to the helping code obtained from Kaggle. The author of this project has evaluated his project according to the "DSS Master Thesis guidelines - vF2022" and the "Data Source/Code/Ethics Statement". In the case of usage of this thesis in the form of publicity, the supervisor(s) of this project have permission from the author to do so. Images used in this thesis, when not produced by the author, were licensed under the public domain licence in which the author at all times respected the Intellectual Property Right (Netherlands Enterprise Agency RVO, 2022).

2 PROBLEM STATEMENT & RESEARCH GOAL

2.1 Context

As already briefly mentioned, customer reviews play an important role in the purchasing journey of products and services with 95% of people first checking reviews before proceeding to purchase (Wu, Liu, Teng, Zhang, & Xie, 2021). Customer reviews are mainly written by users who had a positive, negative or sometimes neutral experience. These users can post a review describing their feelings using text that will be read by potential users. Usually, such reviews are paired with a rating visualising their happiness. Well known organisations such as Facebook and Google Reviews offer within their platforms user experience services wherein users express their feelings about a certain product or service. Some organisations like TripAdvisor (travel-based reviews), Airbnb (stay-based reviews and Zomato (restaurant experience-based reviews) make user experiences their unique selling point and are very dependent on such customer experiences (Baker, 2018). 'The McKinsey Podcast' highlights once more the importance of consumer reviews wherein the podcast states that companies should pay more attention to customer reviews (Podcast, 2022) and if neglecting to do so, a gap is created between organisations that put the focus on this specific task. This raises questions on how organisations can bring added value to their businesses based on customer reviews and how to know which models and techniques to use to investigate customer reviews. These issues will be addressed extensively in section 5 together with the power of NLP. One of the subdomains of NLP is sentiment analysis which is the technique of applying sentiment scores on textual data. In section 6, pre-processing techniques that are useful for sentiment analysis, are applied extensively to classify reviews as positive and negative.

2.2 Societal & Scientific relevance

Comparing different machine learning models and developing an accurate algorithm will provide adequate reasoning on how different models, along with different pre-processing techniques perform on a dataset containing consumer reviews. Haque, Saber, & Shah (2018) propose in the discussion section of their research on sentiment analysis for Amazon reviews, to conduct further research on generalisation of machine- and deep learning- models by applying different pre-processing features on textual data and afterwards, comparing the model performances. In most research done on textual reviews, only a few models are compared with each other without several pre-processing techniques or conversely. Providing a convenient design in which machine- and deep learning- models and techniques result in the highest performance, what to look out for, what problems may arise, and how to encounter these problems are implemented in the topic of classifying reviews as positive or negative.

2.3 Research strategy

Solutions on the societal and scientific problems that arise when dealing with consumer review classification will be answered by establishing the following research question: **RQ1:** To what extent can the text of Amazon reviews be classified as positive or negative with the help of sentiment analysis? This research question however is still broad and vague and is therefore separated into the following research sub questions: **RSQ1:** Which sentiment analysis techniques are best suited for classifying Amazon text reviews and how do they improve their performance? **RSQ2:** How does the performance of Naïve Bayes, Support Vector Machines, Gradient Descent, XGBoost, KNN and Decision Tree machine learning algorithms compare for Amazon's reviews? **RSQ3:** To what extent is the impact of sentiment analysis on the predictive power of machine learning models consistent across the binary positive and negative classification? **RSQ4:** To what extent does a balanced dataset impact sentiment analysis on the predictive power of machine learning models?

Comparing pre-processing techniques with machine learning models

2.4 Findings

Many different pre-processing techniques and NLP models are available for text review classification. Naïve Bayes performed the best while predicting consumer reviews followed by Decision Trees. KNN performed poorly on classifying both validation- and test- data. The models are compared by the Matthews correlation coefficient (MCC), macro F1, and the AUC (Area Under the Curve) score. In general, text pre-processing techniques seem not to have a significant impact on the model performances. Handling imbalance by applying Synthetic Minority Oversampling Technique (SMOTE) increases the performance for all of the models and techniques used. As for the deep learning models RNN showed to have the highest AUC score, however applying the deep learning models raised some issues which will be discussed in section 7.2. RandomizedSearchCrossValidation (RSCV), together with hyperparameter tuning ensures that performances and model consistency in the dataset are robust and consistent.

3 LITERATURE REVIEW

3.1 Introduction to Natural Language Processing

Sentiment analysis is a subject of Natural Language Processing. With the use of sentiment analysis, it is possible to transform textual data into positive, negative or neutral feedback scores. Many techniques and models are available to implement sentiment analysis. In this section some NLP models are implemented on a specific task at hand. Monitoring the attitude from the customer with the help of sentiment analysis, makes positive and negative text reviews able to be classified with the help of star ratings as corresponding labels (Thematic, n.d.).

3.2 Sentiment analysis techniques

Understanding the sentiment analysis pipelines gives a better understanding of the sentiment analysis techniques that can be used in applicated machine learning models.

3.2.1 Feature importance

Text data contains features that need to be extracted before sentiment analysis techniques can be applied. Sharma, Sabharwal, Goyal, & Vij (2020) propose the feature extraction techniques Term Frequency to find word occurrences in a corpus, and Term Presence to indicate whether a word is in the corpus or not. Pang & Lee (2008) prefers Term Presence over Term Frequency for text classification, as Term Presence takes rare words that may carry important context into account.

Haque, Saber, & Shah (2018) applied TF-IDF (Term Frequency-inverse document frequency) and BoW (Bag of Words) and Chi-square techniques on amazon consumer reviews in which TF-IDF and BoW showed to be the best performing feature extraction techniques applied on a 5- and 10-fold cross validation. However, the performance metrics of Haque, Saber, & Shah (2018) doesn't underpin his theory. The author proposes to continue generalising these models further on textual data. Some of the machine learning algorithms in the study are compared next to deep learning networks reporting model performances in a concise manner with the corresponding evaluation metrics.

3.2.2 N-grams

Several N-gram techniques can be used to determine sentiment in textual data. N-Gram techniques are, however, task dependent and represent occurrences in a text with their features. N-Grams can be based on character level, but also on word level. Pang & Lee (2008) showed that unigrams outperformed bigrams in a task in which opinion words were classified. Dave, Pennock, & Lawrence (2003) however, concluded that unigrams performed worse than bigrams and trigrams on classifying consumer reviews as positive, neutral, or negative. Even though both papers are somewhat outdated, they still contain relevant information for the task at hand, as the publishers, which are nowpublishers & Google Research Department publish very accurate papers in the field of data science.

3.2.3 PoS tagging

Sharma, Sabharwal, Goyal, & Vij (2020) opposed another technique to classify textual data with Part of Speech (PoS). PoS is a feature extraction technique that captures features, based on the type of context in a sentence or word. It takes as input multiple word N-grams and tries to capture the context of the word by tagging each word to phrasal definitions such as nouns, subjects or adjectives.

3.2.4 Negations

Negations are N-grams that may be difficult to capture as some word N-grams may have a positive polarity but contain a negative sentiment such as ‘not’ or ‘neither’. In the study of (Kothiya, 2019), negations are used on the minority class to gather more data in a dataset which is especially helpful when the dataset contains imbalance.

3.2.5 Stop words and lemmatization

Miao, Jin, Zhang, Chen, & Lai (2021) compared stop words and lemmatization techniques from the NLTK, spaCy and sklearn package with machine learning models to analyse model performances. Miao, Jin, Zhang, Chen, & Lai (2021) states that there is no optimal data cleaning method for every pre-processing technique.

3.2.6 Relevance

The focus will be on the different varieties of sentiment analysis techniques and on comparing different machine learning algorithms with these techniques. Several sentiment pre-processing techniques and approaches are useful to classify textual data as positive or negative (Sharma, Sabharwal, Goyal, & Vij, 2020). Comparing feature extracting techniques such as PoS tagging and Term Frequency to consumer reviews is optimal for finding the best performing technique for every model.

3.3 Sentiment analysis algorithms and their performances

There are various models that are able to classify textual reviews. Studies show various machine- and deep learning- models on textual datasets to classify a review as positive or negative. In this section, some of the machine- and deep learning- models are proposed for comparable tasks at hand.

3.3.1 Machine learning algorithms

As already mentioned earlier, sentiment analysis has several kinds of techniques and algorithms ready to use. Not only pre-processing, and sentiment analysis techniques can be topic specific, one algorithm and software may also be better in performance than another. Ahmad, Muhammad, Aftab, & Awan (2017) implemented Multi-Layer Perceptron (MLP), NB and Support Vector Machines (SVM) to binary classify movie- and product- reviews. Table 1 shows the most important features of the models implemented. In the study, SVM shows the highest accuracy for binary classifying movie- and product- reviews with an accuracy score of 81.15% and 79.40% for the reviewed datasets. Closely following up is MLP (81.05% and 79.27%) and NB shows the lowest performance with (75.50% and 62.50%) respectively.

Table 1

Different machine learning algorithms applied on movie- and product- reviews with their corresponding features (Ahmad, Muhammad, Aftab, & Awan, 2017)

Tool name	**Features**
MLP	• Robust & non-linear neural network model • Used as universal function approximator • It has one hidden layer and multiple non-linear units
Naïve Bayes (NB)	• Supervised classifier for binary classification • Based on Bayes' Theorem • Useful for large datasets
SVM	• Based on supervised learning model • Can handle linear separation on high dimensional non-linear input data using an appropriate kernel • Multiple derivatives and extensions are available

3.3.2 Deep learning algorithms

Wang, Jiang, & Luo (2016) implemented different recurrent neural networks (RNN) and convolutional neural networks (CNN) on text data to classify sentences as positive, negative, neutral or very positive. Typically, within text classification, keywords need to be converted into high-dimensional matrices which are likely to contain morphology, syntax, and semantics issues before feeding into the deep learning models. Wang, Jiang, & Luo (2016) used the word2vec method to encode word embeddings into an embedding matrix based on sentence level. Wang, Jiang, & Luo (2016) used three different corpora on which to compare the deep learning models. These corpora consist of movie reviews, the Stanford Sentiment Treebank (SST with 5 labels), and the SST with binary labels and neutral reviews removed. Study shows that using word2vec as a pre-processing technique, gives an accuracy score of 82.28% for CNN models combined with a LSTM or GRU layer (on the movie review corpora). The performance however doesn't differ that much from the CNN model of Kalchbrenner et al. (2014) which was 81.5%, and Kim (2014) 81.1%. The complete results are shown in appendix II.

Haque, Saber, & Shah (2018) applied machine learning algorithms on consumer reviews based on item categories. Haque, Saber, & Shah (2018) considered the 3-star reviews as neutral and therefore removed them from the target labels. A Supervised Pool Based Active Learning method was used to label the reviews as positive or negative. Text reviews were tokenized, whereafter stop words were removed and PoS tagging was applied. Six different machine learning approaches were used out of which SVM again showed to be the best performing model in the musical subcategory with an accuracy score of 0.94 and F1 score of 0.98. Appendix IV shows the full results of the musical subcategory.

3.3.3 Relevance

Evaluating some of the proposed deep learning algorithms (CNN and RNN) together with machine learning algorithms (Naïve Bayes, KNN, Decision Tree, SVM, XGBoost and Gradient Boosting) shows that models are robust and consistent across textual data. Comparing these models with the feature extraction techniques mentioned earlier may show surprising results regarding sentiment analysis pre-processing techniques. The research furthermore shows that implementing CNN and RNN models separately does not considerably decrease the classification accuracy for binary labels (Haque, Saber, & Shah, 2018).

3.4 Prediction power for models on sentiment analysis techniques

Miao, Jin, Zhang, Chen, & Lai (2021) implemented negations, stop words and lemmatization techniques on 434,891 steam reviews which were binary labelled as positive and negative. SVM, NB and Random Forest (RF) were compared against each pre-processing technique to show which pre-processing technique and model performed best on review classification. Pre-processing techniques seemed to improve the performance for classifying steam reviews for all the models except for SVM, which performed best on unprocessed textual data.

3.4.1 Pre-processing techniques

Miao, Jin, Zhang, Chen, & Lai (2021) applied data cleaning methods with TF-IDF to capture common words, tokenizing the words afterwards, removing stop words, lemmatizing and PoS tagging. Afterwards models were applied which showed that RF outperformed SVM and NB on all pre-processing techniques with a consistent F1-score of 0.89 to 0.90. Miao, Jin, Zhang, Chen, & Lai (2021) concluded however that there is no consistently best performing pre-processing technique and emphasises the importance of considering several data cleaning methods to find the best performing models for classifying sentiment analysis techniques. Appendix III shows the F1 score for every model compared against the pre-processing techniques.

3.4.2 Relevance

In this thesis similar sentiment analysis techniques will be used to evaluate the predictive power of model performances. It is important to get an understanding of the different metrics available with which to compare models. Miao, Jin, Zhang, Chen, & Lai (2021) implemented the F1 score due to having an imbalanced dataset. The MCC score however may also be an appropriate metric as it takes True Negatives into account which is useful when having a highly imbalanced classification problem (Sisters, 2020).

Comparing pre-processing techniques with machine learning models

3.5 Impact of a balanced dataset in contrast to an imbalanced dataset

Handling imbalance is an important aspect while comparing model performances. Earlier Sisters (2020) proposed the MCC score which takes true negatives into account while Miao, Jin, Zhang, Chen, & Lai (2021) implemented the F1 score to compare model performances on imbalanced textual reviews.

3.5.1 SMOTE

Shaikh, Daudpota, Imran, & Kastrati (2021) shows that balancing data while using SMOTE together with text generation algorithms, increases the performance of deep neural networks with 17% on text classification. Interchangeably Ustuner, Sanli, & Abdikan (2016) showed that machine learning- and deep learning- classification algorithms are affected by imbalanced data and that balancing the data increase performance from 85.94% to 90.94% for machine, and 88.44% to 91.56% for neural networks while classifying images.

3.5.2 Relevance

One of the core procedures before starting with text pre-processing is to check and handle imbalance in the dataset by applying various balancing techniques. This topic highlights the importance of dealing with data imbalance and what the impact can be of ignoring data imbalance. Furthermore, it shows that dealing with data imbalance can increase model performance drastically.

3.6 Additional value of the thesis

The literature review shows that machine learning models and deep learning models can perform well in classifying textual data in a binary or ordinal way. Usually, the accuracy score is established to compare models with each other, where some papers also report precision, recall and the F1 score. In the literature review, various pre-processing techniques are applied on textual data to feed the models while most of the datasets were rarely skewed for the target variable. This thesis will provide a concise overview for machine- and deep learning- model comparisons as opposed earlier (Haque, Saber, & Shah, 2018) to continue generalising machine- and deep learning- models on different pre-processing techniques and their corresponding features. Generalisation will be elaborated on extensively on the applied machine- and some deep learning- models and are compared to their sentiment analysis techniques.

4 METHODOLOGY

In this section the general descriptions of the proposed methods are described. General elaborations on the models, corresponding with their evaluation metrics, pre-processing techniques and generalisation methods are explained in this section. Section 5 will provide a more comprehensive description of the models and pre-processing techniques.

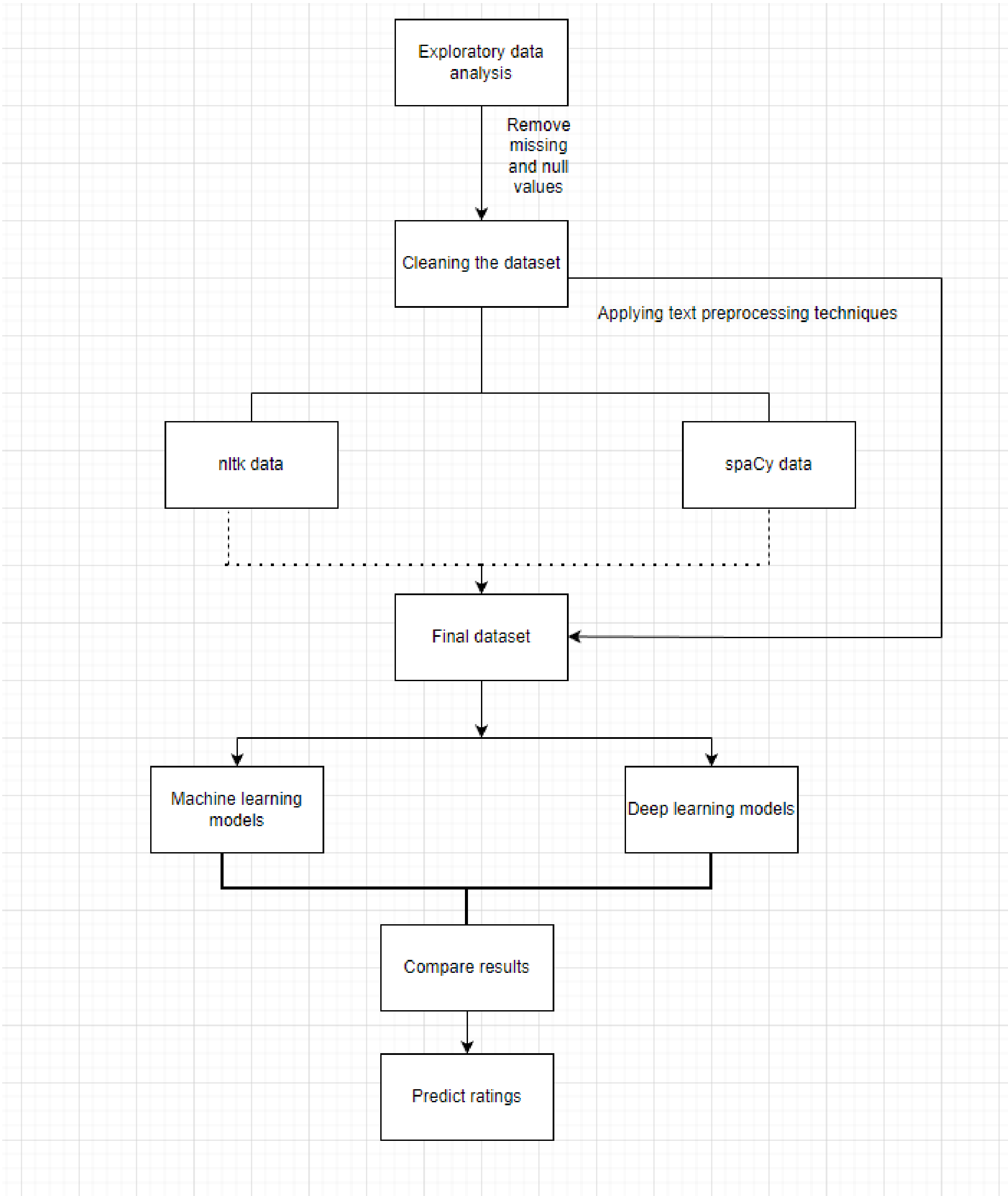
First, the dataset was obtained from the Kaggle website and exploratory data analysis was implemented to find the most important columns and their missing values for classifying sentiment analysis. Pre-processing techniques were implemented in order to omit missing values and remove noise in the text reviews. Several pre-processing techniques from the spaCy and NLTK package were used to answer RSQ1, and afterwards SMOTE was applied to balance the dataset to answer RSQ4. These pre-processing techniques were applied on several machine learning models to answer RSQ2. After applying the pre-processing techniques, RSCV was applied to evaluate the best generalised model performance for sentiment analysis answering RSQ3.

5 EXPERIMENTAL SETUP

5.1 Data science workflow

Figure 1 shows an overview of the data science workflow and methods previously mentioned in section 4.

Figure 1. The Data Science workflow



Comparing pre-processing techniques with machine learning models

5.2 Dataset Description

Sentiment analysis will be applied on the dataset named: 'Consumer Reviews of Amazon Products'. This dataset contains 34,660 unique values with 21 different features. Appendix I shows a clear overview of the features. The dataset is a sample dataset gathered from Kaggle which contains imbalances in the review ratings as can be seen in figure 2. The review ratings of consumer texts are negatively skewed towards positive reviews. While exploring the dataset there are independent variables that are not useful for classifying reviews such as the user ID, Amazon identification number or the source of review generation. Most of these variables only consist of a limited number of categories. In this research, the assumption is made that the variable 'user ID' is unique and therefore isn't being written by the total number of user ID's (which would mean only 8 different users wrote reviews). The main independent variable that influences the target variable is the text reviews variable. This variable contains textual data of different lengths (see figure 5) and is dependent on the ratings given in the review score range of 1 to 5.

In the dataset there are many independent variables that could be useful for predicting review scores such as the geographical features (as certain consumers may be more positive in a certain area than others). Unfortunately, but understandably, these features are anonymized due to the GDPR (General Data Protection Law). Features such as: "Was this review helpful?" & "Reviews did purchase" could also be helpful for prediction although these contain lots of null values.

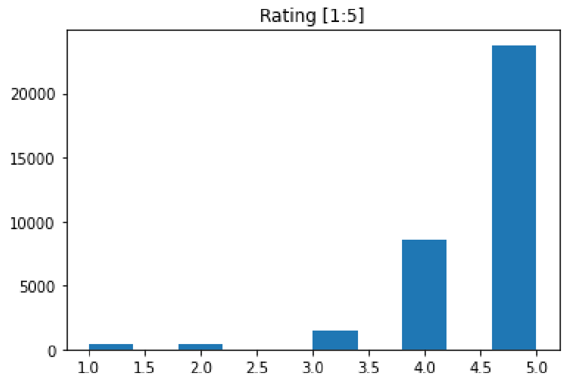


Figure 2. Distribution of the original dataset before labelling reviews as positive or negative

5.3 Label imbalance

Balancing the dataset makes it difficult for pre-processing techniques such as SMOTE to perform well while there is sparsity in negative reviews (Xiong & Lee, 2011). Figure 3 shows the sparsity of the 'negative reviews' while labelling the review ratings from 1-3 as negative and 4-5 as positive.

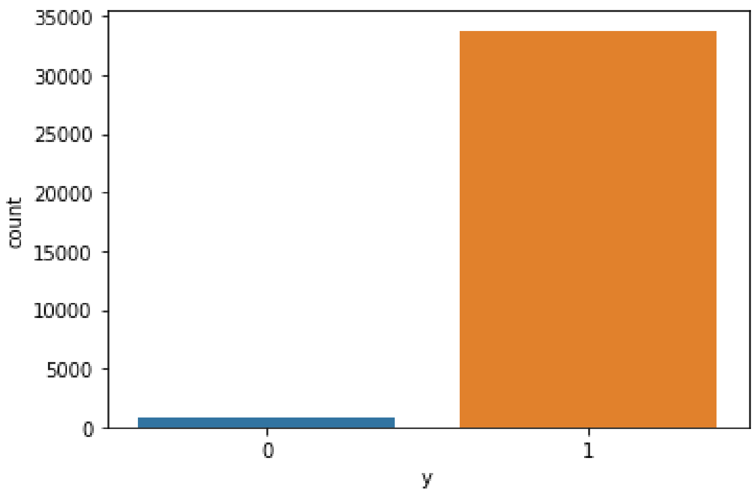


Figure 3. Data imbalance while labelling review ratings from 1-3 as negative and 4-5 as positive

Figure 4 shows the imbalance while categorising the review ratings of 1-4 as negative, and 5 as positive. This shows that the SMOTE algorithm is much more effective for the data in figure 3 as in figure 2**.** The algorithm has more data for which to create 'negative reviews' syntactic samples. Therefore, the review ratings consisting of 1 to 4 are categorised as negative reviews and get the label of 0, while five-star ratings are classified as positive reviews which are labelled as 1. In the context of the subject, the 'positive' label has the meaning of reviews being excellent. Binary classification models will be used to predict these labels. In section 7.1, this implementation method will be further substantiated.

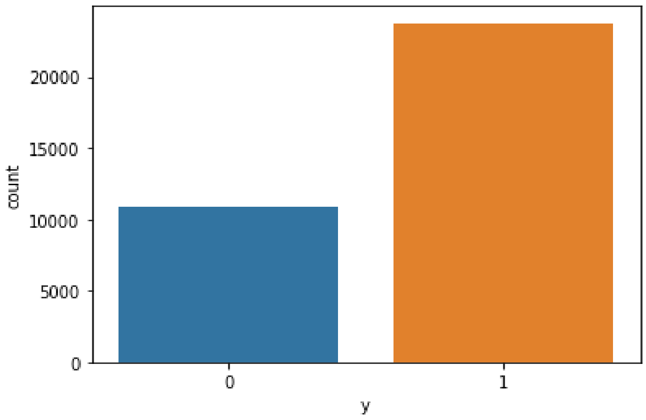


Figure 4. Data imbalance while labelling review ratings from 1-4 as negative and 5 as positive

5.3.1 Oversampling minority class

To handle imbalance there are two oversampling techniques that will be used which are random sampling and the Synthetic Minority Over-sampling Technique (SMOTE). Random sampling takes random samples of the minority class and duplicates these values while SMOTE takes X as K-Nearest Neighbors to generate artificial samples which are between X (Brownlee, 2020). According to Xiong & Lee (2011), SMOTE is preferred as it generates more variety between the generated samples. Random sampling showed to give lower performance measures for the models than synthetic sampling and therefore SMOTE is implemented. Oversampling the minority class will be an applicable imbalance technique, as under sampling the majority class results in loss of information (Brownlee, 2020).

5.4 Stop words

In the dataset the most frequent occurring words are stop words. Getting rid of stop words in NLP is useful as they provide no meaningful information and reduce the training speed for models as is seen in appendix III. NLTK and spaCy both provide a package for stop word removal which will be one of the pre-processing techniques being applied on the dataset. Appendix X shows the most common stop words that were found in the dataset.

5.5 Review length

While visualising the tokenized reviews, most of the reviews contained around five to fifteen words. The review length is important for exploratory data analysis (EDA) but also for the deep learning models as these models require an input of the same shape. In the model the arbitrary number of 15 has been set to give the RNN and CNN sentences of a maximum length of 15 as this accounts for most of the data as is seen in figure 5.

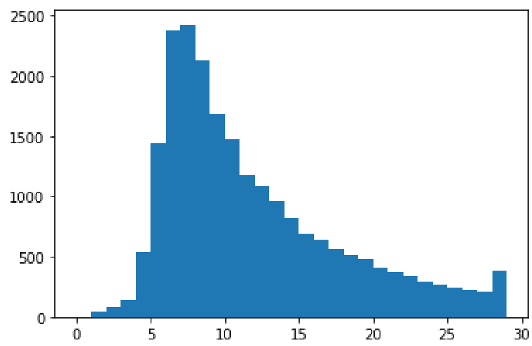


Figure 5. Distribution of the length of the reviews

5.6 Cleaning & Pre-processing techniques

In the dataset there are some values that are missing completely at random (MCAR). The assumption of MCAR is made since there is no causal relation between the missing values and the observations (Tamboli, 2021). 34 values out of the 34,000 values are missing wherein, after applying EDA, 2 out of the 34 values contained negative reviews. In figure 6 the missing values are visualised. The values are omitted such that only a miniscule amount of context is being lost. After omitting the missing values there are 34,626 reviews containing a rating.

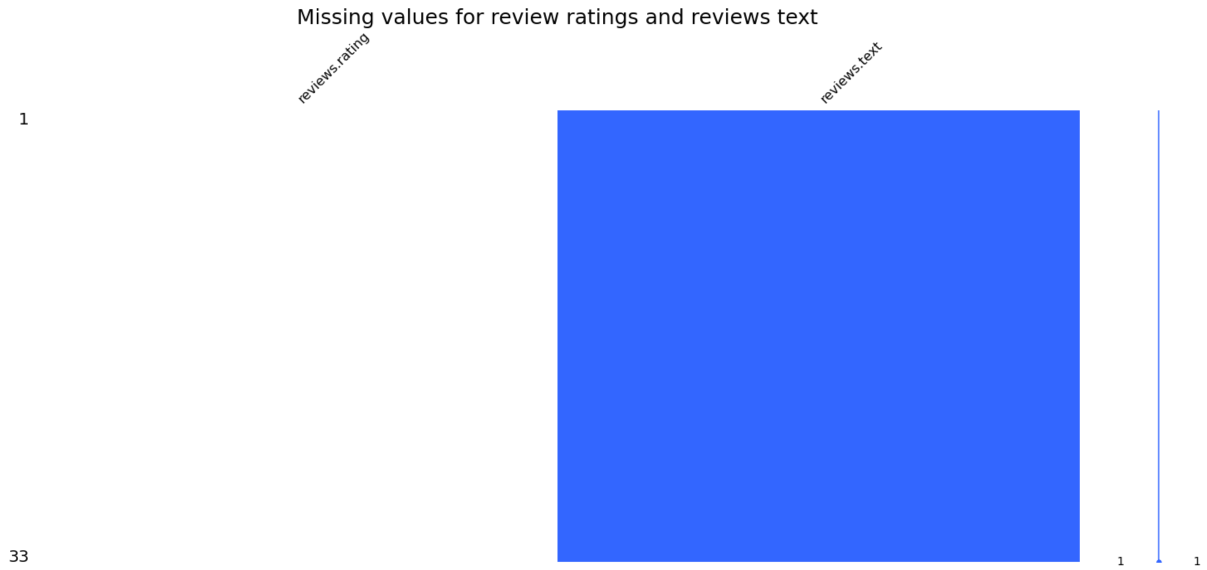


Figure 6. Missing values from the missingmo package

5.7 Splitting the dataset

Splitting the data into a training (validation) and test set is helpful in generalising models. There is no golden standard for splitting the data in some ratio. Joseph (2022) however, proposes 80:20, 70:30 or 60:40 as the most optimal data splits in cases of practical use. The models used in this case are split 70% into training data and 30% for test data. In order with generalisation procedures, the training data will be split into 5 folds which therefore generates 5 validation data folds to test on.

5.8 Generalisation procedure

Cross validation prevents overfitting and makes the models generalisable. As for Cross validation ten folds are used as this is very common in NLP cases (Haque, Saber, & Shah, 2018). When the model uses a lot of computation power, the runtime will increase and consume a lot of time. In that case, five folds will be used and if this approach still takes up a lot of time, RSCV is implemented. While GridSearchCV searches through all the arguments given for machine learning models, RSCV reduces the computational costs of models as this method randomly chooses a combination of hyperparameters and therefore runs less iterations (Torino, 2020).

The data for both the RNN and CNN models is manually cross validated into five folds with the same number of hidden layers for generalisation procedures.

5.9 Hyperparameter tuning

GridSearchCV and RandomizedSearchCV search through a given input of arguments to find the optimal hyperparameters for machine learning models. As training the models had a long runtime, RSCV is applied to check which model hyperparameters performed best on every pre-processing technique. Five cross validation sets are implemented. The number of iterations for every fold is equal to the space of the hyperparameters as is shown in appendix VI.

The RNN model has two LSTM layers with both 150 memory units and two dense layers to classify the text reviews as positive or negative. A simple CNN network with three hidden layers is implemented to compare the results for the given pre-processing techniques.

Appendix VI shows the arguments given for the machine learning- and deep learning-models. The hyperparameters are chosen through a trial-and-error strategy combined with GridSearchCV (see appendix VIII for the hyperparameter scores for every argument given).

5.9.1 Alpha learning rate

The alpha learning rate is a smoothing parameter. As Naïve Bayes is a probability classifier, this hyperparameter uses Laplace smoothing to check whether the probability of Amazon reviews is for example 0.5 for the positive, and 0.5 for the negative class. The downside of implementing such alpha learning rate is that the 0.5 probabilities give less information for binary classification (Jayaswal, 2020).

5.9.2 N_neighbors

As KNN classifies based on the neighbours around a sample, the N_neighbors hyperparameter takes the number of neighbours into account for the sample to classify on.

5.9.3 Max_depth

Decision Tree splits consists into branches and nodes. The Max_depth parameter is an hyperparameter limitation for the 'depth' of a tree. The depth of the tree is the path from the root branch to the leaf.

5.10 Pre-processing techniques

Some of the pre-processing techniques mentioned by Miao, Jin, Zhang, Chen, & Lai, (2021) will be compared with the NLTK and spaCy packages. The following pre-processing techniques are implemented:

- NLTK stop words

  Removing stop words in the text reviews data which are commonly used according to the NLTK author.

- NLTK stemming

  After stop words removal, NLTK stemming can be applied to get the root form of words by removing their affixes. As an example, this technique will convert ‘eating’, ‘eats’ and ‘eaten’ to ‘eat’. The Snowball Stemmer is used over the Porter Stemmer as this algorithm is more aggressive for converting words to their root form than the Porter Stemmer (Ameeruddin, 2022).

- NLTK lemmatization

  Receiving the lemmas of the words in reviews will be done by using the WordNetLemmatizer. To get to the lemmas of words, the MaxEnt Treebank is used to tag tokens to their corresponding PoS. This PoS tagger is trained on the Treebank corpus. See appendix V for the Treebank code.

- SpaCy stop words

  Removing stop words in the text reviews dataset that are commonly used according to the spaCy author (these differ from the NLTK stop words package).

- SpaCy lemmatization

  SpaCy lemmatization is manually coded by appending tokens that haven’t got punctuations and afterwards applying the spaCy lemmatizer (see appendix V for the code).

All these techniques are implemented after the text is pre-processed. In appendix V the detailed code is given. All pre-processed text takes as input a string variable and lowers, strips and applies regular expressions to the text to return cleaned text data without any character n-gram errors such as duplicated letters.

5.11 Vectorization

Machine learning models can't handle string data as an input. The input needs to consist of numbers. As already mentioned by Haque, Saber, & Shah (2018) in section 3.2, vectorization techniques can transform strings into integers**.** The TF-IDF technique will be implemented as this technique takes the context of word occurrences into account while BoW just counts every single word in a word vector (Huilgol, 2020).

5.12 Algorithms and Software

The programming software Python is used to analyse, explore, and apply the mentioned techniques and models onto the data. All code, algorithms and packages are used within Google Colab. Google Colab has the advantage of running various models straight from the cloud (Van Den Reym, 2020). This is helpful as some models require lots of computational power as was seen earlier in appendix III.

5.13 Machine learning models

Machine learning models such as Naïve Bayes, Support Vector Machine, K-Nearest Neighbors, Gradient boosting and XGBoost will be implemented to classify reviews as positive or negative. These machine learning models perform well on text classification tasks as shown in section 3.3 and in the study of projectpro (2022).

5.14 Deep learning models

According to Wang, Jiang, & Luo (2016) CNN and RNN are models that seem to achieve high accuracies classifying text data. To compare the machine learning models with deep learning models RNN and CNN models are implemented. However, due to simplicity of the architecture (see appendix VI) the models are overfitting (see appendix IX) and therefore not taken into consideration for answering the research questions.

5.15 Packages

As of the packages that were used for coding, NLTK, spaCy, and Regular Expression were imported to handle text pre-processing, NumPy, imblearn, Pandas and Matplotlib for exploratory data analysis and scikit learn and TensorFlow for model building.

5.16 Evaluation method

5.16.1 Intrinsic evaluation

The F1 score is a score which is widely used in datasets that contain imbalance. The F1 score is the harmonic mean between the P (precision) ((TP / (TP + FP)) and R (recall) (TP / (TP + FN)) * and very useful for binary classification (Kampakis, n.d.). The F1 score is calculated as follows: (2 * P * R) / (P + R). Another evaluation method that deals with imbalanced datasets is the Matthew's Correlation Coefficient (MCC) score. In a study Tibau (2019) compared the F1 score against the MCC score for SVM classification models and concluded that the F1 score fluctuates when the positive class is named negative (FP), and the negative class is named positive (FN). The MCC score has the advantage of finding a balance between the true positives and the true negatives. The F1 score is therefore independent of the true negatives while the MCC does depend on the true negatives. The MCC score ranges from -1 to 1 wherein a score of 1 means perfect model predictions, 0 means predictions which are averaged due to chance and -1 gives the inverse prediction (scikit-learn, n.d.) As RSCV is used for generalisation, the mean MCC score and STD (Standard Deviation) per fold are reported in section 6. The AUC, F1 and MCC score will be compared for the machine learning models.

The confusion and evaluation metrics are appropriate applications to visualise and summarise the models results. These visualisations make sure that the AUC, MCC and macro F1 score are reported in one glance (Tan, 2021). Appendix VIII shows the confusion and evaluation matrices for all the implemented models.

** True positive, false positive, true negative and false negative are denoted as TP, FP, TN and FN in this order.*

5.16.1.1 Macro F1 vs weighted F1

The macro F1 score is reported in the resulting section as this score penalises the model for not performing well on the minority classes whereas the weighted F1 score favours the majority class. Due to the fact that the dataset is imbalanced and although techniques are applied to handle this imbalance, the macro F1 is favoured over the weighted F1 score as it does not matter in this context to consider the proportion of each label (Leung, 2022).

5.16.2 Extrinsic evaluation

Intrinsic evaluation focuses on achieving the best scores without the model to overfit. The extrinsic evaluation is the evaluation which compares evaluation methods with the earlier defined research questions. Therefore, research questions will always be favoured above any (preferable) model output.

6 RESULTS

In this section, classification performances for text reviews are presented given the possible pre-processing techniques and models mentioned in section 5. Not all the models were successfully implemented for consumer review classification which is further elaborated on in section 7. In the tables the MCC score, STD score per fold, F1 and AUC score which were discussed in section 5.16, are visualised for the corresponding models and pre-processing techniques.

6.1 Sentiment analysis techniques

"Which sentiment analysis techniques are best suited for classifying Amazon text reviews and how do they improve their performance?"

Sentiment analysis techniques such as stemming (NLTK), lemmatization (NLTK & spaCy) and stop words removal (NLTK & spaCy) are applied on machine learning models to train on different kinds of sentiment analysis techniques. The MCC and AUC score and the STD score of 5 folds are reported. First a baseline model is implemented to check whether sentiment analysis techniques improve the performance.

6.1.1 Baseline model

The baseline models are the models applied on the raw textual data without any pre-processing techniques. The baseline model is visualised in Table 2.

Table 2

Baseline model without any pre-processing techniques

	MCC score	**AUC score**	**STD score**
NB	0.3124*	0.5994	0.0082
KNN	0.0429	0.5052	0.0184
DT	0.2407	0.5983	0.02044

6.1.2 Pre-processing techniques

To check whether pre-processing techniques improve the performance of machine learning models, RSCV is implemented to find the optimal hyperparameters for every model. See appendix VI for the hyperparameter options of every model. Table 3 shows the MCC scores and standard deviation per fold for five folds for RSCV on the training data.

Table 3

MCC scores for pre-processing techniques, compared with the standard deviation for 5 RSCV folds without handling imbalance

	NB		**KNN**		**DT**	
	MCC	STD	MCC	STD	MCC	STD
NLTK STOP	0.3105*	0.0076	0.0658	0.0104	0.2120	0.0080
NLTK STEM	**0.3180***	0.0077	0.0834	0.0311	0.2180	0.0130
NLTK LEMMA	0.3175*	0.0135	0.0657	0.0206	0.2258	0.0064
SPACY STOP	0.3060*	0.0030	0.0932	0.0074	0.2248	0.0092
SPACY LEMMA	0.3085*	0.0084	0.0778	0.0097	0.2309	0.0133

Comparing pre-processing techniques with machine learning models

6.2 The impact of a balanced dataset on prediction power

To what extent does a balanced dataset impact sentiment analysis on the predictive power of machine learning models?

After finding the optimal hyperparameters for every model, SMOTE was applied on the dataset to handle imbalance. Earlier table 3 showed the performance of the models for every pre-processing technique before handling imbalance. Table 4 shows the performance of the models for a balanced dataset with the corresponding STD scores per fold for 5 folds for RSCV on the training data.

Table 4

MCC scores for pre-processing techniques, compared with the standard deviation for 5 RSCV folds after handling imbalance

	NB		**KNN**		**DT**	
	MCC	STD	MCC	STD	MCC	STD
NLTK STOP	0.3786*	0.0081	0.1882	0.0341	0.3773	0.1274
NLTK STEM	0.3780	0.0026	0.1857	0.0351	**0.3832***	0.1016
NLTK LEMMA	0.3782*	0.0063	0.1883	0.0410	0.3714	0.1010
SPACY STOP	0.3613	0.0082	0.2045	0.0482	0.3739*	0.1061
SPACY LEMMA	0.3583*	0.0077	0.2181	0.0497	0.3567	0.1051

Comparing pre-processing techniques with machine learning models

6.3 Model performance

How does the performance of Naïve Bayes, Support Vector Machines, Gradient Descent, XGBoost, KNN and Decision Tree machine learning algorithms compare for Amazon's reviews?

After finding the optimal hyperparameters and pre-processing techniques for every machine learning model, the models are tested to predict Amazon reviews on the unseen test data after applying SMOTE. Table 5 shows the proportion of classifications for every model and pre-process technique. For the full classification report of every model see appendix VIII.

Table 5

Classification report of every model for every pre-processing technique after applying SMOTE

	Model	Classification			
	NB	TN	FP	FN	TP
NLTK stop		0.2051	0.1086	0.1842	0.5021
NLTK stem		0.2054	0.1083	0.1835	0.5028
NLTK lemma		0.2052	0.1085	0.1825	0.5038
SpaCy stop		0.2053	0.1084	0.1824	0.5039
SpaCy lemma		0.1992	0.1176	0.1996	0.4836
	KNN				
NLTK stop		0.3027	0.0111	0.6413	0.0450
NLTK stem		0.2943	0.0225	0.5976	0.0856
NLTK lemma		0.2934	0.0234	0.5878	0.0954
SpaCy stop		0.2883	0.0285	0.5684	0.1148
SpaCy lemma		0.0633	0.2535	0.0771	0.6060
	DT				
NLTK stop		0.1419	0.1718	0.1679	0.5184
NLTK stem		0.1750	0.1418	0.2175	0.4657
NLTK lemma		0.1717	0.1452	0.2157	0.4675
SpaCy stop		0.1801	0.1367	0.2260	0.4572
SpaCy lemma		0.0599	0.2570	0.0349	0.6483

6.4 Model consistency

To what extent is the impact of sentiment analysis on the predictive power of machine learning models consistent across the binary positive and negative classification?

To elaborate further on only the classification scores of the models mentioned earlier, evaluation metrics are conducted. Table 6 shows the corresponding AUC, MCC and macro F1 scores of every model and pre-processing technique for the test data.

Table 6

Evaluation matric of every model for every pre-processing technique after applying SMOTE

	Model	Evaluation matric		
	NB	Macro F1	MCC	AUC
NLTK stop		0.6800	0.3669	0.6928
NLTK stem		0.6800	0.3688	0.6937
NLTK lemma		0.6800	0.3697	0.6941
SpaCy stop		0.6800	0.3701*	0.6943
SpaCy lemma		0.6500	0.3198	0.6683
	KNN			
NLTK stop		0.3000	0.0610	0.5151
NLTK stem		0.3500	0.0812	0.5271
NLTK lemma		0.3600	0.0946	0.5329
SpaCy stop		0.3800	0.1036	0.5390
SpaCy lemma		0.3900	0.1045*	0.5407
	DT			
NLTK stop		0.6000	0.2084	0.6038
NLTK stem		0.6100	0.2230	0.6170
NLTK lemma		0.6000	0.2159	0.6131
SpaCy stop		0.6100	0.2253*	0.6189
SpaCy lemma		0.6100	0.2229	0.6175

7 DISCUSSION

7.1 Imbalance

At the time the research question was formulated, the idea was to classify review scores with a rating from 1 to 2 as negative and 3, 4 and 5 as positive. While doing EDA it was clear that the dataset showed imbalance for the negative reviews and that an imbalance technique had to be applied to solve this problem. Random sampling and using a package that negates sentences showed bad model performance for dealing with imbalance. SMOTE was the best way to handle the imbalance in the dataset. These rating classifications however were very low as the models only had a support of 250 for the negative, and around 10,000 for the positive class. Therefore, the research question changed to identifying which reviews are categorised as excellent. This was done by categorising the 5-star reviews as a positive sentiment and everything below as negative (as mentioned earlier in section 5.3). This resulted in more balance between the data after applying SMOTE, which contributed to a higher support for the 'negative' class. Although SMOTE handled imbalance in the dataset and increased their performances by around 10% there are several other pre-processing techniques that could increase the performance of the models. For example, Kasthurirathne & Gregory Dexter (2021) applied general adversarial networks to create synthetic textual data by which to optimise machine learning model performances.

7.2 Models

In the literature review Support Vector Machines, Gradient Descent and XGBoost were appropriate machine learning models with which to classify textual data. Therefore, these models were also considered in classifying reviews. The models however took hours to be trained together with hyperparameter tuning due to the fact that they used a lot of computation power (especially when combined with hyperparameter tuning) and therefore are not considered in the model performances. Even when running the models overnight the issue arose in which the Google Colab connection was lost at some point.

7.2.1 GridSearchCV vs RandomizedSearchCV (RSCV)

Receiving the optimal hyperparameters was essential for reporting model performance. First GridSearchCV was implemented, however, due to long running times, RSCV was applied on all the machine learning models. This implementation technique differs from the former by randomly going through hyperparameters which ensures less iterations. Using GridSearchCV may therefore improve performance even further.

7.2.2 Deep learning models

Due to the small number of epochs, it is not certain that the RNN model is overfitting. Test loss seems to decrease after the 4th epoch. As LSTM is difficult and requires many trials, future research may implement a RNN model which reduces the training and test loss.

As can be seen in appendix VII**,** the CNN model does overfit on the training and test data and is therefore not reported in the resulting section. Applying dropout and regularisation terms may prevent the model from overfitting but due to all the model comparisons for every pre-processing technique it wasn’t possible to improve the CNN model further in the architecture.

Both the RNN and CNN models were tested on the normal dataset as applying SMOTE wasn’t possible due to lack of knowledge. Sagayaraj, Panem, & Zahoor-Ul-Huq (2020) however, implemented a SMOTE technique on a RNN model for medical classification. Future research could implement the deep learning models together with SMOTE to see if the technique increases the performance just as it does with the machine learning models.

7.3 Pre-processing

TF-IDF has been initialised to encode words into vectors. It is still possible to compare different vectorization techniques for every pre-processing technique such as GloVe and Word2Vec. There are many pre-defined models available for predicting consumer reviews which may perform better than the resulting models in this thesis. The aim however was to show the whole NLP pipeline wherein applying pre-defined models may lack information such as the corpora or lexicon on which the models are trained.

7.4 Societal relevance

Machine learning models are tested and compared on common pre-processing techniques used in the world of NLP. When comparing model performances with the MCC score, all the models seem to give resulting scores between 0 and 0.5. In the documentation of (scikit-learn, n.d.), scores close to +1 seem to represent perfect predictions and scores closer to 0 an average random prediction. Given the fact that most scores are more closely related to 0 than 1, and the fact that only 5-star reviews are classified as positive reviews and ratings below as negative, the discussion arises in how far these model algorithms can be used in society.

The author wishes to highlight that there may be better models and pre-processing techniques to tackle review classification and therefore encourages the reader to work with sentence level, word n-gram and even character n-gram classification techniques for text data.

7.5 Scientific relevance

Beside the F1-score, an additional imbalance score is implemented to check model performance. In the scientific context however, the thesis does not give much additional context to model performances as it wasn't possible to implement the classifying techniques proposed before dealing with the imbalance. (Ordinal) regression classifiers were hard to implement given the fact that the distance for 1- and 2-star reviews isn't necessarily the same as the distance for 4- and 5-star reviews. Therefore, the results for the model performances are implemented with binary classification.

8 CONCLUSION

8.1 Applying SMOTE

Consumers write a lot of reviews which can be collected and gathered into a dataset. The performance of models, however, heavily relies on whether target variables are skewed or not. When dealing with imbalance, applying SMOTE is preferred over random sampling or negations for handling imbalance and the MCC score and/or F1 score are suggested. Generally, applying SMOTE increases the performance of the machine learning models, and especially DT models, with an average of 15 to 16%. Applying a 5-fold RSCV generalises the performance of the models since differences in the models are not significant (see appendix VIII for the full results).

8.2 Pre-processing techniques

Although pre-processing techniques are an important aspect for text classification, the proposed pre-processing techniques seemed barely to improve the performance of the models, with about 0.02 to 0.04 for the MCC score and 0.01 to 0.03 for the AUC score.

8.3 Naïve Bayes

The best performing and most consistent machine learning model is Naïve bayes. The best pre-processing technique for this machine learning model is on NLTK stop words which gives a MCC score of 0.3765 and the standard deviation score of 0.0086 between the folds for fitting onto the training data which shows a robust and consistent model. The F1 score for the labels is 0.57 and 0.76 while the average macro score is 0.67.

8.4 Decision Tree

Decision Trees perform well on the RSCV training and validation data while increasing the model hyperparameters and folds. On the test data however, the model performs less well and is less robust as the standard deviation per fold increases with the hyperparameters (see appendix VIII). Decision Trees favour the positive class over the negative class which could be signs of overfitting or underrepresentation of the target variable (due to random splitting). Given these results the model is therefore considered less generalisable and robust than the Naïve Bayes model.

8.5 K-Nearest Neighbors

KNN performs badly in both training, and testing data as is seen from the AUC score of 0.5407 and the MCC score of 0.1045.

This thesis gives businesses insights in the importance of text classification models and compares model performances from start to finish on some of the many available sentiment analysis pre-processing techniques. It deals with an imbalanced dataset and provides techniques and metrices for future research on which to further develop the classification algorithms.

REFERENCES

Ahmad, M., Muhammad, S. S., Aftab, S., & Awan, S. (2017). *Machine Learning Techniques for Sentiment Analysis: A Review.* ResearchGate. Retrieved November 10, 2022, from https://www.researchgate.net/profile/Shabib-Aftab-2/publication/317284281_Machine_Learning_Techniques_for_Sentiment_Analysis_A_Review/links/59302d6ba6fdcc89e78431ec/Machine-Learning-Techniques-for-Sentiment-Analysis-A-Review.pdf

Ameeruddin, M. (2022, May 31). *What is the difference between porter and snowball stemmer in nltk*. Retrieved November 25, 2022, from projectpro.io: https://www.projectpro.io/recipes/what-is-difference-between-porter-and-snowball-stemmer

Baker, K. (2018, Juni). *The Ultimate Guide to Customer Reviews and Testimonials*. Retrieved September 27, 2022, from blog.hutspot.com: https://blog.hubspot.com/service/customer-reviews-testimonials#:~:text=A%20customer%20review%20is%20a,whether%20it's%20worth%20the%20investment.

Brownlee, J. (2020, January 17). *SMOTE for Imbalanced Classification with Python*. Retrieved November 14, 2022, from machinelearningmastery.com: https://machinelearningmastery.com/smote-oversampling-for-imbalanced-classification/

Haque, T. U., Saber, N., & Shah, F. (2018). *Sentiment analysis on large scale Amazon product reviews.* International Conference on Innovative Research and Development. doi:10.1109/ICIRD.2018.8376299.

Huilgol, P. (2020, February 28). *Quick Introduction to Bag-of-Words (BoW) and TF-IDF for Creating Features from Text*. Retrieved November 15, 2022, from analyticsvidhya: https://www.analyticsvidhya.com/blog/2020/02/quick-introduction-bag-of-words-bow-tf-idf/

Jayaswal, V. (2020, November 22). *Laplace smoothing in Naïve Bayes algorithm*. Retrieved January 12, 2023, from towardsdatascience: https://towardsdatascience.com/laplace-smoothing-in-na%C3%AFve-bayes-algorithm-9c237a8bdece

Kampakis, S. (n.d.). *Metrics: Matthew's correlation coefficient*. Retrieved October 29, 2022, from thedatascientist.com: https://thedatascientist.com/metrics-matthews-correlation-coefficient/

Kothiya, Y. (2019, March 15). *How I handled imbalanced text data*. Retrieved October 26, 2022, from Towards Data Science: https://towardsdatascience.com/how-i-handled-imbalanced-text-data-ba9b757ab1d8#:~:text=The%20simplest%20way%20to%20fix,synthetic%20instances%20from%20minority%20class

Leung, K. (2022, January 4). *Micro, Macro & Weighted Averages of F1 Score, Clearly Explained*. Retrieved January 12, 2023, from towardsdatascience: https://towardsdatascience.com/micro-macro-weighted-averages-of-f1-score-clearly-explained-b603420b292f

Miao, Y., Jin, Z., Zhang, Y., Chen, Y., & Lai, J. (2021). *Compare Machine Learning Models in Text Classification Using Steam User Reviews.* Association for Computing Machinery. doi:10.1145/3507473.3507480

Netherlands Enterprise Agency RVO. (2022, September 27). *Protecting your IP rights*. Retrieved November 28, 2022, from Business.gov.nl: https://business.gov.nl/running-your-business/products-and-services/protecting-your-product-idea-or-innovation/ip-rights/#:~:text=Intellectual%20property%20or%20IP%20is,software%2C%20lyrics%2C%20and%20photographs.

Podcast, T. M. (2022). The McKinsey Podcast [Recorded by R. Fusaro, & L. Rahilly]. Retrieved September 27, 2022, from https://www.mckinsey.com/capabilities/operations/our-insights/why-business-must-heed-customer-reviews

scikit-learn. (n.d.). *sklearn.metrics.matthews_corrcoef*. Retrieved November 24, 2022, from https://scikit-learn.org/stable/modules/generated/sklearn.metrics.matthews_corrcoef.html

Sharma, D., Sabharwal, M., Goyal, V., & Vij, M. (2020). *Sentiment Analysis Techniques for Social Media Data: A Review*. Springer Nature Singapore. doi:10.1007/978-981-15-0029-9_7

Sisters, L. (2020, April 8). *Matthews Correlation Coefficient: when to use it and when to avoid it*. Retrieved January 3, 2022, from towardsdatascience: https://towardsdatascience.com/matthews-correlation-coefficient-when-to-use-it-and-when-to-avoid-it-310b3c923f7e#:~:text=F1%20score%2C%20in%20this%20case,matter%20which%20class%20is%20positive.

Tamboli, N. (2021, October 29). *All You Need To Know About Different Types Of Missing Data Values And How To Handle It*. Retrieved November 14, 2022, from AnalyticsVidhya: https://www.analyticsvidhya.com/blog/2021/10/handling-missing-value/#h2_8

Tan, E. (2021, December 27). *How To Visualize Machine Learning Results Like a Pro*. Retrieved November 3, 2022, from towardsdatascience.com: https://towardsdatascience.com/visualize-machine-learning-metrics-like-a-pro-b0d5d7815065

Thematic. (n.d.). *Sentiment Analysis: Comprehensive Beginners Guide*. Retrieved from getthematic.com: https://getthematic.com/sentiment-analysis/#:~:text=Sentiment%20analysis%20can%20help%20brands,their%20customers%20feel%20strongly%20about.

Torino, B. (2020, November 29). *GridSearchCV or RandomSearchCV?* Retrieved November 10, 2022, from towardsdatascience: https://towardsdatascience.com/gridsearchcv-or-randomsearchcv-5aa4acf5348c#:~:text=RandomSearchCV%20has%20the%20same%20purpose,we%20are%20testing%20is%20variable.

Van Den Reym, M. (2020, June 4). *7 Advantages of Using Google Colab for Python*. Retrieved November 17, 2022, from pyhon.plainenglish.io: https://python.plainenglish.io/7-advantages-of-using-google-colab-for-python-82ac5166fd4b

Wang, X., Jiang, W., & Luo, Z. (2016). *Combination of Convolutional and Recurrent Neural Network for*. The COLING 2016 Organizing Committee. Retrieved November 10, 2022, from https://aclanthology.org/C16-1229

Wu, Y., Liu, T., Teng, L., Zhang, H., & Xie, C. (2021). *The impact of online review variance of new products on consumer*. Elsevier. Retrieved September 27, 2022, from https://pdf.sciencedirectassets.com/271680/1-s2.0-S0148296321X00112/1-s2.0-S014829632100494X/main.pdf?X-Amz-Security-Token=IQoJb3JpZ2luX2VjEGIaCXVzLWVhc3QtMSJHMEUCIDOi5pw%2FsrS8Ly vbSdLoykwTjUzZMp4sDTFl5mRTHxXhAiEA72qduiMXkC7f5%2B88nWKJFO69fexmeRL72DnnXDee

Xiong, H., & Lee, W. (2011). *A New Over-Sampling Approach: Random-SMOTE.* Springer. Retrieved November 10, 2022, from https://link.springer.com/content/pdf/10.1007/978-3-642-25975-3.pdf

APPENDIX

APPENDIX I – The Consumer Reviews of Amazon Products dataset

Features:

1. User id
2. Name (Name of the product)
    a. 49 unique values
3. Asins (Amazon standard identification number)
4. Brand
    a. Amazon
    b. Amazon Fire
    c. Amazon Echo
    d. Amazon Coco T
    e. Amazon Fire Tv
    f. Amazon digital services
5. Categories (41 categories (some having overlap))
6. Keys
7. Manufacturer
    a. Amazon
    b. Amazon Digital Services Inc
8. Reviews.date (data of placing the review)
9. Reviews.dateAdded (data review is added to dataset)
10. Reviews.dateSeen (date review is examined)
11. Reviews.didPurchase (did reviewed persons purchased the product)
    a. Null
12. Reviews.doRecommend

   a. 94% True
   b. 4% False
   c. 2% Null

13. Reviews.id
    a. Null
14. Reviews.numHelpful
    a. Ranging from 0-814
       i. Mostly only counted once
       ii. Otherwise, zero
15. Reviews.rating (rating ranging from 1 to 5)
    a. Most values between 4-5
    b. Less values between 1-3
16. Reviews.sourceURL (source from which review is gathered
17. Reviews.text (text placed by consumer)
18. Reviews.title (title of the review)
19. Reviews.userCity
    a. Null
20. Review.userProvince
    a. Null
21. Review.username (username of the consumer that placed the review)
    a. 26790 unique values

APPENDIX II – CNN and RNN vs other deep learning algorithms

Classification accuracy scores of CNN combined with RNN deep learning networks on different corpora (Wang, Jiang, & Luo, 2016).

Model	**MR**	**SST1**	**SST2**
CNN-GRU-word2vec	82.28	50.68	89.95
CNN-LSTM-word2vec	81.52	51.50	89.56
CNN-GRU-rand	76.34	48.27	86.64
CNN-LSTM-rand	77.04	49.50	86.80

Group	Model	MR	SST1	SST2
Other	NB (Socher et al., 2013b)	-	41.0	81.8
	SVM (Socher et al., 2013b)	-	40.7	79.4
CNN	1-layer convolution (Kalchbrenner et al., 2014)	-	37.4	77.1
	Deep CNN (Kalchbrenner et al., 2014)	-	48.5	86.8
	Non-static (Kim, 2014)	81.5	48.0	87.2
	Multichannel (Kim, 2014)	81.1	47.4	88.1
Recursive	Basic (Socher et al., 2013b)	-	43.2	82.4
	Matrix-vector (Socher et al., 2013b)	-	44.4	82.9
	Tensor (Socher et al., 2013b)	-	45.7	85.4
	Tree LSTM1 (Zhu et al., 2015)	-	48.0	-
	Tree LSTM2 (Tai et al., 2015)	-	51.0	88.0
	Tree LSTM3 (Le and Zuidema, 2015)	-	49.9	88.0
	Tree bi-LSTM (Li et al., 2015)	0.79	-	-
Recurrent	LSTM (Tai et al., 2015)	-	46.4	84.9
	bi-LSTM (Tai et al., 2015)	-	49.1	87.5
Vector	Word vector avg (Socher et al., 2013b)	-	32.7	80.1
	Paragraph vector (Le and Mikolov, 2014)	-	48.7	87.8
TBCNNs	c-TBCNN (Mou et al., 2015)	-	50.4	86.8
	d-TBCNN (Mou et al., 2015)	-	51.4	87.9
CNN-RNN	CNN-GRU-word2vec	**82.28**	50.68	**89.95**
	CNN-LSTM-word2vec	81.52	**51.50**	89.56
	AVG-GRU-word2vec	81.44	50.36	89.61
	CNN-GRU-rand	76.34	48.27	86.64
	CNN-LSTM-rand	77.04	49.50	86.80

Comparing pre-processing techniques with machine learning models

APPENDIX III – Model performances

Having datasets containing a lot of values can be problematic for the SVM to train on. Implementing stop words and lemmatization techniques seem to reduce the dataset size to make the models run more quickly.

Table 7

Model performance metric with the F1 score for machine learning models based on word negations and stop words removal on Steam reviews (Miao, Jin, Zhang, Chen, & Lai, 2021)

Model	**Raw**	**Negation**	**NLTK**	**sklearn**	**spaCy**	**best**
SVM	88.60	88.34	87.69	87.35	87.22	raw
NB	87.39	87.24	87.69	87.83	87.67	sklearn
RF	**89.65***	**89.57***	**89.75***	**89.62***	**89.54***	NLTK
Best	RF	RF	RF	RF	RF	

Table 8

Model performance metric with the F1 score for machine learning models based on word negations, stop words removal and lemmatization on Steam reviews (Miao, Jin, Zhang, Chen, & Lai, 2021)

	Raw	**NLTK**	**spaCy**	**best**
SVM	88.60	87.72	86.98	raw
NB	87.39	87.80	87.53	sklearn
RF	**89.65***	**89.87***	**89.42***	NLTK
Best	Random Forest	Random Forest	Random Forest	

Table 9

Size of the data compared to the runtime for every model and pre-processing technique (Miao, Jin, Zhang, Chen, & Lai, 2021)

	Raw	**Negate**	**Stop only (NLTK)**	**Stop only(spaCy)**	**Stop only (sklearn)**	**Stop and lemmatize (NLTK)**	**Stop and lemmatize(spaCy)**	**Best F1 Score**
File size	92.8 MB	92.2 MB	61.8MB	55.2 MB	55.4 MB	60.8 MB	51.9 MB	
Training time (SVM-RBF)	262 min	271 min	223 min	254 min	269 min	224 min	258 min	88.60
Training time (Naïve Bayes)	0.085 sec	0.083 sec	0.072 sec	0.084 sec	0.066 sec	0.085 sec	0.067 sec	87.83
Training time (Random Forest)	30 min	36 min	36 min	35 min	33 min	32 min	32 min	89.87

APPENDIX IV – Machine learning models applied on musical subcategory

Table 10

Experimental results on the musical subcategory of the Amazon reviews dataset (Haque, Saber, & Shah, 2018)

Dataset	Classifier	Accuracy 10-fold	Accuracy 5-Fold	Precision	Recall	F1 score
Musical	Linear support Vector machine	**94.02***	89.76	0.9889	0.971	0.98
	Multinomial Naïve Bayes	91.57	89.77	0.98	0.93	0.96
	Stochastic Gradient Descent	92.89	88.264	0.99	0.96	0.98
	Random Forest	93.56	88.51	0.98	0.97	0.975
	Logistic regression	91.34	87.14	0.96	0.95	0.95
	Decision tree	92.45	86.27	0.969	0.96	0.96

APPENDIX V – Pre-processing definitions

Text pre-processing function

```
def text_process(text:str):
    text = str(text)

    text = text.lower()
    text = text.strip()

    text = re.sub(‘ \d+’, ‘ ‘, text)
    text = re.compile(‘<.*?>’).sub(‘’, text)
    text = re.compile(‘[%s]’ % re.escape(string.punctuation)).sub(‘ ‘, text)
    text = re.sub(‘\s+’, ‘ ‘, text)

    text = text.strip()

    return text
```

NLTK stop words coding

```
def nltkstopwords(text:str):

    text = str(text)
    filtered_sentence = []

    # Stop word lists can be adjusted for your problem
    stop_words = nltk_stopwords

    # Tokenize the sentence
    words = word_tokenize(text)
    for w in words:
        if w not in stop_words:
            filtered_sentence.append(w)
    text = “ “.join(filtered_sentence)

    return text
```

NLTK stemming coding

```
#stemming
def nltkstem(text:str):

    text = str(text)
    # Initialise the stemmer
    snow = SnowballStemmer(‘english’)

    stemmed_sentence = []
    # Tokenize the sentence
    words = word_tokenize(text)
    for w in words:
        # Stem the word/token
        stemmed_sentence.append(snow.stem(w))
    text = “ ”.join(stemmed_sentence)

    return text
```

Maxent treebank PoS tagger

```
def get_wordnet_pos(treebank_tag):
        if treebank_tag.startswith(‘J’):
            return wordnet.ADJ
        elif treebank_tag.startswith(‘V’):
            return wordnet.VERB
        elif treebank_tag.startswith(‘N’):
            return wordnet.NOUN
        elif treebank_tag.startswith(‘R’):
            return wordnet.ADV
        else:
            # As default pos in lemmatization is Noun
            return wordnet.NOUN
```

NLTK lemmatization coding

```
def nltklemmatize(text:str):

    text = str(text)

    # Initialise the lemmatizer
    wl = WordNetLemmatizer()

    lemmatized_sentence = []

    # Tokenize the sentence
    words = word_tokenize(text)
    # Get position tags
    word_pos_tags = nltk.pos_tag(words)
    # Map the position tag and lemmatize the word/token
    for idx, tag in enumerate(word_pos_tags):
        lemmatized_sentence.append(wl.lemmatize(tag[0], get_wordnet_pos
(tag[1])))

    lemmatized_text = " ".join(lemmatized_sentence)

    return lemmatized_text
```

SpaCy lemmatization coding

```
def spacylemmatize(text:str):
  text = str(text)
  doc = nlp(text)

  lemmatized_sentence = []

  for token in doc:
    if not token.is_punct:
      lemmatized_sentence.append(token.lemma_)

  lemmatized_text = " ".join(lemmatized_sentence)

  return lemmatized_text
```

SpaCy stop words coding

```
def spacystopwords(text:str):

    text = str(text)
    filtered_sentence = []

    # Stop word lists can be adjusted for your problem
    stop_words = spacy_stopwords

    # Tokenize the sentence
    words = word_tokenize(text)
    for w in words:
        if w not in stop_words:
            filtered_sentence.append(w)
    text = " ".join(filtered_sentence)

    return text
```

APPENDIX VI– Model hyperparameters

Machine learning models

```
model_params = {
   ‘NB’: {
      ‘model’: MultinomialNB(random_state=42),
      ‘params’ : {
         ‘alpha’: [1, 0.1, 0.01, 0.001, 0.0001, 0.00001],
      }
   },

   'KNN': {
      'model': KNeighborsClassifier(random_state=42),
      'params' : {
         'n_neighbors': [5,8, 10, 15]
      }
   },
   'gradient_boost' : {
      'model': GradientBoostingClassifier(n_estimators=100), random_state=42)
      'params': {
         'max_depth': [3, 5, 9, 14]
      }
   },
   'XGBoost': {
      'model':XGBClassifier(random_state=42),
      'params': {
         'eta': [0.001, 0.01, 0.3, 1],
         'max_depth': [3, 6, 9]
      }
   }
'Decision Tree: {
      'model': DecisionTreeClassifier(random_state=42),

      'params': {
         'eta': [0.001, 0.01, 0.3, 1],
         'max_depth': [3, 6, 9]
}
```

Deep learning models

RNN

Model: "sequential_15"

Layer (type)	Output Shape	Param #
embedding_14 (Embedding)	(None, 15, 15)	204675
lstm_27 (LSTM)	(None, 15, 150)	99600
lstm_28 (LSTM)	(None, 150)	180600
dense_20 (Dense)	(None, 100)	15100
dense_21 (Dense)	(None, 2)	202

===
Total params: 500,177
Trainable params: 500,177
Non-trainable params: 0

CNN

Layer (type)	Output Shape	Param #
embedding_36 (Embedding)	(None, 15, 200)	2219000
conv1d_9 (Conv1D)	(None, 11, 128)	128128
global_max_pooling1d_9 (GlobalMaxPooling1D)	(None, 128)	0
dense_70 (Dense)	(None, 10)	1290
dense_71 (Dense)	(None, 2)	22

===
Total params: 2,348,440
Trainable params: 2,348,440
Non-trainable params: 0

Hyperparameters tuned

Table 11

Naïve Bayes hyperparameters tuned vs the mean MCC score and SD per fold for NLTK stop words

N-folds	Alpha learning rate	Mean MCC-score	STD-score
0	1	0.3765	0.0086
1	0.1	0.3761	0.0080
2	0.01	0.3757	0.0080
3	0.001	0.3758	0.0080
4	0.0001	0.3758	0.0080
5	0.00001	0.3758	0.0080

Table 12

KNN hyperparameters tuned vs the mean MCC score and SD per fold for NLTK lemmatization

N-folds	param_n_neighbors	Mean MCC-score	STD-score
0	5	0.2131	0.0478
1	8	0.1517	0.0348
2	10	0.1317	0.0350
3	15	0.1179	0.0256

Table 13

DT hyperparameters tuned vs the mean MCC score and SD per fold for NLTK stemming

N-folds	Param_max_depth	Min_samples_split	Mean MCC-score	STD-score
0	3	2	0.2616	0.0607
1	3	3	0.2616	0.0607
2	3	5	0.2616	0.0607
3	5	2	0.3004	0.0743
4	5	3	0.3004	0.0743
5	5	5	0.3004	0.0743
6	15	2	0.3822	0.1015
7	15	3	0.3824	0.1017
8	15	5	0.3832	0.1016

Table 14

RNN model scores for NLTK stop words removal

	AUC score
NLTK stop	0.8034*
NLTK stemming	0.7975
NLTK lemmatization	0.8006
SpaCy stop words	0.7974
SpaCy lemmatization	0.7972

APPENDIX VII – Additional exploratory data analysis

Most frequent words in the dataset

Words	Times counted
The	43619
And	33523
For	25918
This	17017
Great	10095
With	9257
Have	7901
You	7701
That	7180
But	6628
Tablet	6479
Love	6476
Was	6432
Easy	5932
Very	5688
Use	5428
Can	5379
Not	5377
Amazon	5284
Bought	4991

Handling missing values

0 reviews.rating 34626 non-null float64

1 reviews.text 34625 non-null object
2 y 34626 non-null int64
3 word count 34626 non-null int64
4 nltk_stop 34624 non-null object
5 spacy_stop 34622 non-null object
6 nltk_stem 34624 non-null object
7 nltk_lemma 34624 non-null object
8 spacy_lemma 34622 non-null object

APPENDIX VIII – Machine learning visualizations

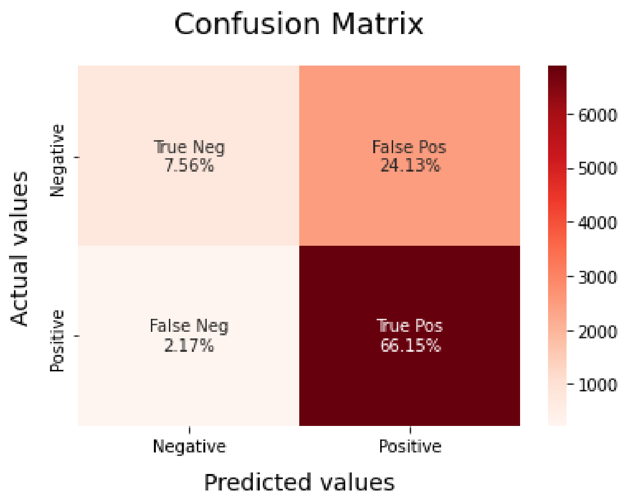
Naïve bayes

NLTK stop words normal

	param_alpha	mean_test_score	std_test_score
0	1	0.335667	0.007557
1	0.1	0.337842	0.008529
2	0.01	0.338538	0.008325
3	0.001	0.338820	0.008272
4	0.0001	0.338820	0.008272
5	0.00001	0.338820	0.008272

	precision	recall	f1-score	support
0	0.76	0.23	0.36	3259
1	0.73	0.97	0.83	7129
accuracy			0.74	10388
macro avg	0.75	0.60	0.60	10388
weighted avg	0.74	0.74	0.68	10388

AUC 0.5994869844983157
MCC 0.31245817705350826



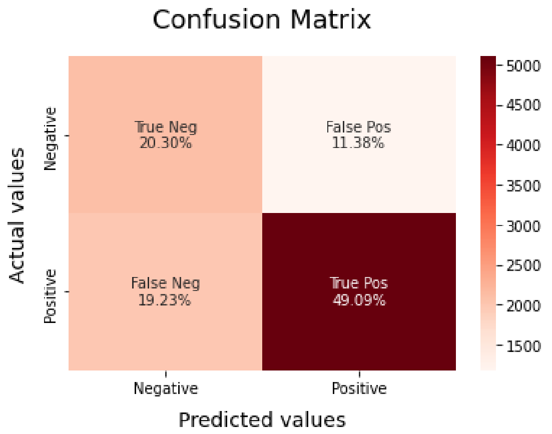
NLTK stop words smote

```
param_alpha  mean_test_score  std_test_score
0           1         0.376527        0.008621
1         0.1         0.376133        0.008034
2        0.01         0.375707        0.008044
3       0.001         0.375766        0.007962
4      0.0001         0.375766        0.007962
5     0.00001         0.375766        0.007962
              precision    recall  f1-score   support

           0       0.51      0.64      0.57      3291
           1       0.81      0.72      0.76      7096

    accuracy                           0.69    10,387
   macro avg       0.66      0.68      0.67    10,387
weighted avg       0.72      0.69      0.70    10,387

AUC  0.6797062476426947
MCC  0.3420109784457811
```



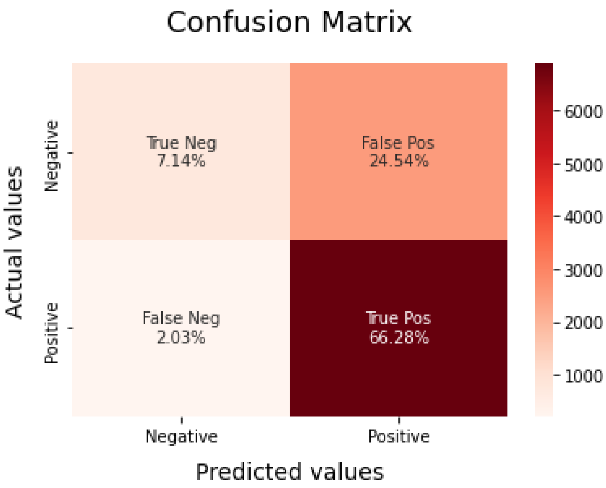
NLTK stemming normal

```
param_alpha  mean_test_score  std_test_score
0           1         0.316378        0.010251
1         0.1         0.318158        0.008052
2        0.01         0.317999        0.007728
3       0.001         0.317999        0.007728
4      0.0001         0.317999        0.007728
5     0.00001         0.317999        0.007728
              precision    recall  f1-score   support

           0       0.78      0.23      0.35      3291
           1       0.73      0.97      0.83      7096

    accuracy                           0.73    10,387
   macro avg       0.75      0.60      0.59    10,387
weighted avg       0.75      0.73      0.68    10,387

AUC  0.5978641614913003
MCC  0.31544963797842623
```

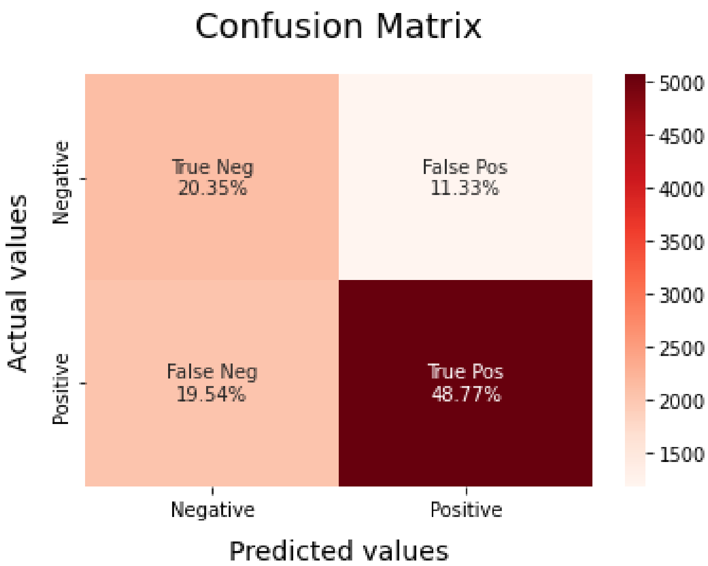


NLTK stemming smote

	param_alpha	mean_test_score	std_test_score
0	1	0.375603	0.002839
1	0.1	0.375490	0.004323
2	0.01	0.375848	0.003847
3	0.001	0.375787	0.003863
4	0.0001	0.375787	0.003863
5	0.00001	0.375787	0.003863

	precision	recall	f1-score	support
0	0.51	0.64	0.57	3291
1	0.81	0.71	0.76	7096
accuracy			0.69	10,387
macro avg	0.66	0.68	0.66	10,387
weighted avg	0.72	0.69	0.70	10,387

AUC 0.6781406415022078
MCC 0.33849886070683205



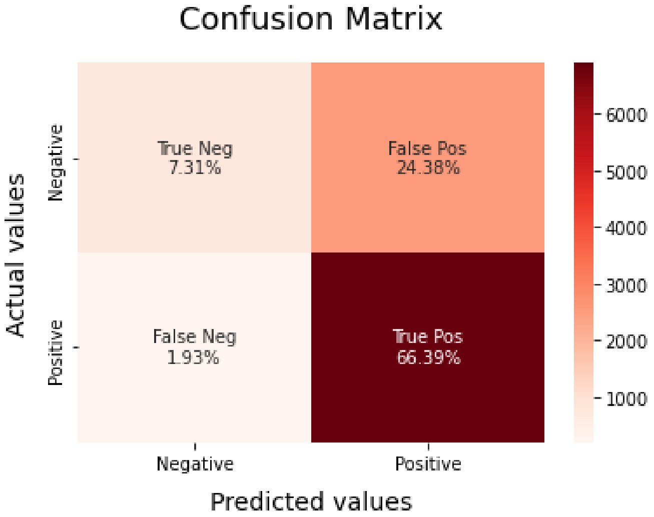
NLTK lemmatization normal

```
param_alpha  mean_test_score  std_test_score
0          1         0.312741        0.011888
1        0.1         0.317470        0.013915
2       0.01         0.317456        0.013468
3      0.001         0.317456        0.013468
4     0.0001         0.317456        0.013468
5    0.00001         0.317456        0.013468
              precision    recall  f1-score   support

           0       0.79      0.23      0.36      3291
           1       0.73      0.97      0.83      7096

    accuracy                           0.74    10,387
   macro avg       0.76      0.60      0.60    10,387
weighted avg       0.75      0.74      0.68    10,387

AUC  0.6012220476260458
MCC  0.3253544627452718
```



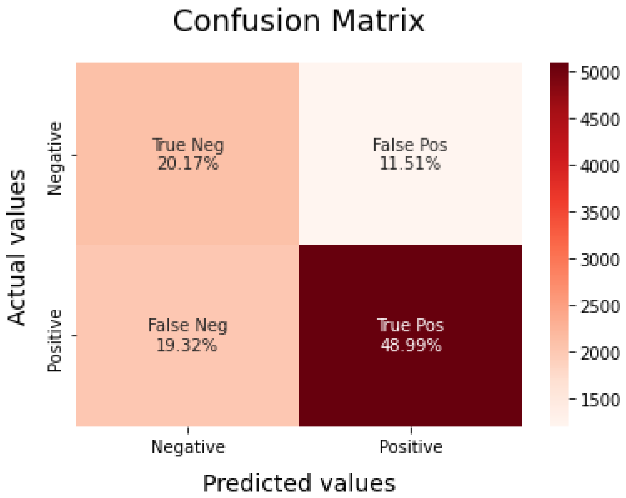
NLTK lemmatizaion smote

```
param_alpha  mean_test_score  std_test_score
0          1         0.371301        0.007877
1        0.1         0.371137        0.008884
2       0.01         0.371313        0.009013
3      0.001         0.371436        0.009162
4     0.0001         0.371436        0.009162
5    0.00001         0.371436        0.009162
              precision    recall  f1-score   support

           0       0.51      0.64      0.57      3291
           1       0.81      0.72      0.76      7096

    accuracy                           0.69    10,387
   macro avg       0.66      0.68      0.66    10,387
weighted avg       0.71      0.69      0.70    10,387

AUC  0.6768746122543221
MCC  0.3366788011008144
```



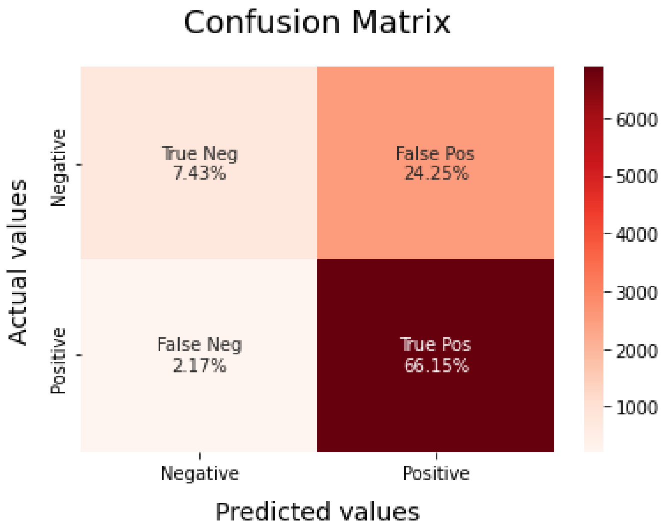
Spacy stop words normal

```
param_alpha  mean_test_score  std_test_score
0          1         0.301973        0.003719
1        0.1         0.305993        0.003220
2       0.01         0.305833        0.003241
3      0.001         0.305983        0.003015
4     0.0001         0.305983        0.003015
5    0.00001         0.305983        0.003015
              precision    recall  f1-score   support

           0       0.77      0.23      0.36      3291
           1       0.73      0.97      0.83      7096

    accuracy                           0.74     10,387
   macro avg       0.75      0.60      0.60     10,387
weighted avg       0.75      0.74      0.68     10,387

AUC  0.6014355753811854
MCC  0.32041339376889816
```

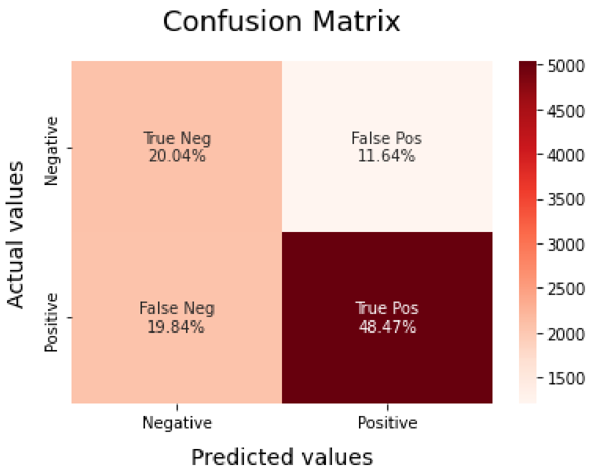


SpaCy stop words SMOTE

	param_alpha	mean_test_score	std_test_score
0	1	0.364577	0.007175
1	0.1	0.363650	0.007087
2	0.01	0.363051	0.007056
3	0.001	0.363055	0.007054
4	0.0001	0.363055	0.007054
5	0.00001	0.363055	0.007054

	precision	recall	f1-score	support
0	0.50	0.63	0.56	3291
1	0.81	0.71	0.75	7096
accuracy			0.69	10,387
macro avg	0.65	0.67	0.66	10,387
weighted avg	0.71	0.69	0.69	10,387

AUC 0.6710945681519447
MCC 0.3251232655171551

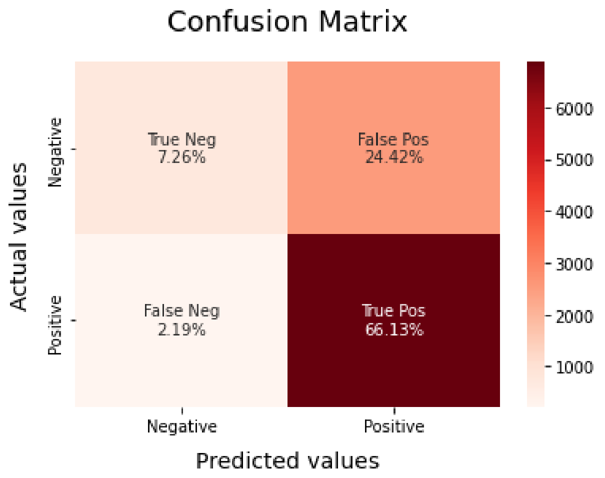


Spacy lemmatization normal

	param_alpha	mean_test_score	std_test_score
0	1	0.307084	0.009421
1	0.1	0.307962	0.008200
2	0.01	0.308379	0.008377
3	0.001	0.308521	0.008420
4	0.0001	0.308521	0.008420
5	0.00001	0.308521	0.008420

	precision	recall	f1-score	support
0	0.77	0.23	0.35	3291
1	0.73	0.97	0.83	7096
accuracy			0.73	10,387
macro avg	0.75	0.60	0.59	10,387
weighted avg	0.74	0.73	0.68	10,387

AUC 0.5985599198319218
MCC 0.31359136166393603

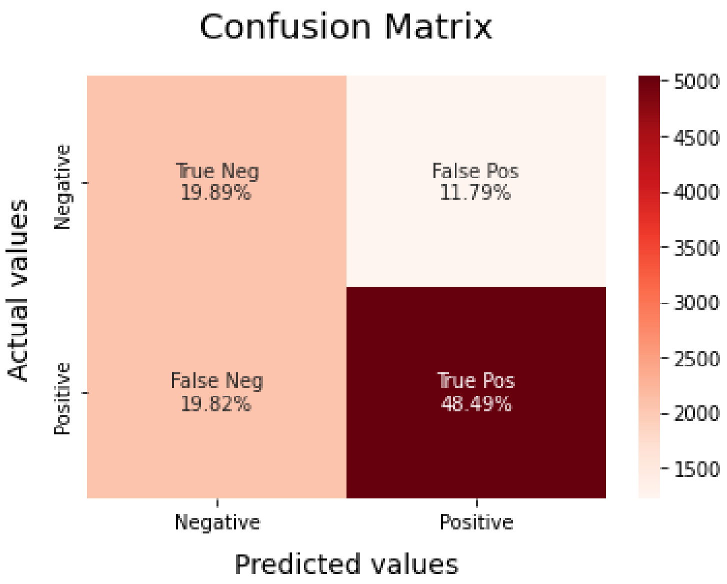


Spacy lemmatization SMOTE

	param_alpha	mean_test_score	std_test_score
0	1	0.355484	0.010170
1	0.1	0.355993	0.009096
2	0.01	0.356290	0.009008
3	0.001	0.356288	0.008865
4	0.0001	0.356288	0.008865
5	0.00001	0.356288	0.008865

	precision	recall	f1-score	support
0	0.50	0.63	0.56	3291
1	0.80	0.71	0.75	7096
accuracy			0.68	10,387
macro avg	0.65	0.67	0.66	10,387
weighted avg	0.71	0.68	0.69	10,387

AUC 0.6688046205410746
MCC 0.32100852209483466



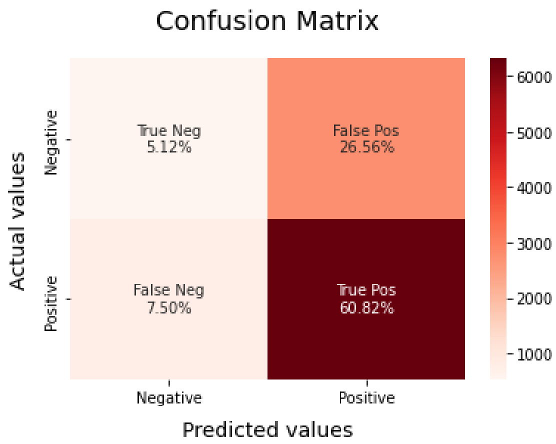
KNN

KNN stop words normal

	param_n_neighbors	mean_test_score	std_test_score
0	5	0.065776	0.010346
1	8	0.065652	0.008378
2	10	0.065208	0.010504
3	15	0.042052	0.013754

	precision	recall	f1-score	support
0	0.41	0.16	0.23	3291
1	0.70	0.89	0.78	7096
accuracy			0.66	10,387
macro avg	0.55	0.53	0.51	10,387
weighted avg	0.60	0.66	0.61	10,387

AUC 0.5259364175879213
MCC 0.0726711680441926



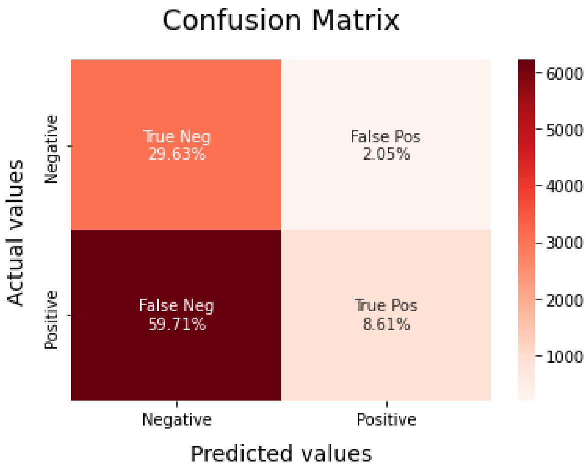
KNN stop words SMOTE

```
param_n_neighbors  mean_test_score  std_test_score
0                  5         0.185931        0.032463
1                  8         0.130931        0.020446
2                 10         0.113089        0.017913
3                 15         0.084904        0.005940
              precision    recall  f1-score   support

           0       0.33      0.94      0.49      3291
           1       0.81      0.13      0.22      7096

    accuracy                           0.38     10,387
   macro avg       0.57      0.53      0.35     10,387
weighted avg       0.66      0.38      0.30     10,387

AUC  0.530632251122514
MCC  0.09237019391843233
```

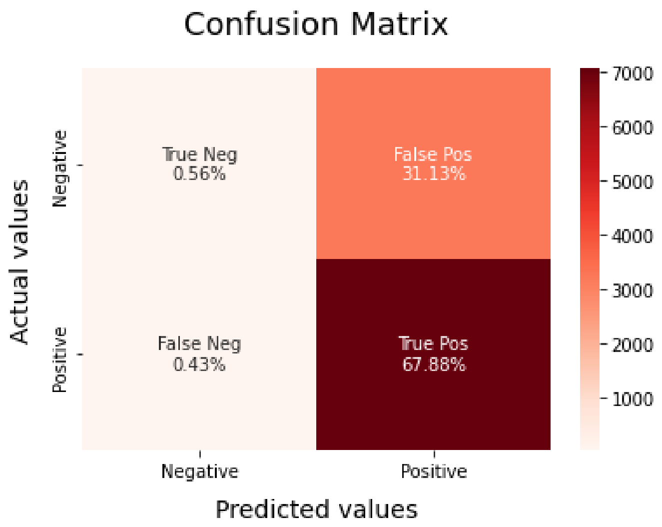


KNN stemming normal

	param_n_neighbors	mean_test_score	std_test_score
0	5	0.061731	0.018977
1	8	0.062068	0.012768
2	10	0.065438	0.008834
3	15	0.083440	0.031129

	precision	recall	f1-score	support
0	0.56	0.02	0.03	3291
1	0.69	0.99	0.81	7096
accuracy			0.68	10,387
macro avg	0.62	0.51	0.42	10,387
weighted avg	0.65	0.68	0.57	10,387

```
AUC  0.505641110822211
MCC  0.052974360859102094
```



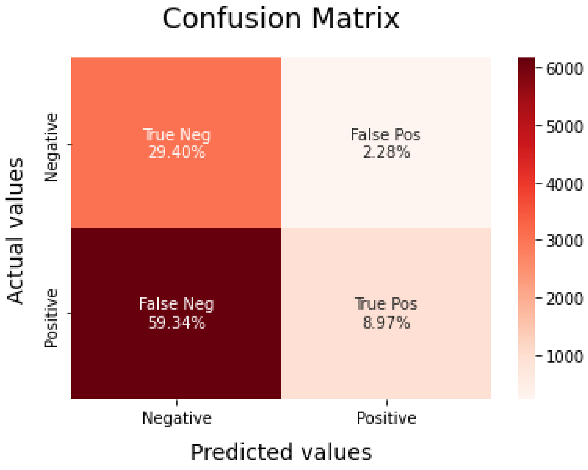
KNN stemming SMOTE

```
param_n_neighbors  mean_test_score  std_test_score
0                 5         0.184080        0.036417
1                 8         0.120034        0.020725
2                10         0.103830        0.014379
3                15         0.087022        0.017363
              precision    recall  f1-score   support

           0       0.33      0.93      0.49      3291
           1       0.80      0.13      0.23      7096

    accuracy                           0.38     10,387
   macro avg       0.56      0.53      0.36     10,387
weighted avg       0.65      0.38      0.31     10,387

AUC  0.5296635078347323
MCC  0.08733687324087752
```



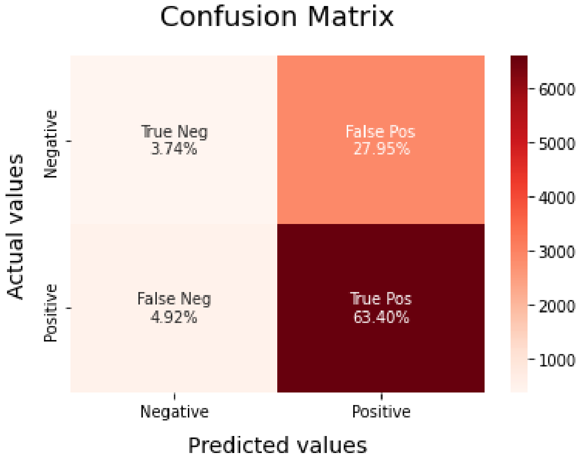
KNN lemmatization normal

```
param_n_neighbors  mean_test_score  std_test_score
0                  5         0.061232        0.013156
1                  8         0.066499        0.012679
2                 10         0.065710        0.020639
3                 15         0.048629        0.016178
              precision    recall  f1-score   support

           0       0.43      0.12      0.19      3291
           1       0.69      0.93      0.79      7096

    accuracy                           0.67     10,387
   macro avg       0.56      0.52      0.49     10,387
weighted avg       0.61      0.67      0.60     10,387

AUC  0.5229424471509707
MCC  0.07592298880418193
```

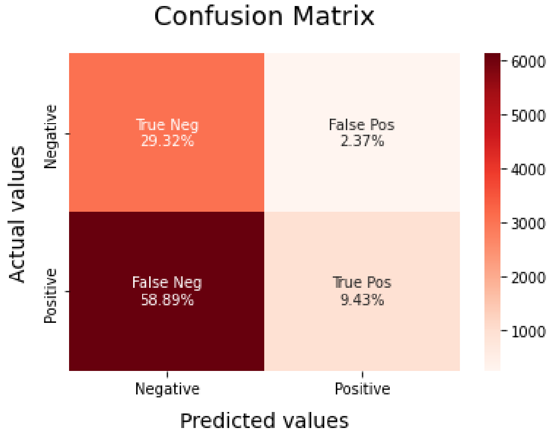


KNN lemmatization SMOTE

	param_n_neighbors	mean_test_score	std_test_score
0	5	0.185682	0.037832
1	8	0.128851	0.024402
2	10	0.105428	0.020713
3	15	0.082890	0.018629

	precision	recall	f1-score	support
0	0.33	0.93	0.49	3291
1	0.80	0.14	0.24	7096
accuracy			0.39	10,387
macro avg	0.57	0.53	0.36	10,387
weighted avg	0.65	0.39	0.32	10,387

AUC 0.5316078672077892
MCC 0.09118696143994234

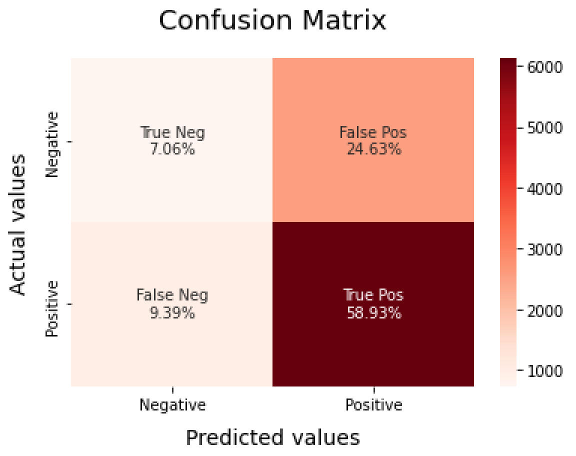


KNN spaCy stop words normal

	param_n_neighbors	mean_test_score	std_test_score
0	5	0.093216	0.007428
1	8	0.081169	0.005772
2	10	0.074001	0.011408
3	15	0.056697	0.007083

	precision	recall	f1-score	support
0	0.43	0.22	0.29	3291
1	0.71	0.86	0.78	7096
accuracy			0.66	10,387
macro avg	0.57	0.54	0.53	10,387
weighted avg	0.62	0.66	0.62	10,387

AUC 0.5426636505148645
MCC 0.10709765852733638

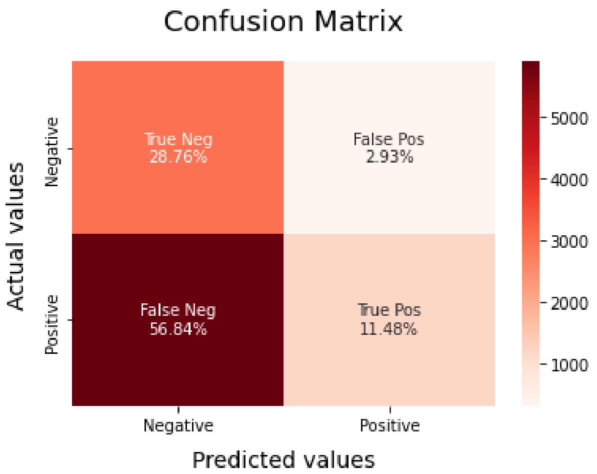


KNN spaCy stop words SMOTE

	param_n_neighbors	mean_test_score	std_test_score
0	5	0.206393	0.050543
1	8	0.146032	0.032158
2	10	0.129039	0.028514
3	15	0.108250	0.024177

	precision	recall	f1-score	support
0	0.34	0.91	0.49	3291
1	0.80	0.17	0.28	7096
accuracy			0.40	10,387
macro avg	0.57	0.54	0.38	10,387
weighted avg	0.65	0.40	0.34	10,387

AUC 0.5378044114024891
MCC 0.10018486539538907

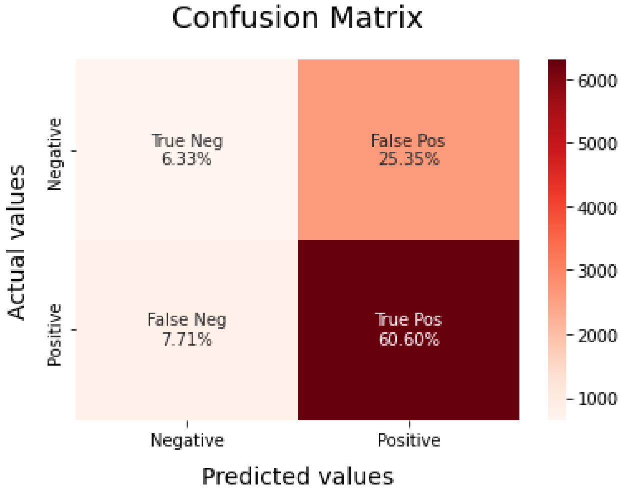


KNN spaCy lemmatization stop words

	param_n_neighbors	mean_test_score	std_test_score
0	5	0.077956	0.009655
1	8	0.074444	0.013513
2	10	0.075703	0.008293
3	15	0.063688	0.011176

	precision	recall	f1-score	support
0	0.45	0.20	0.28	3291
1	0.71	0.89	0.79	7096
accuracy			0.67	10,387
macro avg	0.58	0.54	0.53	10,387
weighted avg	0.62	0.67	0.62	10,387

AUC 0.5435293660720005
MCC 0.11656773179913808



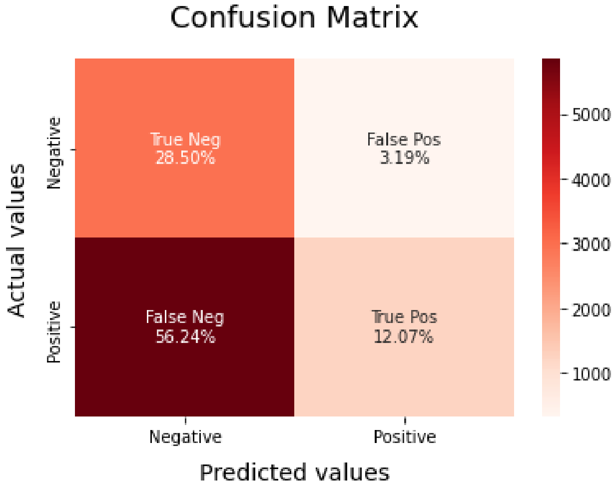
KNN spaCy lemmatization SMOTE

```
param_n_neighbors  mean_test_score  std_test_score
0                 5         0.213117        0.047802
1                 8         0.151664        0.034773
2                10         0.131706        0.034950
3                15         0.117885        0.025570
              precision    recall  f1-score   support

           0       0.34      0.90      0.49      3291
           1       0.79      0.18      0.29      7096

    accuracy                           0.41     10,387
   macro avg       0.56      0.54      0.39     10,387
weighted avg       0.65      0.41      0.35     10,387

AUC  0.5380709731744222
MCC  0.09851206056991095
```



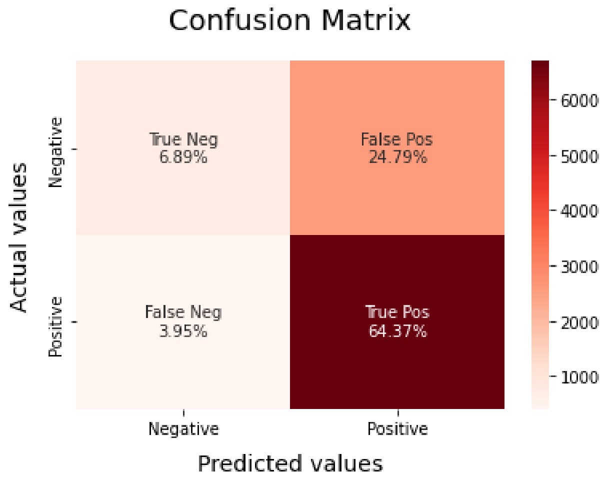
Decision Tree

NLTK stop words normal

	param_max_depth	param_min_samples_split	mean_test_score	std_test_score
0	3	2	0.124558	0.013701
1	3	3	0.124558	0.013701
2	3	5	0.124558	0.013701
3	5	2	0.173146	0.012626
4	5	3	0.173146	0.012626
5	5	5	0.173146	0.012626
6	15	2	0.211991	0.008040
7	15	3	0.211991	0.008040
8	15	5	0.210949	0.010949

	precision	recall	f1-score	support
0	0.64	0.22	0.32	3291
1	0.72	0.94	0.82	7096
accuracy			0.71	10,387
macro avg	0.68	0.58	0.57	10,387
weighted avg	0.69	0.71	0.66	10,387

AUC 0.5798920101523851
MCC 0.2391145212962987

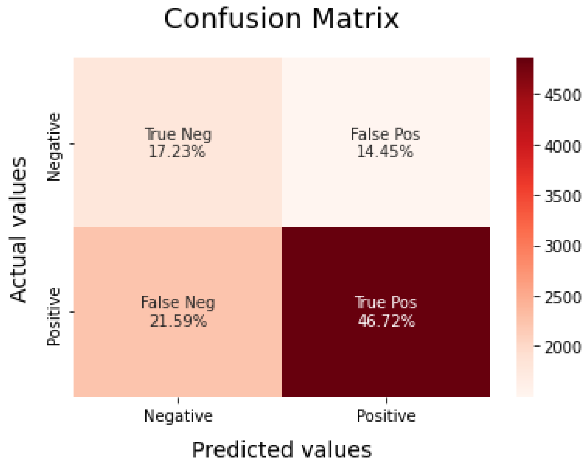


NLTK stop words SMOTE

	param_max_depth	param_min_samples_split	mean_test_score	std_test_score
0	3	2	0.262138	0.068549
1	3	3	0.262138	0.068549
2	3	5	0.262138	0.068549
3	5	2	0.297470	0.082422
4	5	3	0.297470	0.082422
5	5	5	0.297470	0.082422
6	15	2	0.377156	0.127644
7	15	3	0.377341	0.127402
8	15	5	0.376601	0.125373

	precision	recall	f1-score	support
0	0.44	0.54	0.49	3291
1	0.76	0.68	0.72	7096
accuracy			0.64	10,387
macro avg	0.60	0.61	0.61	10,387
weighted avg	0.66	0.64	0.65	10,387

AUC 0.6139070265083586
MCC 0.2174771028812177

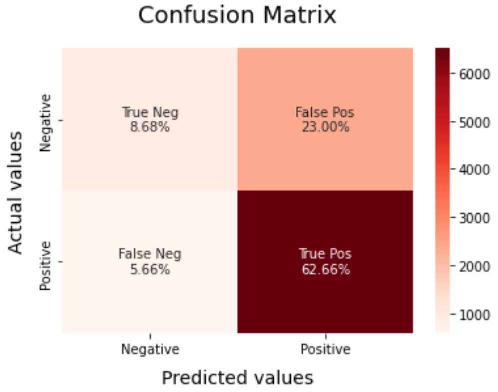


NLTK stemming normal

	param_max_depth	param_min_samples_split	mean_test_score	std_test_score
0	3	2	0.122604	0.009216
1	3	3	0.122604	0.009216
2	3	5	0.122604	0.009216
3	5	2	0.190759	0.013319
4	5	3	0.190759	0.013319
5	5	5	0.190759	0.013319
6	15	2	0.217954	0.013032
7	15	3	0.217954	0.013032
8	15	5	0.215546	0.010697

	precision	recall	f1-score	support
0	0.61	0.27	0.38	3291
1	0.73	0.92	0.81	7096
accuracy			0.71	10,387
macro avg	0.67	0.60	0.60	10,387
weighted avg	0.69	0.71	0.68	10,387

AUC 0.5956086206890645
MCC 0.2537948841332824

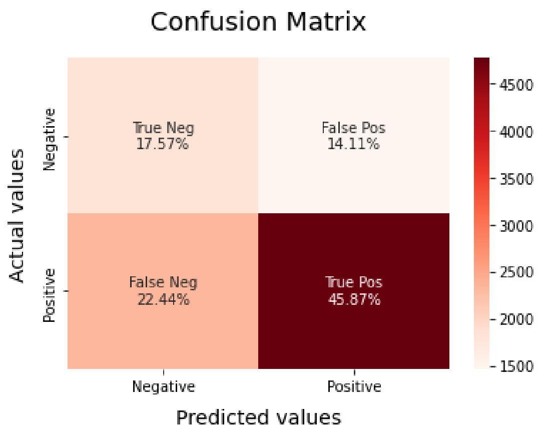


NLTK stemming SMOTE

	param_max_depth	param_min_samples_split	mean_test_score	std_test_score
0	3	2	0.261595	0.060694
1	3	3	0.261595	0.060694
2	3	5	0.261595	0.060694
3	5	2	0.300363	0.074321
4	5	3	0.300363	0.074321
5	5	5	0.300363	0.074321
6	15	2	0.382163	0.101470
7	15	3	0.382386	0.101701
8	15	5	0.383150	0.101567

	precision	recall	f1-score	support
0	0.44	0.55	0.49	3291
1	0.76	0.67	0.72	7096
accuracy			0.63	10,387
macro avg	0.60	0.61	0.60	10,387
weighted avg	0.66	0.63	0.64	10,387

AUC 0.6130238827357725
MCC 0.21466157810369293

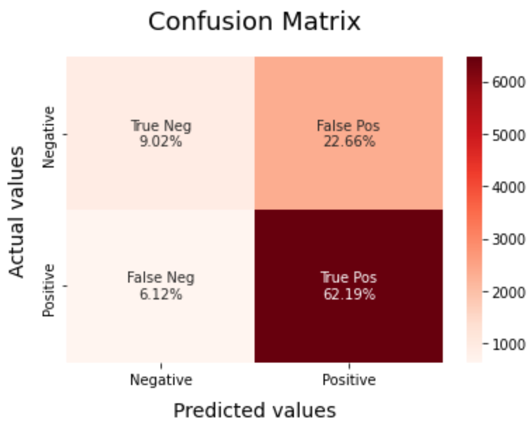


NLTK lemmatization normal

	param_max_depth	param_min_samples_split	mean_test_score	std_test_score
0	3	2	0.121280	0.005747
1	3	3	0.121280	0.005747
2	3	5	0.121280	0.005747
3	5	2	0.185230	0.014550
4	5	3	0.185230	0.014550
5	5	5	0.185230	0.014550
6	15	2	0.225190	0.006114
7	15	3	0.225790	0.006398
8	15	5	0.223228	0.008405

	precision	recall	f1-score	support
0	0.60	0.28	0.39	3291
1	0.73	0.91	0.81	7096
accuracy			0.71	10,387
macro avg	0.66	0.60	0.60	10,387
weighted avg	0.69	0.71	0.68	10,387

AUC 0.5975439662062192
MCC 0.25319216721142845

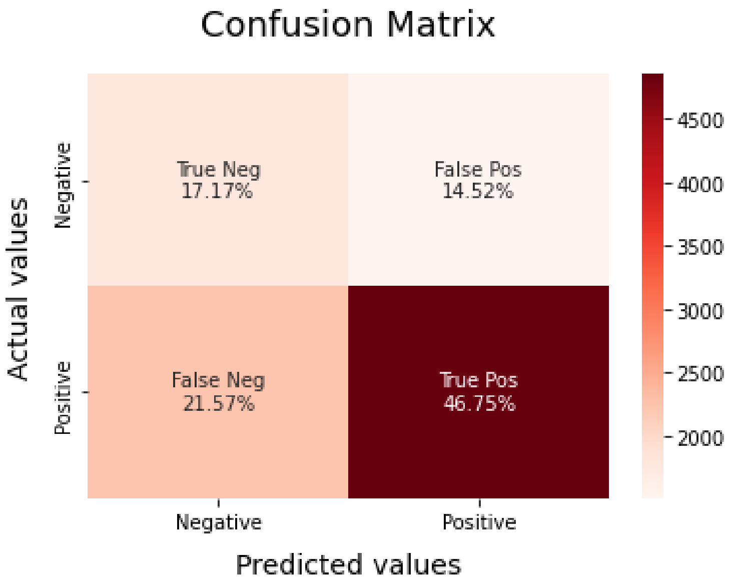


NLTK lemmatization SMOTE

	param_max_depth	param_min_samples_split	mean_test_score	std_test_score
0	3	2	0.272692	0.068259
1	3	3	0.272692	0.068259
2	3	5	0.272692	0.068259
3	5	2	0.311250	0.076833
4	5	3	0.311250	0.076833
5	5	5	0.311250	0.076833
6	15	2	0.371350	0.101029
7	15	3	0.370882	0.101007
8	15	5	0.371084	0.100805

	precision	recall	f1-score	support
0	0.44	0.54	0.49	3291
1	0.76	0.68	0.72	7096
accuracy			0.64	10,387
macro avg	0.60	0.61	0.60	10,387
weighted avg	0.66	0.64	0.65	10,387

AUC 0.6130549066721204
MCC 0.21594843164084157

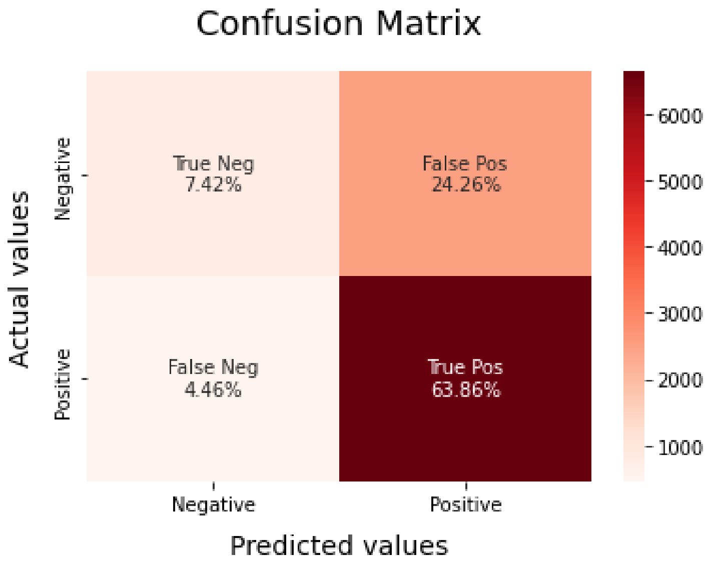


NLTK spaCy stop words normal

	param_max_depth	param_min_samples_split	mean_test_score	std_test_score
0	3	2	0.117356	0.008895
1	3	3	0.117356	0.008895
2	3	5	0.117356	0.008895
3	5	2	0.173208	0.008851
4	5	3	0.173208	0.008851
5	5	5	0.173208	0.008851
6	15	2	0.224756	0.009203
7	15	3	0.224756	0.009203
8	15	5	0.219952	0.008928

	precision	recall	f1-score	support
0	0.62	0.23	0.34	3291
1	0.72	0.93	0.82	7096
accuracy			0.71	10,387
macro avg	0.67	0.58	0.58	10,387
weighted avg	0.69	0.71	0.67	10,387

AUC 0.5845136346025185
MCC 0.2430458492759667

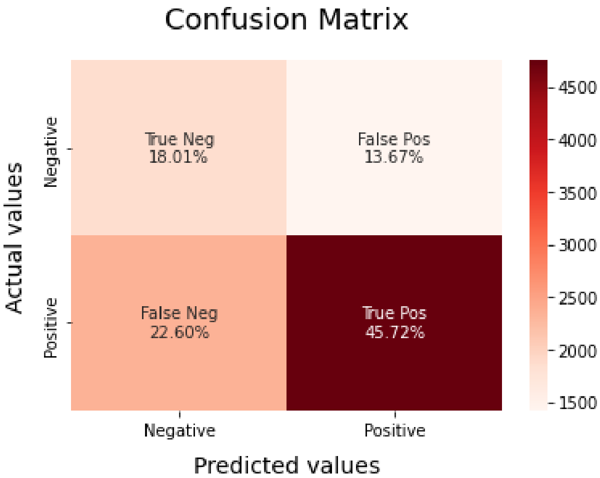


NLTK spaCy stop words SMOTE

	param_max_depth	param_min_samples_split	mean_test_score	std_test_score
0	3	2	0.246346	0.056993
1	3	3	0.246346	0.056993
2	3	5	0.246346	0.056993
3	5	2	0.281677	0.082824
4	5	3	0.281677	0.082824
5	5	5	0.281677	0.082824
6	15	2	0.373288	0.105944
7	15	3	0.372777	0.105997
8	15	5	0.373895	0.106094

	precision	recall	f1-score	support
0	0.44	0.57	0.50	3291
1	0.77	0.67	0.72	7096
accuracy			0.64	10,387
macro avg	0.61	0.62	0.61	10,387
weighted avg	0.67	0.64	0.65	10,387

AUC 0.6188852442365277
MCC 0.22525167116052078

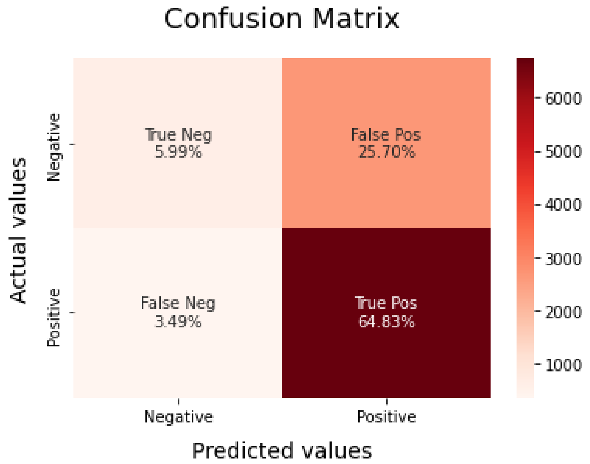


NLTK spaCy lemmatization normal

	param_max_depth	param_min_samples_split	mean_test_score	std_test_score
0	3	2	0.163965	0.011298
1	3	3	0.163965	0.011298
2	3	5	0.163965	0.011298
3	5	2	0.195722	0.015375
4	5	3	0.195722	0.015375
5	5	5	0.195722	0.015375
6	15	2	0.230876	0.013275
7	15	3	0.230876	0.013275
8	15	5	0.228458	0.011206

	precision	recall	f1-score	support
0	0.63	0.19	0.29	3291
1	0.72	0.95	0.82	7096
accuracy			0.71	10,387
macro avg	0.67	0.57	0.55	10,387
weighted avg	0.69	0.71	0.65	10,387

AUC 0.5689928238573514
MCC 0.2192168302541571

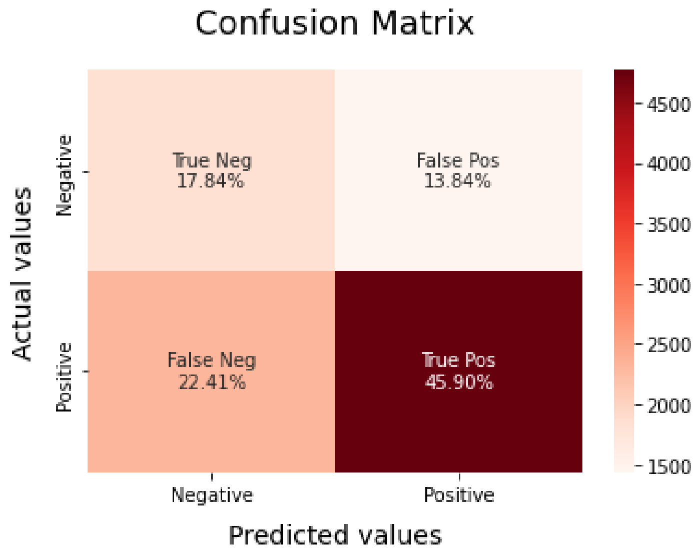


NLTK spaCy lemmatization SMOTE

	param_max_depth	param_min_samples_split	mean_test_score	std_test_score
0	3	2	0.274037	0.070046
1	3	3	0.274037	0.070046
2	3	5	0.274037	0.070046
3	5	2	0.300487	0.085236
4	5	3	0.300487	0.085236
5	5	5	0.300487	0.085235
6	15	2	0.356579	0.105144
7	15	3	0.356573	0.105151
8	15	5	0.354953	0.104132

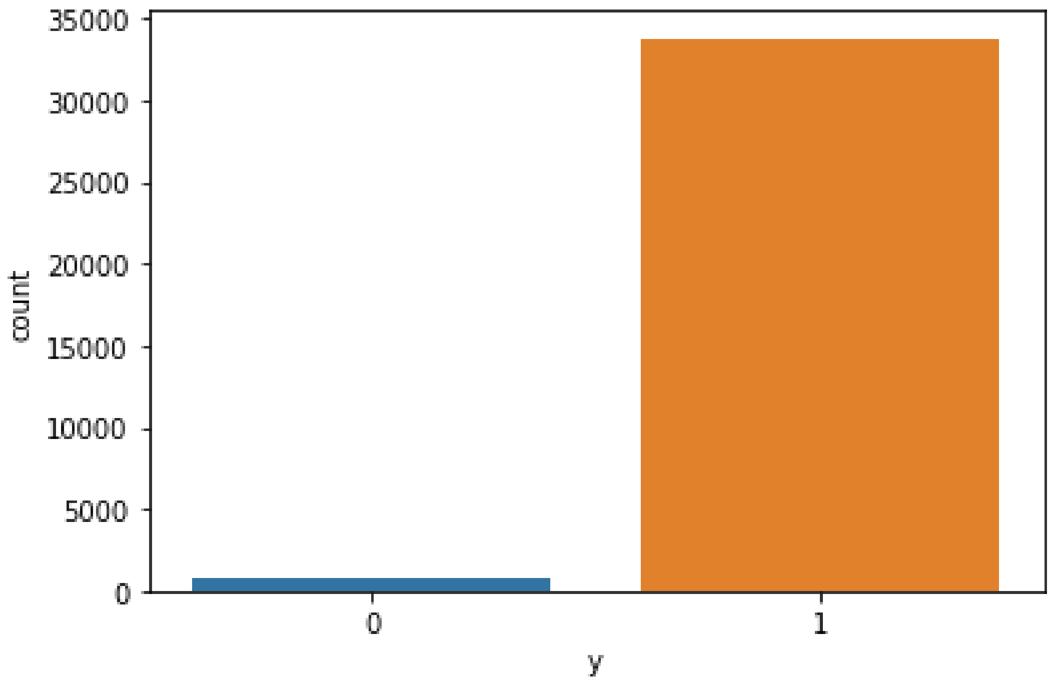
	precision	recall	f1-score	support
0	0.44	0.56	0.50	3291
1	0.77	0.67	0.72	7096
accuracy			0.64	10,387
macro avg	0.61	0.62	0.61	10,387
weighted avg	0.67	0.64	0.65	10,387

AUC 0.6174892955643778
MCC 0.2229220718274372

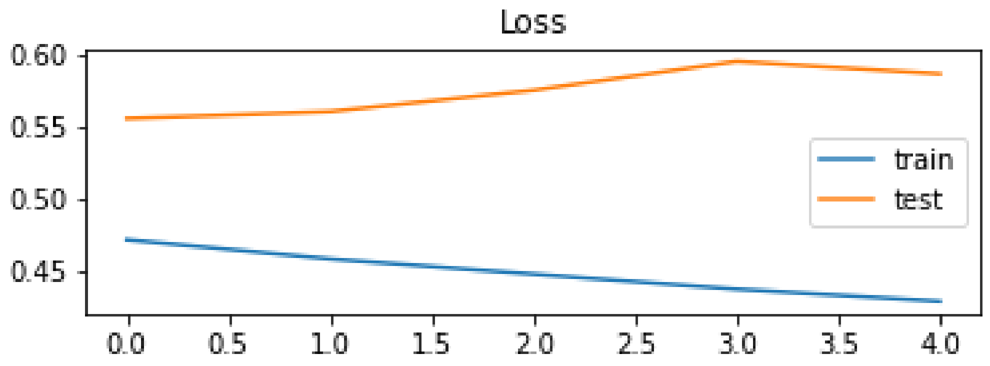


APPENDIX IX – Additional discussion

Original imbalance



RNN Loss



CNN loss