universität wien

BACHELORARBEIT / BACHELOR'S THESIS

Titel der Bachelorarbeit / Title of the Bachelor's Thesis

„A Benchmark Study using Various Algorithms for Review Rating Prediction and Sentiment Analysis“

verfasst von / submitted by

Sebastian Radu Herman

angestrebter akademischer Grad / in partial fulfilment of the requirements for the degree of

Bachelor of Science

Wien, 2019 / Vienna, 2019

Studienkennzahl lt. Studienblatt / degree programme code as it appears on the student record sheet:	A UA 033 526
Studienrichtung lt. Studienblatt / degree programme as it appears on the student record sheet:	Bachelorstudium Wirtschaftsinformatik UG2002
Betreut von / Supervisor:	Dipl.-Ing.Nour Jnoub

Contents

Abstract

Online reviews are the most common form of feedback regarding products or services nowadays. Most people use online reviews when deciding whether to purchase a product or a service, based on what other people had to say about the product or the service and what their overall experience was like. In this paper, we benchmark various approaches for predicting a restaurants review rating and overall sentiment just by analyzing the text content. This is achieved by using multiple classifiers combined with multiple text pre-processing and feature engineering methods. This could be very helpful for platforms such as Yelp and Amazon which have a big focus on reviews. Such businesses could use our algorithms in order to improve their methods of checking if reviews are rated according to text and it could also help when recommending a product or a service to other customers based on the overall sentiment that reviews for a specific product or service have. We achieve this by predicting a negative or positive sentiment for the sentiment analysis approach. For predicting a product rating, we try to predict whether it has received a good, neutral or negative review.

1 Motivation

1.1 Introduction

Nowadays most businesses also offer, besides the classic walk in stores, an online platform in the form of a website/web shop which customers can access by using just their web browser. Since almost everyone has access to a computer or a smartphone, many people make use of the internet to search for products or services they wish to purchase.

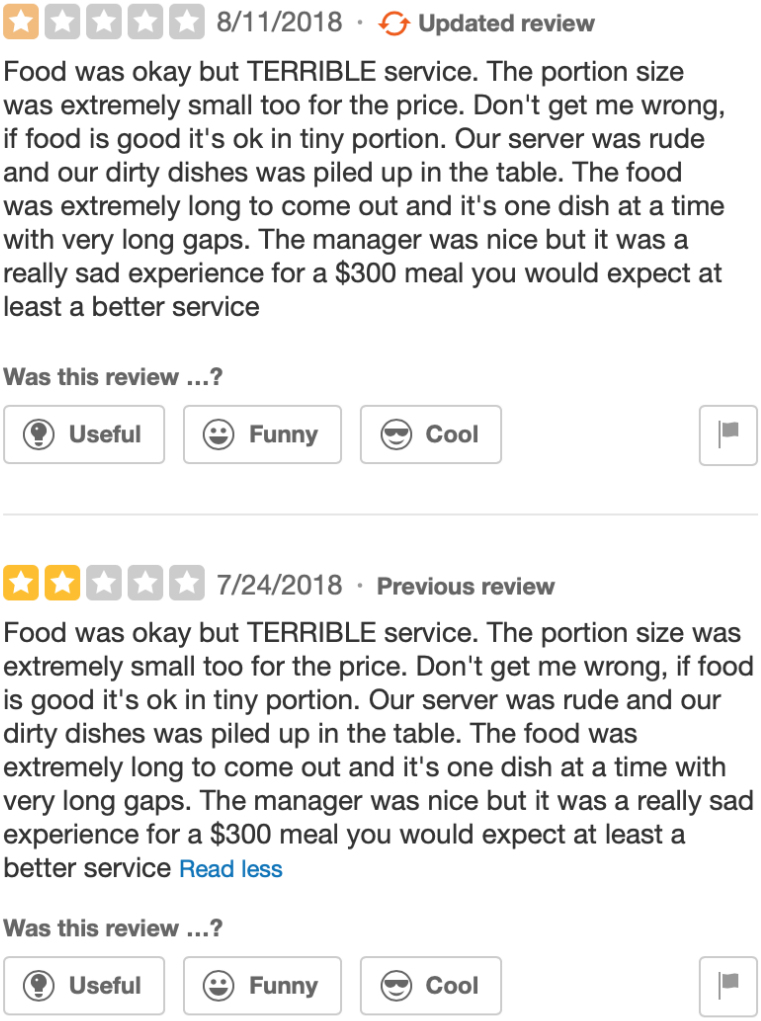
In our opinion, one of the main reasons why online platforms have become so popular is because almost all the merchants/service providers make use of customer written reviews. Customer written reviews provide customers desiring to purchase a product with a general insight about the quality of the product or the experience someone else had with the said product. Reviews have become so popular since the internet has become so widely available resulting in the creation of online businesses and platforms with the sole purpose of reviewing other businesses or products. These new platforms usually focus on reviewing restaurants, hotels and in some cases even services like accountants, psychologists, doctors etc. Notable example of services focused on reviewing other businesses are: Yelp, TripAdvisor, Foursquare, Trustpilot etc.

The majority of websites which offer users the possibility to write a review are based on a system comprised of a number based rating of the business, usually a number between 1 and 5 followed by a textual description of the rating. Reviews are helpful not only for customers, but also for businesses. Customer written reviews allow the service provider to get an insight about how well the business performs, how satisfied customers are, what other aspects might need improvement and so on. In case of products, the merchant can see how well the product is being received by customers, how satisfied customers

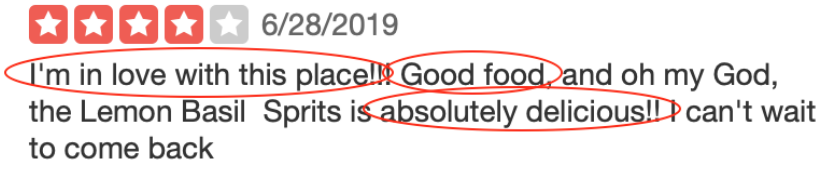
are by using the product etc. Some websites, for example Yelp and Amazon, also offer the user the possibility to rate other users reviews as helpful, funny or useful. The logic behind these votes is that the additional helpful votes of other users are supposed to improve the credibility of the original review. In most of the cases, reviews end up influencing a lot of people, positive reviews usually influencing people to buy a product, a service or visit a restaurant or place. On the other hand negative reviews usually influence customers to a negative extent, by having them not purchase products or services, which in turn influences the economical aspect of the business and in some cases, may even lead to bankruptcy. Needless to say, reviews have gotten to be a very important part of the business community, many owners using reviews as the number one feedback source when it comes to the success of their business.

1.2 Problem

The major problem with reviews lies within the fact that they are usually inconsistent, being it due to length, lack of language skills or just due to the fact that every person has a different style of expressing him or herself. This can lead to inconsistencies between the awarded rating and the textual description of the rating which in turn could have a big impact on the business itself because most businesses are shown to a user in order according to their average rating. Other cases which we have noticed, especially on Yelp's online platform, are reviews on which the rating is updated, but the content is not as seen in this screenshot. As we can clearly see the image below, the user didn't even bother to change the body of the review, they just changed the rating.



In the second screenshot we can see that the user used words and phrases such as "in love", "good food", "absolutely delicious", "can't wait to come back", yet they only rated the restaurant with 4 out of 5 stars without specifying any downside.



Normally this would lead us to think it is a 5-star review, but here we

can clearly see that the author of the review doesn't agree with us. Websites generally used automated algorithms to recommend reviews or businesses to other users, but do not perform that well in terms of comparing the rating with the textual description as to decide if a review is worth showing to another member or not.

This became our motivation. Trying to solve the discrepancy issue between the number based rating and textual content to help businesses and customers alike when deciding upon purchasing or visiting a place. We have tried to come up with a machine learning solution which is able to distinguish the general sentiment or rating of a review in order to be able to detect such inconsistencies and take further action if necessary.

1.3 Goals and Methods

Our goal is to try and close this discrepancy gap between the number-based rating and its textual content. For this we have chosen to approach this issue in two ways:

- predicting the overall sentiment, whether it may be positive or negative
- predicting a rating of good, neutral or bad just based on textual analysis

Others have also tried to achieve this, however using different techniques which will be discussed in the "Related Work" part of this paper. We however, tried to incorporate many of the ideas used in the other papers by using multiple classifiers, 8 to be exact, specifically tuned for this task. The ones that performed the best are: Multilayer Perceptron Classifier, Logistic Regression and SVM. In addition, we also tried various pre-processing techniques and feature engineering methods and benchmarked all combinations in order to see which of them provides the better result. In order to validate our results, we decided to measure the accuracy of our experiment with following parameters which are all used among all other papers: f1-score, precision-score, recall-score and support-score. In this paper, we will try to present and analyze all the results obtained using the 8 classifiers and text pre-processing techniques. In our opinion, we achieved better results than what others have achieved at this point in time, although not by a huge margin, but still be large enough in order to be able to call it improvement.

2 Related Work

We have found a number of papers dealing with the problem of trying to rate reviews based just on textual analysis or just trying to detect the predominant sentiment of the review. However, we have tried to incorporate techniques used by many authors and have them side by side in order to see which one performs best.

The authors of [30] have chosen to try to predict the overall sentiment of a review by using 4 classifiers: Naive Bayes, Linear SVM, Logistic Regression and SGD. They have chosen to remove all the 3 star rated reviews and keep only the reviews rated from 1 to 2 stars which they have marked as negative and those ranging from 4 to 5 which have been marked as positive. The authors pre-process the textual data, followed by two approaches when extracting the features. For the first approach the authors of [30] have chosen to implement a custom dictionary created from the training dataset and for the second approach they have chosen to perform a lexical analysis of the text-based reviews. The best accuracy they have managed to achieve using their first proposed method is 94.4%. When analyzing the results of the second approach the authors have noticed a 14% drop in accuracy compared to the first approach.

The authors of [36] have chosen to take a different approach, by trying to predict all ratings ranged from 1 to 5 just by analyzing the textual features. They have chosen to use the Naive Bayes, Perceptron and Multiclass SVM classifiers. They have used two approaches when trying to predict the rating. The first one makes use of the "Bing Liu Opinion Lexicon"[16], while their second approach makes use of the Bag-Of-Words model which they have created themselves using the dataset used for training and testing. The authors [36] come to the conclusion that trying to predict all 5 ratings is quite a challenge due to the fact that ratings the textual description of 1 and 2 starred ratings, 4 and 5 starred ratings respectively can't be that easily distinguished even by seasoned human readers.

A different approach can be seen in the paper written by [21], where the authors chose to use only common words, eliminating all rare words with a frequency lower than 1e-4. On top of that they also tried to take into consideration what other reviews the author of a review has also written and also take that rating into consideration when making a prediction. After that the authors tried to predict the sentiment and rating. For predicting the sentiment they proceeded to transform the ratings to binary values, by that we mean label ratings ranging from 1 to 3 as 0 or negative and those ranging from 4 to 5 as 1 or positive. The authors of [21] proceeded to use following classifiers: Logistic Regression, Naive Bayes and SVM for binary classification. For the second approach they chose to use Multinomial Logistic Regression, Naive Bayes and Support Vector Regression. Using these classifiers the authors tried to predict all the ratings from 1 to 5. They have used the RMSE metric to measure their success and have achieved a RMSE of 0.17 when predicting all ratings and a RMSE of 0.0178 when predicting the polarity of the review.

The authors of [9] also try to treat the prediction as a multilabel classification. They use the number of the rating as the label they are trying to predict. The authors combine 4 feature engineering methods, unigram, bigram, trigram and Latent Semantic Indexing with four different classifiers: Logistic Regression, Naive Bayes, Perceptron and Linear SVM. For measuring performance, the authors [9] use accuracy and RMSE. Out of all classifiers they found that Logistic Regression is able to predict all 5 labels with the greatest accuracy of 64% followed by the linear SVM which was able to achieve 63% accuracy. They

also offer possible solutions which may increase the accuracy, some of them used by us, namely using Part-Of-Speech tags in text pre-processing.

The authors of [19] provide yet another approach to rating prediction. They try to find effective clues from reviews and predict users' social rating by fusing user sentiment similarity, interpersonal sentiment influence and item reputation similarity into a unified matrix. On top of that they also describe how they managed to build a new relationship named interpersonal sentiment influence between a user and their friends. The authors also described techniques for text pre-processing which we have used in our project, for example they recommend removing stop words or words that have no meaning. This means that the method proposed by the authors is vastly more complicated, but they have managed to get decent results. As for measuring performance, the authors decided upon using RMSE. However, due to the nature of their project a direct comparison between the results we have achieved and the results they have achieved is not possible.

[10] chose to use another approach when trying to predict the rating of Yelp reviews. They chose to transform reviews rated as 2 to a rating of 1 due to the lexical similarities. They also converted reviews rated with 4 stars to 5 stars out of the same reason. They ended up with 3 ratings: 1 for bad, 2 for average and 3 for good. For the preparation of the dataset they chose to use 3 techniques: feature reduction, stemming and the bi-gram technique. For the prediction, the authors of [10] chose to use two machine learning algorithms, SVM and KNN. However, they chose to benchmark the results differently, by training the algorithms both on the pre-processed data and the non-processed data. They have concluded that the accuracy provided by the SVM is greater in both cases than the one provided by the KNN algorithm. However, the authors fail to provide a classification report as most other scientific papers do, instead opting for displaying the results using confusion matrixes while textually describing the accuracy, again not mentioning the metrics used to measure accuracy. They state the accuracy of the linear SVM to be highest, ranking in at 76.4%.

Another paper which tries to solve the issue of removing user bias from the numerical rating of a review is the paper written by [14]. The authors try to predict all 5 ratings just based on the textual information. For the reviews they chose to use 1000 out of the 4243 restaurants they have found in their version of the Yelp Open Dataset. They decided to split the data into 90% training and 10% testing while using 10-fold cross validation to validate the accuracy obtained by the classifiers. The authors of [14] chose to use 3 different methods of pre-processing the text. The first one called "Baseline" consists of building a bag of words model using the top K words. They have experimented with values ranging from 50 to 1000. The second approach called "Feature Engineering I" also consists of building a Bag-Of-Words model but makes use of "Part of Speech Tagging" in order to try and improve the accuracy. For the third approach they tried to analyze only the adjectives, hoping it would yield better results. As for machine learning algorithms, the authors chose the following classifiers: Linear Regression, Support Vector Regression, Support Vector Regression With Normalized Features and Decision Tree Regression. The accur-

acy of their benchmark was measured using the RMSE metric, the best results being achieved by the Linear Regression classifier in all 3 text pre-processing scenarios. The best results being produced while using Feature Engineering I and II methods for text pre-processing.

[34] chose to tackle the task of predicting a review's rating in yet another way. The author decided to treat reviews rated with 4 or 5 stars as positive and reviews rated with 1, 2 or 3 stars as negative. As the author himself describes, his project aims to detect the overall sentiment rather than to predict the rating itself. The author chose to use a dataset consisting of 10.000 reviews split into 60% for training and 40% for testing. For measuring accuracy, the precision metric and recall metric have been used. The author chose several text pre-processing and feature engineering methods: Basic Unigram, Basic Bigram, Unigram + Removing Common Symbols, Unigram + Removing Stopwords and Common Symbols, Stemming + Removing Stopwords and Common Symbols and finally Character-based 5-gram. As for the supervised learning algorithms, the author chose to use the Perceptron Learning Algorithm, the Naive Bayes Classifier and the SVM Classifier. The best result was produced by the Perceptron Algorithm and the text pre-processing and feature engineering method which performed the best according to the author is Stemming + Removing Stopwords and Common Symbols.

The authors of [28] chose a completely different approach when trying to tackle the issue of rating prediction. They propose a bag-of-opinions model, which according to them, exploits domain independent corpora of opinions, but is applied for learning prediction on domain-specific reviews [28]. The authors summarize their concept as follows [28]:

1. Introducing a bag-of-opinions model for capturing the influence of n-grams, structured by using root words, modifiers and negators.

2. Developing a constrained ridge regression method for learning scores of opinions from domain-independent corpora of rated reviews.

3. Transferring the regression model to newly given domain-dependent applications by deriving a set of statistics over opinion scores in documents and using them as features together with unigrams

4. Evaluating the model on Amazon reviews from different product categories.

The authors state that the method proposed by them is very fast, the training of over 350.000 reviews taking only 11 seconds to complete. For measuring accuracy, the authors of [28] have chosen to use the MSE, or Mean Squared Error. We can observe that the method proposed by them gives a MSE score of 0.884 for the book category, 0.928 for the DVD category and 0.627 for products in the music category.

The authors of [12] try a completely different approach when trying to solve the issue of rating prediction based purely on text. They propose a system called NARRE (Neural Attentional Regression model with Review-level Explanations)

which not only predicts the rating of a review but can also rate it's usefulness simultaneously. For their research, they use both the Amazon Review Dataset and the Yelp Open Review Dataset. The model proposed by the authors of [12] consists of two parallel neural networks, one for user modelling and one for item modelling. On top of these layers, the authors also put a prediction layer in order for the two layers for user modelling and item modelling to be able to interact with each other. For the training they propose to make use of user data, item data and the text review. For the first stage, in order to process the reviews, the authors make use of a Convolutional Neural Network specially programmed for processing text. To evaluate the performance of their system, the authors use the RMSE metric and compare it to five other approaches: PMF, NMF, SVD++, HFT and DeepCoNN (another method proposed by [37]). The NARRE method seems to outperformed all other methods to which the results have been compared to. On average the performance increase measured using the RMSE metric is 0.1%.

[27] is another paper and master thesis which tries to tackle the issue of sentiment analysis performed on Yelp reviews. The author uses a dataset consisting of 32.637 reviews split into training and testing with a ratio of 70/30. Due to the fact that a rating ranging from 1 to 5 is very difficult to predict, the author chooses to binarize the data for the first approach, trying to predict only if a certain review is positive or negative. For the classification of the reviews, the author chooses to use 4 classifiers: Logistic Regression, Naive-Bayes, Random Forest and Support Vector Machine. For measuring the accuracy of the predictions, the F1 Score metric is being used. For feature engineering the author chooses to use the Bag-Of-Words model using the Count Vectorizer and the TFIDF Vectorizer, both implemented using the scikit-learn framework. As we can see from the results table, the best performing classifier is the Logistic Regression classifier, which we also have found to be most accurate at predicting rating and sentiments in our Project. For the second part of the project, the author of [27] tries to predict all the ratings ranging from 1 to 5. This however proves to be a much more challenging task than only trying to predict the polarity of a review. The best score for the second method is 0.6% with an RMSE of 0.71%.

3 Design & Algorithms

3.1 Methodology

Due to the fact that so many papers and authors have tried to predict either the rating or the sentiment, we decided to try to do them both using the same methods and classifiers in order to benchmark and see which of the combinations of feature extraction and various classifiers perform best and in what condition. The two approaches we decided to take are:

- predict the polarity of a review i.e. if a review is positive or negative

- predict the rating of a review based solely on the textual content

For our first approach, we decided that it would be best to drop all the reviews that have been rated with 3 stars, because these reviews can be either biased towards positivity or negativity, which would make training machine learning algorithms more difficult. Next, we opted to merge the ratings rated with 1 or 2 stars into a category represented by the label 0 which stands for negative and the reviews rated with 4 or 5 stars into a category represented by the label 1 which stands for positive.

For our second approach, we decided it would be best if we only stick to 3 classes as this would make guessing whether a review is positive, negative or neutral less confusing for the classifiers we have used. We have decided to follow this path due to the fact that reviews rated with either 2 or 4 stars are very close in content to those rated 1 or 5 respectively. Usually reviews which have been rated with 2 stars still are still regarded as bad by the majority of people interested in reading reviews, while those rated with 4 stars tend to be seen as good.

3.2 Data

This project makes use of reviews extracted from a database containing Yelp Restaurant Reviews, curated and offered to us by Prof. Bing Liu from the University of Illinois at Chicago. [24] Many of the reviews in the database are either fake or could not be checked whether they have been written by real human reviewers or bots. In order to tackle this challenge, we decided to only pick reviews which have been rated by 7 or more people as being useful reviews. We presume that reviews which have already been rated by other reviewers as useful to be representative for the rating they stand for. Due to limited hardware capabilities, we have decided to use only 5000 reviews, 1000 for each rating category. We can see the distribution of class labels in the table below.

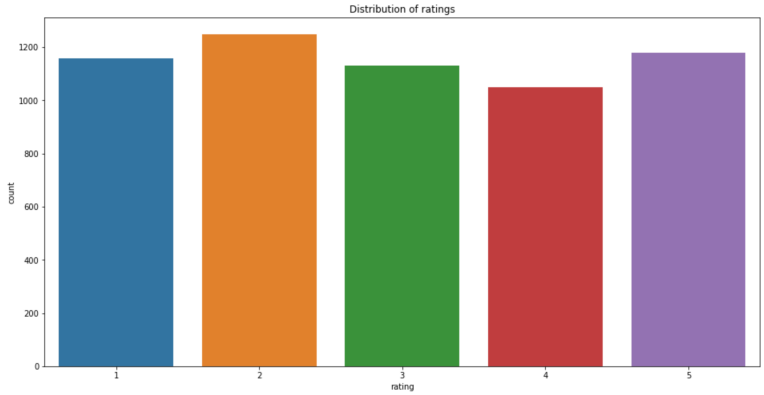


Figure 1: Distribution of ratings

3.2.1 Data exploration

Before moving on with classifying, we decided to also do a textual analysis of the reviews we have selected for this project.

In order to see how different reviews are, we decided to study the word clouds of both reviews rated with 1 star and reviews rated with 5 stars. Word clouds are graphics generated using the Python programming language which display the predominant words in a block of text, in an array or a string. We decided to study these particular word clouds do to the fact that reviews rated with 1 star should contain words which are opposite to the reviews rated with 5 stars. However, as you can see in the two word cloud graphics, the distribution of predominant words is very similar.

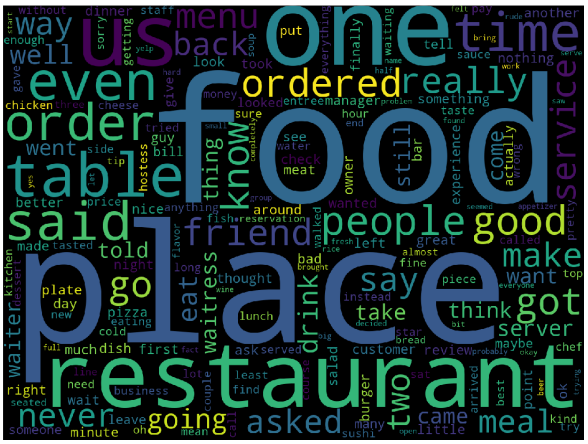


Figure 2: Worldcloud 1 star reviews

Figure 3: Worldcloud 5 star reviews

3.2.2 Text Pre-Processing & Feature Extraction

After selecting the data using SQL and exporting it to a CSV format, we loaded it into Python. As we can see, the exported dataset contains 6 columns. The first column named "reviewContent" contains the text review. The following 3 columns, "usefulCount", "coolCount" and "funnyCount" contain the number of votes from each category (useful, cool, funny), votes which have been by other users to the review itself in order to mark it as such. The next column, "rating", contains the actual labels we will use for classification. Finally, the last column, "restaurantID" contains the ID of the restaurant for which the review was written. This column was needed in order to select only restaurant reviews from the Yelp Restaurant Database provided by Prof. Bing Liu from the University of Illinois at Chicago [16], as we have found that the restaurant database also contains reviews from other types of businesses within the HoReCa (Hotel, Restaurant, Catering) industry.

	reviewContent	usefulCount	coolCount	funnyCount	rating	restaurantID
0	"'Check, Please." The bartender was unable to ...	18	11	25	1	VZHyAmdFDreQqL0BT-zdoA
1	"2 stars for disappointing food, one star for ...	14	10	7	3	tFcmrGLZNEymSnijoTPmqw
2	"A Divine Dialogue" God: "Britton. Times up. T...	14	16	23	5	INvIaBFnAvGxzTXFWHzGvA
3	"A Place To Go When You Have Time" It was a mi...	13	10	10	5	FySId5SjNhkrtPA5qktdxg
4	"A Thaiphoon of Flavor" My First: I never enli...	27	22	28	5	RgeMUiZncTs-VSHQLm0wNg

Figure 4: Pandas Dataframe

In order to construct our feature vectors, we decided it would be best for a benchmarking project to choose as many text pre-processing methods as possible. We decided to try and use some of the ideas that have been implemented in previous projects in order to be able to compare their performances side by side. For text cleaning the authors of [30], [36] propose removing all punctuation and extra spaces while at the same time lowercasing all words. The authors of [13] choose to only strip HTML tags leaving the punctuation and white spaces behind.

For our project, we decided to take a slightly different approach, namely construct a baseline method for cleaning the text, on top of which we will further add other methods like stemming and lemmatizing in order to construct the final feature vectors. Before diving proceeding with the pre-processing, we noticed that the reviews extracted from the database contain a special character, which shows up on the screen as \xa0 which is actually a non-breaking space in the Latin 1 encoding (ISO 8859-1). Due to the fact that we opted to use the UTF-8 encoding, we decided removing this character is of utmost importance in order not to temper our feature vectors. Next, we decided it would be best if we removed the links and URL's from reviews, as we have found that many reviews also contain links, links which have not necessarily inserted by the user, but links which were auto generated by the website when typing the name of the restaurant for example, we assume for tagging purposes. Next, we decided

that it would be ideal to lowercase all words before moving on with further techniques to pre-process the text.

A lot of reviews are written in informal style, so it was quite common to stumble upon reviews full of abbreviations such as: "we'll" instead of "we will", "you're" instead of "you are". We decided it would be best to get rid of these contractions and solve them into two separate words, as to make bigrams and trigrams relevant further up the road. After removing the contractions, we decided the next ideal step would be to remove all punctuation. Next, we decided to remove numbers, as most reviews contain numbers which are irrelevant when trying to predict a rating or a sentiment. Moreover, we decided to remove the numbers as to not bias the classifier towards positivity or negativity when numbers are encountered. Finally, we have decided to remove one lettered words, or simply put, remove single characters as they cannot have any meaning. In the end we removed all spare white spaces in the text. We decided to use this text pre-processing method as a base for our next methods.

For feature extraction papers like [30] propose removing stop words, removing words that are less than 3 characters. For marking negations, the author of [30] suggests appending a "!" between the negation and the next punctuation symbol and finally applying a stemming algorithm, the Porter Algorithm. The author of [36] suggests appending "not_" to every word between the negation and the following punctuation symbol in order to handle negations. For our paper, we decided to use multiple approaches when trying to build our feature vectors, so as a result we ended up with 6 text pre-processing methods, 5 of them being built on top of the baseline method.

For our second method we decided to use the baseline method we discussed earlier and also remove stop words. Stop words are words like "the", "an", "is", so basically words which do not have a big impact when trying to predict the rating or the polarity of a review.

For our third method, we decided to use the second method and add the Porter Algorithm on top of it. As described in [35], the porter algorithm is quite a simple algorithm for stemming used in the English language. According to [35], the Porter Algorithm "consists of ca. 60 suffixes, two recording rules and a single type of context-sensitive rule to determine whether a suffix should be removed". As an example, the author of [35] offers two scenarios. Firstly, using the Porter Algorithm if we try to stem the word FULNESS, the algorithm will output FUL, while at the second iteration the result would be null, but only if the resulting stem has a non-zero measure. Another example would be HOPEFULNESS, which at the first iteration returns HOPEFUL, while at the second instead it returns HOPE, due to the fact that the returned stem does not have a non-zero measure.

Our fourth approach makes use of the Snowball Stemming Algorithm, which is fitted onto the second method, namely Baseline + Stop Words Removal. The Snowball Algorithm is a stemming algorithm designed to support multiple languages. However, our reviews are all in the English language and we use the Snowball Stemming Algorithm as a more aggressive version of the Porter Stemming Algorithm.

Our fifth approach makes use of a lemmatizer. The lemmatizer we chose to use is the WordNetLemmatizer built using the NLTK framework. The main difference between a lemmatizer and a stemmer is that a lemmatizer takes into consideration what word it is processing. Unlike a stemmer, a lemmatizer doesn't chop word endings off, it instead checks what type of word it is processing and then returns the root word of the processed word from a dictionary called lemma.

Finally, for our sixth approach, we decided to use the Lancaster Stemming Algorithm built into the NLTK framework. From our research, the Lancaster Algorithm is the most aggressive stemming algorithm used for our project. The stemming is so aggressive, even human readers might have problems understanding a piece of text after it has been stemmed. In the example below, we can see how different a piece of text is after it has been stemmed using the Lancaster Algorithm.

Original Review	'Most amazing meal ever. PERFECT wine pairings. Impeccable service. 14 courses beyond imagination. I left with a buzz - and mesmerized by what is possible. We flew in to eat here; the meal was more expensive than the flights.'
Baseline Processing	'most amazing meal ever perfect wine pairings impeccable service courses beyond imagination left with buzz and mesmerized by what is possible we flew in to eat here the meal was more expensive than the flights'
Lancaster Algorithm	'amaz meal ev perfect win pair impecc serv cours beyond imagin left buzz mesm poss flew eat meal expend flight'

Table 1: Baseline Processing and Stemming Example

3.3 Feature Engineering

Most other papers trying to solve this issue usually make use of different language models, usually resorting to using unigrams and bigrams as we can see in the paper written by [34]. In order to engineer the features that have to be used, most other authors decided upon using both the Bag-Of-Words model using the Count Vectorizer and the TFIDF Vectorizer as seen in all the papers we have discussed in the related works section.

For our approach we have decided to use both the Count Vectorizer and the TFIDF Vectorizer. So, what is a Count Vectorizer and a TFIDF Vectorizer and how do they work? The author of [4] proposes a simple description on how these two vectorizers work.

3.3.1 Count Vectorizer vs TFIDF

According to the author of [4], the Count Vectorizer is a counter for term frequency, counting the words and building a sparse matrix of documents multiplied by words. The TFIDF, which stands for Term Frequency-Inverse Document Frequency is a statistical measure used to measure how important a word is within a document. The TFIDF is measured by multiplying two values, namely Term Frequency with Inverse Document Frequency.

$$TF(x) = \frac{Number\ of\ times\ x\ appears\ in\ a\ document}{Total\ number\ of\ words\ in\ a\ document} \tag{1}$$

$$IDF(x) = log_e(\frac{Total\ number\ of\ documents}{Number\ of\ documents\ with\ x\ in\ them}) \tag{2}$$

$$TF - IDF_{score} = TF * IDF \tag{3}$$

3.3.2 N-Grams

Due to the fact that our project aims to be a benchmark of multiple combinations of feature vectors and machine learning algorithms, we decided to use the same combinations for both approaches, both for analyzing the polarity of a review and for predicting the overall rating.

For this, we have chosen to use following models: unigram, unigram + bigram, unigram + bigram + trigram on both Vectorizers. Basically a unigram tokenizes a sentences into separate words, and these words are analyzed separately one from the other. When applying the bigram technique, the words are split into combinations of two adjacent words. The same goes for trigrams. However, the approach we took by using combinations of them, for example in the case of unigram + bigram, the classifier will take into consideration both words individually and sequences of adjacent words. The table below shows how N-Grams work when applying them to a simple sentence.

Original Sentence	"This place was delicious!! A coworker introduced it to me and I fell in love. I ordered 3 taco's for lunch and it was less than $5.00."
Unigram	'00','and', 'coworker', 'delicious', 'fell', 'for','in', 'introduced', 'it', 'less', 'love', 'lunch', 'me', 'ordered', 'place', 'taco', 'than', 'this', 'to', 'was'
Bigram	'and fell', 'and it', 'coworker introduced', 'delicious coworker', 'fell in', 'for lunch', 'in love', 'introduced it', 'it to', 'it was', 'less than', 'love ordered', 'lunch and', 'me and', 'ordered taco', 'place was', 'taco for', 'than 00', 'this place', 'to me', 'was delicious', 'was less'
Trigram	'and fell in', 'and it was', 'coworker introduced it', 'delicious coworker introduced', 'fell in love', 'for lunch and', 'in love ordered', 'introduced it to', 'it to me', 'it was less', 'less than 00', 'love ordered taco', 'lunch and it', 'me and fell', 'ordered taco for', 'place was delicious', 'taco for lunch', 'this place was', 'to me and', 'was delicious coworker', 'was less than'

Table 2: Comparison of N-Grams

When looking into the table, we can clearly see why text needs to be preprocessed before being vectorized. Numbers for example, especially floating-point numbers which have a separator do not work well when they are being split into N-Grams.

As for the number of features used, we have decided to use all the features present into the text due to the limited number of reviews we have chosen to make our training and predictions on. In order to increase the accuracy we decided to use only those words which appear in a minimum of 5 documents, as to not have words which do not have a contextual meaning, like for example personal names.

3.4 Rating Prediction & Sentiment Analysis

The two approaches we have decided to take are rating prediction and sentiment analysis.

3.4.1 Rating Prediction

Due to the similarities in ratings which are closely related one to another, by that we mean for example reviews rated with 1 star and reviews rated with 2 stars, we have chosen to drop reviews rated as 2 or 4 stars in order for the classifier to perform best when trying to detect whether a review was rated as 1 (bad), 3 (neutral), 5 (positive). We consider this approach to be best when trying to predict a rating using supervised learning algorithms. A possible rating classification which takes into account all the ratings would only be feasible if being performed by a complex neural network, as supervised learning algorithms do not generally possess the complexity needed to process all the subtle similarities that were discussed earlier.

3.4.2 Sentiment Analysis

Sentiment analysis generally implies detecting the polarity of a review. Out of this consideration and due to the fact that all other papers we have cited in the related work section of our paper only use two labels when predicting the polarity of a review, we also decided for this approach. Thus, in our sentiment analysis approach we drop all the reviews which have been rated with 3 stars. We dropped these reviews as 3 stars is generally considered to have a neutral sentiment and we would like that our classifier to perform best when predicting a positive or a negative sentiment. The second step we took was binarizing the ratings. In order to achieve this, we decided to merge reviews which have been rated with 2 stars into a category we labeled as 0 which stands for bad and the reviews rated with 4 or 5 stars into another category which we have labeled using the number 1 which we consider to stand for positive.

3.5 Dataset splitting

For this project, we have decided to split the data into 80% training and 20% testing.

3.6 Machine Learning Algorithms

As previously mentioned, we have decided to use 8 classifiers for our two approaches. Due to the fact that consistency is desired when benchmarking algorithms, we decided to use the same classifiers tuned in a similar fashion both for rating prediction and sentiment analysis. We will try and briefly explain how the used classifiers work in order to give the reader a better understanding of our project.

3.6.1 Multinomial Naive Bayes

Multinomial Naive Bayes is an algorithm which has been used for years for text classification as it is considered to be one of the best performing text classifiers. However, in our research we have found that at least 3 other classifiers outperform the Multinomial Naive Bayes Classifier. The authors of [18] offer a very good in depth yet relatively short explanation about how the Multinomial Naive Bayes Classifier works. We will try to summarize their work as best as possible. The authors of [18] chose to explain it as follows: The set of classes will be noted with the letter C. N will stand for the size of the vocabulary used. The Multinomial Naive Bayes Algorithm then assignes a test document t_i to the class that has the highest probability $Pr(c|t_i)$, which is a result of the following formula:

$$Pr(c|t_i) = \frac{Pr(c)Pr(t_i|c)}{Pr(t_i)}, \quad c \in C \tag{4}$$

$Pr(c)$ can be estimated by dividing the number of documents belonging to class c by the total number of documents. $Pr(t_i|c)$ is the probability of obtaining a document like t_i in class c and is calculated as the following:

$$Pr(t_i|c) = (\sum_n f_{ni}!) \prod_n \frac{Pr(w_n|c)^{f_{ni}}}{f_{ni}!} \tag{5}$$

where f_{ni} is the count of word n in the test document t_i and $Pr(w_n|c)$ the probability of word n given class c. For more information regarding the way Multinomial Naive Bayes Classifiers work and the complexity of the algorithm, we recommend checking out the original paper on this subject, written by [18].

3.6.2 Random Forest

"The Random Forest Classifier is actually a combination of tree classifiers where each classifier is generated using a random vector sampled independently from the input vector and each tree casts a unit vote for the most popular class to classify an input vector"[25][11]. The Random Forest Classifier is an ensemble learning method that uses multiple decision trees that are randomly built to solve classification and regression problems.[33]

The authors of [23] provide a very good explanation on how the Random Forest algorithm words. We will try and summarize their explanation step by step:

1. n_{tree} bootstrap samples from the original data are being drawn.
2. For each of these bootstrap samples, grow an unpruned classification or regression tree, while taking the following into consideration: randomly sample m_{try} of the predictors and choose the best split from among those variables.
3. Predict new data by aggregating the predictions of the n_{trees}.[23]

3.6.3 Decision Tree

The Decision Tree is a machine learning algorithm used for a long time in machine learning problems. "A decision tree is comprised out of decision nodes and leaf nodes. Each decision node corresponds to a test X over a single attribute of the input data and has a number of branches, each of which handles an outcome of the test X. Each leaf node represents a class that is the result of decision for a case."[31]

A decision tree works by splitting the data into subsets. After this phase, follows the pruning phase with the goal of reducing the size of tree branches. This is achieved by turning some branch nodes into leaf nodes and by removing the original leaf nodes from the root branch. The goal of the Decision Tree Algorithm is to find the smallest tree that fits the data. Usually the smallest tree yields the lowest error.[3]

The authors of [22] have a very good explanation on how one of the most famous Decision Tree Algorithms works, namely the ID3 algorithm which was originally proposed by [29]. For further information, we advise the readers to take a look at the paper written by [22].

3.6.4 SVM

Support vector machines are considered to be a staple of text classification due to numerous advantages they offer. One of the advantages of SVM's is it's high dimensional input space which basically means that they can handle datasets containing very large feature sets. Another advantage of SVM's is the fact that they usually take all features into consideration which boosts the performance when trying to classify text, as text usually imposes to use the greatest number of feature possible for classification. Other features worth mentioning are the fact that document vectors are mostly sparse, meaning that only a few entries in the matrix are not 0 and the fact that most text classification problems are usually linearly separable.[17]

The basic idea behind Support Vector Machines is that SVM's try to find a hyperplane which divides the dataset into two or more "sections". The hyperplane is basically a line which divides the dataset. The further from the dataplane the datapoints lie, the better the classification. The best hyperplane can is the one where the margin is the greatest between the hyperplane and the training set. The margin is the distance between the closest datapoint to the hyperplane and the hyperplane. [7]

3.6.5 Gradient Boosting Classifier

As [20] himself described it, Gradient Boosting is basically comprised of two separate algorithms, namely Gradient Descent and Boosting. The Gradient Boosting Classifier is suited for all types of problems including: regression, classification and ranking. The Gradient Boosting Algorithm is based on the Adaboost algorithm.

The idea behind a Gradient Boosting Classifier is that a decision tree is being trained in which each observation is assigned an equal weight. After completing the evaluation for the first tree the weights of the observations that are difficult to classify are being increased while those of the observations that are easy to classify are being lowered. A second decision tree sprouts as a result of weights manipulation. The new model is now comprised of both trees. This process which generates new trees is repeated for a number of set times. All sprouting trees help classify the data that could not be classified by previous trees. The Gradient Boosting Classifier identifies the weak points of classification by using gradients in the loss function. The loss function is a metric for determining how good a model's coefficients are at fitting the underlying data. [8]

3.6.6 K Nearest Neighbour Algorithm

The authors of [32] provide a wonderful explanation for the K Nearest Neighbour Algorithm, "The KNN approach is a non-parametric that has been used in the early 1970's in statistical applications. The basic theory behind KNN is that in the calibration dataset, it finds a group of k samples that are nearest to unknown samples (e.g., based on distance functions). From these k samples, the label (class) of unknown samples are determined by calculating the average of the response variables (i.e., the class attributes of the k nearest neighbor). As a result, for this classifier, the k plays an important role in the performance of the KNN, i.e., it is the key tuning parameter of KNN. The parameter k was determined using a bootstrap procedure. In this study, we examined k values from 1 to 20 to identify the optimal k value for all training sample sets."[32]

3.6.7 Multilayer Perceptron Classifier

The Multilayer Perceptron Classifier is the simplest neural network form. It's main points are that it is a feedforward neural network which makes use of the backpropagation algorithm. The most basic architecture of a MLP Classifier is an input layer, followed by a hidden layer and an output layer. The input layer is always on the bottom, while the output layer is always on top. Every neuron in the upper layer is connected to neurons in the bottom layer. This means that the MLP Classifier is a fully connected network. Each neuron besides those in the input layer contains a nonlinear activation function. The backpropagation algorithm is the main component of the MLP's training.[33]

According to [15], the connections between layers should have weights somewhere in the range between -1 and 1. Each of these nodes can perform two actions: summation and activation. The summing is calculated using the following formula:

$$S_j = \sum_{i=1}^{n} \omega_{ij} I_i + \beta_j \tag{6}$$

n stands for the number of inputs, I_i denotes the input variable i, β_j is a bias term, w_{ij} stands for the connection weight.[15] The most used activation

function in MLP's is the S-shaped sigmoid function which can be described using the following formula[15]:

$$f_j(x) = \frac{1}{1 + e^{-S_j}} \tag{7}$$

From these two formulas we can conclude the output of the neuron j using the following formula [15]:

$$y_i = f_i(\sum_{i=1}^{n} \omega_{ij} I_i + \beta_j) \tag{8}$$

The main focus of MLP classifiers is to solve the problem of classifying non linear data which simple perceptron based classifiers cannot. An example of a non linear problem would be the XOR operation which MLP's are able to solve with two neurons which converge into one, usually the one being in the hidden layer.

3.6.8 Logistic Regression

The authors of [26] have managed to explain the Logistic Regression Classifier in a very simple manner. According to [26] the logistic regression model analyzes the relationship between multiple independent variables and a categorical dependent variable. It can estimate the probability of an occurrence of an event by fitting the data to a logistic curve. There are two types of logistic regression models, the binary logistic regression which is used when the dependent variable is dichotomous and the independent variables are either continuous and categorical. The other type of logistic regression is the multinomial logistic regression and it is typically used when the dependent variable is not dichotomous and is comprised of more than two categories[26].

The basic idea behind a Logistic Regression Classifier is that the output of the linear function is squashed between the values 0 and 1 using the sigmoid function. The sigmoid function is an S-shaped curve which can take any value between 0 and 1, but never 0 neither 1. Generally speaking, a threshold is assigned and values are compared to it. If a value is greater than the threshold, the label 1 gets assigned to it. Analog, if a value is smaller than the threshold, the value of 0 gets assigned to it[6].

The formula on which the logistic regression classifier is:

$$log(\frac{p(X)}{1 - p(X)}) = \beta_0 + \beta_1 X \tag{9}$$

The left hand side of the formula is called the logit, while the value inside the logarithmic function is called odds. The odds is a value standing for the ratio for the probability of success to the probability of failure.[5] The sigmoid function can be expressed as the inverse of the formula above:

$$p(X) = \frac{e^{\beta_0 + \beta_1 X}}{1 + e^{\beta_0 + \beta_1 X}} \tag{10}$$

4 Implementation

For our project we have decided upon using multiple technologies. The main programming languages used in this project are Python and SQL for data querying. The main frameworks we have chosen to use for our project are:

- scikit-learn: a free machine learning library written in Python which contains implementations of the most common classifiers.
- Jupyter Lab: an interactive Python environment.
- NLTK: a Python package built for manipulating human language data. The most used features of NLTK are classification, tokenization, stemming, tagging and parsing.
- Pandas: a Python package used for data manipulation and analysis.
- Numpy: a Python package used for scientific computing.
- Contractions: a Python package used for solving contractions into separate words.
- Matplotlib: a 2D Python plotting library.
- String: a Python package used for interacting with the String datatype
- Re: a Python implementation of regex
- Seaborn: a statistical data visualization library written in Python
- Wordcloud: a Python package used for generating word clouds.

4.1 Strategy

Due to the fact that our hardware capabilities were limited we decided to only use 5000 reviews in order to try to speed up the computations. Another extra step that we took in order to tackle this challenge was to split our approach in two separate steps. In the first step we decided to pre-process the text and export it to CSV format to avoid performing pre-processing each time before training each separate classifier.

In the second part of out project we performed feature engineering followed by training and testing the classifier and analyzing the results. Another very important aspect of this project is that we made use of Python Notebooks, to be more specific Jupyter Notebooks. For each text engineering method, we decided to use a separate notebook. For example, when using a count vectorizer, we decided to make a separate notebook for unigrams, unigrams+bigrams, unigrams+bigrams+trigrams in order to have a better code separation and a better overview of the results. Another very important aspect is the fact that we also decided to split the classifications, by that we mean separate files for sentiment analysis and rating prediction out of the same reasons mentioned above.

In the diagram below we can see each of the major steps we took when working on this project. We will briefly go through each step and try to explain the core aspect of each step and discuss the code behind.

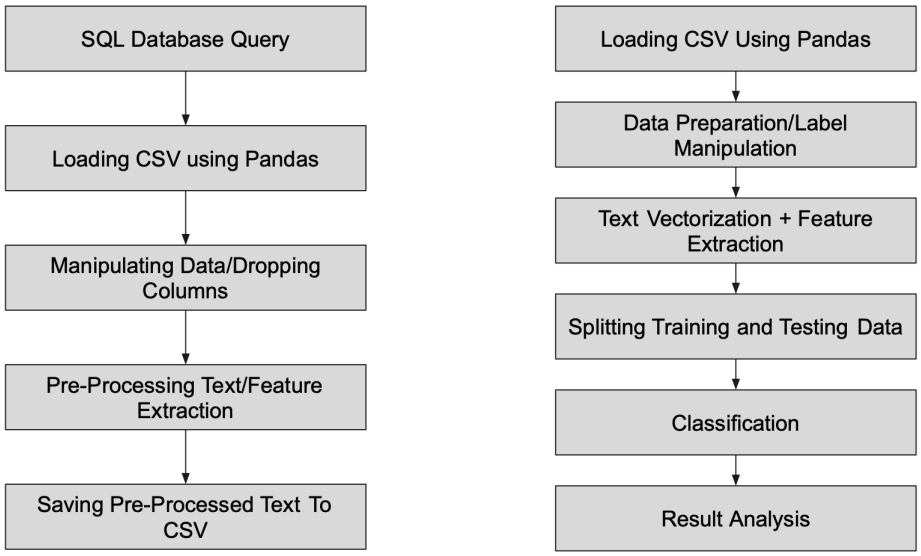


Figure 5: Workflow

4.2 Data Pre-Processing

4.2.1 SQL Database Query

In order for us to be able to solve the issue of rating prediction and sentiment analysis, we needed to find a large database containing reviews. Prof. Bing Liu from the University of Illinois at Chicago was kind enough to give us access to his database of Yelp Restaurant Reviews[16]. We received the database under the form of an SQLite Database.

The database consists of 3 tables: restaurant, review, reviewer; each of these tables containing relevant information according to their name. After browsing the database, we came to the conclusion that although the database should only contain restaurant reviews, it also contained reviews from other types of businesses listed on yelp. In order to tackle this issue and the issue mentioned before, namely the fact that a large part of the reviews are not checked whether they are real or fake, we had to come up with an SQL solution for extracting the data. For this, we have decided to only use reviews which had a useful count higher than the value 7, which basically means that 7 people have found the review useful. This should mean that the selected reviews should be real and should contain useful information.

Listing 1: SQL Query example for selecting 1 star reviews

```
SELECT *
FROM ( SELECT reviewContent, usefulCount, coolCount,
    funnyCount, rating,
        restaurantID
        FROM review
        WHERE rating = 1
        AND (usefulCount >= 7)
        )
WHERE restaurantID in (
  select restaurantID
  from restaurant
  where categories LIKE "Restaurant%"
)
```

As we can see in the code snippet above, we had to create a select statement for each rating category and then combine all of these queries into a single one using the UNION parameter. In order to only select restaurants, we had to create an additional where clause to check the ID of the restaurant from the review table with the one in the restaurant table to make sure the business is indeed a restaurant. The results were exported to CSV format.

4.2.2 Loading CSV using Pandas

In order to be able to interact with the data using the Python programming language, we had to make use of the Pandas package to import the dataset and transform it to a pandas dataframe. This simple operation was achieved using the Pandas method **pd.read_csv** which reads a CSV file and automatically converts it to a pandas dataframe.

4.2.3 Manipulating Data & Pre-processing

As mentioned earlier, we decided to try 6 different methods for text cleaning for this project. Because we already discussed the methods that were used in order to clean text, we would like to discuss the following piece of code which is the code used to clean the text and stem it using the Porter Algorithm.

```
def stem_clean_remove_stopwords(review):
    # remove xa0
    review = re.sub(r"\xa0", "", review)
    # remove links
    review = re.sub(r"http\S+", "", review)
    # lowercase words
    review = review.lower()
    # remove contractions
    review = contractions.fix(review)
    # remove punctuation and tokenize
```

```
    review = [word.strip(string.punctuation) for word in
        review.split(" ")]
    # remove numbers
    review = [word for word in review if not any(
        character.isdigit() for character in word)]
    # remove stopwords
    stopwords_engl = stopwords.words("english")
    review = [word for word in review if word not in
        stopwords_engl]
    # remove whitespaces
    review = [word for word in review if word]
    # stem review
    stemmer = PorterStemmer()
    review = [stemmer.stem(word) for word in review]
    # remove words with only a letter
    review = [word for word in review if len(word) > 1]
    # join all back together
    full_string_return = " ".join(review)
    return full_string_return
```

As mentioned before, due to the fact that reviews contained the \xa0, we needed to remove it before continuing with text cleaning. The stemming is performed after removing stopwords and whitespaces in order for us to be sure that only actual words are being stemmed and not spaces or other types of characters. As we can see, in order to process the text, we chose to tokenize it into separate words and after stemming we join all words back together and return the string in it's original format.

The text cleaning is performed using a lambda function which is applied to all rows in the dataset. We chose to create another column and append the result to that column in order to be able to compare the cleaning methods side by side. In order to achieve this, we have repeated the following process 6 times. The example below illustrates how the lambda function operates on the dataset.

```
yelp_data_5k['reviewCleanNoStopwords'] = yelp_data_5k['
    reviewContent'].apply(default_clean_remove_stopwords)
```

4.2.4 Saving Pre-Processed Text to CSV

We chose to export the dataset containing the pre-processed reviews to CSV format using pandas. Pandas provides a function which automatically converts a dataset to the correct CSV format, the method we are talking about being **pd.to_csv**.

4.2.5 Loading CSV Data

The process we are about to describe is present in used for all the combinations of text pre-processed methods and classifiers we have chosen to use for this

project. This means that the next part will be present in 12 different files, with subtle changes adapted for each method used. The data is loaded into the Jupyter Lab environment using the same Pandas method described above.

4.2.6 Data Preparation

This is one of the most important steps in our project, as this is the step which decides which approach the classifiers will take. In this step we drop and manipulate the data necessary to perform sentiment analysis and rating prediction respectively. Let us start with sentiment analysis. For this approach we have decided to drop the reviews which have been rated with 3 stars as these reviews are generally seen as neutral. In order to achieve this, we had to select the indexes from all rows containing the rating 3 and save them into a variable. Next, we deleted those rows from the dataset using the Pandas **drop** method. In order to check if the rows have been successfully removed from the dataset we called the **value_counts** method using the Pandas package. This method returns all different values from a column.

```
# Get names of indexes for which column rating has value
    3
indexNamesRating3 = yelp_data_5k[ yelp_data_5k['rating']
    == 3 ].index

# Delete these row indexes from dataFrame
yelp_data_5k.drop(indexNamesRating3, inplace=True)
```

The last step requires us to transform the reviews which have been rated as 1 or 2 stars to the label 0, which stands for negative and those rated as 4 or 5 stars to the label 1 which stands for positive. We have achieved this using a lambda function as we can see below:

```
yelp_data_5k['rating'] = yelp_data_5k['rating'].apply(
    lambda x: 1 if x>3 else 0)
```

For our other approach, the rating prediction, we chose to drop reviews rated as 2 or 4 stars due to the fact that they are very similar and quite hard to distinguish even for a human reader. We decided to use the rating 1, 3 and 5 which we refer to as bad, neutral and good. In order to achieve this we selected the indexes of the rows of the reviews rated with 2 or 4 stars and chose to drop them, exactly in the same fashion as described above.

4.3 Text Vectorization & Feature extraction

For this next step, we have decided to save all the rows containing processed reviews, each into it's own variable as seen in the example below.

```
x_lemmatized = yelp_data_5k["reviewCleanLemmatized"]
```

For the vectorization part, we have decided to use two different approaches. We decided to try and use both a Count Vectorizer[1] and a TFIDF Vectorizer[2],

both being implemented in the scikit-learn framework. After some experimenting, we decided that the best results are achieved when building the bag-of-words model using words that have a minimum text frequency of at least 5. This means that all words which do not meet this criteria are excluded from the vocabulary. We decided it would be best if we would perform the vectorization for each type of text cleaning separately. This is also the step where we decide what N-Gram to use. In the code below, we can see how we decided to code the count vectorizer using a minimum document frequency of 5 and using a combination of unigrams, bigrams and trigrams.

```
cv = CountVectorizer(ngram_range=(1,3), min_df=5).fit(
    x_porterStemmer)
x_porterStemmer = cv.transform(x_porterStemmer)
```

4.4 Splitting training and testing data

This is quite a simple step, as we made use of the **train_test_split** method built into the scikit-learn framework.

```
x_train_cleanWithStopwords, x_test_cleanWithStopwords,
    y_train, y_test = train_test_split(
    x_cleanWithStopwords, y, test_size=0.2, random_state
    =50)
```

As we can see, the method returns 3 variables, which are used for training and testing. Basically it splits the dataset into two parts, with a ratio of 0.2 for the test size which means that we use 80% of the data for training and 20% of the data for testing. The variables contain the text and ratings of said reviews.

4.5 Classification & Result Analysis

In this part we will discuss how we chose to structure and run all classifiers. It is worth mentioning that due to the fact that our whole project was programmed using the Jupyter Lab environment, all the code can be found inside Jupyter Notebooks. Another important reason to mention this is the fact that all classifiers can be found and run from the same notebook which corresponds to each different feature engineering method. In order to understand easier what we mean, you can find two folders named *2 ratings* and *3 ratings* respectively (git repository link `https://gitlab.com/sebastian.herman/bachelor-final`), which contain each of the six feature engineering methods and classifiers, each inside its own notebook.

In order to have a better overview of the classifiers, we chose to group them by type and run them consecutively 6 times, each time for one of the text cleaning methods. We will discuss the basic format of our classifiers and then proceed to discuss the parameters we chose to use, where applied. For this first explanation, we choose the Multinomial Naive Bayes classifier.

The initialization and fitting of the classifier is quite easy, due to the fact that scikit-learn provides intuitive methods which allow us to code using few lines of code:

```
mnb = MultinomialNB()
mnb.fit(x_train_cleanWithStopwords, y_train)

prediction = mnb.predict(x_test_cleanWithStopwords)
```

As we can see, training the classifier is also easy when it comes to the coding, requiring only the use of the **predict** method.

Next, we have chosen to output a confusion matrix, print the accuracy score and classification report to screen. In addition we have also decided to export the classification report to CSV format in order to be able to compare all the results using Excel. However, we will discuss this part more thoroughly in the Evaluation and Discussion part of this paper.

4.5.1 Classifier Parameters

As mentioned before, we have tried to find parameters that work both when trying to predict 2 and 3 values out of consistency reasons. In our opinion this is the best approach when trying to compare multiple combinations of classifiers and feature engineering methods.

For the Multinomial Naive Bayes Classifier, we have chosen to use the default parameters which sets **alpha = 1.0**, **fit_prior = True** and **class_prior = None**.

For the Random Forest Classifier, we have decided to leave the default values unchanged, the only exception being the number of estimators. For this we have found the value of 200 to yield the best accuracy after some experimentation.

For the Decision Tree Classifier, we have also found that the default parameters performed best.

For the Support Vector Machine Classifier, we have decided to change two parameters, namely the random state parameter in order to retain some consistency and to use a **linear kernel** instead of the default **rbf** one.

For the Gradient Boosting Classifier we had to experiment quite a while until we found some parameters which yielded a good accuracy. For this we have chosen to modify following hyperparameters: **learning_rate = 0.1**, **max_depth=5**, **max_features = 0.5**, and a random state for consistency purposes.

For the K Neighbours classifier we have chosen to use 10 neighbours, instead of the 5 default ones.

The Multilayer Perceptron Classifier seemed to perform best at this task with default hyperparameter values.

However, the Logistic Regression Classifier required us to experiment quite a while until we have found a well suited value for the inverse of the regularization strength, parameter noted with **C**. Other parameters we chose to modify are the **solver** parameter which we have assigned the "saga" solver and we also had to

modify the **multi_class** in order to be able to use the classifier for multinomial classification.

4.6 Accuracy Metrics

In order to compare the results to the other papers we have found to be related, we have chosen metrics that are common amongst all works. For this, we chose following metrics:

- Precision: calculated by dividing the number of true positives by the number of true + false positives
- Recall: calculated by dividing the number of true positives by the number of true positives + false negatives
- F1: the harmonic mean of precision and recall

5 Evaluation

As mentioned beforehand, the main metrics we have used to measure the accuracy of our project are the precision, recall and F1 metrics. We chose to use these metrics as these metrics have been used by others in their research and allow us to make a direct comparison between our results and their results. We chose the F1 score as the reference metric when comparing a classifier to another.

Due to the fact that we have tested a large number of classifiers combined with a large number of text pre-processing and feature engineering methods, we will discuss only the best results of each classifier in the following part. However, we will provide all our results in the appendix of this paper, so that anyone can analyze our results.

The score which we will use for comparison with other papers is the weighted average of the F1 score which can be found in bold text in each table in the top right hand corner. We will try to compare our results with results obtained by other peers where applicable.

5.1 Sentiment Analysis

5.1.1 Multilayer Perceptron Classifier

This classifier performed the best when trying to classify the polarity of a review. The best score was achieved by using the bag-of-words model using the Count Vectorizer. As for text engineering, we have found the unigram + bigram + trigram to perform the best when using the baseline text processing method (including stop words). As we can see in the table below, using this method we were able to achieve an accuracy of 95%.

Due to the fact that no other author which has tried to solve this problem has used a Multilayer Perceptron Classifier, we have nothing to compare it's performance against. However the accuracy of 95% is higher than any achieved

by any other author before us. So we can only assume that this classifier is performing the best out of all others. We encourage others to also try and use this classifier when dealing with the issue of text classification.

The combination we have found to give the best results for this classifier is: unigram + bigram + trigram + CV + baseline text pre-processing without removing stop words. The weighted average F1-score we achieved using this method is 0.9525.

		0	1	micro avg	macro avg	weighted avg
0	f1-score	0.9557	0.9489	0.9525	0.9523	**0.9525**
1	precision	0.9519	0.9533	0.9525	0.9526	0.9525
2	recall	0.9595	0.9445	0.9525	0.9520	0.9525
3	support	495.0	433.0	928.0	928.0	928.0

Table 3: MLP + unigram + bigram + trigram + TFIDF + baseline including stopwords

5.1.2 Logistic Regression Classifier

The Logistic Regression Classifier is the second best performing classifier when trying to determine a reviews polarity. When comparing our results to the results achieved by [30], we can state that we have achieved a better accuracy, 0.900 being the highest score achieved by [30] using a Linear Regression classifier. The author of [27] has achieved an F1 score of 0.9164 when using a Logistic Regression Classifier in combination with a TFIDF vectorizer.

The combination we have found to give the best results for this classifier is: unigram + bigram + TFIDF + Snowball Stemming Algorithm. The combination we have found to give the best results for this classifier is: unigram + bigram + trigram + CV + baseline text pre-processing without removing stop words. The weighted average F1-score we achieved using this method is 0.9483.

		0	1	micro avg	macro avg	weighted avg
0	f1-score	0.9508	0.9454	0.9482	0.9481	**0.9483**
1	precision	0.9646	0.9306	0.9482	0.9476	0.9487
2	recall	0.9373	0.9607	0.9482	0.9490	0.9482
3	support	495.0	433.0	928.0	928.0	928.0

Table 4: Logistic Regression + unigram + bigram + TFIDF + Snowball

5.1.3 SVM

The Support Vector Machine performs third best in out sentiment polarity classification. We can observe that the F1 score obtained by this classifier is very close to the ones obtained by LR and MLP classifiers. The author of [27] has achieved an overall F1 score of 0.9130 using the SVM classifier coupled with the TFIDF Vectorizer. [30] has achieved a score of 0.923 using the SVM

classifier. Overall, out results seem to be a bit higher than other comparable findings.

The combination we have found to give the best results for this classifier is: unigram + bigram + trigram + TFIDF + baseline text pre-processing without removing stop words. The weighted average F1-score we achieved using this method is 0.9439.

		0	1	micro avg	macro avg	weighted avg
0	f1-score	0.9476	0.9396	0.9439	0.9436	**0.9439**
1	precision	0.9438	0.9440	0.9439	0.9439	0.9439
2	recall	0.9515	0.9353	0.9439	0.9434	0.9439
3	support	495.0	433.0	928.0	928.0	928.0

Table 5: SVM + unigram + bigram + trigram + TFIDF + baseline including stopwords

5.1.4 Multinomial Naive Bayes

[30] achieved a score of 0.897 using the MNB classifier. [27] 0.8935 using the Count Vectorizer method.

The combination we have found to give the best results for this classifier is: unigram + bigram + trigram + TFIDF + baseline text pre-processing and removing stop words. The weighted average F1-score we achieved using this method is 0.8966.

		0	1	micro avg	macro avg	weighted avg
0	f1-score	0.9016	0.8909	0.8965	0.8962	**0.8966**
1	precision	0.9147	0.8769	0.8965	0.8958	0.8971
2	recall	0.8888	0.9053	0.8965	0.8971	0.8965
3	support	495.0	433.0	928.0	928.0	928.0

Table 6: MNB + unigram + bigram + trigram + TFIDF + baseline without stopwords

5.1.5 Random Forest

[27] achieved a score of 0.8886 using the Random Forest classifier coupled with the TFIDF vectorizer.

The combination we have found to give the best results for this classifier is: unigram + bigram + trigram + TFIDF + Snowball. The weighted average F1-score we achieved using this method is 0.8932.

		0	1	micro avg	macro avg	weighted avg
0	f1-score	0.9007	0.8847	0.8933	0.8927	**0.8932**
1	precision	0.8944	0.8920	0.8933	0.8932	0.8933
2	recall	0.9070	0.8775	0.8933	0.8923	0.8933
3	support	495.0	433.0	928.0	928.0	928.0

Table 7: Random Forest + unigram + bigram + trigram + TFIDF + Snowball

5.1.6 Gradient Boosting Classifier

The combination we have found to give the best results for this classifier is: unigram + CV + baseline text pre-processing and removing stop words. The weighted average F1-score we achieved using this method is 0.8846.

		0	1	micro avg	macro avg	weighted avg
0	f1-score	0.8926	0.8754	0.8846	0.88407	**0.8846**
1	precision	0.8864	0.8826	0.8846	0.8845	0.8846
2	recall	0.8989	0.8683	0.8846	0.8836	0.8846
3	support	495.0	433.0	928.0	928.0	928.0

Table 8: Gradient Boosting + unigram + count vectorizer + baseline without stopwords

5.1.7 Decision Tree

The combination we have found to give the best results for this classifier is: unigram + bigram + trigram + TFIDF + Porter Algorithm. The weighted average F1-score we achieved using this method is 0.7508.

		0	1	micro avg	macro avg	weighted avg
0	f1-score	0.7692	0.7298	0.7510	0.7495	**0.7508**
1	precision	0.7608	0.7393	0.7510	0.7501	0.7508
2	recall	0.7777	0.7205	0.7510	0.7491	0.7510
3	support	495.0	433.0	928.0	928.0	928.0

Table 9: Decision Tree + unigram + bigram + trigram + tfidf + Porter

5.1.8 K-Nearest-Neighbours Classifier

The combination we have found to give the best results for this classifier is: unigram + bigram + trigram + TFIDF + baseline text pre-processing without removing stop words. The weighted average F1-score we achieved using this method is 0.7507.

		0	1	micro avg	macro avg	weighted avg
0	f1-score	0.7701	0.7285	0.7510	0.7493	**0.7507**
1	precision	0.7588	0.7416	0.7510	0.7502	0.7507
2	recall	0.7818	0.7159	0.7510	0.7488	0.7510
3	support	495.0	433.0	928.0	928.0	928.0

Table 10: KNN + unigram + bigram + trigram + tfidf + baseline with stop words

5.2 Rating Prediction

In this section we can see the best results of all classifiers while trying to predict 3 labels: 1 - bad, 3 - neutral, 5 - good. Due to the fact that this approach is not very common, we have found only one other paper with which we can directly compare our results. The other author who tried to predict 3 labels in the same way we have chosen to, has used 3 methods overall, two combinations using the KNN classifier and one combination where the author makes use of a Support Vector Machine.[10]

5.2.1 SVM

We have found that using the 3 label method, the SVM classifier performs classifier best. This is also the case for the analysis made by [10]. The authors have managed to achieve a score of 0.764 while using the linear SVM classifier coupled with the bag-of-words model. Our approach seemed to perform best when using a combination of n-grams, namely unigram + bigram + trigram in conjunction with the baseline processing algorithm without the removal of stop words applied by the TFIDF vectorizer.

The combination we have found to give the best results for this classifier is: unigram + bigram + trigram + TFIDF + baseline text pre-processing without removing stop words. The weighted average F1-score we achieved using this method is 0.8278.

		1	3	5	micro avg	macro avg	weighted avg
0	f1-score	0.8871	0.7482	0.8400	0.8285	0.8251	**0.8278**
1	precision	0.8715	0.7570	0.8475	0.8285	0.8253	0.8275
2	recall	0.9032	0.7397	0.8325	0.8285	0.8251	0.8285
3	support	248.0	219.0	227.0	694.0	694.0	694.0

Table 11: SVM + unigram + bigram + trigram + TFIDF + baseline with stopwords

5.2.2 Multilayer Perceptron Classifier

The combination we have found to give the best results for this classifier is: unigram + bigram + TFIDF + baseline text pre-processing without removing

stop words. The weighted average F1-score we achieved using this method is 0.8228.

		1	3	5	micro avg	macro avg	weighted avg
0	f1-score	0.8762	0.7405	0.843	0.8242	0.8202	**0.8228**
1	precision	0.8544	0.7658	0.8421	0.8242	0.8207	0.8224
2	recall	0.8991	0.7168	0.8458	0.8242	0.8206	0.8242
3	support	248.0	219.0	227.0	694.0	694.0	694.0

Table 12: MLP + unigram + bigram + TFIDF + baseline with stopwords

5.2.3 Logistic Regression Classifier

The combination we have found to give the best results for this classifier is: unigram + bigram + TFIDF + baseline text pre-processing without removing stop words. The weighted average F1-score we achieved using this method is 0.8176.

		1	3	5	micro avg	macro avg	weighted avg
0	f1-score	0.8747	0.7348	0.8351	0.8184	0.8149	**0.8176**
1	precision	0.8627	0.7488	0.8333	0.8184	0.8149	0.8171
2	recall	0.8870	0.7214	0.8370	0.8184	0.8151	0.8184
3	support	248.0	219.0	227.0	694.0	694.0	694.0

Table 13: Logistic Regression + unigram + bigram + TFIDF + baseline with stopwords

5.2.4 Gradient Boosting Classifier

The combination we have found to give the best results for this classifier is: unigram + bigram + CV + baseline text pre-processing without removing stop words. The weighted average F1-score we achieved using this method is 0.7820.

		1	3	5	micro avg	macro avg	weighted avg
0	f1-score	0.8505	0.7039	0.7826	0.7838	0.7790	**0.7820**
1	precision	0.8102	0.7190	0.8142	0.7838	0.7811	0.7827
2	recall	0.8951	0.68949	0.7533	0.7838	0.7793	0.7838
3	support	248.0	219.0	227.0	694.0	694.0	694.0

Table 14: Gradient Boosting + unigram + bigram + count vectorizer + baseline with stopwords

5.2.5 Random Forest

The combination we have found to give the best results for this classifier is: unigram + CV + Porter Algorithm. The weighted average F1-score we achieved using this method is 0.7724.

		1	3	5	micro avg	macro avg	weighted avg
0	f1-score	0.8349	0.6908	0.7829	0.7752	0.7695	**0.7724**
1	precision	0.7885	0.7333	0.7954	0.7752	0.7724	0.7733
2	recall	0.8870	0.6529	0.7709	0.7752	0.7703	0.7752
3	support	248.0	219.0	227.0	694.0	694.0	694.0

Table 15: Random Forest + unigram + count vectorizer + Porter

5.2.6 Multinomial Naive Bayes

The combination we have found to give the best results for this classifier is: unigram + bigram + TFIDF + baseline text pre-processing without removing stop words. The weighted average F1-score we achieved using this method is 0.7690.

		1	3	5	micro avg	macro avg	weighted avg
0	f1-score	0.8235	0.6633	0.8114	0.7723	0.7661	**0.7690**
1	precision	0.8285	0.7180	0.7586	0.7723	0.7684	0.7708
2	recall	0.8185	0.6164	0.8722	0.7723	0.7690	0.7723
3	support	248.0	219.0	227.0	694.0	694.0	694.0

Table 16: MNB + unigram + bigram + TFIDF + baseline with stopwords

5.2.7 K-Nearest-Neighbours Classifier

The author of [10] also used the KNN classifier when trying to predict 3 labels. As we both ourselves and the author of [10] have concluded, the KNN classifier doesn't perform that good at this particular task. The other mentioned author managed to achieve a score of 0.5912 using the KNN classifier, only managing to push its performance to 0.5968 while using trigrams. Our classifier performed best when using a combination of unigrams + bigrams + baseline processing without the removal of stop words all while using the TFIDF method for vectorization.

The combination we have found to give the best results for this classifier is: unigram + bigram + TFIDF + baseline text pre-processing without removing stop words. The weighted average F1-score we achieved using this method is 0.6432.

		1	3	5	micro avg	macro avg	weighted avg
0	f1-score	0.7063	0.5476	0.6666	0.6469	0.6402	**0.6432**
1	precision	0.6551	0.5894	0.6869	0.6469	0.6438	0.6448
2	recall	0.7661	0.5114	0.6475	0.6469	0.6417	0.6469
3	support	248.0	219.0	227.0	694.0	694.0	694.0

Table 17: KNN + unigram + bigram + TFIDF + with stopwords

5.2.8 Decision Tree

The combination we have found to give the best results for this classifier is: unigram + bigram + trigram + CV + Snowball Algorithm. The weighted average F1-score we achieved using this method is 0.6130.

		1	3	5	micro avg	macro avg	weighted avg
0	f1-score	0.6555	0.5569	0.6206	0.6109	0.6110	**0.6130**
1	precision	0.6796	0.5176	0.6490	0.6109	0.6154	0.6185
2	recall	0.6330	0.6027	0.5947	0.6109	0.6101	0.6109
3	support	248.0	219.0	227.0	694.0	694.0	694.0

Table 18: Decision Tree + unigram + bigram + trigram + count vectorizer + snowball

5.3 Approach Performance and Comparison

After benchmarking all our classifiers with all the combinations of text pre-processing and feature engineering method, we have found the first approach which predicts the polarity of a review to be vastly more accurate than our second approach. This is due to the fact that it is much easier to distinguish between a good and a bad word for example, unlike having to guess a third label, which in most cases consists of both positive and negative words. However, we were pleased to see that guessing 3 labels can also be done with a high enough accuracy.

For the sentiment analysis classification, we have found that the Multilayer Perceptron Classifier and the Logistic Regression classifier perform best overall when using either combination of text pre-processing and feature extraction. The Multilayer Perceptron performed best overall in scenarios where the traditional bag-of-words approach was used, being implemented with the help of the Count Vectorizer. The Logistic Regression Classifier performed best when using the TFIDF Vectorizer.

For the rating prediction part, we have found the same classifiers to perform the best. The Multilayer Perceptron Classisifer seems to perform best when using a Count Vectorizer and the Logistic Regression performs best when using a TFIDF Vectorizer.

In both scenarios, using a larger n-gram number yields generally better results, with a few exceptions. As for the text pre-processing methods we have found that the larger the number of n-grams, the more important stop words become. This is because when taking larger grams into consideration, binding words start to become increasingly important, especially when it comes to negations. For those who wish to explore further, we recommend starting with an n-gram range of (1,2) as this is a good baseline accuracy to compare further research against. We would also advise those who wish to research in this area further not to exclude stopwords from their benchmarks as stopwords can really boost the accuracy the higher the n-gram range used. As for stemming/lem-

matizing we have found the Porter Stemmer to be more than enough, as the performance increase provided by the Snowball and Lancaster stemmer is not great enough to justify using such aggressive stemming methods which in turn could decrease the accuracy especially when it comes to negations.

To our surprise, the Multinomial Naive Bayes classifier, a classifier used traditionally for text classification problems, didn't perform as good as expected. However, we can say that after performing our benchmarks, we highly recommend using a Multilayer Perceptron classifier, a Logistic Regression classifier or a Support Vector Machine classifier no matter if the purpose of the classification is sentiment analysis or rating prediction. We have noticed minimal changes in accuracy (in the range 0.1 to 0.5) when processing the reviews with a different random seed for splitting, which makes us believe that these classifiers truly work best when trying to solve such problems. However, we do recommend that instead of taking our advice of X classifier being generally the best, readers try to fine tune the classifiers because by tuning the hyperparameters even more, there is a chance for the accuracy to go even higher.

Using the same methods for both approaches, we have found that the polarity of a review can be detected more easily due to the fact that the "neutral" reviews are mostly comprised of both positive and negative words. Our opinion is that while sentiment analysis is a technology that can be implemented with success in detecting the polarity of a review, an approach which tries to predict a rating is not suitable. We would advise the reader to try and solve this issue by using a custom built neural network which could boost the accuracy of rating prediction.

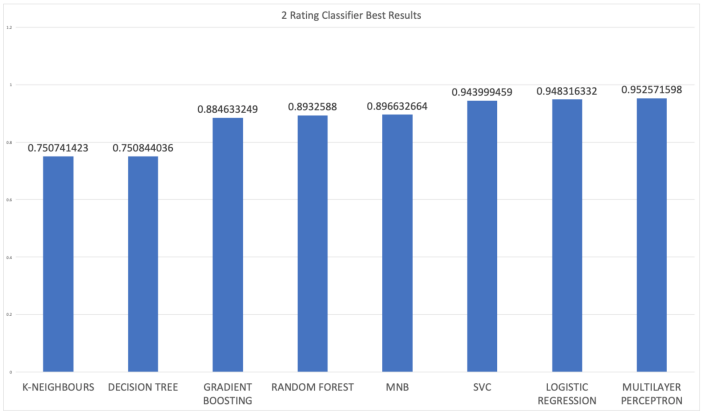


Figure 6: Classifier ranking for sentiment analysis

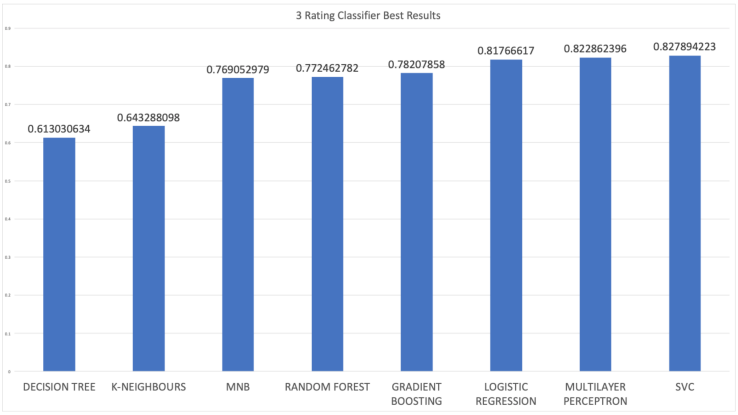


Figure 7: Classifier ranking for rating prediction

5.3.1 Benchmark Results

Due to the fact that our benchmark includes 438 different classification reports we are unable to present all results in this paper. We would advise those interested to visit the following link where all results can be found in form of Excel Sheets: `https://gitlab.com/sebastian.herman/bachelor-final`

6 Conclusion

In this paper we explore two different ways to classify reviews. The first approach tries to classify the polarization or sentiment of a review. Our second approach tries to also take "neutral" ratings into consideration in order to see how it impacts the overall accuracy.

In our project, we have managed to achieve a very good accuracy for sentiment analysis, going up to 95% accuracy when using a certain combination of classifiers and feature engineering methods, as described above. As for rating prediction, our opinion is that while having achieved a prediction accuracy of about 82%, it is not enough to make this technique reliable in a real world scenario. It is very likely that in a few years this will no longer be an issue, as progress in the machine learning field is being made at a very high rate.

We find this issue to be of utmost importance due to the fact that nowadays most people who have access to internet use the internet and available platforms to form an opinion about a product they are about to purchase or a location they are about to visit. Both techniques would be useful to review service providers, not necessarily to users. This is because online business depend a lot on customer written reviews when it comes to conducting business. Such systems would allow websites to monitor more closely the accuracy of a rating written by a customer. For example sentiment analysis could be used to predict if the general opinion for some product or restaurant in order to be able to

recommend and promote it to other customers. The review rating prediction could be useful for such websites in order to minimize spam and check if the rating which has been attributed to a specific product or business is fair and has some kind of backing in the text review.

We would have liked to have access to more powerful hardware that would have allowed us to train our models using a greater number of reviews and a greater number of features. Other areas which would have benefited massively form more powerful hardware would be hyperparameter tuning. Hyperparameter tuning usually implies training the models multiple times and observing which of the hyperparameter combinations performs best using the given dataset. We would have also liked to be able to perform n-crossfold validation for our classifiers, but again, due to limited hardware this would have taken a very long time.

We also feel the need to state that the nature of this project was to benchmark various combinations of classifiers and feature engineering methods which. Because of this we have used classifiers built in the scikit-learn framework. We are pretty sure that building a custom neural network tailored specifically to predict reviews in a certain domain would yield better results. But building a custom neural network for text classification is a very complex task, requiring a lot of time and knowledge in various domains like mathematics and statistics. We recommend users who wish to approach this problem using the neural network approach to take a look at our results and start the design of their neural network taking our results into considerations. This could result in a quicker and better development process.

6.1 Future Work

This paper can be used as a starting point for those who would like to predict the rating of text based reviews or just the sentiment polarity. We would strongly advise people who wish to use this project as a starting point to use powerful hardware. The scikit-learn framework can only use the CPU of a machine, so a CPU with a high core clock and high core number would be recommended. Moreover, we are sure that tuning hyperparameters would yield better results, but this is a very resource intensive task and can take a lot of time to complete, especially when experimenting with larger n-gram numbers, as larger n-grams exponentially increase the time needed for calculations.

7 Appendix

All benchmark results can be found as mentioned above, by visiting the following link: `https://gitlab.com/sebastian.herman/bachelor-final`. The README.md file contains all necessary instructions for accessing the benchmark Excel spreadsheets.

References

[1] https://scikit-learn.org/stable/modules/generated/sklearn.feature_extraction.text.CountVectorizer.html. Accessed: 14.07.2019.

[2] https://scikit-learn.org/stable/modules/generated/sklearn.feature_extraction.text.TfidfVectorizer.html. Accessed: 14.07.2019.

[3] Decision trees explained easily. https://medium.com/@chiragsehra42/decision-trees-explained-easily-28f23241248. Accessed: 13.07.2019.

[4] An introduction to bag-of-words in nlp. https://medium.com/greyatom/an-introduction-to-bag-of-words-in-nlp-ac967d43b428. Accessed: 12.07.2019.

[5] Logistic regression. simplified. https://medium.com/data-science-group-iitr/logistic-regression-simplified-9b4efe801389. Accessed: 13.07.2019.

[6] Support vector machine vs logistic regression. https://towardsdatascience.com/support-vector-machine-vs-logistic-regression-94cc2975433f. Accessed: 13.07.2019.

[7] Support vector machines for dummies. http://blog.aylien.com/support-vector-machines-for-dummies-a-simple/. Accessed: 13.07.2019.

[8] Understanding gradient boosting machines. https://towardsdatascience.com/understanding-gradient-boosting-machines-9be756fe76ab. Accessed: 13.07.2019.

[9] Asghar, N. Yelp dataset challenge: Review rating prediction. *arXiv preprint arXiv:1605.05362* (2016).

[10] Bandla, N. Predict user ratings based on review texts.

[11] Breiman, L. 1 random forests–random features.

[12] Chen, C., Zhang, M., Liu, Y., and Ma, S. Neural attentional rating regression with review-level explanations. In *Proceedings of the 2018 World Wide Web Conference* (2018), International World Wide Web Conferences Steering Committee, pp. 1583–1592.

[13] Dave, K., Lawrence, S., and Pennock, D. M. Mining the peanut gallery: Opinion extraction and semantic classification of product reviews. In *Proceedings of the 12th international conference on World Wide Web* (2003), ACM, pp. 519–528.

[14] Fan, M., and Khademi, M. Predicting a business star in yelp from its reviews text alone. *arXiv preprint arXiv:1401.0864* (2014).

[15] Heidari, A. A., Faris, H., Aljarah, I., and Mirjalili, S. An efficient hybrid multilayer perceptron neural network with grasshopper optimization. *Soft Computing* (2018), 1–18.

[16] Hu, M., and Liu, B. Mining and summarizing customer reviews. In *Proceedings of the tenth ACM SIGKDD international conference on Knowledge discovery and data mining* (2004), ACM, pp. 168–177.

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

[18] Kibriya, A. M., Frank, E., Pfahringer, B., and Holmes, G. Multinomial naive bayes for text categorization revisited. In *Australasian Joint Conference on Artificial Intelligence* (2004), Springer, pp. 488–499.

[19] Lei, X., Qian, X., and Zhao, G. Rating prediction based on social sentiment from textual reviews. *IEEE Transactions on Multimedia 18*, 9 (2016), 1910–1921.

[20] Li, C. A gentle introduction to gradient boosting. *URL: http://www. ccs. neu. edu/home/vip/teach/MLcourse/4_ boosting/slides/-gradient_ boosting. pdf* (2016).

[21] Li, C., and Zhang, J. Prediction of yelp review star rating using sentiment analysis, 2014.

[22] Li, Y. H., and Jain, A. K. Classification of text documents. *The Computer Journal 41*, 8 (1998), 537–546.

[23] Liaw, A., Wiener, M., et al. Classification and regression by randomforest. *R news 2*, 3 (2002), 18–22.

[24] Mukherjee, A., Venkataraman, V., Liu, B., and Glance, N. What yelp fake review filter might be doing? In *Seventh international AAAI conference on weblogs and social media* (2013).

[25] Pal, M. Random forest classifier for remote sensing classification. *International Journal of Remote Sensing 26*, 1 (2005), 217–222.

[26] Park, H. An introduction to logistic regression: from basic concepts to interpretation with particular attention to nursing domain. *Journal of Korean Academy of Nursing 43*, 2 (2013), 154–164.

[27] Qiao, R. *Yelp Review Rating Prediction: Sentiment Analysis and the Neighborhood-Based Recommender*. PhD thesis, UCLA, 2019.

[28] Qu, L., Ifrim, G., and Weikum, G. The bag-of-opinions method for review rating prediction from sparse text patterns. In *Proceedings of the 23rd international conference on computational linguistics* (2010), Association for Computational Linguistics, pp. 913–921.

[29] Quinlan, J. R. Induction of decision trees. *Machine learning 1*, 1 (1986), 81–106.

[30] Salinca, A. Business reviews classification using sentiment analysis. In *2015 17th International Symposium on Symbolic and Numeric Algorithms for Scientific Computing (SYNASC)* (2015), IEEE, pp. 247–250.

[31] Stein, G., Chen, B., Wu, A. S., and Hua, K. A. Decision tree classifier for network intrusion detection with ga-based feature selection. In *Proceedings of the 43rd annual Southeast regional conference-Volume 2* (2005), ACM, pp. 136–141.

[32] Thanh Noi, P., and Kappas, M. Comparison of random forest, k-nearest neighbor, and support vector machine classifiers for land cover classification using sentinel-2 imagery. *Sensors 18*, 1 (2018), 18.

[33] Wan, S., Liang, Y., Zhang, Y., and Guizani, M. Deep multi-layer perceptron classifier for behavior analysis to estimate parkinsonâĂŹs disease severity using smartphones. *IEEE Access 6* (2018), 36825–36833.

[34] Wang, J. Predicting yelp star ratings based on text analysis of user reviews.

[35] Willett, P. The porter stemming algorithm: then and now. *Program 40*, 3 (2006), 219–223.

[36] Xu, Y., Wu, X., and Wang, Q. Sentiment analysis of yelp's ratings based on text reviews, 2014.

[37] Zheng, L., Noroozi, V., and Yu, P. S. Joint deep modeling of users and items using reviews for recommendation. In *Proceedings of the Tenth ACM International Conference on Web Search and Data Mining* (2017), ACM, pp. 425–434.