

National College of Ireland

BSc (Honours) in Computing - Evening

Data Analytics

2021/2022

Dejan Mijakovac

x18144861

x18144861@student.ncirl.ie

Opinion Mining for Automated Restaurant Reviews Rating

Technical Report

Contents

Executive Summary	2
1.0 Introduction	3
1.1. Background	3
1.2. Aims.....	4
1.3. Technology	5
1.4. Structure	7
2.0 Data	8
3.0 Methodology.....	13
4.0 Analysis	21
5.0 Results.....	39
6.0 Conclusions	56
7.0 Further Development or Research	57
8.0 References	58
9.0 Appendices.....	60
9.1. Project Proposal	60
1.0 Objectives.....	62
2.0 Background.....	62
3.0 State of the Art	63
4.0 Data	63
5.0 Methodology & Analysis.....	64
6.0 Technical Details.....	65
7.0 Project Plan	66
8.0 Bibliography.....	68
9.2. Reflective Journals	69
9.3. Outputs of models' performance on test set	76

Executive Summary

Restaurant industry is a high grossing one, both in Ireland and worldwide. With most businesses moving online during the COVID-19 pandemic, customers started to spend more time online, writing online reviews. Not only were more reviews being written, but people also relied more on those reviews when deciding whether to use a particular restaurant's service. All this put additional pressure on restaurants to maintain a positive online presence and high ratings. This pressure has increased recently with the high levels of inflation, which forces customers to reduce their outdoor dining and food orders. As a result of this, they are even more careful when it comes to online ratings.

The aim of this project was to, using the KDD methodology, develop a model which can predict, solely on the basis of the textual review, the rating of a review, on a scale of 1 to 5. An additional aim was to develop a model which can predict the review sentiment, it being negative, positive or neutral. While the sentiment analysis classification is a well-known topic studied in the Natural Language Processing field, the idea to classify the reviews according to rating is one that is not usually entertained due to the complexity of the classification process.

The textual reviews were vectorized using two different approaches – Count Vectorizer and Tfidf Vectorizer - in order for them to be used by the machine learning algorithms. Four out of five algorithms used are the ones typically used for sentiment analysis (Logistic Regression, Support Vector Machines, Random Forest and Naïve Bayes), while Neural Networks were introduced in this project due to the high number of features and reviews that needed to be processed. As Neural Networks perform better with large, complex datasets and highly-dimensional features, their choice was a logical one, especially for the ratings classification.

The final results show that Logistic Regression is the best-performing algorithm in both use cases – ratings and sentiment classification. Furthermore, Support Vector Machines and Neural Networks also perform very well. When sentiment classification is compared to the results obtained using a tool already on the market (Vader), all models developed during the project outperform it. This means that these models, especially the best-performing ones, can be used either on restaurants' websites to predict ratings or sentiment, or internally, when reviewing customer feedback, to identify problems that need to be rectified and positive feedback which helps to note what customers enjoy.

1.0 Introduction

1.1. Background

Restaurant industry is a very lucrative industry. Even during the year that was mostly impacted by the COVID-19 pandemic (2020), the restaurant industry had an estimated worth of \$1.2 trillion (Aureliano-Silva, et al., 2021). If more local data is examined, Bord Bia reported that people on the Irish market spent around €8.55 billion on restaurant dining in 2019, with a significant increase of 4.5% when compared to 2018 (O'Brien, 2019). In addition to this, whenever COVID-19 restrictions were eased, the spending in restaurants increased significantly. For example, in 2021, in the month of June, a total of €332 million was spent on meals in restaurants in the Republic of Ireland, an increase of 41% compared to the same period in the previous year (Ihle, 2021). This means that the restaurants can make a good profit once there are no restrictions or with minimal restrictions.

Quality online presence can help restaurants attract customers more than any other factor, especially in these times. As lockdowns come and go, restaurants sometimes only open for a short period of time. During that period, good online reviews are what will most likely result in increased number of customers. Recent survey conducted by ReviewTrackers showed that 94% of customers claim they avoided a business based on a bad review (Review Trackers, 2021). Similarly, a survey of US customers' attitudes towards local businesses showcased just how much they rely on online reviews. 93% of them do an online search for local businesses and 87% read their online reviews. The same survey also showed that restaurants are number one on the list of industries for which the customers would read online reviews (Murphy, 2020). All this shows just how much a restaurant's revenue can be impacted by online reviews.

COVID-19 pandemic resulted in people relying on online reviews even more, both as active and as passive participants. As a direct result of COVID-19, 31% of people surveyed claim they are reading more reviews than before (Murphy, 2020) and 70% of them use rating filters when doing online search, mostly defaulting their filter to four stars and above (Review Trackers, 2021). There are also factors showing more people are now actively engaging in writing reviews. On one hand, the overall number of online reviews at the start of the pandemic (February – May 2020) increased by 81% (PowerReviews, 2020). In addition to this, around 72% of customers have written an online review (Murphy, 2020). Not only have people been spending more time at home and online, but writing reviews with the help of smartphones and better internet coverage has become faster and easier. Along with that, the customers seem to be writing shorter and more focused reviews than before – since 2010, an average online review reduced by 65% in the number of characters, resulting in around 200 characters on average (Review Trackers, 2021).

When all this is reviewed, it becomes logical that restaurants would start paying more attention to reviews by trying to respond to them sooner, to pinpoint the issues that the customers are most sensitive about and improve both their online presence and their

business practices. Having previously worked in customer support for an online website-building service, most customer complaints related to e-commerce were about the ability to automate the reviews process. Some businesses had problems with the fact that the customer wrote a really nice review, but clicked on the wrong number of stars so it got registered as a 1-star review, which reduced their overall rating. So, for them, a system where the star rating would be generated solely based on the customers' textual review would be very beneficial.

However, this would only solve part of their problem. Most customers will not go directly to the restaurant's website to read the reviews. At this point, four services are proving to be the ones where most customers post and / or read reviews. Those four services host 88% of all online reviews. Google is the absolute winner with 73% of the market, while Yelp (6%), Facebook (3%), and TripAdvisor (3%) are the other three major forces (Review Trackers, 2021). For that reason, businesses would probably profit even more by having a model that could analyse the reviews for them and classify them correctly. Finally, the restaurant owners could focus on negative reviews to isolate what needs to be changed, or on the extremely positive ones, which would showcase what needs to remain the same. And this is what this project is focused on – trying to analyse restaurant reviews and build models which can correctly classify reviews, according to their rating and sentiment.

1.2. Aims

The main goal of this project is to determine the overall rating of reviews on the scale from 1 to 5, based solely on the text of the review. In addition to this, the goal is to build a model that can classify the review as positive, negative or neutral. The actual field that is the basis of the project, sentiment analysis, is also known as emotion AI or opinion mining and its main purpose is to determine whether a review is positive or negative. In addition to this, it can also be used to predict ratings for restaurant reviews (Zahoor, et al., 2020) and this is how it is used in this project.

In order for these two goals to be achieved, a number of smaller goals needed to be achieved as well. The first subset of these smaller goals is related to the actual process of performing sentiment analysis. Before any of the actual work started, research was conducted on previous works done on sentiment analysis. This step is crucial in order to avoid steering the work in the wrong direction and prevent resource wastage. Then, quality datasets, which meet the project requirements, were acquired. Once the data was found, it was examined, processed and transformed so it could be ready for machine learning. Using classification machine learning algorithms and the process of training, validating and tuning the model, the best-performing algorithm was identified and tested against the final test data. At last, in the final step of the process, the results were collated and presented using summary tables and charts.

The other subset of smaller goals relates all the other work that was done in order to ensure that the first subset of goals is achieved. First of all, a project plan was created so the

project execution could be monitored. Then, all the documentation was updated timely to reflect all stages of the process. Furthermore, all the technology and tools needed to complete the project were identified and explored. Finally, in order to present the final results, as well as the results of the previous phases, data visualisation techniques were used. Fulfilling all these goals ensured that well-performing models were being developed.

1.3. Technology

The technologies and tools used can be divided into two distinct groups. The first group consists of tools used to manipulate the data, implement machine learning algorithms and handle data visualization. Python (3.9) is the main programming language used in the project. In order to run Python code, PyCharm IDE (integrated development environment) was used locally at the start of the process. Google Colab notebooks were used mostly for presentations as they offered a cleaner outline and were more convenient when a single line of code needed to be run, especially on a project that has hundreds of lines of code. However, as the project developed, the move was made towards Jupyter Notebooks and Anaconda. One reason for this change was that with the increasing amounts of code, running it in PyCharm was no longer feasible as the runtime was just too slow. Google Colab was abandoned as the runtime of the code with the free account was also too slow. Python libraries used to complete the project and their descriptions are available in Table 1.

Table 1 – Description of libraries used	
Library	Usage
Pandas	Importing / exporting csv files, manipulating DataFrames
Openpyxl	Importing / handling xlsx files
Re	Cleaning data using Regular Expressions
Matplotlib	Data visualization library
Seaborn	Data visualization library
Plotly	Data visualization library
NumPy	Library used to process multidimensional arrays
NLTK	Language processing library, items used from it for this project: RegexpTokenizer - text tokenization FreqDist - frequency distribution bigrams - used for analysing bigrams trigrams - used for analysing bigrams stopwords - used to identify stop-words in reviews WordNetLemmatizer - used for lemmatizing reviews SentimentIntensityAnalyzer – pre-built tool for sentiment analysis, used only in the end to verify how the models built perform compared to it

Scikit-Learn	<p>Machine learning library used for text vectorization and building, training and testing models, items used for this project:</p> <p>CountVectorizer, TfidfVectorizer - used to vectorize textual data</p> <p>GridSearchCV – used to find best hyperparameters for the models</p> <p>LogisticRegression - used to build, train and test models using LogisticRegression algorithm</p> <p>MultinomialNB - used to build, train and test models using Naïve Bayes algorithm</p> <p>RandomForestClassifier - used to build, train and test models using Random Forest algorithm</p> <p>LinearSVM - used to build, train and test models using Support Vector Machines algorithm</p> <p>confusion_matrix - builds confusion matrix for the current implementation of one of the algorithms</p> <p>accuracy_score - calculates the accuracy for the current implementation of one of the algorithms</p> <p>classification_report - lists the full set of metrics that outline the model's performance</p> <p>make_pipeline – collates all the pre-processing and model-building steps into one single call</p>
tensorflow	<p>Library used for neural networks / deep learning, items used for this project:</p> <p>keras - deep learning API that enables Tensorflow to build neural networks</p> <p>Sequential – a model used for building neural networks</p> <p>Dense – a type of layer added to the network</p> <p>Activation – type of function passed into the layer</p> <p>Dropout – layer used to prevent overfitting</p> <p>EarlyStopping – stops training when the relevant metric stops improving</p> <p>load_model – used to save and load neural networks models</p>
selenium	Automation / testing library used to scrape data from websites
time	Library used to set delays between actions performed by the machine during automated tasks

Probably the two main choices that were the most difficult to make, in terms of libraries used, are Scikit-Learn and TensorFlow. Scikit-Learn was, in the end, an obvious choice for machine learning implementation as it has a very clean and uniform structure, as well as good online documentation. Basically, once one model is built, it is easy to move to a completely different algorithm as the basic syntax is similar enough (VanderPlas, 2017), which is important when approaching the project in an iterative way. When it comes to TensorFlow, the choice was primarily made due to its syntactic similarity to Scikit-Learn. This library is

primarily used to build either small or very large Neural Networks and relies on the Keras API, Google's deep learning tool (Géron, 2019).

The second group of tools are those used to manage the project completion. This set of tools mostly consists of programs from the Microsoft Office package. Word was used for all the documentation that was submitted, from project proposal, ethics declaration, reflective journals, midterm submission document to the final technical document. Excel was used mostly for initial review of files stored in xlsx or csv files. In order to create slides for video presentations, PowerPoint was the preferred tool. As for the video-creation, videos were recorded using Microsoft Teams and then edited using Kdenlive. An online tool was used for project management – TeamGantt. As the name itself suggests, the tool is used to create and host Gantt charts.

1.4. Structure

The document is divided into several main sections, each of which contains crucial information on the project execution. Those are:

<u>Data</u>	This section provides basic information on the datasets chosen. In addition to this, it contains plots and charts which are the result of data exploration. Its overall goal is to showcase the strengths and weaknesses of each dataset and provide additional insight into what changes needed to be made in the later stages of data manipulation.
<u>Methodology</u>	Methodology explains the process of project completion. The stages that are covered in detail in this section are data selection, pre-processing and transformation. It also includes basic information on the data mining and evaluation stages.
<u>Analysis</u>	More detailed information on vectorization process, model building, splitting data into training, validation and testing sets, and the reasoning behind the choice of machine learning algorithms is provided here.
<u>Results</u>	This section outlines the overall results for all of the chosen machine learning algorithms, taking into consideration different hyperparameters, vocabulary sets, as well as different usage of vectorizers and unigrams, bigrams and trigrams.
<u>Conclusions</u>	Final overview of the results with the clarification on the best machine learning approach to use and potential future applications.

Further Development or Research

Details how the results could be further improved and what other approaches can be taken to attempt this type of analysis.

2.0 Data

Zenodo Dataset

The first dataset acquired for the project was Social website reviews and ratings of Dublin restaurants situated across 65 locations (Basheer & Kaushik, 2019). It is a public secondary dataset and it was downloaded directly from the website in .xlsx format. Its size is 1.37MB. As seen from the title, it is a collection of restaurant reviews generated from social media and focused on the Dublin area. The full dataset contains 10 000 reviews and 11 columns. The columns' descriptions can be seen in Table 2.

Table 2 – Zenodo dataset description		
Column name	Description	Data Type
Restaurant ID	Restaurant names are anonymized and replaced by an ID.	String
Location ID	Restaurant location anonymized and replaced by an ID.	String
Review	A textual review written by a user for the restaurant identified by the Restaurant ID and the Location ID.	String
Review Sentiment	Overall review sentiment – defaults to either Positive or Negative. The dataset has 5000 positive and 5000 negative reviews.	String
Cuisine	Dominant type of cuisine offered by the restaurant.	String
Price Range	Price range for a meal at the restaurant. It has three default values – under 30€, 31-50€ and more than 50€.	String
Food Rating	User's evaluation of food quality. User can choose the rating between 1 and 5.	Integer
Service Rating	User's evaluation of the service in the restaurant. User can choose the rating between 1 and 5.	Integer
Ambient Rating	User's evaluation of the ambient / atmosphere. User can choose the rating between 1 and 5.	Integer
Overall Rating	User's evaluation of food quality. User can choose the rating between 1 and 5.	Integer
Restaurant Rating	Overall restaurant rating which is a general average of all the users' overall ratings. This is not a value derived from the values entered in the columns 7-10.	Float

TripAdvisor Dataset

The second dataset was acquired primarily to account for the imbalance in the ratings distribution of the first one. Unlike the first dataset, which has a low count of four-star reviews, this dataset is perfectly balanced. The data was acquired by scraping the reviews of 40 different restaurants using Python and Selenium. The data was saved in a csv file, the size of which is 1.04MB. There are 2000 reviews in the dataset, which consists of two columns only – one containing the reviews (string) and the other one containing the ratings (integer). Restaurant and user information were not used to create the dataset.

Yelp Dataset

The final dataset is the Yelp's public dataset made available by the company for academic purposes (Yelp, 2021). The dataset contains information collected from different areas of the USA and Canada and is split into several different JSON files, each of them focussing on a different area (businesses, users, tips...). For the purpose of this paper, only review.json file was used. As this file is 6.45GB and holds more than 8 million reviews, from wide range of businesses, a subset of this dataset only including restaurant reviews was used. This subset contains 208 214 reviews and only two columns – stars (integer) and text (string). The description of the original dataset is available in Table 3.

Table 3 – Yelp dataset description		
Column name	Description	Data Type
review_id	A unique ID for each review.	String
user_id	ID of the review author.	String
business_id	ID of the restaurant reviewed.	String
stars	The rating user assigned to the restaurant, ranges from 1 to 5.	Integer
date	Date of the review.	String
text	The user's textual review of the restaurant.	Integer
useful	Number of times review was voted useful.	Integer
funny	Number of times review was voted funny.	Integer
cool	Number of times review was voted cool.	Integer

Exploratory Analysis

Zenodo Dataset

This dataset was imported using Pandas and openpyxl libraries, both of which are necessary for working with .xlsx files. Immediately after reviewing general information on the dataset, it became clear that an additional column exists that was not mentioned in the dataset description. Upon more detailed examination, it transpired that the column in

question (named Unnamed: 11), contains only one non-null value. Further investigation showed that the column's only value is located in row 2589 and that the value entered is almost the mirror image of the value entered in the Review column. As there are no other entries in that column and there are no other null values in the dataset, it was possible to move to the next step. However, this check did find a useful piece of information which is useful for data cleaning. The actual review text in that row contains an escape character (\n) which needs to be removed. If that column is ignored, the resulting count of unique values for non-numerical columns can be seen in Table 4.

Table 4 – Unique values distribution in Zenodo dataset

Restaurant ID	211
Location ID	65
Review	9982
Cuisine	43

Note that Price Range and Review Sentiment are not included in the table as their unique values have previously been identified.

Regarding other columns, it is interesting to note that not all reviews are unique. This is not surprising as there could easily be several reviews with the exact same text, especially if the review text is short and generic. An additional check was made to identify if these are complete duplicates, but full matches were not found, meaning that even if the review text is the same, the rest of the information is different. Furthermore, there are over 40 different types of cuisines included in the dataset, with Irish, Italian and British dominating. The ten most popular ones are presented in the chart in Figure 1. All this shows how data was collected on a very diverse set of restaurants, cuisines and locations.

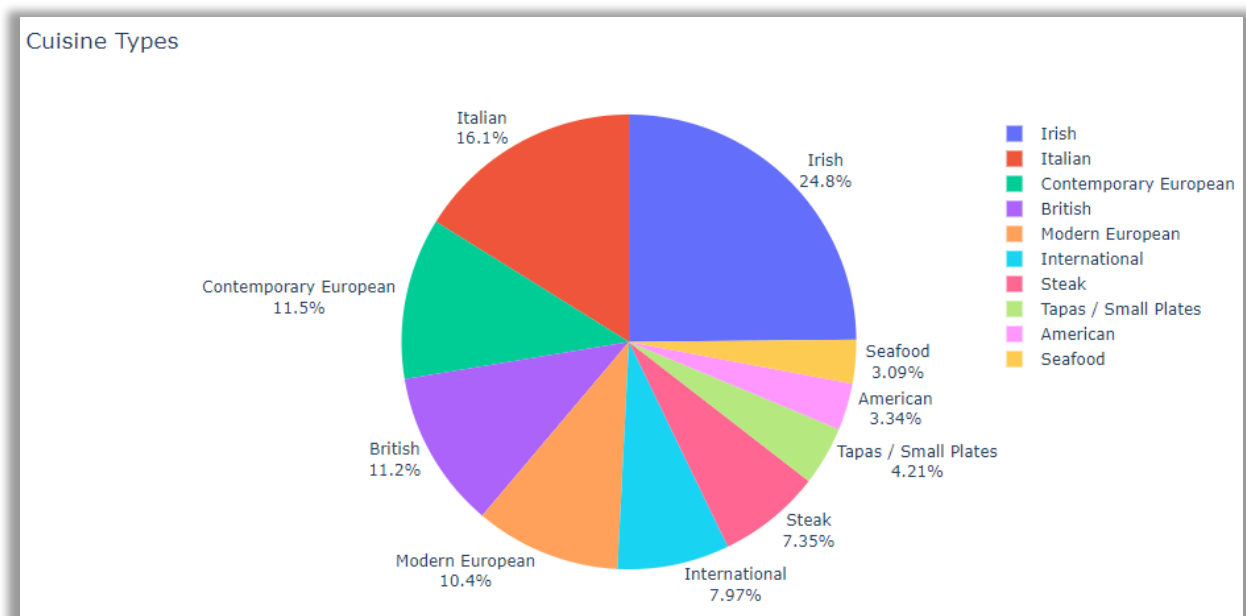


Figure 1 – Top 10 cuisine types in Zenodo dataset

When it comes to the numerical columns, Food Rating (Figure 2), Service Rating (Figure 3) and Ambient Rating (Figure 4) are mostly dominated by extremely positive reviews – five-star ratings account for around 50% in each of them. However, there seems to be one main difference between them – users were much more inclined to give 1 star when it comes to service rating compared to the other categories. Also, in general, 4 stars do not seem to be awarded in any of the categories as often as other ones – only Ambient Rating has over 10% of 4-star ratings.

5	4759
2	1685
3	1444
1	1378
4	734

Figure 2 – Food rating

5	4851
1	1811
2	1281
3	1201
4	856

Figure 3 – Service rating

5	4919
3	1764
2	1204
4	1083
1	1030

Figure 4 – Ambience rating

Similar results can be seen when the column of interest, Overall Rating, is reviewed. Again, the number of 4-star reviews is quite low, while 5-star ratings dominate the dataset. In case other datasets had the same imbalance, this might have impeded the model from correctly classifying 4-star reviews as the number of 4-star reviews it can be trained on is fairly limited. The distribution of rating preferences can be seen in the chart in Figure 5.

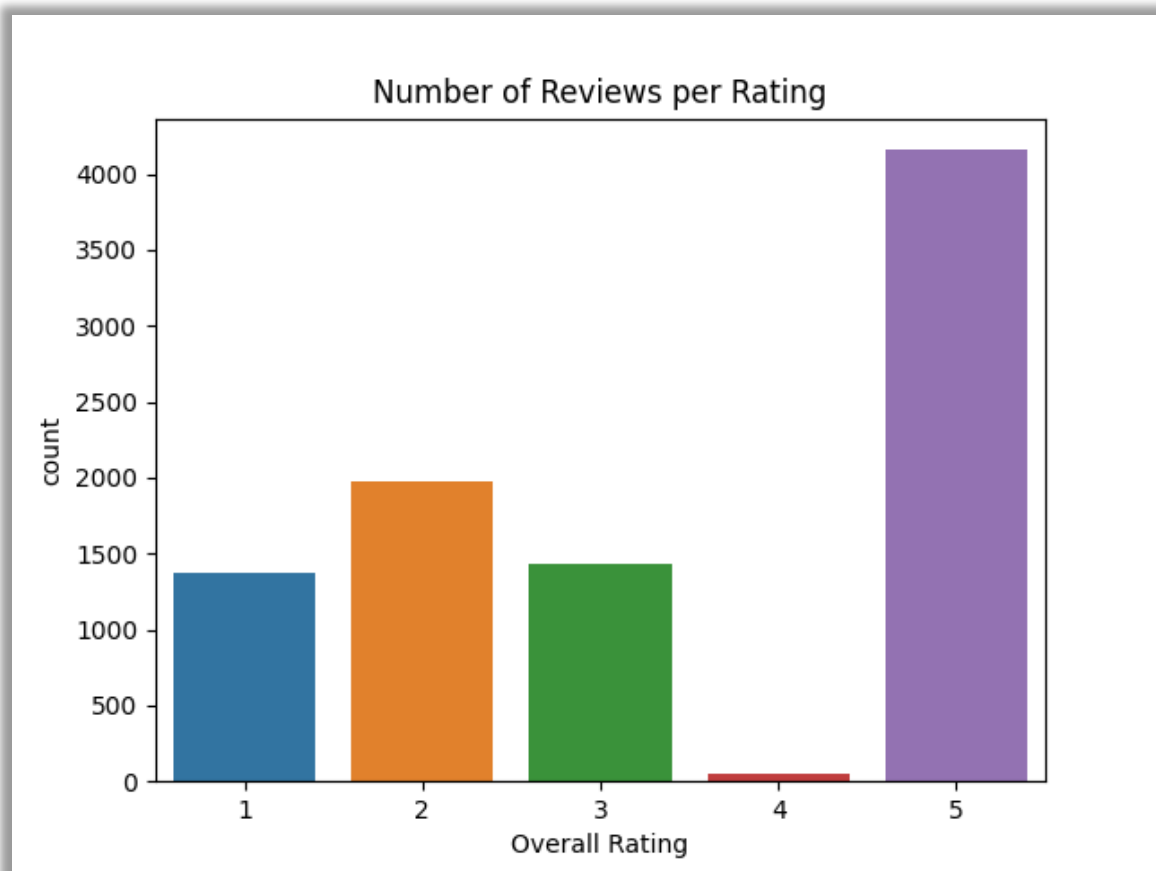


Figure 5 – Distribution of ratings in Zenodo dataset

When all this is taken into consideration, it is no surprise that the dataset has almost identical number of positive and negative reviews, as evidenced in Figure 6. Ultimately, the balanced nature of the dataset was the main driver behind the decision to use it for the analysis.

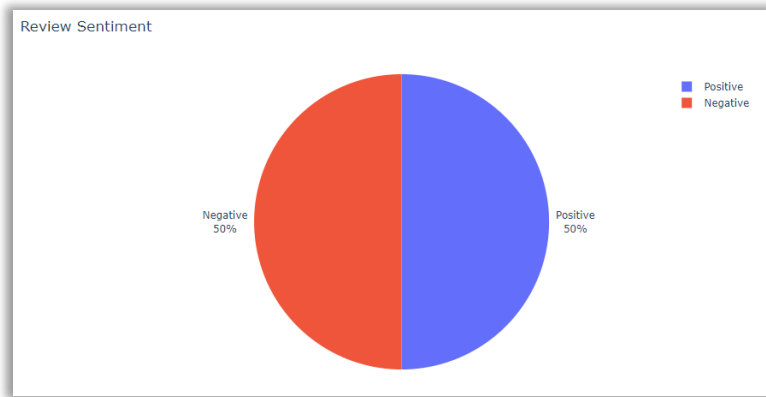


Figure 6 – Distribution of positive and negative reviews

TripAdvisor Dataset

This dataset was imported using Pandas. The dataset itself is perfectly balanced as it contains 2000 reviews and each rating has 400 occurrences in it. This can be seen in Figure 7.

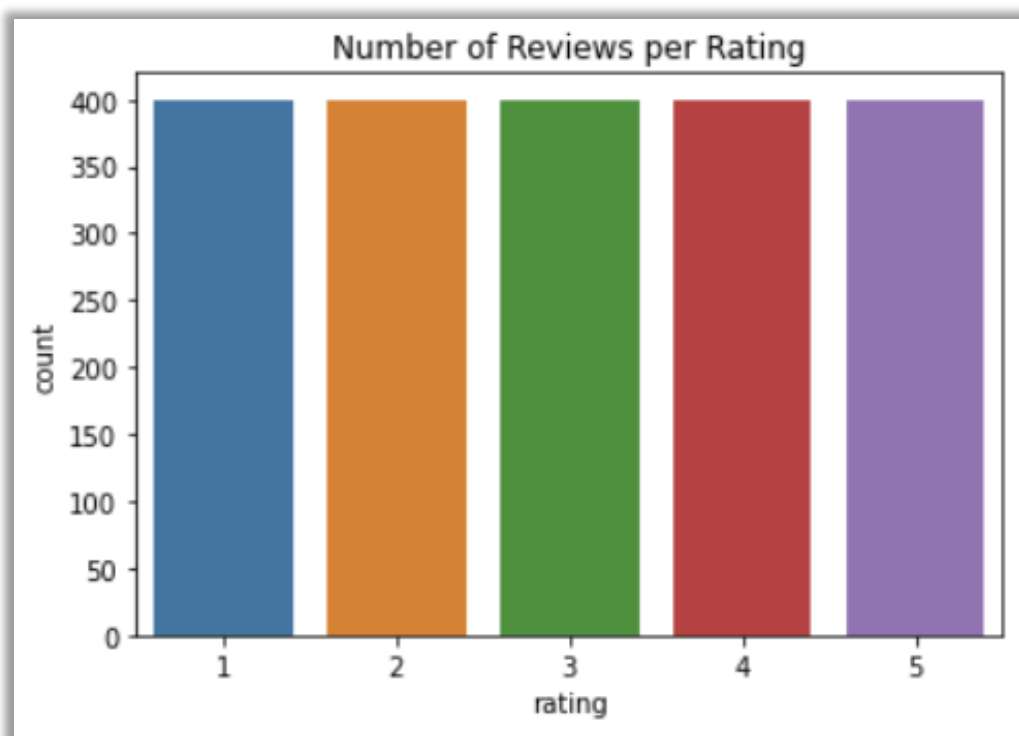


Figure 7 - Distribution of ratings in TripAdvisor dataset

Yelp Dataset

The full dataset containing all reviews and ratings was imported using Pandas. Then, a smaller subset was selected, containing 208 214 reviews – those only belonging to restaurants. The smaller dataset does echo the overall distribution of the full dataset, as the positive reviews dominate. However, based on the overall high number of reviews available, this should still enable the model to correctly predict the correct rating. The distribution of ratings (or stars in this dataset) is available in Figure 8.

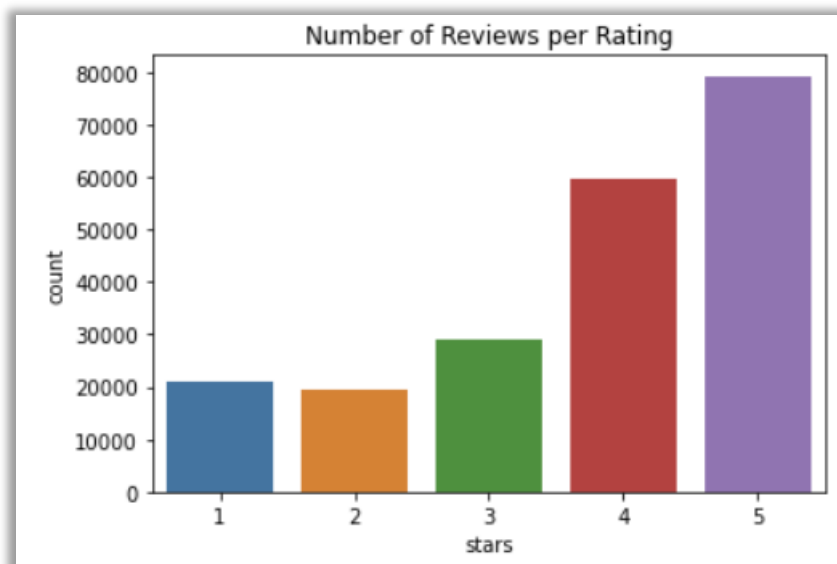


Figure 8 - Distribution of ratings in Yelp dataset

3.0 Methodology

In order to complete the project, the KDD process methodology was followed. KDD stands for knowledge discovery from data and its main goal is to identify “valid, novel, potentially useful, and ultimately understandable patterns in data” (Fayyad, et al., 1996). This methodology relies on an interactive and iterative process where decisions are made by the researcher and that is especially important for this project. As there is a need to keep tuning the model and revisit previous phases in order to make different choices in regards to pre-processing and transformation, this process is a perfect fit here. KDD consists of 5 major stages, each of which is explained in its own section.

Data Selection

When it comes to the data selection, KDD defines this phase as the one where the target dataset is selected, based on a subset of variables on which data exploration is to be performed and which can be used for feature generation and machine learning (Fayyad, et al., 1996).

In previous works conducted on a similar topic, the number of reviews used ranged from 700 (Huda, et al., 2019) to 4000 (Zahoor, et al., 2020). All of them specified the importance of having a balanced dataset which contains approximately the same number of positive and negative reviews. An unbalanced dataset of 200 positive and 500 negative ones proved to be a problem in one of them (Zahoor, et al., 2020). All of the datasets, among other columns, contained two most important ones – a textual review column and another column that served as the target column, which served for the validation of their model. That column had either a numerical rating (usually 0, 1 and 2 for negative, neutral and positive) or the information on whether the review was positive or negative.

Based on that information, it was important to acquire datasets with the following characteristics:

- contain textual reviews of restaurants only
- contain a column that denotes if the review is positive, negative or neutral and /or a column that contains numerical rating (1-5 or 1-10)
- contain at least 2000 rows as most previous research was done on smaller samples than that
- the datasets are publicly available and have clear permissions available on their websites or allow for the licence to be granted by contacting the author(s) – clear permissions were preferable as licence-granting process can sometimes be fairly long and defer the start of the project

The Zenodo dataset met all those requirements. It is actually the only one that offers both the split into negative / positive reviews and the star rating in range between 1 and 5. The Yelp dataset met most requirements. It was publicly available on their website for academic purposes and it did offer ratings from 1 to 5. The only issue with that dataset is that it does not offer great balance between the ratings, but this is offset by the sheer volume of reviews available. The TripAdvisor dataset was created using Selenium to scrape the reviews from their website, with the goal to create a dataset that contains a perfect split between the 5 different ratings. As the dataset does not contain any user or business information, the scraping process does not violate their policies. The full process of data exploration on these datasets is explained in the previous section. Table 5 contains the information on the other datasets considered and the reason why they were not chosen.

Table 5 – Other datasets considered

Name	URL	Suitability
<i>TripAdvisor reviews of hotels and restaurants by gender</i>	https://figshare.com/articles/dataset/TripAdvisor_reviews_of_hotels_and_restaurants_by_gender/6255284	The dataset had clear licensing instructions, but it had neither the numerical rating nor the information on positivity / negativity of the review so training the model would not be possible.

<i>Wake County Restaurant Reviews</i>	https://waketechnalytics.opendatasoft.com/explore/dataset/wake-county-restaurant-reviews/information/	There was no clear licensing information and the numerical ratings did not align with the actual review text. For example, 1-star reviews were sometimes very positive.
<i>TripAdvisor Restaurants Info for 31 Euro-Cities</i>	https://www.kaggle.com/damienbeneschi/krakow-ta-restaurants-data-raw?select=TA_restaurants_curated.csv	Licensing not clear and the rating column is the overall restaurant rating and not the individual review rating.
<i>RecSys2013: Yelp Business Rating Prediction</i>	https://www.kaggle.com/c/yelp-recsys-2013/data?select=final_test_set.zip	Licensing not fully transparent, the dataset seems like a modified version of the older Yelp dataset.
<i>Restaurant Customer Reviews</i>	https://www.kaggle.com/vigneshwarsofficial/reviews	Again, licensing not fully clear. This dataset is widely used across Kaggle for sentiment analysis and opinion mining. Its main drawbacks are that it only identifies reviews as positive or negative and the total number of reviews is under 1000.

Pre-processing

KDD defines this phase as the phase where noise or outliers are identified and removed if necessary. In addition to this, necessary information is collected that is later used for model-building (Fayyad, et al., 1996). This is further expanded by the processes used for this particular type of data analysis – sentiment analysis. In order to achieve better performance of the model, data pre-processing needs to be accompanied by the extraction of relevant features. Some researchers claim that this process includes noise removal, normalization, tokenization and vectorization (Krishna, et al., 2019), while others claim that tokenization and vectorization are part of the transformation phase, or actual processing phase, as they refer to it (Haque, et al., 2019). For the purpose of this paper, only vectorization has been classified as the transformation phase as all the other preceding processes seem to fit pre-processing definition better than transformation.

The steps covered in this phase are outlined below:

a) Isolating null values

This check is done to verify whether there are any null values in the columns considered important for the analysis. As mentioned before, there were null values in the extra column of the Zenodo dataset. However, as that column will not be used for further processing of the data, no action is taken. However, some of the items in the textual review column were quite short and uninformative – for example, one of them only contained the word “This”. As these entries cannot really help in predicting either the sentiment or the rating, seven of them were removed in total.

b) Isolating escape characters in the reviews and removing them

The next stages were all performed on textual reviews. During the data exploration stage, the row that contained the only entry in the unnamed column of the Zenodo dataset also pointed to the fact that there are some escape characters in the text. Specifically, the new line character (`\n`) was discovered. As this character was joined with the next word, there was a possibility that only the backslash character would be removed if a regular expression is applied to it. This would mean that instead of `\nbe`, the word would later be recorded as `nbe`. In order to prevent this, a check was done for the most common escape characters (`\n` and `\t`). Only new line escape characters were found and immediately replaced by an empty space using a function defined for that purpose. This would ensure that these characters are not present when the next step is performed. To present this in action, a sample review was taken as an example. This is what the review looked like before any pre-processing was done:

“Cocktails were very impressive and a perfect end to a seriously tasty meal.
:)

A very deserved 4 5 star rating!!!.”

Once the newline character was removed, the structure of the review changed:

“Cocktails were very impressive and a perfect end to a seriously tasty meal.
:) A very deserved 4 5 star rating!!!.”

c) Lowercasing the review text

The next step was done in order to ensure that the words are properly recorded. At this point, the words `Restaurant` and `restaurant` would be considered as two different words as one is capitalized and the other one is not. This is solved by lowercasing the words. This was also done using a specific

function that takes in all the reviews and turns all of them to lowercase. The resulting review is:

“cocktails were very impressive and a perfect end to a seriously tasty meal. :)
a very deserved 4 5 star rating!!!.”

d) Removing special characters

Once the review text was lowercased, special characters (dots, question marks, commas...) were removed using regular expressions. At this point, the review contains only lowercased words, with no special characters or numbers:

“cocktails were very impressive and a perfect end to a seriously tasty meal
a very deserved star rating”

e) Removing extra whitespace

By examining the reviews in more detail, it transpired that the reviews contained plenty of whitespace. So, a special function was written to isolate those sections and replace all the excessive whitespace with only one space. This was crucial for the next steps where the reviews would be split into individual words. The resulting sample review now looks like this:

“cocktails were very impressive and a perfect end to a seriously tasty meal a
very deserved star rating”

f) Tokenizing the review text

Tokenization is defined as breaking a piece of text into individual words that bear a specific meaning (Haque, et al., 2019). By definition, a token is a sequence of characters that is treated as a group (Bird, et al., 2009). In this case, that group of characters are meaningful words.

As all the reviews, at that point, contained reasonably clean text, it was necessary to split each of the reviews from sentence level to word level. This created a list of words for every row of the dataset, which allowed for further pre-processing. The process was completed using the NLTK library's RegexpTokenizer. The tokenized sample review is now a list of following words:

cocktails, were, very, impressive, and, a, perfect, end, to, a, seriously, tasty,
meal, a, very, deserved, star, rating

g) Removing stop-words

The previous steps were done so the individual words can be checked against a list of words often referred to as stop-words. These words include common words, like I, the, is, are – the words that do not add much additional meaning

to the sentence. These words were removed from the reviews with the help of stop-words collected by the NLTK library, with some additional words added after examining the data (yelp, com, biz, ca, www, http, us were very common, but convey no relevant meaning). The tokenized sample review with no stop-words is a list of following words:

cocktails, impressive, perfect, end, seriously, tasty, meal, deserved, star, rating

h) Lemmatizing the reviews

This part of the process ensures that the words are cleaned of suffixes. For example, the word cocktails becomes cocktail, which means that the plural and singular version of the word are now considered the same word. The sample review now contains these words:

cocktail, impressive, perfect, end, seriously, tasty, meal, deserved, star, rating

Transformation

This phase is defined by the KDD methodology as the one where useful features are found to represent the data and transformation methods are used to reduce the number of variables under consideration for the data mining phase. It is also the phase where these features are chosen based on the type of algorithms which are to be used (Fayyad, et al., 1996). For this reason, as the classification algorithms would best fit this use case, the process of transformation starts with creating the main feature that the classification will depend on – vectorized lists of words.

The steps taken in this phase are:

a) Creating the review corpus from the lemmatized text

Now that the reviews have gone through the cleaning process, each of the reviews is joined back into a string. This string contains the lowercased review sentence cleaned of punctuation, special characters, excessive whitespace and stop-words. All of the individual strings form a list of reviews – the review corpus. This corpus can now be used to prepare the data for the machine learning process. Note that corpus is sometimes used as the term for the whole dataset, especially in the context of text analysis. Each entry in the dataset (or row) is then called a document (Muller & Guido, 2017).

The final state of the sample review, after the textual transformation:

“cocktail impressive perfect end seriously tasty meal deserved star rating”

b) Count Vectorizer

In order for the machine learning algorithms to be able to perform the classification, the words in the corpus need to be transformed into numerical values. This process, of converting arbitrary data into well-behaved vectors, is called vectorization (VanderPlas, 2017). The first way of doing this is using the Count Vectorizer (also referred to as CV), which creates a numerical array out of the text. The process is done using the Scikit-Learn library and the theory behind it is explained in more detail in the Analysis section.

c) Tfidf Vectorizer

Similar to Count Vectorizer, Scikit-Learn's Tfidf Vectorizer uses the textual reviews and transforms them to a numerical array so they can be used for machine learning. As with the previous vectorization technique, the theory behind it is further explained in the Analysis section.

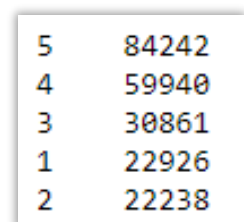
In addition to transforming the actual textual reviews, it was also necessary to organise the three datasets in a way that enables them to be joined into one dataset. The following steps were applied to get the dataset that contains both the independent and the target variables:

a) Column name change

As every dataset has different names for the textual review and the ratings columns, it was necessary to unify the naming convention first. The final choice for those two column names was review and rating so this needed to be reflected in each individual dataset. Columns in the Zenodo dataset were changed from Review and Overall Rating to review and rating, while the columns Restaurant ID, Location ID, Review Sentiment, Cuisine, Price Range, Food Rating, Service Rating, Ambience Rating, Restaurant Rating and Unnamed: 11 were dropped. The TripAdvisor dataset did not require major changes – only the reviews column was renamed to review. Finally, in the Yelp dataset, the columns text and stars were renamed to review and rating, while the rest were dropped (review_id, user_id, business_id, useful, funny, cool, date).

b) Joining the datasets

The three datasets were then combined to form a joint dataset, consisting of two columns (review and rating) and 220 207 reviews. When joining the data, the items were also randomized so that the reviews from the same dataset are not grouped together. An overview of the ratings distribution (Figure 9) shows that the joint datasets is now skewed towards higher ratings, with the lowest numbers being assigned to 1 and 2-star ratings.

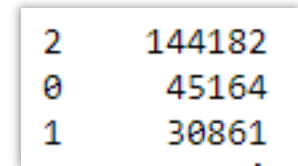


5	84242
4	59940
3	30861
1	22926
2	22238

Figure 9 - Ratings distribution

c) Feature engineering the sentiment column

Feature engineering is denoted as using important information relevant for the problem and turning it into numbers that can be used to build the feature matrix (VanderPlas, 2017). This entailed creating the sentiment variable, that would classify the reviews as positive, negative or neutral. In this case, reviews that had the ratings 1 or 2 were labelled as negative (0), those that had rating 3 were labelled as neutral (1) and the ones rated with 4 or 5 stars were labelled as positive (2). Again, as seen from Figure 10, the positive reviews dominate the dataset. Now, the final dataset contains three columns – review, rating and sentiment.



2	144182
0	45164
1	30861

Figure 10 - Sentiment distribution

Data mining

The data mining process, which is sometimes used to denote the whole process of acquiring data, data-cleaning, transformation, model-building and evaluation, denotes the part of the process which involves machine learning and model-building in KDD. Machine learning itself deals with building mathematical models to help understand data. The models learn by observing data using customizable parameters. Once these models have been tuned, they can be used on new data to gain insights or predict a certain outcome (VanderPlas, 2017). When considering this problem, in both cases (rating and sentiment), the reviews need to be grouped in either 5 or 3 different groups. This means that the best fit for the problem are classification algorithms. These algorithms work on the same basic principle – a set of variables or features are selected as independent variables which the model uses in order to try to predict the class label, an item from a predefined list of possibilities. This is done by first training the model on a part of data, which is called the training set. Later on, the model needs to be validated on previously unseen data – usually referred to as test or holdout set (Muller & Guido, 2017). The performance of the model is measured using some kind of metric or a fitness function (Géron, 2019).

Classification algorithms are divided into two subgroups – binary and multiclass (or multinomial). While binary classification attempts to group items into two categories, multinomial classification is used when there are more than two labels available for classification (Muller & Guido, 2017). During this part of the process, several different machine learning algorithms are used to classify the reviews. These algorithms are listed below, but the reasoning behind the choice, the theoretical background and their historical performance are explained in the Analysis section:

- Logistic Regression
- Naïve Bayes Classifier
- Support Vector Machines
- Random Forest Classifier
- Neural Networks

Interpretation / Evaluation

Interpretation / evaluation is done using statistical approach once the machine learning results are obtained. The terms used to evaluate whether an algorithm performs well or not are accuracy, precision, recall, and F1-Score (Haque, et al., 2019). In addition to this, data visualization (different plots and charts using different tools), confusion matrices and summary tables containing results are also used to interpret and evaluate results. The overall process, as well as all the terms, are outlined in more detail in the Analysis and Results sections.

4.0 Analysis

For the actual analysis, several things needed to be considered – the algorithms to be used, the features to be used for the algorithms to classify the data, the ratio of train / validation and test data, and the process to be employed. All of these are explained in more detail below. Note that the in-depth explanations of algorithms and their performance review from previous researches are included in this section, instead of the Introduction. This was done as it was necessary to explain the full theoretical background behind sentiment analysis, pre-processing and transformation and provide detailed information on the dataset structure and its size, in order to be able to offer a fully feasible explanation on the decision-making process used for the choice of algorithms.

Choice of Machine Learning Algorithm

The first decision made regarding the analysis process was compiling a list of suitable machine learning algorithms. As the goal is to classify the reviews according to their ranking and sentiment, the choice had to be made between the classification algorithms. While exploring previous works on similar topics, four algorithms seemed to be the ones used the most often, with satisfactory results (Haque, et al., Huda, et al., Krishna, et al., Zahoor, et al.). Neural Networks were included as the fifth option primarily because of the sheer volume of the data and because they generally perform better than the traditional algorithms when the datasets are large and the number of features high, especially with the great increase in computers' processing power in the last couple of decades (Géron, 2019). These advances are best observed on one of the recent projects, where the researchers worked on the ImageNet dataset, containing close to 1.5 million images, and managed to get the classification accuracy of 91% (Yu, et al., 2022).

Logistic Regression

Logistic Regression is one of the algorithms most widely used in similar classifications. Its default use is for binary classification. In these cases, it uses discrete values and maps the function of any real value into 0 and 1 (Zahoor, et al., 2020). It usually does not default to 0 or 1, but it relies on the estimated probability – if probability of belonging to a certain class is greater than 50%, then it assumes that the entry (document) belongs to class 1. If this is not the case, it labels the item as class 0 (Géron, 2019).

However, Logistic Regression can also be used for multiclass classification. In that case, each class can be compared directly to all the other classes at the same time to determine if it belongs to that class or not – this is the so-called one-versus-rest approach. Another approach, called one-versus-one, compares the class against each of the other classes separately. So, while in the first case Logistic Regression would try to label the review as 1-star or any other class, the second approach would separately try to classify review as 1-star or 2-star, then 1-star or 3-star, and so on. Scikit-Learn implements the multiclass version of the classifier by default in cases when it automatically detects that there are more than 2 classes. However, to ensure the algorithm employs multinomial classification, this can be manually set by defining the parameter `multi_class` to be `multinomial` (Géron, 2019).

Logistic Regression is a popular choice as the training process is faster than in most algorithms. Also, it scales well to large datasets (Muller & Guido, 2017), which is important as this project uses between 50 and 100 times more examples than similar projects. In the researches reviewed, its accuracy ranged between 64%, when classifying reviews as good, bad or excellent (Haque, et al., 2019), and 94% (Huda, et al., 2019), when performing binary classification.

Naïve Bayes

Naïve Bayes range of classifiers is somewhat similar to the linear models, like Logistic Regression. The multinomial version of this algorithm, used in this project, is most commonly used is text classification, specifically using bag of words or TfIdf approaches (Albon, 2018), and some version of it has been used by all the researches consulted. The algorithm works in a way that it takes in the data counts, for example, the number of times a word appears in a sentence, and then uses the average value of each feature for each class. Finally, it classifies an entry by comparing it to the statistics of each of the classes. Once it finds the best match, it assigns that label to the entry (Muller & Guido, 2017).

Many researchers use Naïve Bayes as the first algorithm to train their data as it is very fast, works well with large datasets, it is easily interpretable and does not have many parameters that need to be tuned. This ensures quick results that can indicate where to go next, in terms of algorithm selection (VanderPlas, 2017). The main problem with Naïve Bayes is that the performance of models built using it is often lower than those of other algorithms used for sentiment analysis (Muller & Guido, 2017). When it comes to the researches

consulted, the algorithm proved to be fairly dependable for binary classification as it achieved above 80% accuracy in all of the resources reviewed, peaking at 92.75% (Krishna, et al., 2019).

Support Vector Machines

This algorithm is mostly referred to as SVM. It is well-known for its “fast and dependable classification which resolves two-group classification problems” (Zahoor, et al., 2020). Similar to Logistic Regression, it does indeed default to two categories, but it can also be used for multiclass classification problems. SVM algorithms work well for small and medium-sized datasets (Géron, 2019). In this case, as the dataset is quite large, the recommended version of the algorithm to be used is LinearSVC, as the basic implementation of the algorithm does not scale well if the sample size is over 10 000 (scikit-learn developers, 2022a).

This algorithm is a very popular one as it is fast at prediction once it has been trained. In addition to this, it is memory-efficient and works well with larger datasets (VanderPlas, 2017). Furthermore, it performed extremely well in previous researches. Out of the four main papers reviewed, it had the best results in three of them, with its accuracy peaking at 95.23% when performing binary classification (Krishna, et al., 2019). However, its accuracy level did drop to 75.58% when the reviews were classified in three categories (Haque, et al., 2019).

Random Forest

Random Forest belongs to the so-called Ensemble methods. These methods use several different algorithms to classify data, based on the majority vote (Géron, 2019). In case of Random Forest, it is an expanded version of the Decision Tree algorithm. On its own, the Decision Tree algorithm “applies a series of questions and conditions to formulate the tree-like structure where the leaf nodes will result in required classes” (Krishna, et al., 2019). This basically means that this tree is a flowchart consisting of the root node, which is split based on a certain condition. Each resulting node is then split further based on other conditions. This process ends once the leaf node is reached – the one that finally provides the label and classifies the document (Bird, et al., 2009).

Random Forest classifier forms a specified number of decision trees and integrates them in order to obtain more accurate classifications (Zahoor, et al., 2020). It is named after the way it operates – each tree is grown using different splitting criteria and different features, so this results in diverse trees, which are obtained completely randomly. By using that approach, the model obtains different perspectives on the possible class of a document and the final decision is made by aggregating the results of every tree built. For that reason, it outperforms regular Decision Trees (Géron, 2019). This algorithm is also often used for sentiment analysis and it was the best performing one, with its accuracy for binary classification at 95%, in one of the previous papers (Zahoor, et al., 2020).

Neural Networks

As mentioned previously, this approach has not really been used in previous research on the topic, but it seemed like it would be a good idea to include it as it has the potential to perform well with large datasets and high-dimensional data. Neural Networks (also called Artificial Neural Networks or ANNs) stand for a machine learning approach which tries to emulate the network of biological neurons found in human brain. This approach is considered to be very versatile, powerful, and scalable, which makes it ideal for a dataset with more than 200 000 textual reviews (Géron, 2019).

The functioning of this algorithm is based on MLPs (multilayer perceptrons). A typical network built using MLPs consists of one input layer, one or more hidden layers (middle layers) and a final layer - the output layer (Géron, 2019). In case of multiclass classification tasks, MLPs perform a series of calculations between every layer of the network in order to assign the correct label. The main problem with this approach is that the model, especially with larger datasets, can take a long time to train and tune. However, sometimes even a simple network can yield pretty good results (Muller & Guido, 2017).

Train / validation / test split

The only way of being certain that a model can perform well on new data is to evaluate it on the test set (Muller & Guido, 2017). In this case, the decision was made to split the data into three sets. The first set (training set / train set) is the one where all the iterative work is done – changes made to pre-processing, hyperparameters, use of algorithm. For training purposes, 80% of the dataset was chosen – a total of 176 165 reviews. These reviews were saved into a separate file (train_reviews.csv) for easier access. The changes made to the model are verified using the validation set, which consists of 10% of the data from the original dataset (22 021 reviews). This part of the dataset is primarily used to verify if the changes made to the model were effective or not. That dataset is constantly reused and was also saved in a separate file for easier access (validate_reviews.csv). Finally, as it is important for the final testing to be on a set of data never before encountered by the model, the safest way to do that was to separate a further 10% of the dataset and put it aside until the models are finely tuned. This way, the models can be trained and validated safely until the best configuration is found. The total of 22 021 test reviews were saved in a file named test_reviews.csv.

As for the previous researches, the splits between training and validation sets differ. Some use 80:20 ratio (Krishna, et al., 2019), some 70:30 (Huda, et al., 2019), some 90:10 (Haque, et al., 2019) and some experiment with ranges from 70:30 to 80:20 (Zahoor, et al., 2020). The split used for the train / validation / test sets is 80:10:10. That way, the data can be trained, validated and tested, each time using more reviews than the previous researches used for the full set combined. The large training dataset was primarily selected due to the fact that the classification is done based on 5 or 3 categories, while all the previous papers classified the reviews only as positive / negative, only seldom trying to detect the neutral ones. This provides models with plenty of examples they can learn from.

Features

In order to classify the review rating, only textual reviews are used as the independent variable. Even though some of the datasets contain other information that could be helpful in better classifying the reviews (type of cuisine and food, service, and ambience ratings), the goal is for the model to be trained to categorize the review solely based on the actual text of the review. The target variables are the rating and the sentiment columns. The rating column was already ready for machine

learning as it contained integer values in the range from 1 to 5, while the sentiment column was feature engineered, as previously explained. When the training set is considered, it is visible that both the rating (Figure 11) and sentiment distributions (Figure 12) closely mirror the ones seen in the overall set, with higher ratings and positive reviews dominating the set.

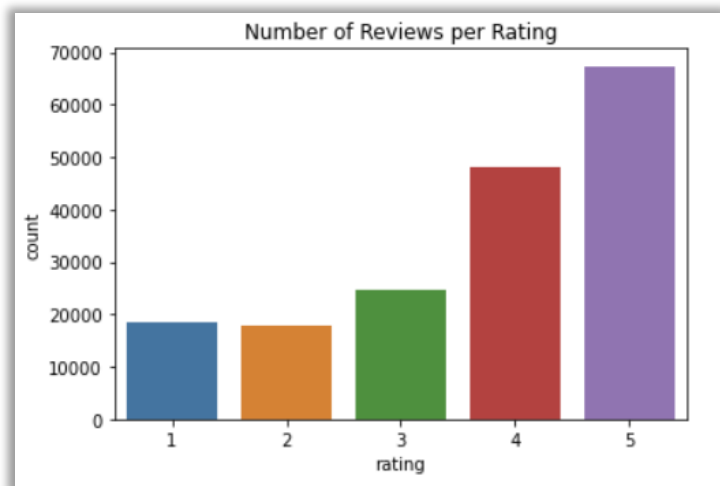


Figure 11 - Train set ratings distribution

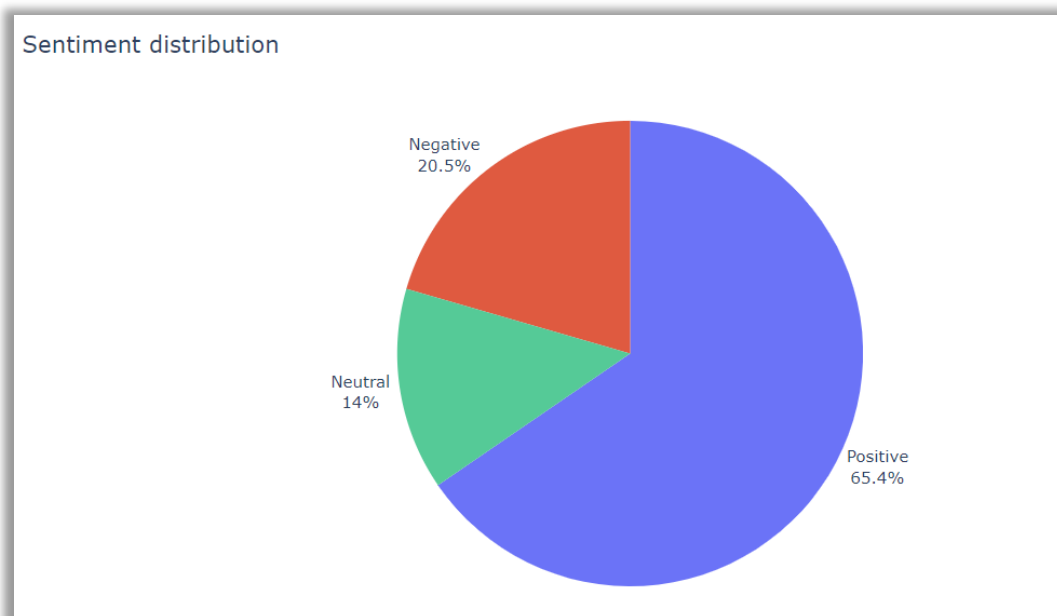


Figure 12 - Test set sentiment distribution

However, the review column, where the actual textual reviews are, needed to be converted into a format that can be used by machine learning algorithms. This was done by cleaning the review text first, as explained in the Methodology section, followed by vectorizing the review text using two different methods, explained below.

Count Vectorizer

Count vector is “a data set matrix notation in which each row represents a corpus document, each column represents a corpus term, and each cell represents the frequency count of a particular term in a particular document” (Huda, et al., 2019). This approach is based on the so-called bag of words model, where the words are represented based on their counts, while completely ignoring grammar and word order. Using this approach, all the words from all the reviews are collected into one list. After that, each of the reviews is considered as a separate document where the words absent from that particular review are recorded as 0 and those present are recorded as the number of times they appear in the review. The words with the lowest counts are removed from the bag of words as they are not used often so their potential influence on the overall rating is very low (Krishna, et al., 2019).

Scikit-Learn’s implementation is called CountVectorizer() and it stores the data in a SciPy sparse matrix, which omits items that are recorded as 0. Due to fact that many of the words do not appear in most of the reviews, omitting those entries prevents the waste of memory, while also not compromising the model-build process as those entries would not significantly contribute to the classification (Muller & Guido, 2017). However, as this vectorizer only relies on raw word counts, this results in overreliance on words that appear very frequently, but do not add much to the meaning. For that reason, this approach sometimes does not work well (VanderPlas, 2017).

TF-IDF Vectorizer

This vectorizer is the result of term frequency (TF) and inverse document frequency (IDF). Term frequency is calculated by dividing the number of times a term occurs in a document (one review) by the total number of terms in a document (in that same review). Inverse document frequency shows the importance of a given word across all the documents (all the individual reviews combined). It can be derived by calculating the logarithm of the quotient of the number of documents and the number of documents containing the term queried (Krishna, et al., 2019). The formula for this vectorizer can be seen in Figure 13.

$$\text{tfidf}(w, d) = \text{tf} \log \left(\frac{N + 1}{N_w + 1} \right) + 1$$

Figure 13 - TF-IDF formula

In this formula, N is the number of documents in the dataset, while N_w is the total number of documents in the dataset that the word (w) appears in. tf (the term frequency) stands for the number of times the word (w) appears in the document (d) (Muller & Guido, 2017).

This vectorizer is implemented as `TfidfVectorizer()` in Scikit-Learn and it is also saved as a SciPy sparse matrix. Its implementation is mostly preferred over `CountVectorizer()` as it generally performs better with most algorithms.

N-grams

Both of these vectorizers can be set up to generate data that takes different token lengths as input, also called n-grams. This means that, in order to properly test a model, it needs to be tested using different n-gram lengths as different models will react differently to the input length, as well as the pre-processing done. Most commonly used options are unigrams (one-word combinations), bigrams (two-word combinations) and trigrams (three-word combinations) (Bird, et al., 2009).

Most often occurring n-grams can be extracted using frequency distribution, which calculates the frequency of each vocabulary item in the corpus. NLTK library offers a very streamlined way to achieve this, using the `FreqDist()` method (Bird, et al., 2009).

When the top 20 unigrams of the training set are explored (Figure 14), it is visible that some of the words do reflect the important items that might constitute a positive or a negative review. However, there are some words which, at first glance, do not seem like they would impact the review a lot – these not only include words like *would*, *back*, *get* but also words like *food*, *place* and *service*. For review analysis, it seems like the words *great*, *good* and *like* are the most important ones.

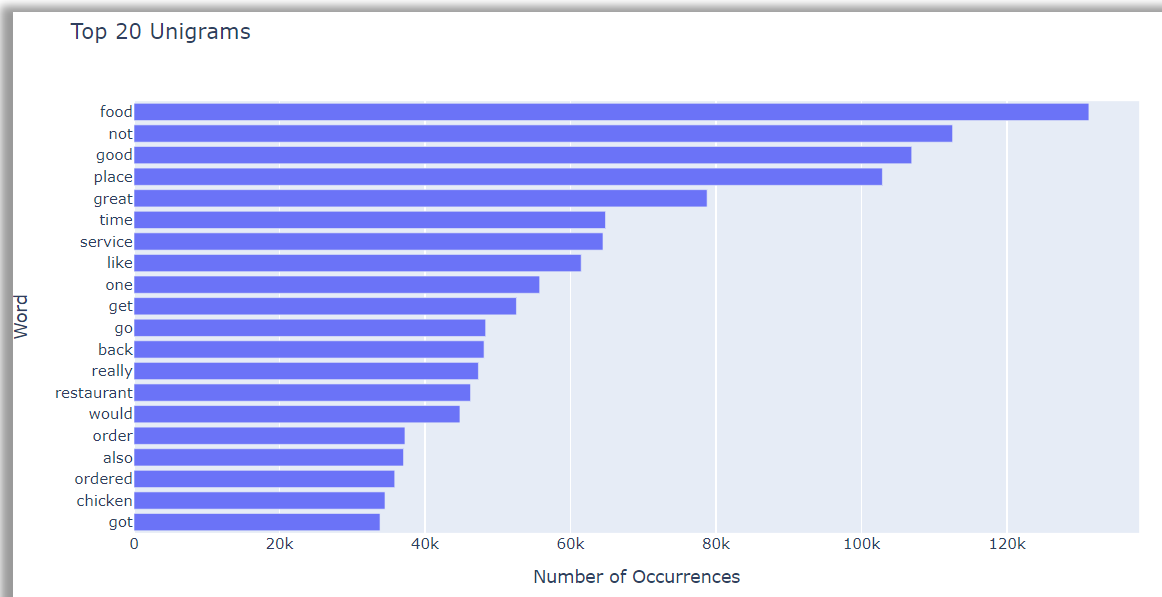


Figure 14 – Most common unigrams in the training set

After reviewing bigrams (Figure 15), some of these words that could have easily been discarded when only unigrams are considered, show how important they are. Food now features in combinations like great food, good food and food great, while place now appears in great place and love place. This just shows how much more is gained when bigrams are also taken into account.

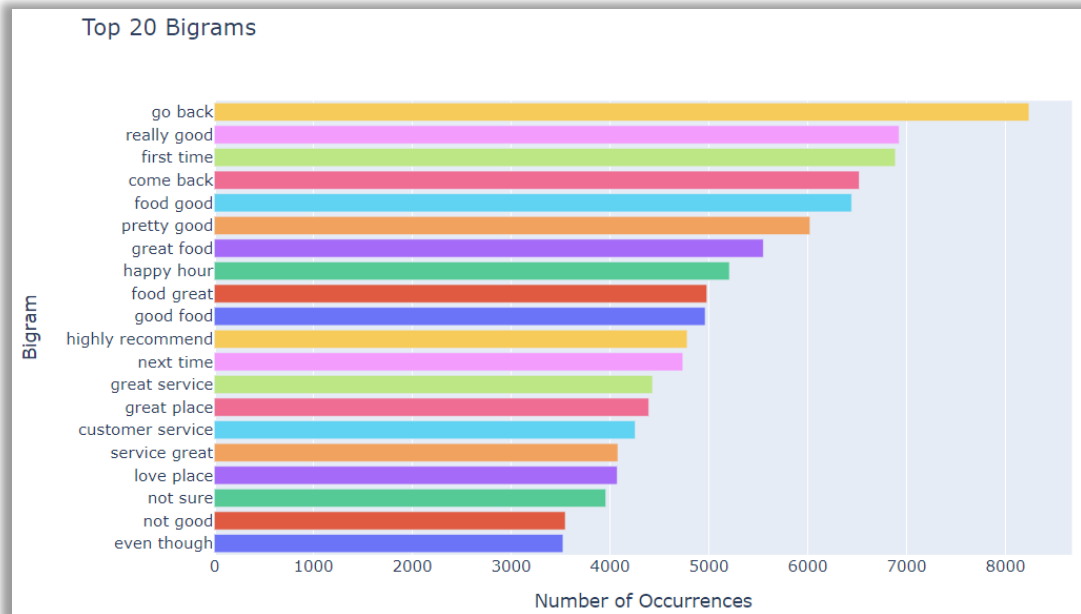


Figure 15 - Most common bigrams in the training set

Finally, trigrams in Figure 16 uncover even more information. Combinations like would highly recommend and would definitely recommend further consolidate the necessity of using different input lengths for machine learning. In addition to this, words like back gain extra meaning here in expressions like would go back and definitely come back.

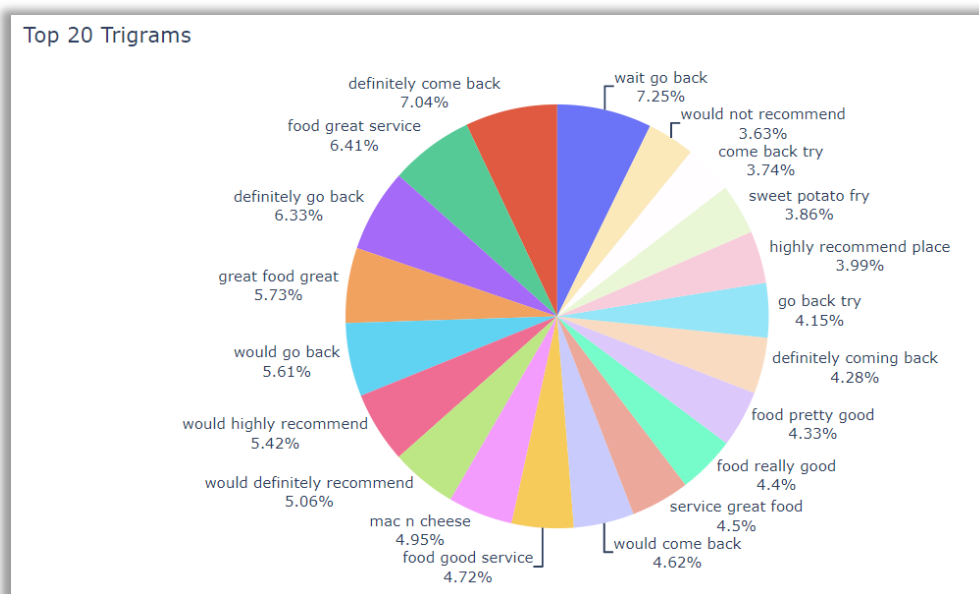


Figure 16 - Most common trigrams in the training set

Process

Due to the complex nature of the data itself, the memory and processing power needed to run the algorithms, and the overall number of different algorithms, the training / validation / testing process was split into several different stages in order to streamline it as much as possible. All the stages are explained below.

Different n-gram counts and number of features

As some researchers say, it is helpful to start with the so-called kitchen sink approach, where all the features are used in order to reduce the number of features to the ones that are actually important for model-building (Bird, et al., 2009). A different approach is taken here. Due to the large amount of reviews, the total number of features (words or expressions) extracted by both vectorizers is immense. For that reason, some initial testing was done, which confirmed that, for most models, performance plateaus when 3000 features are used. For that reason, instead of using tens of thousands of features to train the model, the parameter `max_features` is set when using both Count Vectorizer and Tfidf Vectorizer. This parameter helps to limit the number of most important features used to build the model. All the models are tested using 6 different configurations of maximum features – 500, 1000, 1500, 2000, 2500 and 3000.

In order to get the best performing models, another vectorizer parameter is used to create the feature set used by the machine learning algorithms – `ngram_range`. This parameter allows for different sets of n-grams to be defined when building the features (Muller & Guido, 2017). For the purpose of this stage, 6 different variations were used:

- unigrams only
- unigrams and bigrams
- unigrams, bigrams and trigrams
- bigrams only
- bigrams and trigrams
- trigrams only

In this stage, the main metric used to evaluate the models' performance is accuracy. This metric displays the overall percentage of reviews that were correctly classified. Basically, it is calculated by dividing the number of correctly classified reviews by the number of total reviews (Bird, et al., 2009). As with both use cases, ratings and sentiment classifications, there is no preference on which class is more important to predict so this metric is simple and compact enough to eliminate any models that are not performing well.

For the purpose of this stage, a function was written which uses either Count Vectorizer or Tfidf Vectorizer to build the feature set and then train the selected model using all the combinations of n-grams and feature sizes outlined previously. By the end of this stage, only one n-gram range / feature size combination of each vectorizer for each of the machine

learning algorithms is kept for further tuning. These combinations are selected based on the accuracy levels obtained on the validation set.

Note that all models are built using the Scikit-Learn's default settings, except for:

- Logistic Regression
 - multi_class hyperparameter is set to multinomial due to the nature of the classification task
 - max_iter hyperparameter is set to 1000 as the default number of iterations (100) is not enough for the model to classify the reviews
- Support Vector Machines
 - dual hyperparameter for the LinearSVC is set to False as the number of reviews is higher than the number of features
- Neural Network
 - Sequential is the model used to build the network as it is simple to manage – it consists of a single stack of layers sequentially connected
 - Input layer - dense layer with relu activation and 10 units – units signify the number of neurons used, dense means that every neuron in the current layer is connected to every neuron in adjacent layers, activation functions are aimed at introducing nonlinearity between the layers - relu stands for Rectified Linear Unit function and it is used due to its very fast computational time, the input_shape parameter is set to the number of features for each observation – this would range from 500 to 3000 (Albon, 2018)
 - Hidden layer – middle layer, dense, activation is also set to relu, 10 units used
 - Output layer – dense, 6 units for ratings classification, 3 for sentiment, activation function used is softmax as that is the one used for multiclass classification
 - When compiling, loss function is set to the one required for multiclass classification (sparse_categorical_crossentropy), optimizer is set to fast and efficient adam, the number of epochs to 200 (number of iterations over the dataset) and the batch_size to 128 (number of documents taken in at once) (Géron, 2019)

Hyperparameter tuning on smaller samples

In order to get better results, one of the most common approaches is further tuning of hyperparameters as the default hyperparameters do not always provide the best performance, especially when more complex algorithms like Random Forest or Neural Networks are considered. This step is not only performed in order to get better results, but also to avoid overfitting of the model – when model performs well on training and validation sets, but does not perform well on the data that was never encountered before (test set) (Géron, 2019).

The approach taken here is one of the most popular approaches used to tackle the problem – a combination of the so-called grid search and cross-validation. Grid search (implemented in Scikit-Learn as GridSearchCV) is done by compiling a dictionary consisting of hyperparameters and the values that are most likely to improve the performance of the model. Once that data is compiled, grid search searches through all the possible combinations of the hyperparameters defined in order to find the best option. As an extension of its function, it also uses cross-validation, evaluation of the same set of hyperparameters on several different versions of the dataset. This further reinforces whether the hyperparameter combination is a good fit or not as it is run on several different versions of training and validation data (Bird, et al., 2009).

As the overall size of the dataset did not allow for the efficient hyper-parametrization to be run, due to the processing power required, a smaller subset of the training set (15 000 reviews) was used for grid search. The cross-validation was set to 3, which means that the subset was split in training and validations sets in three different ways, ensuring that the resulting set of hyperparameters is the one that has the lowest possibility of overfitting. The main metric used was accuracy, same as in the previous stage. Note that due to the complexity of the hyper- parametrization of Neural Networks, this approach was not included at this stage and its hyperparameters are only manually tuned.

Logistic Regression Hyperparameters

Logistic Regression already had two hyperparameters set by default (max_iter and multi_class). Except for these two, the two most important hyperparameters to set were solver and C. With solver, it is a choice between 5 different algorithmic solutions used to classify the data using Logistic Regression. One of them (liblinear) is not suited for multiclass classification and two of them were not performing at all, even on smaller sets (sag and saga). This meant that the two possible solvers can be newton-cg and lbfgs. The choice of these two solvers also impacted the choice of the penalty hyperparameter, which is directly linked to the solver chosen so the l2 and none were used (scikit-learn developers, 2022b).

Finally, another important hyperparameter was C, which determines the strength of the regularization. Its higher values mean less regularization, placing more importance on classifying each review correctly. Low values, on the other hand, will force the algorithm to try and fit the majority of reviews correctly (Muller & Guido, 2017). In this case, to account for those differences, the values of C were set to 0.6, 0.7, 0.8, 1, 5 and 10.

Naïve Bayes

Multinomial Naïve Bayes only has a single parameter that can be tuned – alpha. This hyperparameter controls model complexity, in a similar way C does for Logistic Regression. When changing the alpha value, the algorithm adds virtual data points that have positive values for all the features. The result of this process is smoothing of the results. High alpha values result in more smoothing and models of smaller complexity and vice versa (Muller &

Guido, 2017). The alpha values used for hyper-parametrization are 0.01, 0.1, 0.6, 0.8, 1, 3, 5, 10, 20, 50, 100.

Support Vector Machines

Similar to Logistic Regression, SVM's result depend on the value set for C so its values need to be carefully chosen (VanderPlas, 2017). In addition to this, several other hyperparameters can be tested – penalty (the type of normalization used for penalization), loss (the type of loss function) and class_weight. With the first two, all the available options were used for tuning (penalty as l1 and l2, loss as hinge and squared_hinge), while the last one seemed particularly important as the number of classes is not balanced, for the sentiment or the rating column. That option was used with two settings – balanced and None. The balanced option is supposed to help in dealing with unbalanced datasets (scikit-learn developers, 2022a). Note that the default hyperparameter dual is still kept as False.

Random Forest

In terms of the grid search, this algorithm is the most demanding when it comes to the number of parameters that need to be adjusted and the processing power required. The most important parameter to adjust was max_features as it limits the number of features every tree examines. High numbers of features (close to the actual number of features used) reduce the randomness of the forest, which in turn eliminates its main advantage over the regular Decision Tree algorithm. Another important parameter is max_depth and this denotes how deep each of the trees is explored. Usually, with high number of features, low depth number is required to keep the randomness intact. The same way, in case of lower number of features, deeper trees are recommended as this allows them to get different results. max_depth was tested on values 70, 90 and 100, while max_features used were 4, 8, 12.

In addition to this, other important parameters include n_estimators (number of trees used by the algorithm – set to 150 and 200), bootstrap (set to True, used in combination with random_state to control the randomness of the tree), n_jobs (set to -1, this enables all processing power to be used and speeds up the training process), min_samples_leaf (minimum samples needed to constitute a leaf node - set to 2 and 4), min_samples_split (minimum samples to split a node – set to 16 and 20), random_state (controls the randomness of the trees – set to 42) and class_weight (works in the same way as for SVM) (Muller & Guido, 2017).

Using hyperparameters obtained on full training / validation sets

Once the previous stage is completed, the hyperparameters obtained are used on the full training and validation datasets in order to further eliminate any models that are underperforming. The main metric considered is still accuracy, but other metrics are also calculated. Those metrics are based on the following terms:

- True positives (TP) - relevant items that are correctly identified as relevant
- True negatives (TN) - irrelevant items that are correctly identified as irrelevant
- False positives (FP) - irrelevant items that are incorrectly identified as relevant
- False negatives (FN) - relevant items that are incorrectly identified as irrelevant

Based on these terms, the following metrics can be calculated:

- Precision – shows how many of the identified items are relevant, formula

$$TP / (TP+FP)$$

- Recall – shows how many relevant items are identified, formula

$$TP / (TP+FN)$$

- F-Score – harmonic mean of precision and recall, formula

$$(2 \times \text{Precision} \times \text{Recall}) / (\text{Precision} + \text{Recall})$$

Finally, for multiclass classification tasks, it is also a good idea to build a confusion matrix. Confusion matrix is a table where each cell shows how many predictions were correctly made per label. The diagonal entries, going from upper left to bottom right, show the number of items that were correctly classified. All the others show those incorrectly classified (Bird, et al., 2009). In this stage, in addition to accuracy, more focus is put on precision, in order to verify the correctly predicted items per class, and confusion matrices as they show where most errors are made in terms of classification.

Manual hyperparameter tuning on best performing models

Once the best performing models are singled out, some hyperparameters can be manually adjusted, especially when certain trends are noticed either in confusion matrices or during the model tuning. At this stage, VotingClassifier, a classifier that works as an ensemble of the best models found can be tested (Géron, 2019). In addition to this, further experimentation can be done using the feature selection. In this case, only features that appear in more than a designated number of reviews can be selected when the model is built. This can be done using the min_df parameter when vectorizing the data using either Count Vectorizer or the Tfidf Vectorizer (Muller & Guido, 2017).

When it comes to Neural Networks, the hyperparameters which are tweaked the most are the number of layers, the number of neurons (inputs) per layer, types of activation functions, batch sizes, the class weight, in case of skewed datasets, and the type of optimizer.

In terms of the number of layers, the minimum is three, as initially tested. However, increasing the number of hidden (middle) layers can sometimes significantly improve the results. The number of inputs per layer is usually set up to resemble a pyramid. While the input layer needs to have the number of layers that is at least the same as the overall number of classes available, the hidden layers would have higher values. The first hidden layer usually defaults to 300, the second to 200 and the third to 100. This basic setup can be used and then further changed based on the results obtained. Another approach is to use the same number of inputs in all hidden layers so that is the second approach taken.

As the training set is skewed, `class_weight` argument can also be tested to make sure that the model knows that the dataset is not fully balanced. As for the optimizer, in addition to adam, SGD (Stochastic Gradient Descent) is also worth exploring as it traditionally performs well with Neural Networks. The batch size can also be explored as some researchers claim that the batch sizes under 32 perform well with Neural Networks. Finally in order to avoid overfitting, there are two important strategies to explore – EarlyStopping and Dropout. The first one observes the validation set while the training is undergoing. It is then used to interrupt the training process if there is no progress observed on the validation set for a pre-defined number of epochs. The second one, Dropout, is said to be able to increase model's accuracy by 1–2%. The rate set is usually between 10% and 50%. If the model is overfitting, the dropout rate can be further increased, but it can also be decreased if the model starts underfitting. In addition to this, the dropout rate should be increased for larger layers, but decreased for smaller ones. Finally, it can sometimes help to use Dropout only after the last hidden layer (Géron, 2019).

Using different vocabulary sets to validate model performance

At this stage, the same models are run on validation set using different vocabulary sets. The reasoning for this is the fact that both vectorizers extract the most important features across the dataset. This approach usually favours unigrams, which might not always convey the clearest message on the review sentiment. Furthermore, as the dataset is skewed towards positive reviews, this also means that the models are more likely to correctly predict positive and 5-star reviews than any other category. By extracting the most common unigrams, bigrams and trigrams from each of the rating groups and using them to build a more balanced vocabulary might lead to better classification. As noted in the overview of the most popular bigrams and trigrams, only one item in both groups is negative (not good and would not recommend), with everything else highly positive or slightly neutral. When the comparison between the unigrams, bigrams and trigrams in 1-star reviews and 5-star reviews is made (Figures 17-22), there is noticeable difference.

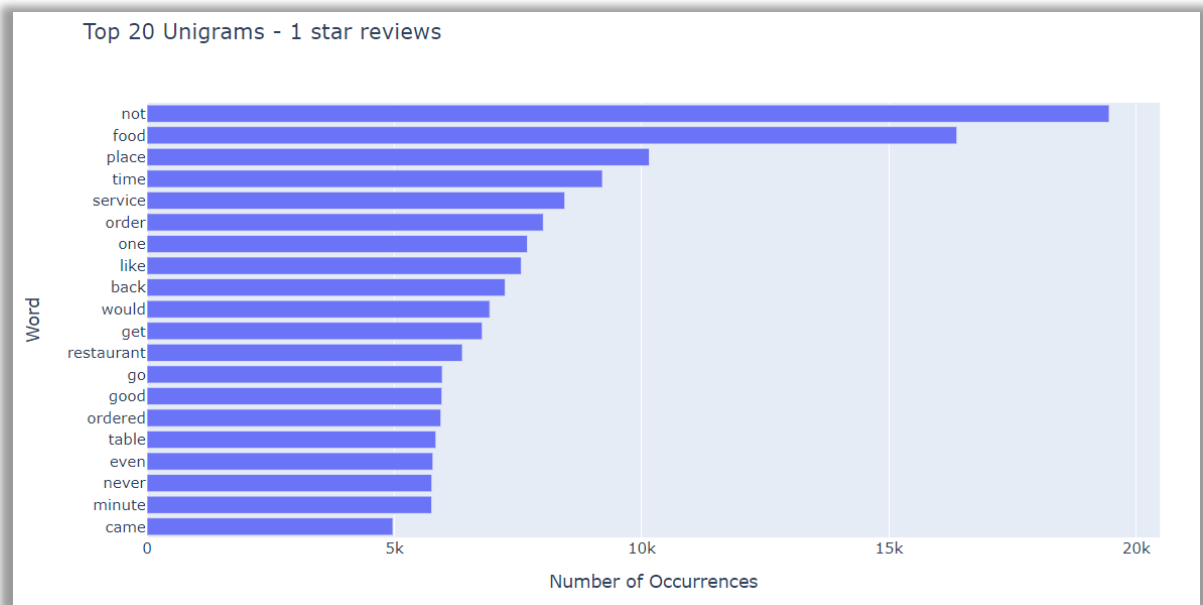


Figure 17 - 1-star unigrams

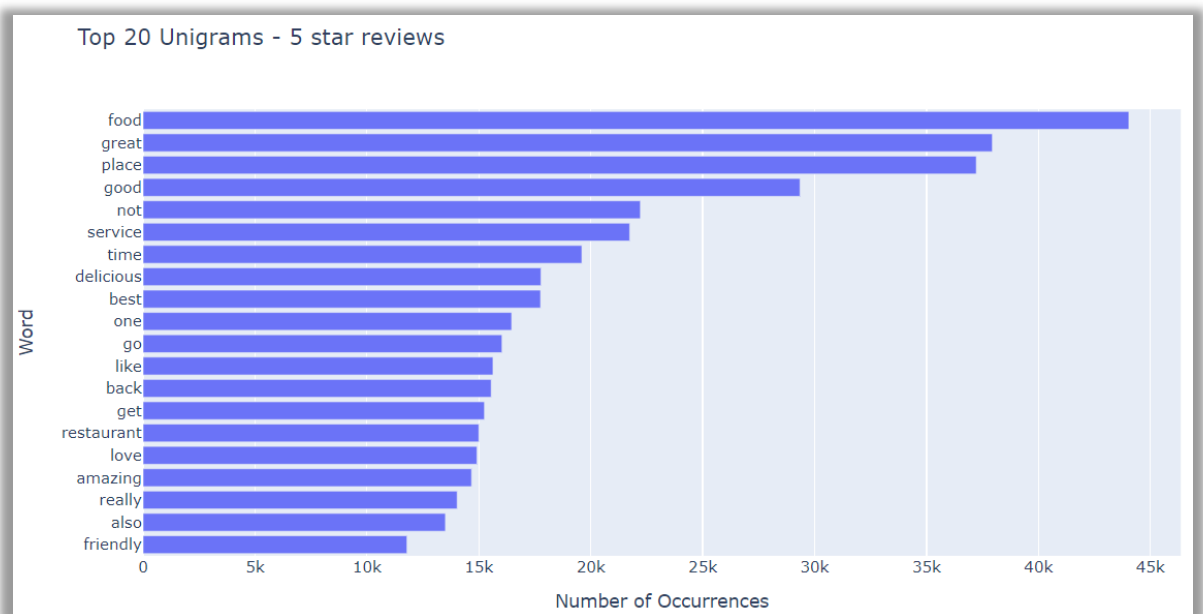


Figure 18 - 5-star unigrams

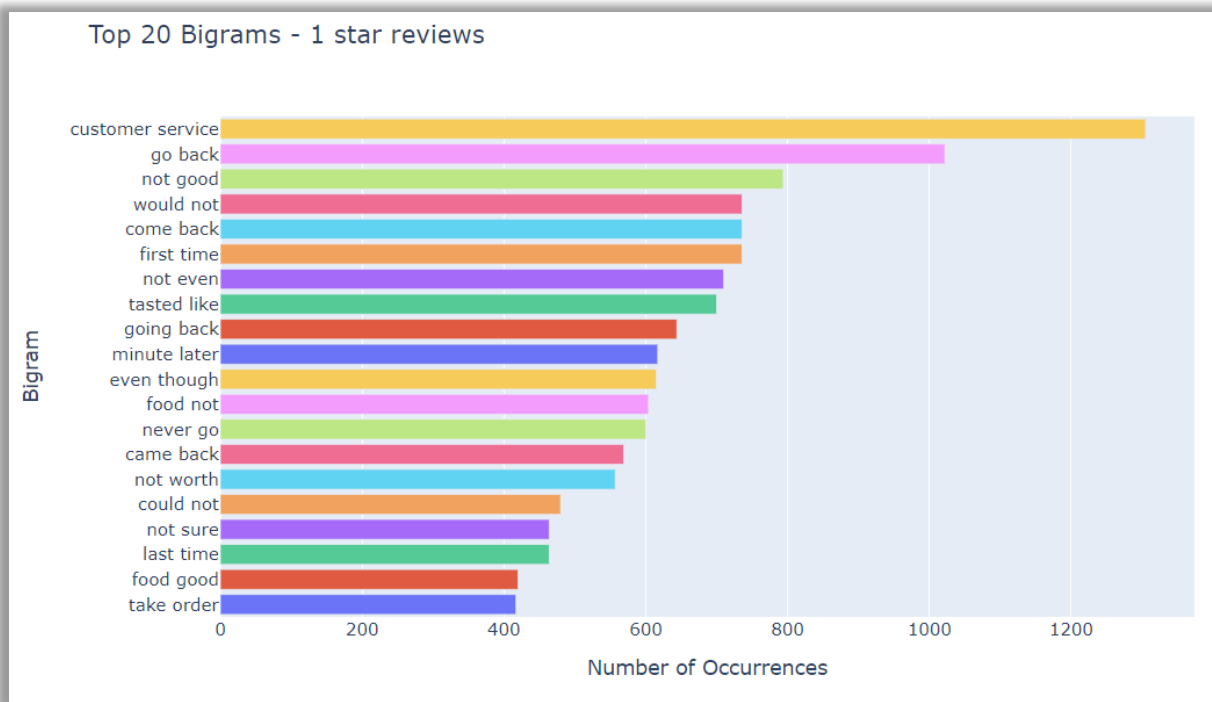


Figure 19 - 1-star bigrams

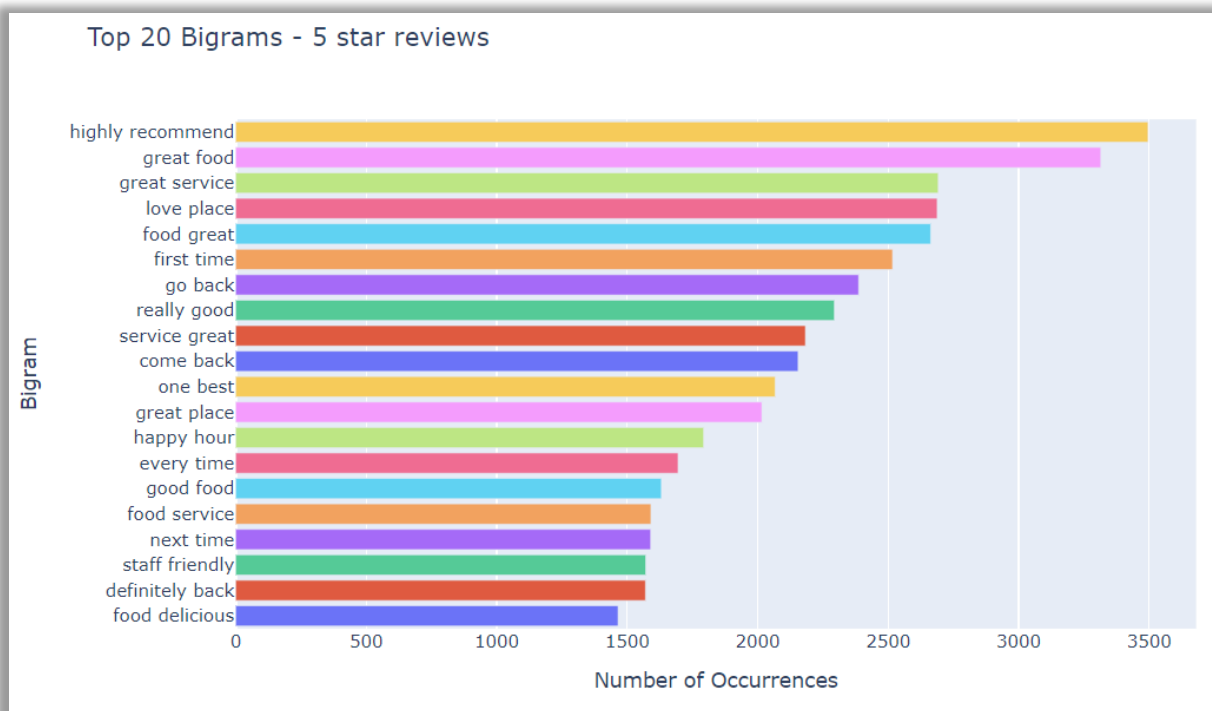


Figure 20 - 5-star bigrams

Top 20 Trigrams - 1 star reviews

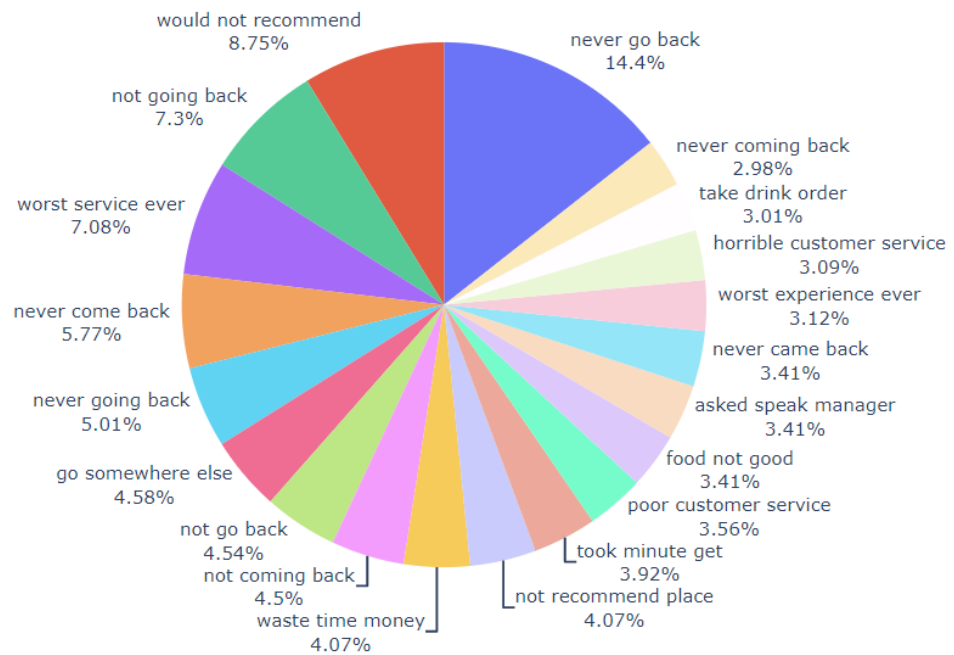


Figure 21 - 1-star trigrams

Top 20 Trigrams - 5 star reviews

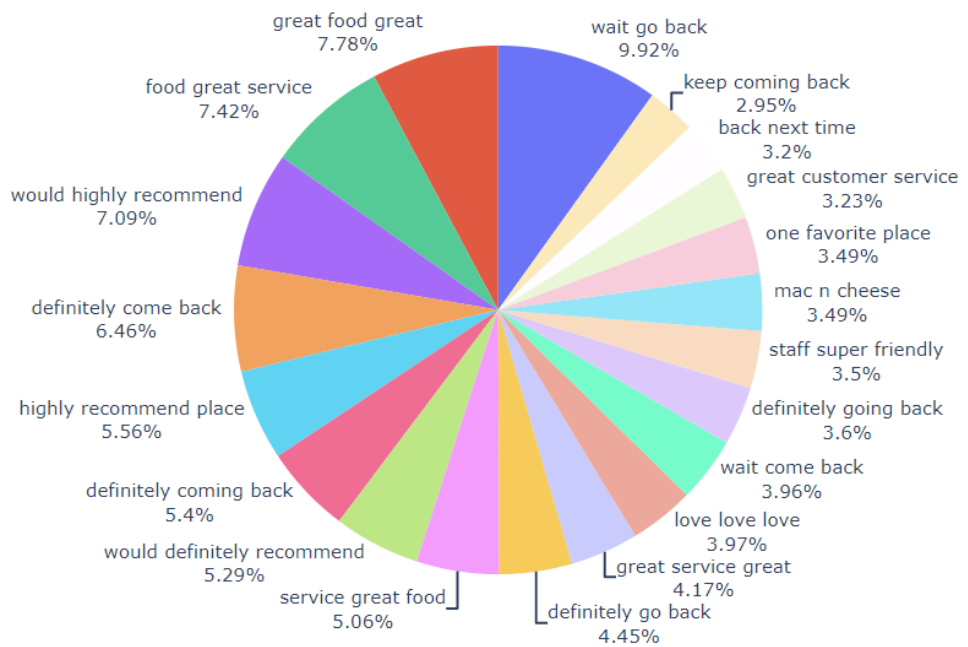


Figure 22 - 5-star trigrams

When observing unigrams, it is evident that the difference is not that significant – it would be difficult to differentiate 1-star reviews from 5-star reviews. However, bigrams and trigrams are much different. 1-star bigrams are dominated by negation (not good, would not, not worth), while the 5-star bigrams are full of praise (highly recommend, great food, great service). Similarly, 1-star trigrams suggest that those people are not becoming returning customers (never go back, would not recommend, not going back). On the other hand, 5-star trigrams are dominated by extreme positivity (great food great, would highly recommend, definitely come back). For this reason, 5 different sets of vocabulary were created so that they can be used to test whether the models perform better using customized vocabulary. Each set contains the exact same number of bigrams, trigrams and unigrams per rating. That means that the first example in Table 6 has 4000 unigrams extracted, 800 for each of the categories, as well as 3000 bigrams and 3000 trigrams. Unique features is the overall count of features used per vocabulary set once the duplicates were removed.

Table 6 – Customized vocabulary sets

Name	Unigram count	Bigram count	Trigram count	Unique features
Voc1	800	600	600	4139
Voc2	600	800	600	4226
Voc3	600	600	800	4453
Voc4	800	700	500	4049
Voc5	500	700	800	5422

Using test set to verify the models' performance

Last stage of the process is using the highest performing models in terms of accuracy and precision on the test set to verify how they perform on new data, the data they never encountered before. Once collected, the results are presented using summary tables and compared to the results obtained using the train and validation sets.

5.0 Results

In this section, the overall results are presented for each individual stage of the training process. That way, a clear overview of the progress in the quality of results can be easily observed.

Stage 1 - Different n-gram counts and number of features

While training different models using different n-gram ranges with different lengths of input features, it transpired that all of the algorithms performed best when n-gram ranges that consisted of unigrams and bigrams (1, 2) or unigrams, bigrams and trigrams (1, 3) were used to build the models. Another trend that emerged was that the preferred vectorizer was the Tfidf Vectorizer, as it was outperformed by Count Vectorizer on only two occasions. The final major trend that could be observed was that the performance with most algorithms stopped significantly increasing after the number of features reached 1500. However, it did continue to slightly increase after that. The minor outlier was Random Forest, where the performance mostly plateaued around 2500, when ratings were considered. The difference between Random Forest and other algorithms can be best seen in Figures 23 and 24, where its performance is compared to the one of Logistic Regression. Note that Random Forest had performed really badly with n-gram ranges that did not include unigrams – accuracy was well under 40% and for that reason, those results have not been added to the graph.

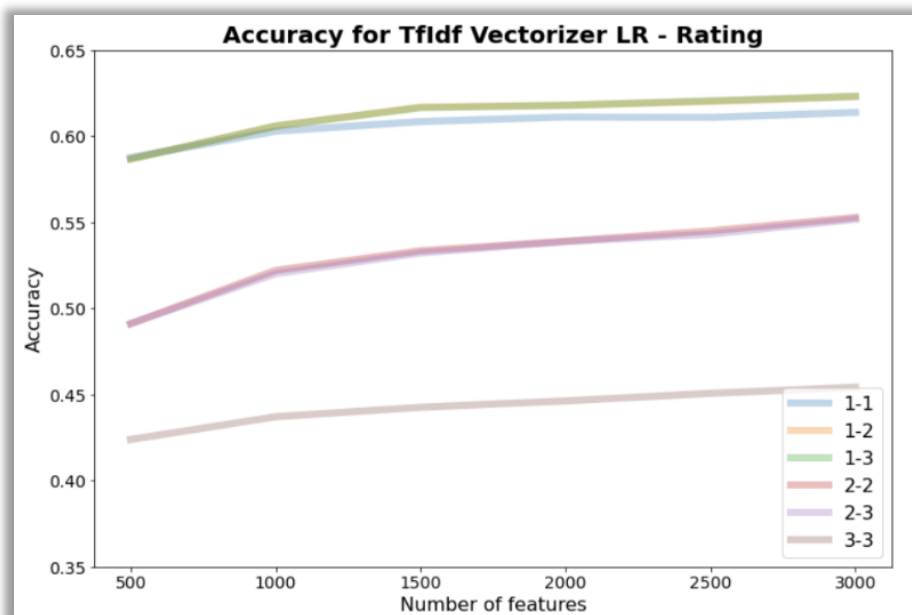


Figure 23 - Logistic Regression performance based on the number of features

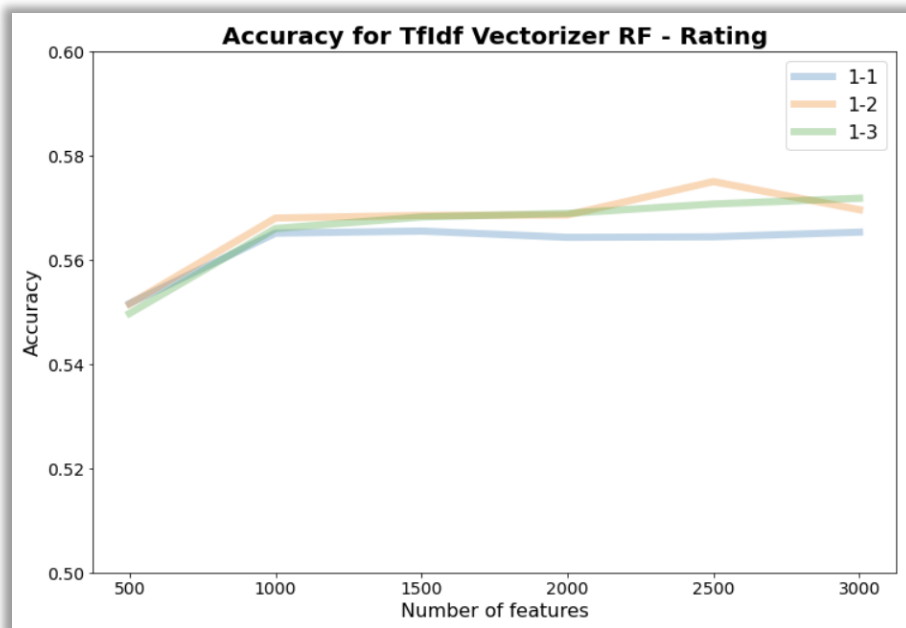


Figure 24 - Random Forest performance based on the number of features

However, there was an even bigger outlier when it came to the number of features needed to perform well – Neural Networks. In this case, the performance peaked when using 1000 or 1500 features and degraded from there. This can be seen in Figure 25.

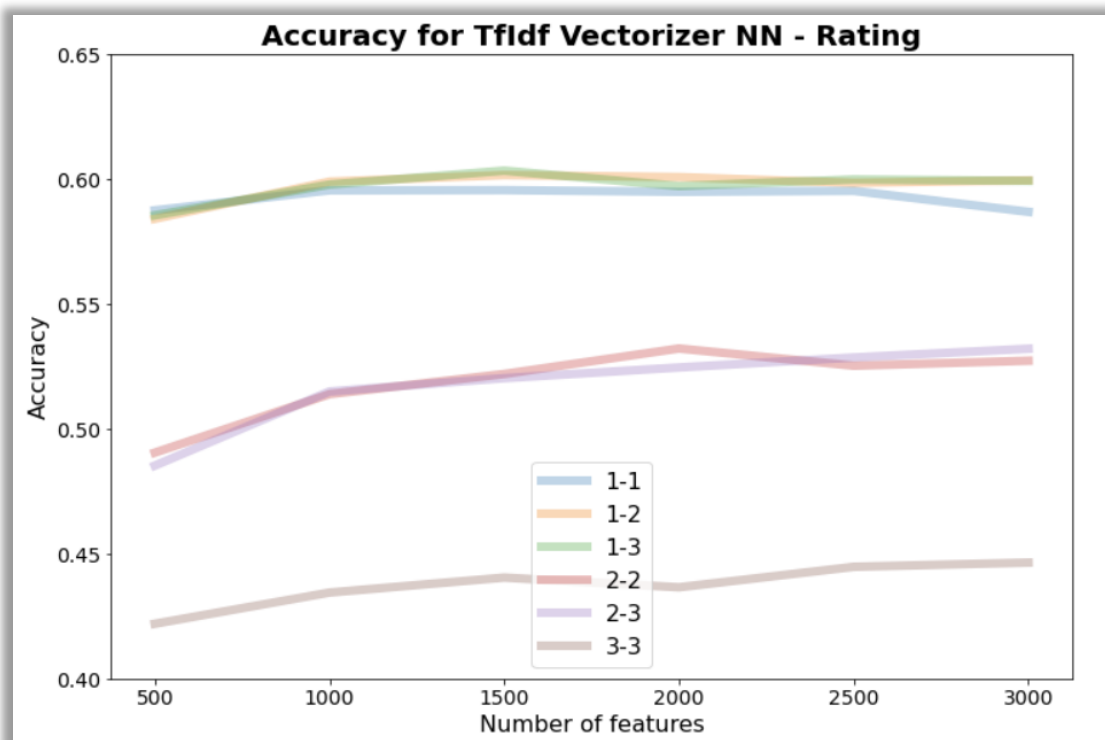


Figure 25 - Neural Networks performance based on the number of features

As can be seen from the graphs, as well as from the results in Table 7, the performance significantly degraded when n-gram ranges that did not contain unigrams were used to build models. For that reason, all those options were eliminated going forward. In addition to this, as most models performed the best when using the combination of Tfidf Vectorizer, 3000 features and n-gram range that consisted of unigrams, bigrams and trigrams, those combinations are explored again in the next phase, alongside the combinations that outperformed them. This does not apply to Neural Networks as they did not perform similarly to other algorithms in this stage. Finally, the best performing algorithm at this stage is Logistic Regression, both when classifying based on sentiment and ratings. The worst performance so far, when ratings are considered, belongs to Random Forest, almost 5% below the best performing option. In terms of sentiment, the lowest performer (Naïve Bayes) is a bit over 4% less accurate than Logistic Regression. This first stage shows that the performance of all algorithms when classifying reviews according to sentiment is quite good, with all of them able to correctly classify 4 out of 5 reviews. However, the ratings are proving to be a much more complex challenge.

Table 7 – Stage 1 results

Algorithm	Analysis Type	Vectorizer	Number of features	N-gram range	Highest Accuracy
Logistic Regression	Rating	CV	3000	1, 2	61.74
	Rating	TFIDF	3000	1, 3	62.32
	Sentiment	CV	3000	1, 3	83.94
	Sentiment	TFIDF	3000	1, 2	84.17
SVM	Rating	CV	3000	1, 2; 1, 3	60.45
	Rating	TFIDF	3000	1, 2	61.29
	Sentiment	CV	3000	1, 3	83.44
	Sentiment	TFIDF	3000	1, 2	84.05
Random Forest	Rating	CV	3000	1, 3	57.3
	Rating	TFIDF	2500	1, 2	57.5
	Sentiment	CV	3000	1, 2	80.56
	Sentiment	TFIDF	3000	1, 2	80.49

Naïve Bayes	Rating	CV	3000	1, 2	59.52
	Rating	TFIDF	3000	1, 3	58.08
	Sentiment	CV	3000	1, 2	79.47
	Sentiment	TFIDF	3000	1, 3	79.88
Neural Networks	Rating	CV	1500	1, 3	60.3
	Rating	TFIDF	1500	1, 3	60.33
	Sentiment	CV	1000	1, 2	81.37
	Sentiment	TFIDF	1500	1, 2	81.97

Stage 2 - Hyperparameter tuning on smaller samples

The performance of four algorithms improved during this stage. However, as all the resulting hyperparameters are run on the full train / validation dataset in the next stage, all the results are available in that section, in joint summary tables for each algorithm. As mentioned previously, hyperparameters for Neural Networks are adjusted manually in stage 4 of the process so that approach will not be discussed here.

When it comes to Logistic Regression, the default Scikit-Learn implementation seemed to perform well in general as the hyperparameters obtained mainly reflected the defaults. There was one major difference when it comes to vectorizers – Count Vectorizer was responding better to lower values of C (around 0.6), while Tfidf Vectorizer was performing better when C was set to the default value of 1. In addition to this, models built using Count Vectorizer outperformed those built using Tfidf Vectorizer in this stage, both in sentiment and rating classification.

All models built using LinearSVC returned the exact same parameters. As the best fit for the values of the most important parameter (C) was also the lowest one tried (0.6), this was an indication that an even lower value of C might result in better performance. Similar to the Logistic Regression results, Count Vectorizer performed better at this stage. This seems to indicate that this vectorizer performs better on smaller datasets, as it was completely outperformed by Tfidf Vectorizer in the previous stage, when the dataset was 10 times larger.

Random Forest is the algorithm where the highest number of hyperparameters needed to be tuned in order to get better performance. Due to high processing requirements, the parameter values were incrementally changed, especially the important ones (max_depth, max_features, min_samples_split) to get better performance. Finally, models generally performed well when the number of features was low, but the depth of each tree was higher. The main difference, compared to the previous two algorithms, was the fact that

Tfidf Vectorizer performed better here – the main reason was probably that higher complexity model also preferred higher complexity vectorizer.

Finally, MultinomialNB has only one hyperparameter that needs to be adjusted to get better results – alpha. Generally, the main difference was again between the two vectorizers. Count Vectorizer seemed to prefer models of lower complexity so the alpha value returned was always higher. With Tfidf, the preferred alpha value was low, signifying that this vectorizer works better with more complex models. Finally, there was no significant difference between the results obtained in this step compared to the previous step, unlike the other algorithms. With all the other algorithms, accuracy was significantly higher in this stage so this indicates that Naïve Bayes performs at a similar level no matter the dataset size.

Stage 3 - Using hyperparameters obtained on full training / validation sets

In this stage, the hyperparameters obtained in the previous stage were used on full training and validation sets to evaluate if there was any improvement. In addition to accuracy, precision was also observed in this stage. Finally, in case of all algorithms, only best performing combinations were kept for the next stage. Those advancing further are indicated in green.

Hyperparameters obtained for Logistic Regression improved the performance of the model using Tfidf Vectorizer to classify the reviews according to ratings slightly compared to stage 1. All the other combinations showed no improvement, while the performance of some also deteriorated. The accuracy results were generally lower than they were when performing grid search on a smaller sample, especially for models using Count Vectorizer. When looking at precision, all models seem to perform well when classifying the extremes – negative and positive sentiment or 1 and 5 stars. This is not related to the number of reviews available per rating / sentiment as the number of 1-star reviews is lower than the number of both 2-star and 3-star reviews, but its precision is still similar to the one obtained for 5-star reviews. This just shows that the models struggle to identify those reviews which are not full of praise or highly critical. Finally, models that are further explored in the next stage were all built using Tfidf Vectorizer. This further reinforces that Count Vectorizer might be more suitable for smaller datasets and simpler models. The full overview of the results can be seen in Table 8.

Table 8 – Stage 2 and 3 results – Logistic Regression

Analysis Type	Vectorizer	Number of features	N-gram range	Hyperparameters	Grid search accuracy	Validation set accuracy	Validation set precision
Rating	CV	3000	1, 2	C=0.6 max_iter=1000 multi_class=multinomial solver=newton-cg	83.63	61.7	1 - 0.66 2 - 0.48 3 - 0.5 4 - 0.55 5 - 0.7
Rating	CV	3000	1, 3	C=0.6 max_iter=1000 multi_class=multinomial solver=newton-cg	83.73	61.69	1 - 0.65 2 - 0.48 3 - 0.5 4 - 0.55 5 - 0.7
Rating	TFIDF	3000	1, 3	C=0.7 max_iter=1000 multi_class=multinomial solver=newton-cg	71.15	62.36	1 - 0.67 2 - 0.49 3 - 0.5 4 - 0.55 5 - 0.72
Sentiment	CV	3000	1, 3	C=0.6 max_iter=1000 multi_class=multinomial solver=newton-cg	93.69	83.97	0 - 0.8 1 - 0.55 2 - 0.89
Sentiment	TFIDF	3000	1, 2	C=1 max_iter=1000 multi_class=multinomial solver=newton-cg	87.27	84.17	0 - 0.8 1 - 0.55 2 - 0.89
Sentiment	TFIDF	3000	1, 3	C=1 max_iter=1000 multi_class=multinomial	87.28	84.13	0 - 0.8 1 - 0.55 2 - 0.89

Support Vector Machines performance was similar to the one of Logistic Regression. However, almost all models here recorded improvements in terms of accuracy compared to stage 1. The other interesting difference is that these models are better at classifying those reviews that should be neutral or the rating of which is 2, 3 or 4 stars. Overall, the best performing models again use Tfidf Vectorizer built using unigrams and bigrams or unigrams, bigrams and trigrams. Summary results are available in Table 9.

Table 9 – Stage 2 and 3 results – Support Vector Machines							
Analysis Type	Vectorizer	Number of features	N-gram range	Hyperparameters	Grid search accuracy	Validation set accuracy	Validation set precision
Rating	CV	3000	1, 2	C=0.6 dual=False penalty=l1	81.83	60.5	1 - 0.63 2 - 0.49 3 - 0.52 4 - 0.54 5 - 0.66
Rating	CV	3000	1, 3	C=0.6 dual=False penalty=l1	79.39	60.48	1 - 0.63 2 - 0.49 3 - 0.52 4 - 0.54 5 - 0.66
Rating	TFIDF	3000	1, 2	C=0.6 dual=False penalty=l1	73.28	61.48	1 - 0.63 2 - 0.5 3 - 0.51 4 - 0.54 5 - 0.69
Rating	TFIDF	3000	1, 3	C=0.6 dual=False penalty=l1	73.22	61.43	1 - 0.63 2 - 0.5 3 - 0.51 4 - 0.53 5 - 0.69
Sentiment	CV	3000	1, 3	C=0.6 dual=False penalty=l1	93.29	83.42	0 - 0.79 1 - 0.57 2 - 0.87
Sentiment	TFIDF	3000	1, 2	C=0.6 dual=False penalty=l1	87.61	84.06	0 - 0.78 1 - 0.59 2 - 0.88
Sentiment	TFIDF	3000	1, 3	C=0.6 dual=False penalty=l1	87.59	84.04	0 - 0.78 1 - 0.59 2 - 0.88

Random Forest classifier has the most complex list of hyperparameters of all the algorithms tested in this stage. For that reason, the highest number of different combinations were tuned. This tuning process also resulted in more significant improvements compared to the other algorithms. However, the results are still not comparable to the ones obtained using Logistic Regression or SVM. Not only is accuracy not as high, but the bias towards predicting top and bottom categories is much more visible with models built using Random Forest. The overall results are available in Table 10. Note that the parameters used for all the models have been omitted from the table (class_weight=balanced, n_jobs=-1, random_state=42).

Table 10 – Stage 2 and 3 results – Random Forest

Analysis Type	Vectorizer	Number of features	N-gram range	Hyperparameters	Grid search accuracy	Validation set accuracy	Validation set precision
Rating	CV	3000	1, 3	max_depth=90 max_features=12 min_samples_leaf=2 min_samples_split=20 n_estimators=200	87.83	58.7	1 - 0.55 2 - 0.45 3 - 0.45 4 - 0.54 5 - 0.68
Rating	TFIDF	2500	1, 2	max_depth=90 max_features=8 min_samples_leaf=2 min_samples_split=16 n_estimators=200	90.47	58.54	1 - 0.55 2 - 0.47 3 - 0.45 4 - 0.53 5 - 0.68
Rating	TFIDF	2500	1, 3	max_depth=110 max_features=8 min_samples_leaf=2 min_samples_split=20 n_estimators=200	91.32	58.9	1 - 0.55 2 - 0.49 3 - 0.46 4 - 0.53 5 - 0.69
Rating	TFIDF	3000	1, 3	max_depth=110 max_features=12 min_samples_leaf=2 min_samples_split=16 n_estimators=200	92.09	59.24	1 - 0.56 2 - 0.47 3 - 0.47 4 - 0.53 5 - 0.69
Sentiment	CV	3000	1, 2	max_depth=90 max_features=12 min_samples_leaf=2 min_samples_split=16 n_estimators=200	94.94	81	0 - 0.73 1 - 0.45 2 - 0.9
Sentiment	CV	3000	1, 3	max_depth=90 max_features=12 min_samples_leaf=2 min_samples_split=16 n_estimators=200	94.71	80.89	0 - 0.73 1 - 0.45 2 - 0.90
Sentiment	TFIDF	3000	1, 2	max_depth=90 max_features=12 min_samples_leaf=2 min_samples_split=16 n_estimators=150	96.62	81.52	0 - 0.73 1 - 0.48 2 - 0.89
Sentiment	TFIDF	3000	1, 3	max_depth=90 max_features=12 min_samples_leaf=2 min_samples_split=20 n_estimators=200	96.65	81.5	0 - 0.74 1 - 0.48 2 - 0.89

MultinomialNB is the only classifier where models built using Count Vectorizer performed better than the ones built using Tfidf. This does only happen when predicting rating, but it does come with a price. This classifier, when using Count Vectorizer, mainly focusses on predicting the highest class – the chasm between the classes is most obvious here. When using Tfidf, the overall accuracy is a bit lower, but the model produces more balanced results, by paying more attention to classification of all the classes as best as possible. The overall results for sentiment analysis are the worst compared to other algorithms. When it comes to ratings, only Random Forest performs worse.

Table 11 – Stage 2 and 3 results – Naïve Bayes

Analysis Type	Vectorizer	Number of features	N-gram range	Hyperparameters	Grid search accuracy	Validation set accuracy	Validation set precision
Rating	CV	3000	1, 2	alpha=10	64.40	59.5	1 - 0.58 2 - 0.42 3 - 0.45 4 - 0.54 5 - 0.72
Rating	CV	3000	1, 3	alpha=5	64.80	59.48	1 - 0.58 2 - 0.42 3 - 0.44 4 - 0.54 5 - 0.72
Rating	TFIDF	3000	1, 3	alpha=0.1	65.23	58.14	1 - 0.62 2 - 0.48 3 - 0.46 4 - 0.47 5 - 0.67
Sentiment	CV	3000	1, 2	alpha=10	82.22	79.47	0 - 0.75 1 - 0.41 2 - 0.92
Sentiment	CV	3000	1, 3	alpha=10	82.23	79.42	0 - 0.75 1 - 0.41 2 - 0.92
Sentiment	TFIDF	3000	1, 3	alpha=0.1	81.49	79.91	0 - 0.8 1 - 0.54 2 - 0.81

Stage 4 - Manual hyperparameter tuning on best performing models

This stage of the process was the most significant one in terms of the improvement in overall results. The first model that underwent the manual tuning was Logistic Regression and there was little that the model responded to in terms of the hyperparameter changes. The only thing that actually had effect was changing the number of features used to build the model. In general, model preferred more features, but filtered in a different way than before. Instead of using 3000 top features, using all the features that appeared a minimum of 10 or 15 times in the dataset yielded best results. Overall, this model's accuracy increased by 0.67% when classifying reviews according to ratings and 0.71% according to sentiment.

With Support Vector Machines, the only hyperparameter that ensured better results are obtained was C. That, in addition to the use of all the features generated by Tfidf Vectorizer, resulted in accuracy improvement of 1.54% for ratings classification and 1.10% for sentiment classification. This meant that SVM's performance for ratings was only 0.01% lower than the one for Logistic Regression and that it became the best sentiment classifier.

Random Forest did not improve much during this stage (0.03% for ratings, 0.47% for sentiment). Additional increase in `max_depth` and tweaking both `max_features` and `min_samples_split` did help, but the performance decayed if more than 3000 features were used. In addition to this, trying to limit the number of features based on the number of times they appeared in the dataset did not help either.

Multinomial Naïve Bayes was one of the models which improved the most during this phase, with accuracy for ratings classification increasing by 2.15% and the one for sentiment by 3.16%. These improvements were the result of changes made to the `alpha` hyperparameter and the limiting of number of features used, based on the minimum number of occurrences in the dataset. However, even with these massive improvements, Naïve Bayes was still not able to compete with the best models as its initial results were quite low.

The majority of manual adjustments were made to the Neural Networks as they did not undergo the grid search stage of the process. When working with different parameters, several basic configurations transpired – the model performed best when the number of hidden layers was set to 6 and when each of them had 50 units. In addition to this, smaller batch sizes worked better than the larger one, with the ideal one being set to 16. Finally, SGD optimizer was performing better than the one used in the previous steps (adam). In spite of all these changes, the biggest improvements in performance were achieved when monitoring accuracy across the validation set during the training and forcing the training to stop if the performance has not improved for 30 epochs. After that, the model would roll back to the best performing epoch. Along with that change, it transpired that the dropout rate of 20% added only before the output layer was also improving the overall accuracy. Finally, all these changes made led to an improvement of accuracy in the range between 1.43 (sentiment) and 1.66 (ratings).

Finally, the Voting Classifier was used on the validation set, consisting of the highest performing versions of the Logistic Regression, SVM, Random Forest and Naïve Bayes models. Its performance equalled the one of Logistic Regression in terms of accuracy, while also making it third best performing sentiment classifier, just behind SVM and Logistic Regression. Its performance was increased by using the features which appeared at least 20 (rating) or 50 (sentiment) times across the dataset. The overall results for this stage are available in Tables 12 and 13, based on rating or sentiment classification.

Table 12 – Stage 4 results – best performing models - rating

Algorithm	Vectorizer setup	Hyperparameters	Accuracy	Precision
Logistic Regression	TFIDF, (1, 3), min_df=10	C=1 max_iter=1000 multi_class=multinomial solver=newton-cg	63.03	1 - 0.69 2 - 0.50 3 - 0.51 4 - 0.55 5 - 0.72
SVM	TFIDF, (1, 3)	C=0.6 dual=False penalty=l1	63.02	1 - 0.66 2 - 0.51 3 - 0.51 4 - 0.55 5 - 0.72
Random Forest	TFIDF, (1, 3), max_features=3000	max_depth=190 max_features=12 min_samples_leaf=2 min_samples_split=16 n_estimators=200	59.27	1 - 0.56 2 - 0.47 3 - 0.47 4 - 0.53 5 - 0.70
Naïve Bayes	CV, (1, 3), min_df=15	alpha=5	61.65	1 - 0.61 2 - 0.46 3 - 0.47 4 - 0.55 5 - 0.74
Neural Networks	TFIDF, (1, 3), max_features=1500	6 hidden layers 50 units per layer SGD optimizer Learning rate=0.01 batch_size=16 epochs=100 EarlyStopping with settings: monitor=val_accuracy mode=max patience=30 restore_best_weights=True	61.99	1 - 0.66 2 - 0.47 3 - 0.49 4 - 0.55 5 - 0.71
Voting Classifier	TFIDF, (1, 3), min_df=5	Logistic Regression, SVM, Random Forest and Naïve Bayes best performing models	63.03	1 - 0.63 2 - 0.50 3 - 0.51 4 - 0.56 5 - 0.73

Table 13 – Stage 4 results – best performing models - sentiment

Algorithm	Vectorizer setup	Hyperparameters	Accuracy	Precision
Logistic Regression	TFIDF, (1, 3), min_df=15	C=1 max_iter=1000 multi_class=multinomial	84.89	0 - 0.82 1 - 0.57 2 - 0.89
SVM	TFIDF, (1, 3)	C=0.72 dual=False penalty=l1	85.16	0 - 0.80 1 - 0.58 2 - 0.90
Random Forest	TFIDF, (1, 3), max_features=3000	max_depth=190 max_features=12 min_samples_leaf=2 min_samples_split=20 n_estimators=200	81.99	0 - 0.74 1 - 0.50 2 - 0.89
Naïve Bayes	TFIDF, (1, 3), min_df=5	alpha=0.1	83.07	0 - 0.79 1 - 0.53 2 - 0.87
Neural Networks	TFIDF, (1, 2), max_features=1500	6 hidden layers 50 units per layer SGD optimizer Learning rate=0.01 batch_size=16 epochs=100 EarlyStopping with settings: monitor=val_accuracy mode=max patience=30 restore_best_weights=True	83.40	0 - 0.80 1 - 0.53 2 - 0.88
Voting Classifier	TFIDF, (1, 3), min_df=50	Logistic Regression, SVM, Random Forest and Naïve Bayes best performing models	84.66	0 - 0.78 1 - 0.58 2 - 0.90

Stage 5 - Using different vocabulary sets to validate model performance

When using different manually created vocabulary sets to verify the performance of the models built, only one model showed an improvement in results – Neural Networks. When classifying the reviews according to ratings, this model outperformed all the others when the vocabulary set 4 was used, reaching accuracy of 63%. This was 1.01% higher than its result in the previous stage. In general, most algorithms performed best with vocabulary sets 1 and 4, both of which had a slightly higher number of unigrams and bigrams. The performance of all the models per vocabulary set can be seen in Figure 26.

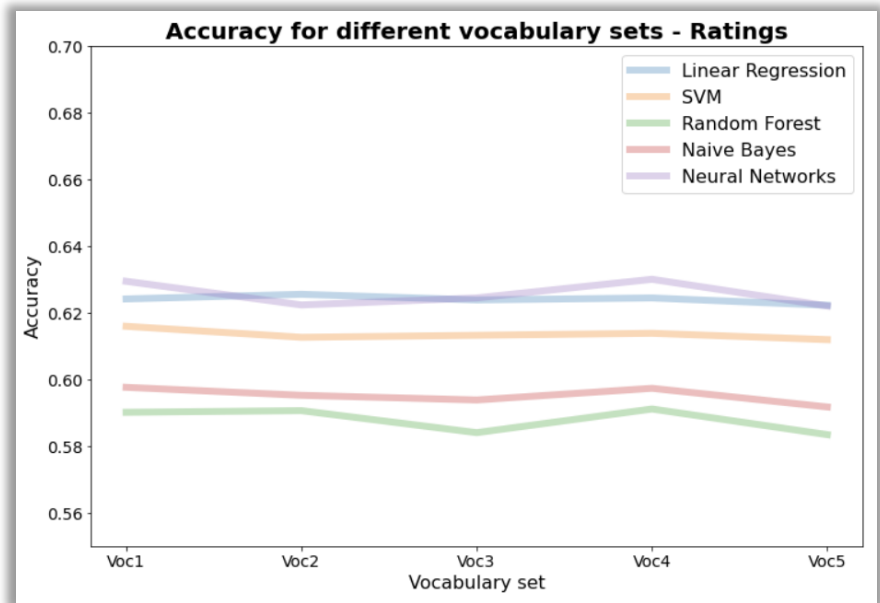


Figure 26 - Vocabulary testing - ratings

When trying to classify the reviews by sentiment, Neural Networks was again the highest performing model, but SVM and Logistic Regression were much closer now. Again, only Neural Networks actually improved the overall accuracy in this step, by 0.78%. This would indicate that this model apparently performs better with a vocabulary set that is more curated for a slightly

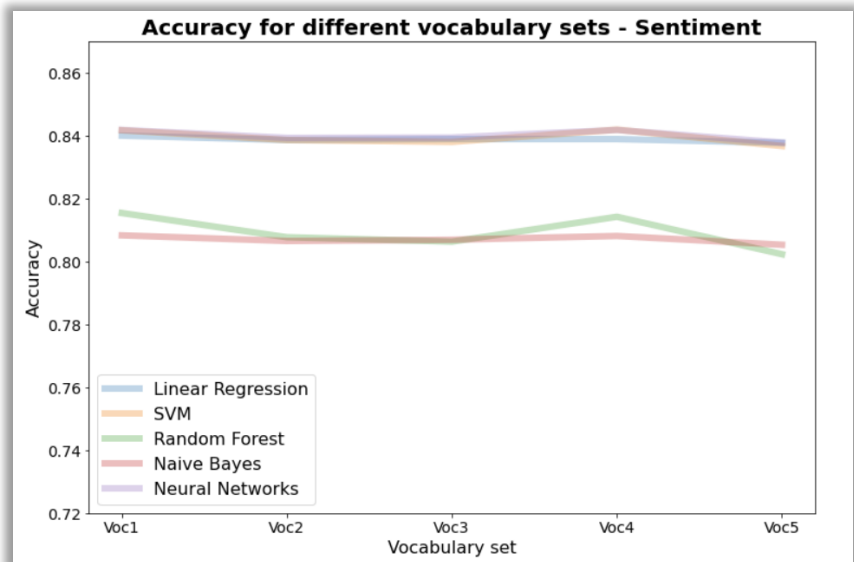


Figure 27 - Vocabulary testing - sentiment

imbalanced distribution in the dataset, while the other models prefer the features generated directly by Scikit-Learn. The performance of all the models is charted in Figure 27.

Stage 6 - Using test set to verify the models' performance

In the final stage of the project, the models were run on the test set, the set none of them have encountered before. With this set, there were no major changes in performance – all the models performed similarly to how they performed on the validation set. Four out of six models increased their accuracy when classifying based on the ratings and half of them when classifying based on the sentiment. The best performing model in both categories emerged to be Logistic Regression, with 63.31% accuracy when classifying the reviews according to rating and 85.09% when classifying according to sentiment. The close second is SVM, followed by the Voting classifier and Neural Networks. Naïve Bayes shows a really solid performance, while Random Forest is the model showing lowest scores in both sentiment and ratings prediction.

The major difference in performance between the highest and the lowest performing models can best be explained by looking at the confusion matrices. When comparing the ratings classification results of Logistic Regression model in Figure 28 and Random Forest in Figure 29, it is immediately noticeable that the Logistic Regression model outperforms Random Forest in correctly classifying 4 out of 5 classes. Not only is it better at correctly predicting the correct labels, but it is also closer when mislabelling. For example, the number of 5-star reviews classified as 1-star reviews by Random Forest is higher than the sum of 5-star reviews classified as 1, 2 and 3-star reviews by Logistic Regression. In addition to this, Logistic Regression is much better at classifying those middle ratings (2, 3 and 4), which is what most models struggled with the most.

True Class	Predicted Class				
	1	2	3	4	5
1	1629	405	128	50	63
2	487	849	585	156	89
3	143	380	1278	1026	244
4	37	46	445	3231	2180
5	36	11	63	1506	6954

Figure 28 - Logistic Regression ratings confusion matrix

True Class	Predicted Class				
	1	2	3	4	5
1	1757	309	104	43	62
2	724	676	471	169	126
3	344	325	1185	887	330
4	149	85	599	2858	2248
5	142	35	153	1671	6569

Figure 29 – Random Forest ratings confusion matrix

When other metrics for ratings classification are considered, this further reinforces the Logistic Regression's dominance. The most important indicator after accuracy, precision, is also the highest in Logistic Regression, with both Recall and F-1 Score looking solid. The worst performance, when those supporting metrics are considered, is the one by Random Forest. The graphical representation is available in Figure 30.

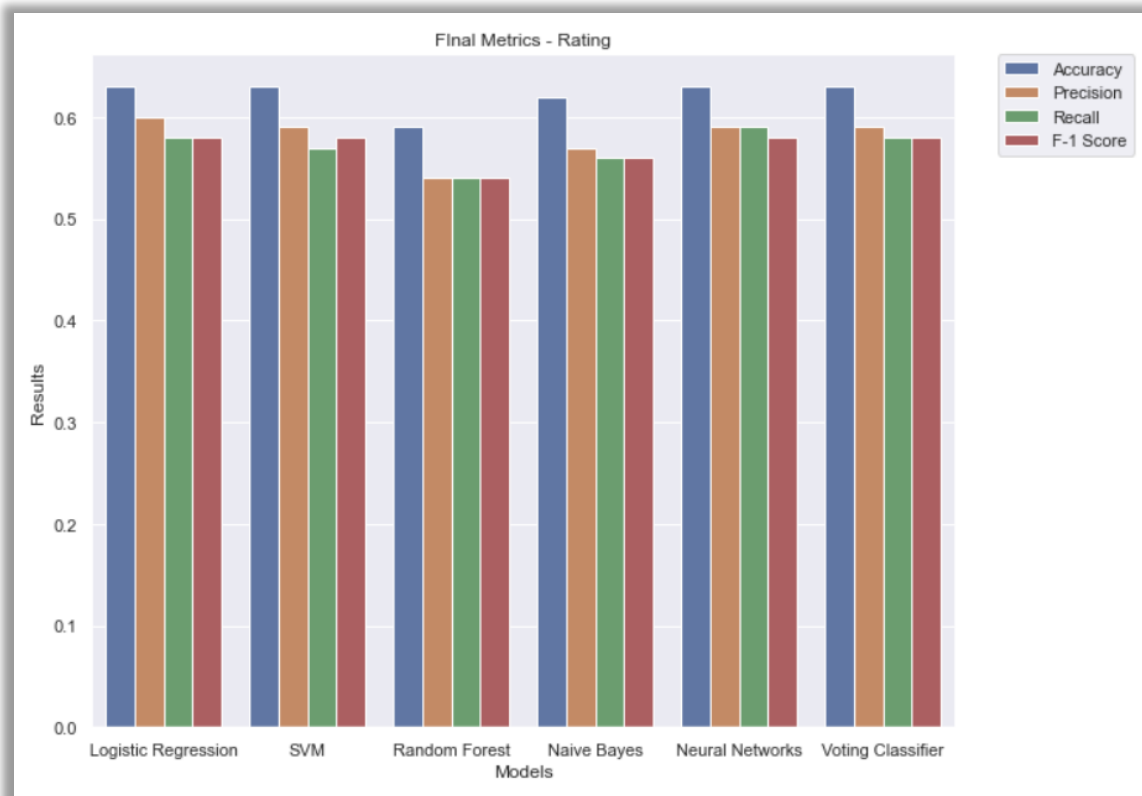


Figure 30 – Test set metrics for rating

Sentiment classification confusion matrices for Logistic Regression (Figure 31) and Random Forest (Figure 32) show a similar tendency to the one discussed when ratings were examined. Logistic Regression outperforms Random Forest in each of the classes this time, while also keeping its mislabelling closer. However, in spite of that, it is worth noting that both models do still struggle to label the neutral reviews correctly.

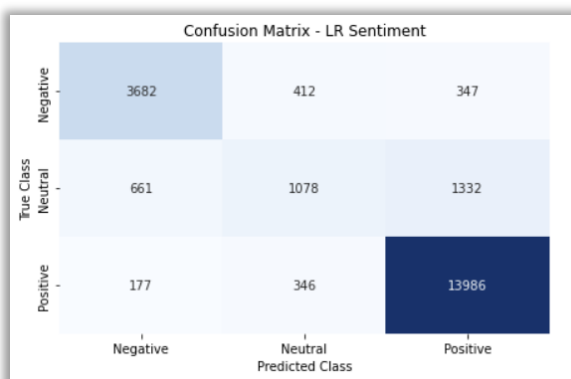


Figure 31 - Logistic Regression sentiment confusion matrix

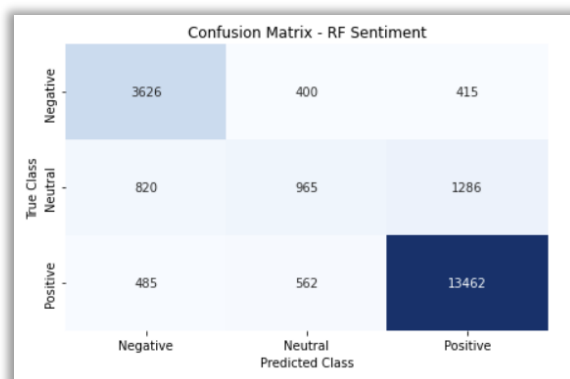


Figure 32 – Random Forest sentiment confusion matrix

Finally, when supporting metrics are examined for sentiment analysis, precision is again the highest in Logistic Regression. However, a bit more solid performance across Recall and F-1 Score is seen with the Voting Classifier and SVM. However, the Logistic Regression model still holds the highest results for the two most important metrics, which makes it the best choice for this type of analysis. Another thing worth noticing is that the performance of Random Forest holds up better in this analysis as it does outperform Naïve Bayes in terms of Recall and F-1 Score. The graphic representation of these results can be seen in Figure 33.

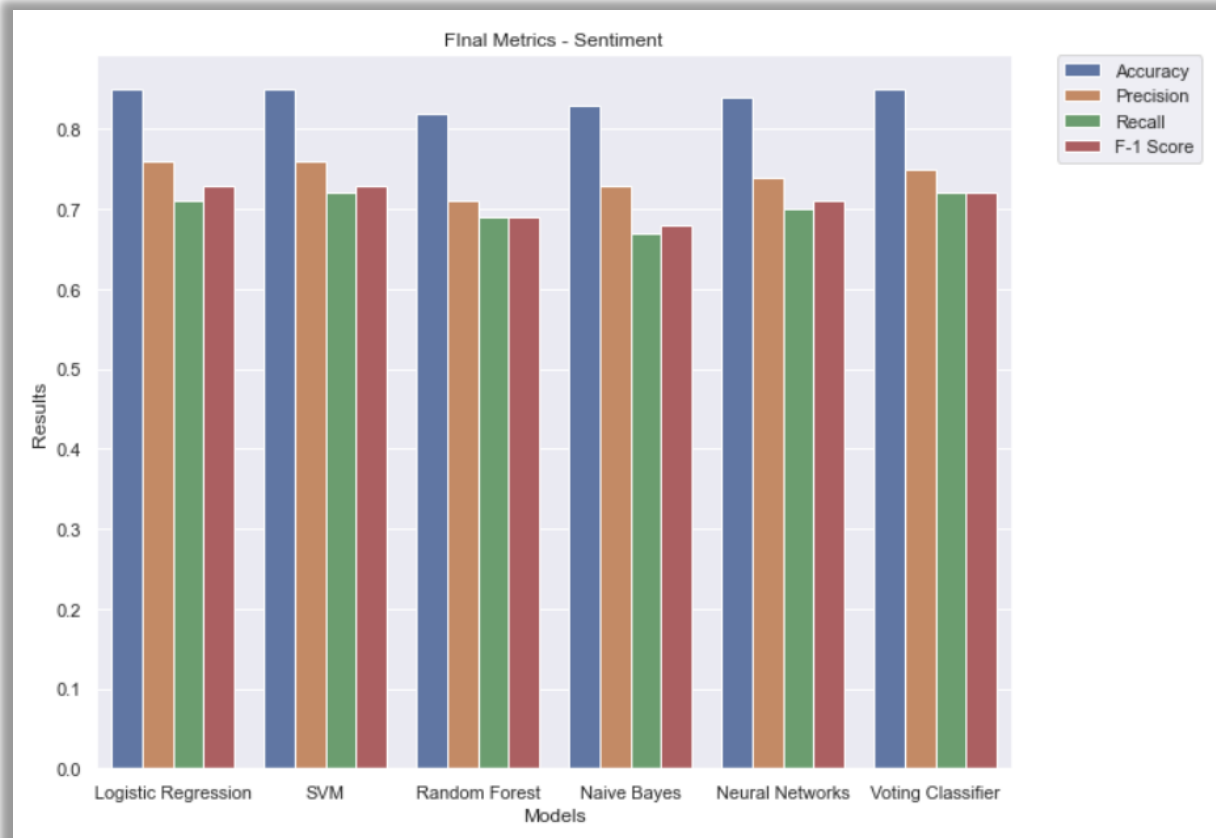


Figure 33 - Test set metrics for sentiment

After verifying the performance of all the models built, their performance for sentiment analysis was also compared to one of the tools already available on the market. Vader, that is often used for sentiment analysis, is available through the NLTK library and classifies the reviews based on the compound score – the score calculated on the basis of overall positivity, negativity or neutrality of the review. In case the compound score is equal to or higher than 0.05, the review is classified as positive. Negative reviews have the compound score that is equal to or less than -0.05, while the neutral reviews are those in the middle (Rai, 2021).

The Vader tool was run on the test set using the same pre-processing steps first, but as that library actually covers most of pre-processing steps utilised in this project (like lowercasing words) by default, it achieved better scores when no manual pre-processing was done. Overall, Vader performs worse than all the models used in the paper, achieving

accuracy of 73.69% when trying to classify the reviews as positive, negative or neutral. This is 11.44% less than the highest-performing Logistic Regression and 8.29% less than the underperforming Random Forest model. The biggest problems it had, as seen from Figure 34, was correctly classifying neutral reviews. This was also the case with models built during this project, but Vader's precision was really low for those reviews – only 18%. This might indicate that Vader is more capable of working with shorter pieces of text, but really underperforms when it has to handle longer reviews. Longer reviews are better handled by more robust models, like the ones built using machine learning algorithms. The final accuracy results obtained for the test set are available in Table 14.

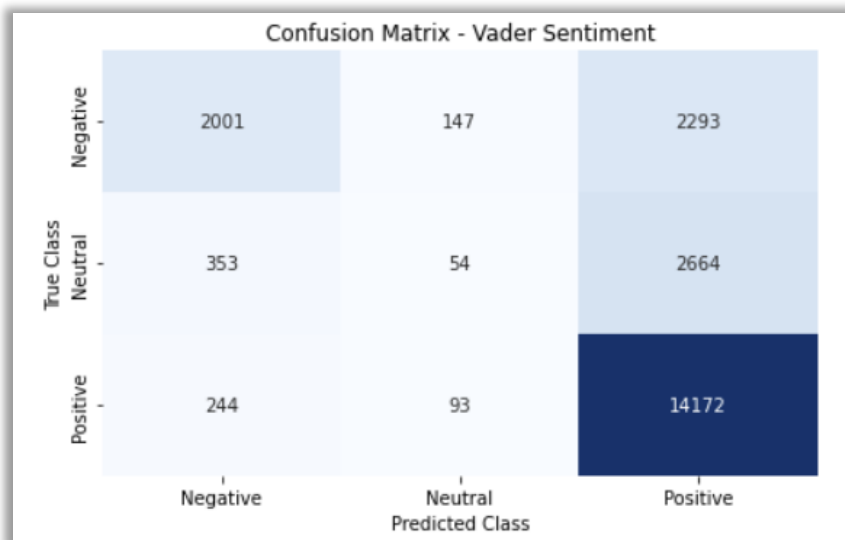


Figure 34 - Vader sentiment confusion matrix

Table 13 – Stage 6 results – accuracy on the test set			
Ratings Analysis		Sentiment Analysis	
Algorithm	Accuracy	Algorithm	Accuracy
Logistic Regression	63.31%	Logistic Regression	85.13%
SVM	63.26%	SVM	85.09%
Random Forest	59.24%	Random Forest	81.98%
Naïve Bayes	62%	Naïve Bayes	83.12%
Neural Networks	63%	Neural Networks	83.92%
Voting Classifier	63.16%	Voting Classifier	84.86%
		Vader	73.69%

6.0 Conclusions

The primary goal of the project was to build a model that could classify the reviews according to ratings ranging from 1 to 5. In addition to this, an additional model was supposed to be developed to classify the reviews based on their sentiment – positive, negative or neutral. The approach that was followed was KDD, an iterative process that focussed on completing all the individual stages with great care, but also included willingness to return to the previous steps and make the necessary changes that would result in better model performance. Using KDD and extensive literature review, it was also easier to identify the potential opportunities for innovation.

The main strengths of this project are that it is very well researched, it uses really large datasets and a wide range of technologies, that the correct methodology was followed and, finally, that the models produced perform well. When classifying ratings, the models achieved satisfactory levels of accuracy, in spite of the fact that the reviews analysed were very long and complex. Considering the fact that only the review text was used to determine ratings, the models were able to identify 1-star and 5-star reviews quite consistently, especially Logistic Regression and SVM. When the sentiment analysis is considered, all the models performed at an accuracy level higher than 80%, which definitely indicates that they have high potential for usability outside of this project. Specifically, when compared to a tool that is already being used on the market to perform the same type of analysis (Vader) and which actually underperforms in comparison with the models developed. Finally, using Neural Networks that has not been referenced in previous research as an option for sentiment analysis was a great choice, both as an innovation point and because this model performed really well in both ratings and sentiment classification.

While this project has many advantages, there are a couple of limitations. The main one is that the models developed have certain problems classifying the middle ranges – neutral sentiment or 2, 3 and 4-star ratings. However, the mislabelling errors are major, as most errors are usually move the review classification by only one class – a 2-star review will most often get misclassified as a 1-star or 3-star review. Nevertheless, this would probably be a problem for a business that wanted to implement the ratings model on a public-facing website. It might be of better use internally, for the business to classify the reviews or comments left on their website. Except for that, the other disadvantage would be the fact that the models produced run offline so the business would need to re-train the data as new reviews are collected in order to have the most up-to-date features when model is run. This can be done once a week and then the model can be redeployed with the newest information. However, in spite of these minor issues, the models perform well overall, especially the sentiment classifiers, so they can be readily used in any environment.

7.0 Further Development or Research

One of the potential reasons as to why the accuracy was lower for the ratings classification is the imbalanced nature of the dataset. The dataset that heavily relies on 5-star and 4-star ratings might have prevented better prediction of those mid-range reviews. With a more balanced dataset, it would be good to also add reviews from other sources – from Facebook, Twitter, Instagram and Google to other review-centric websites. That way, an even more robust model, with more important features could be built.

With all the additional reviews, all the pre-processing and training could be done in the cloud, using one of the online methods. That way, the model's performance could be updated in real time, which would not only cut down on the processing time, but also automate the process even further.

If the model is running in the cloud, it could also be deployed on a live website, where it could run on a trial basis, collect reviews and perform classification. In that first stage, the user could be given the predicted classification and asked to evaluate if it is correct or not. If not, there would be an option to change the rating – that way, the model would be able to learn from its mistake.

Finally, in addition to exploring some new pre-processing options, other tools and approaches could also be researched. In addition to Scikit-Learn and TensorFlow, it would be beneficial to explore PyTorch and do more work with deep learning as customizing the input vocabulary really did show major improvements with Neural Networks. Overall, all these improvements could have been made during the project development or could be done in the future, but the main idea behind them is the same – use new technologies, find more data and move everything online.

8.0 References

- Albon, C., 2018. *Machine Learning with Python Cookbook: Practical Solutions from Preprocessing to Deep Learning*. 1st ed. Sebastopol: O'Reilly Media, Inc..
- Aureliano-Silva, L., Leung, X. & Spers, E. E., 2021. The effect of online reviews on restaurant visit intentions: applying signaling and involvement theories. *Journal of Hospitality and Tourism Technology*, 12(4), pp. 672-688.
- Basheer, A. & Kaushik, A., 2019. *Social websites reviews and ratings of Dublin restaurants situated across 65 locations*. [Online]
Available at: <https://zenodo.org/record/3356793#.YcMa-Gj7SUI>
[Accessed 05 11 2021].
- Bird, S., Klein, E. & Loper, E., 2009. *Natural Language Processing with Python*. 1st ed. Sebastopol: O'Reilly Media, Inc..
- Fayyad, U., Piatetsky-Shapiro, G. & Smyth, P., 1996. The KDD Process for extracting useful knowledge from volumes of data. *Communications of the ACM*, 39(11), pp. 27-34.
- Géron, A., 2019. *Hands-On Machine Learning with Scikit-Learn, Keras and TensorFlow*. 2nd ed. Sebastopol: O'Reilly Media, Inc..
- Haque, F., Manik, M. H. & Hashem, M., 2019. *Opinion Mining from Bangla and Phonetic Bangla Reviews Using Vectorization Methods*. Khulna, 4th International Conference on Electrical Information and Communication Technology.
- Huda, S. M. A., Shoikot, M., Hossain, A. & Ila, I. J., 2019. *An Effective Machine Learning Approach for Sentiment Analysis on Popular Restaurant Reviews in Bangladesh*. Ipoh, 1st International Conference on Artificial Intelligence and Data Sciences (AiDAS).
- Ihle, J., 2021. *Consumers embrace the 'going out economy' as lockdowns ease*. [Online]
Available at: <https://www.independent.ie/business/irish/consumers-embrace-the-going-out-economy-as-lockdowns-ease-40702879.html>
[Accessed 05 11 2021].
- Krishna, A., Aich, A., V. A. & Hegde, C., 2019. Analysis of Customer Opinion Using Machine Learning and NLP Techniques. *International Journal of Advanced Studies of Scientific Research*, 3(9), pp. 128-132.
- Muller, A. C. & Guido, S., 2017. *Introduction to Machine Learning with Python: A Guide for Data Scientists*. 1st ed. Sebastopol: O'Reilly Media, Inc..
- Murphy, R., 2020. *Local Consumer Review Survey 2020*. [Online]
Available at: https://www.brightlocal.com/research/local-consumer-review-survey/?SSAID=314743&SSCID=c1k5_o3fca
[Accessed 05 11 2021].
- O'Brien, B., 2019. *Irish consumers to spend €8.55 billion on 'dining out' in 2019*. [Online]
Available at: <https://www.agriland.ie/farming-news/irish-consumers-to-spend-e8-55-billion-on-dining-out-in-2019/>
[Accessed 01 12 2021].

PowerReviews, 2020. *PowerReviews Market Trends Snapshot – June 2020*. [Online]
Available at: <https://www.powerreviews.com/insights/market-trends-june-2020/>
[Accessed 05 11 2021].

Rai, A., 2021. *Python / Sentiment Analysis using VADER*. [Online]
Available at: <https://www.geeksforgeeks.org/python-sentiment-analysis-using-vader/>
[Accessed 8 May 2022].

Review Trackers, 2021. *Online Reviews Statistics and Trends: A 2022 Report by ReviewTrackers*.
[Online]
Available at: <https://www.reviewtrackers.com/reports/online-reviews-survey/>
[Accessed 10 12 2021].

scikit-learn developers, 2022a. *sklearn.svm.SVC*. [Online]
Available at: <https://scikit-learn.org/stable/modules/generated/sklearn.svm.SVC.html>
[Accessed 15 April 2022].

scikit-learn developers, 2022b. *sklearn.linear_model.LogisticRegression*. [Online]
Available at: https://scikit-learn.org/stable/modules/generated/sklearn.linear_model.LogisticRegression.html
[Accessed 15 April 2022].

van Lohuizen, A.-W. & Trujillo-Barrera, A., 2020. The influence of online reviews on restaurants: The roles of review valence, platform and credibility. *Journal of Agricultural and Food Industrial Organization*, 18(2).

VanderPlas, J., 2017. *Python Data Science Handbook: Essential Tools for Working with Data*. 1st ed. Sebastopol: O'Reilly Media, Inc..

Yelp, 2021. *Yelp Open Dataset*. [Online]
Available at: <https://www.yelp.com/dataset/>
[Accessed 20 February 2022].

Yu, J. et al., 2022. *CoCa: Contrastive Captioners are Image-Text Foundation Models*, s.l.: arXiv.

Zahoor, K., Bawany, N. Z. & Hamid, S., 2020. *Sentiment Analysis and Classification of Restaurant Reviews using Machine Learning*. Giza, 21st International Arab Conference on Information Technology (ACIT).

9.0 Appendices

9.1. Project Proposal



National College of Ireland

Project Proposal

Opinion Mining for Automated Restaurant Reviews Rating

7/11/2021

BSc (Honours) in Computing - Evening

Data Analytics

2021/2022

Dejan Mijakovac

x18144861

x18144861@student.ncirl.ie

Contents

1.0	Objectives.....	62
2.0	Background	62
3.0	State of the Art.....	63
4.0	Data	63
5.0	Methodology & Analysis	64
6.0	Technical Details	65
7.0	Project Plan	66
8.0	Bibliography	68

1.0 Objectives

The main objective of the project is to develop a model that can correctly classify restaurant rating on the scale from 1 to 5, based on a non-numerical textual review provided by users. In order to complete the project, a number of other goals will have to be completed. Those goals can be grouped in two different categories – the ones required to complete the technical part of the project and the ones required to manage project completion. In terms of the first category, it will be important to find datasets that contain restaurant reviews, with classification already provided. Ideally, the dataset would contain both a numerical rating and categorical (positive / negative / neutral). Next goal would be to do some exploratory analysis and use data visualisation to find the general trends. After that, the data needs to be cleaned and transformed so it is ready for the machine learning process. The primary goal of the machine learning process would be to find the algorithm that performs best on this dataset. Once the right algorithm is found and the model fully tuned, it needs to be checked against the final test data.

The second set of goals is primarily concerned with managing the project work by creating a detailed project plan in order to manage project execution. Once the project plan is in place, the main focus shifts to updating the project documentation regularly. One last goal in this category is to research every stage of the project plan in detail.

2.0 Background

Restaurant industry is a very lucrative industry. Even during 2020, when many restaurants were close for a longer period of time, due to the COVID-19 pandemic, the restaurant industry was estimated to be worth around \$1.2 trillion (Aureliano-Silva, et al., 2021). Similarly, if only the Irish data is considered, around €8.55 billion was spent on dining in restaurants 2019. That number increased by 4.5% in comparison to 2018 (O'Brien, 2019).

Online reviews are extremely important for the restaurant industry and this became really clear during the COVID-19 pandemic. Recent survey of US customers' attitudes towards online presence of local business showed just how important a positive online presence is. 93% of people surveyed claim to have searched local businesses online, while 87% read their online reviews. The same survey also indicated that customers were most likely to read online reviews on restaurants when compared to all other industries (Murphy, 2020). This points to the conclusion that restaurants should indeed pay attention to online reviews.

Restaurant staff probably does not receive all the feedback in live interaction and for that reason they can use online reviews to make the business more successful. Currently, four websites hold 88% of all online reviews. Out of those 88%, Google is the industry leader with 73% of the market, followed by Yelp (6%), Facebook (3%), and TripAdvisor (3%). (Review Trackers, 2021). In addition to this, many online reviews are classified incorrectly as the users selected the incorrect star rating when writing their review. With that in mind, restaurants would benefit from a model that could analyse those third-party reviews and classify them correctly, allowing the restaurant owners to focus on quality feedback provided. This is what

this project is set out to do – build a model able to predict the rating based on the textual review that can be used on third-party reviews or even incorporated on a website.

3.0 State of the Art

This topic has been explored in the past, with most researchers attempting to classify the review sentiment by its polarity. This means that they tried to classify the reviews into either two or three categories. While most work uses positive / negative classification, some also add the neutral category. The first research reviewed used four different machine learning algorithms and a dataset of 4000 reviews. Its best performing algorithm was random forest with an accuracy of 95% (Zahoor, et al., 2020). The second researched paper had a dataset of 1500 reviews and used three different machine algorithms. Their best result was obtained when Support Vector Machine (SVM) algorithm was used, with a much lower level of accuracy– 75.58% (Haque, et al., 2019). Third paper reviewed had the lowest number of reviews in the dataset – only 700. Although they used four different algorithms to classify the data, the highest performing was once again SVM with an accuracy of 95% (Huda, et al., 2019). The last paper reviewed had just over 1000 reviews in the dataset and while 5 algorithms were used for classification, the best performing was, once again, SVM with 95% of accuracy (Krishna, et al., 2019).

There are two main differences between their approach compared to the one undertaken in this project. While their classification only focussed on two categories (positive / negative), this project will attempt to provide a classification into 5 categories. In addition to this, primarily because more are categories are considered, the dataset used will also be larger compared to the ones used in previous research. For that reason, the machine learning accuracy might come closer to the levels achieved than otherwise would be possible.

4.0 Data

As mentioned in the previous section, the number of reviews for this type of analysis ranges from 700 to 4000. All the researchers specified the importance of having a balanced dataset which contains approximately the same number of positive and negative reviews. One of them had an unbalanced dataset of 200 positive and 500 negative reviews, which proved to be a problem when testing the model (Zahoor, et al., 2020). All of the datasets, among other columns, contained two most important ones – a textual review column and another column that served as the target column, which served for the validation of their model. That column had either a numerical rating or the information on whether the review was positive or negative.

Based on that information, it is important to acquire a dataset that has the following characteristics:

- contains textual reviews of restaurants only
- contains a column that denotes if the review is positive, negative or neutral and /or a column that contains numerical rating (1-5 or 1-10)
- contains at least 2000 rows as most previous research was done on smaller samples than that
- the dataset is publicly available and has the clear permissions available on their website or allows for the licence to be granted by contacting the author(s) – clear permissions are preferable as licence-granting process can sometimes be fairly long and differ the start of the project

Once the dataset is found, it will be downloaded directly, or accessed using API, depending on the process outlined on the website.

5.0 Methodology & Analysis

In order to complete the project, the plan is to follow the KDD (Knowledge, Discovery and Data Mining) approach. This methodology is a data analytics methodology that has been used for the last 30 years and differs from the others as it relies on the iterative process (Fayyad, et al., 1996). In this case, as there will be a need for constant refinement for the model to become precise in its classification, there will exist a need to go back to one of the previous steps in order to get better results. Through this repetition, the model should become much better at classifying.

In terms of the actual organisational part, the classic stages of KDD – data selection, pre-processing, transformation, data mining and interpretation / evaluation (Fayyad, et al., 1996). For the selection phase, relevant sources of data need to be found and the dataset explored. In addition to becoming familiar with the dataset, this phase is also used to research the rest or the process needed to complete the project.

During the pre-processing phase, the idea is to clean the datasets of unnecessary columns, deal with the missing data and eliminate the datasets that prove to be irrelevant for the project. In the transformation phase, review text needs to be transformed in a format acceptable for machine learning. During the data mining phase, the idea is to use different machine learning approaches to train the model to successfully classify the data on both validation and test part of the dataset. Finally, in the interpretation / evaluation phase, the results are evaluated using charts and summary tables.

The machine learning algorithms planned for use based on the literature review are:

- Logistic Regression
- Naïve Bayes
- Support Vector Machine
- Decision Tree
- Random Forest

Regarding work organization, it will be done in two-week sprints, consisting of activities logically linked to one another. The focus of the first semester would be on finding the data, cleaning it from irrelevant information and showing some trends using data visualisations. The main driver for those tasks would be the mid-term presentation as the first major milestone. During second semester, there will be two major milestones. First one would be the reading week where at least one model used should be performing well. The second major milestone would be Easter break. At that point, all the project work should be done. This will allow a couple of extra weeks to work on documentation or account for delays in project work.

6.0 Technical Details

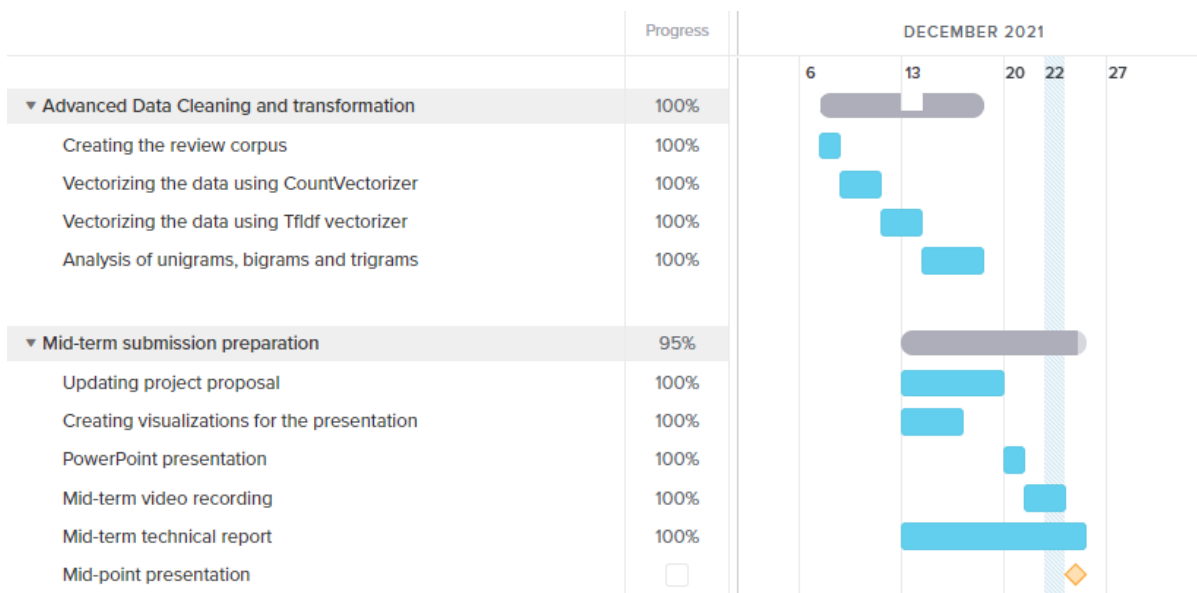
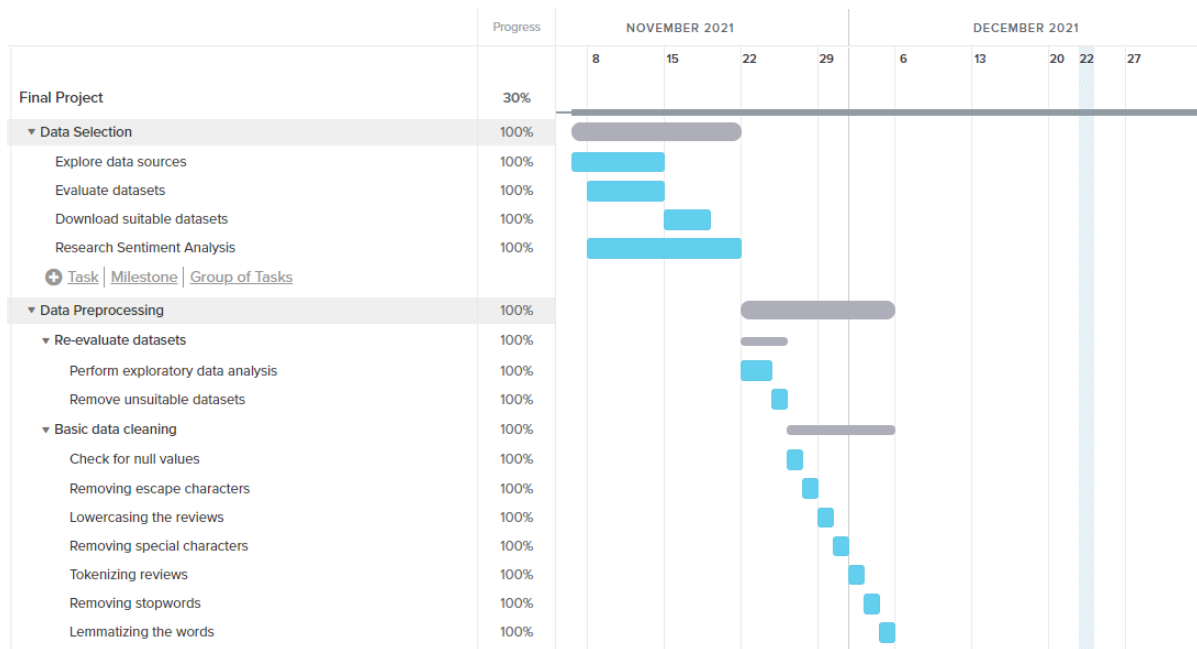
For the project management part of the project, Microsoft Office products will mostly be used. Microsoft Word will be used for all the documentation. Excel will be used to review datasets in the data selection phase. In order to create slides for video presentations, PowerPoint will be used. The videos themselves will be recorded using Microsoft Teams and then hosted using Microsoft Stream. The only non-MS Office tool used will be an online project management tool – TeamGantt. As the name itself suggests, the tool is used to create and host Gantt charts. Gantt chart created using that tool is used for the project plan in this document.

The technical side of the project, the code used to import and clean the data, create data visualisations and train, validate and test the models will be done using Python 3.9. In order to run Python code, PyCharm IDE (integrated development environment) will be used locally. Google Colab notebooks will mostly be used for presentations as they offer a cleaner outline and are more convenient when a single line of code needs to be run, especially on a project that potentially has hundreds of lines of code. Python libraries used to complete the project and their descriptions are available in the table below.

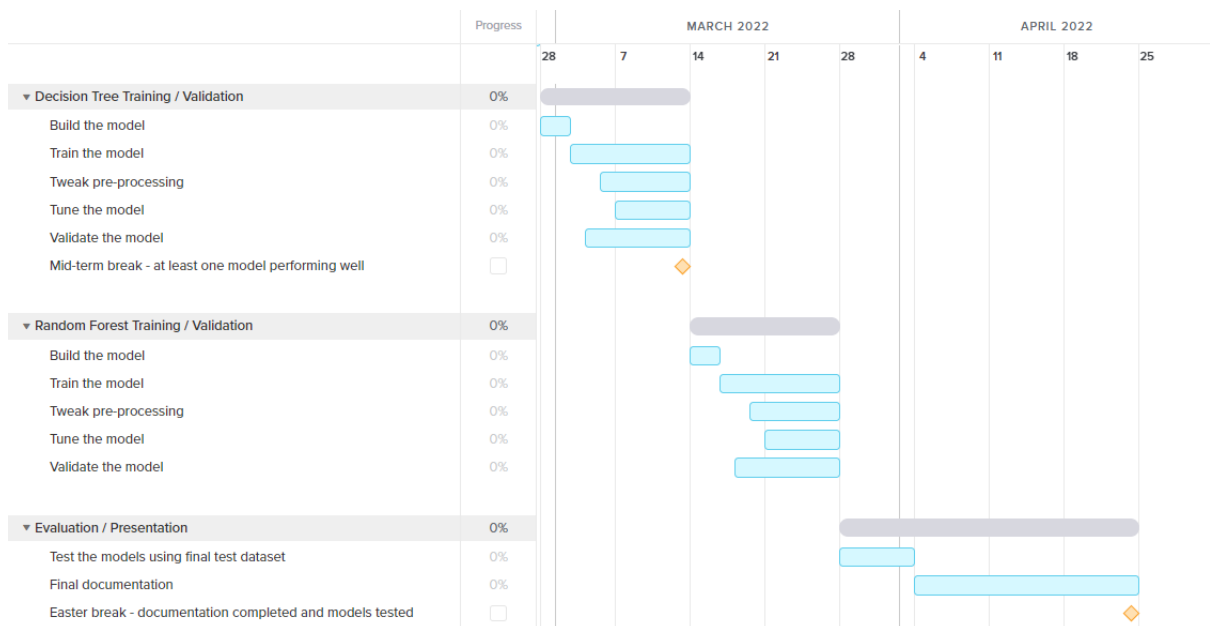
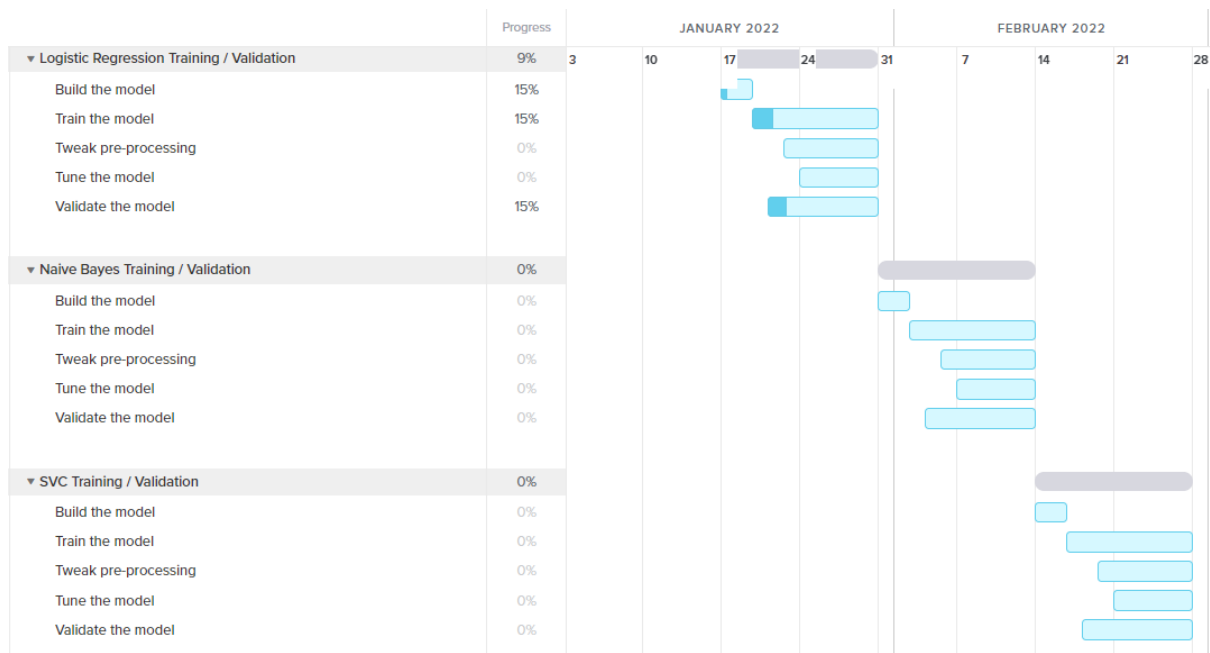
Library	Usage
Pandas	Importing / exporting csv files, manipulating DataFrames
Openpyxl	Importing / handling xlsx files
Re	Cleaning data using Regular Expressions
Matplotlib	Data visualization library
Seaborn	Data visualization library
Plotly	Data visualization library
NLTK	Language processing library
Scikit-learn	Machine learning library used for building, training and testing models

7.0 Project Plan

Project plan has been adjusted for the change of project idea so it starts on the 8/11.



The gap between the two semesters is purely down to exam preparation.



Note that some extra time is provisioned here in case of delays in project work – the final two weeks before the project deadline are meant to be used in case of delays or in case some additional adjustments are needed.

8.0 Bibliography

- Aureliano-Silva, L., Leung, X. & Spers, E. E., 2021. The effect of online reviews on restaurant visit intentions: applying signaling and involvement theories. *Journal of Hospitality and Tourism Technology*, 12(4), pp. 672-688.
- Fayyad, U., Piatetsky-Shapiro, G. & Smyth, P., 1996. The KDD Process for extracting useful knowledge from volumes of data. *Communications of the ACM*, 39(11), pp. 27-34.
- Haque, F., Manik, M. H. & Hashem, M., 2019. *Opinion Mining from Bangla and Phonetic Bangla Reviews Using Vectorization Methods*. Khulna, 4th International Conference on Electrical Information and Communication Technology.
- Huda, S. M. A., Shoikot, M., Hossain, A. & Ila, I. J., 2019. *An Effective Machine Learning Approach for Sentiment Analysis on Popular Restaurant Reviews in Bangladesh*. Ipoh, 1st International Conference on Artificial Intelligence and Data Sciences (AiDAS).
- Krishna, A., Aich, A., V. A. & Hegde, C., 2019. Analysis of Customer Opinion Using Machine Learning and NLP Techniques. *International Journal of Advanced Studies of Scientific Research*, 3(9), pp. 128-132.
- Murphy, R., 2020. *Local Consumer Review Survey 2020*. [Online]
Available at: https://www.brightlocal.com/research/local-consumer-review-survey/?SSAID=314743&SSCID=c1k5_o3fca
[Accessed 05 11 2021].
- O'Brien, B., 2019. *Irish consumers to spend €8.55 billion on 'dining out' in 2019*. [Online]
Available at: <https://www.agriland.ie/farming-news/irish-consumers-to-spend-e8-55-billion-on-dining-out-in-2019/>
[Accessed 01 12 2021].
- Review Trackers, 2021. *Online Reviews Statistics and Trends: A 2022 Report by ReviewTrackers*. [Online]
Available at: <https://www.reviewtrackers.com/reports/online-reviews-survey/>
[Accessed 10 12 2021].
- Zahoor, K., Bawany, N. Z. & Hamid, S., 2020. *Sentiment Analysis and Classification of Restaurant Reviews using Machine Learning*. Giza, 21st International Arab Conference on Information Technology (ACIT).

9.2. Reflective Journals

Month: October

What?

The first month of a new project, especially such a large one, always brings about a varied set of tasks to complete. The first thing to do was to prepare the project pitch, which not only meant making a decision on how to record the video and where to stream it, but also to decide on the project topic and do some preliminary research about the topic chosen. Once the video was done and uploaded, we were supposed to start working on the project proposal and wait for a supervisor to be assigned to us so they can inform us whether our project idea has been accepted or not, give us feedback on the things we might need to change with the project itself and maybe give us some pointers on what to focus first. In the meantime, we were also supposed to research datasets we might want to use so we can include them in the Ethics Form.

So What?

I did decide on a topic I feel strongly about – the impact of socio-economic factors on covid-19 vaccine hesitancy in the EU. That was the first step. Then, I had a look at some datasets that would be required for the project and the first glance at them seems to indicate there indeed might be a link between the factors I want to explore and the vaccine hesitancy. The video upload went ok, not great. While filming, I realized that I might need to leave out some technical details in order to fit it within 3 minutes and I was pretty certain this might lead to some amendments being suggested for the project idea. In terms of the technical part, I did decide to use Python and R to analyse the data. I do want to do some machine learning to verify the data, but I definitely need more input from the supervisor to see if this would be best done using logistic regression, neural nets or something else. I also have a “nice-to-have” feature – if there is enough time, I’d like to display the charts / graphs on a website powered by Python and either Flask or Django (maybe use React to handle frontend). Finally, the supervisor was assigned to me at the end of the month and, as I suspected, mentioned that there will be a need for some changes to be implemented. However, he is still waiting for the other reviewer to upload the feedback so it is still uncertain what needs to be changed. Overall, the biggest successes are definitely the completion of all the deliverables so far and the fact that I was able to make a decision on the topic. The challenges are definitely fully defining the scope of the project and organising the workload while taking into consideration all the continuous assessments and projects in the first semester.

Now What?

The things I would like to complete in the following month are:

- Check in with the supervisor on Tuesday to see if there is feedback available from the other marker and then set up a meeting so we can discuss the changes that need to be implemented

- Research the approach on how to complete the project in a bit more detail and complete the project proposal before 7/11
- File the Ethics Form for the datasets I have already found by 7/11
- After 7/11, start searching for other compatible datasets that would be suitable for the project
- Do some initial data cleaning for a smaller part of the dataset, maybe do all the socio-economic factors vs vaccination rates for two countries (one at the top end and one of the ones with lower vaccination rates, I might just compare Croatia and Ireland as I am familiar with situation in both countries)
- Try plotting some graphs using R to show the comparison between the two

Month: November

What?

The main event this month happened during the first week of November – the proposed project idea was rejected by one of the markers so the supervisor informed me that we would need to look at the college approved projects and pick one of them. After examining the options provided, I have chosen Opinion Mining for Automated Restaurant Reviews Rating as this was the topic that suited the specialization the most. Once the topic was chosen, the most important first milestones to meet were writing up a project proposal and the ethics form. As I have been notified of the change in topic only a couple of days before the submission date, I needed to focus on quickly exploring the topic so I can include some basic pieces of information in the submission documents and build a foundation on which I can explore the topic further during November and build the project.

So What?

After having a couple of meetings with the supervisor about how to proceed with the project, I researched the basic information behind the topic and included the information in the project proposal documentation. I have also managed to find datasets that do fit in with the topic, but most of them did not have a clearly formulated disclaimer that would allow me to use it with no issues. Finally, I managed to submit two of them as part of the ethics form, right on time. Once the upload was done, it allowed me to focus more on further exploration of the topic. During November, I have decided to move the data exploration and pre-processing from R into Python. On one hand, the decision was made because I already have experience coding in Python and my knowledge of R is still pretty basic. On the other hand, Python also offers more versatility in later stages of the project, not only in terms of different libraries that can be used for machine learning, but also when it comes to the avenues that can be taken in the final stages of the project – if I do decide to deploy the model to work with restaurant reviews on a live website, that would be much easier done using Python and Django or Flask than R. In addition to doing the technical research, I also focussed heavily on researching the process relating to sentiment analysis and opinion mining – which steps need to be taken in each of the phases and which machine learning algorithms would work best.

While I still have not decided on the best machine learning approach, the exploration and pre-processing phases are pretty clear at this point. In order to complete these initial steps, I have also in the process of finishing an online course on Pandas as this is going to be the first library that will enable me to pre-process and clean the dataset. As far as the datasets are concerned, the decision was made to use one main dataset that has both the numerical scale attached to each of the reviews (1-5) and the general sentiment (positive/negative). Other datasets can be kept as backup for now, but this one should be more than enough as it has 10000 reviews stored. The initial data exploration showed that the dataset is appropriate for the project as it is well balanced, with the same amount of positive and negative reviews, and no faulty or missing values.

Now What?

The things I need to complete in the following month are:

- Finish pre-processing phase – remove stopwords, special characters, lemmatize the words
- Analyse unigrams, bigrams and trigrams – generate charts showing most common expressions
- Generate wordclouds to see which words dominate the dataset
- Finish Pandas course and identify which other libraries / frameworks will be necessary to complete the project
- Shortlist the machine learning approaches that would be most suitable for the project
- Complete documentation for mid-point submission – update proposal to reflect the changes made, create PowerPoint presentation and record the video showcasing the progress, fill out the documentation with relevant information gathered so far

Month: December

What?

The focus of this month's efforts was mainly on getting as much as possible done for the mid-point submission. The information was provided that, as a part of the submission, a draft of the final documentation, sections 1 to 4, will also need to be included in the submission, along with the project proposal and the video showcasing the work already completed. This created plenty of additional work as each part of the process needed to be documented in detail to have the document draft ready.

So What?

Regarding the actual work on the code, the goals set for this month were met, and even surpassed. All the pre-processing steps and data visualisations were completed, two different ways of transforming the reviews from textual into numerical form were used. At this point, in terms of technologies, a wide array of libraries were employed – NLTK for pre-processing of textual data, Pandas for manipulating DataFrames, Plotly, Seaborn and Matplotlib for data visualisations and scikit learn is the library chosen for the machine learning part of the project. In terms of actual machine learning algorithms, five were chosen for the project, based on the review of previous works done on sentiment analysis – logistic regression, support vector machine, decision trees, random forest and naïve Bayes. The work has been started just before the submission on logistic regression, generating first results, the accuracy of which is above 65%.

This is an encouraging first step as the results when using 5 categories with this algorithm are usually much lower. Finally, Google Colab is proving to be a great help when running machine learning scenarios – as the dataset is quite large for this type of project, in some cases the code cannot be run locally so Google Colab is a great backup option there.

As for the documentation, most of the planned work was completed – Project Proposal is now much more solid, the introduction, pre-processing and transformation stages are well-organised, with enough visual representation of the work done so far. The sections that will definitely need a lot more work are those relating to machine learning algorithms, but these are still a work in progress as that part of the work has only started.

The video itself did not go as planned. The original plan was to record shorter videos, one for each section and then combine them using wave.video, an online video-editing platform. However, their service experienced prolonged downtime periods around the time of submission so a one-take video was recorded in the end using Microsoft Teams. The downside of this is lower video quality, but it exposed even more that 10 minutes are far from enough to explain the whole project and the work done so far coherently. The positive thing is that everything was completed and submitted on time.

When it comes to the supervisor meetings, they are happening regularly, once a week. I am really satisfied with the collaboration as I feel the supervisor has good sense of direction for the project and always brings in some fresh ideas and a new perspective on the problem.

Now What?

The things I need to complete in the following month are:

- Explore other machine learning algorithms in more detail
- Complete all the planned scenarios with logistic regression – train and validate using different combinations of pre-processing and experiment with model-tuning

Month: January

What?

Most of January went towards exam preparation and execution. Once this was done, the plan was to continue with the logistic regression and further exploration of machine learning to prepare for the rest of the work on the project. However, the announcement was made that the final project bio, along with project overview and the images that are going to be displayed on the project showcase page need to be completed / selected by early / mid February. In addition to this, the final submission date has also been moved so there will be one additional week available to complete the project and the project documentation.

So What?

Due to those announcements, the project plan has been tweaked slightly – even though the logistic regression work was continued, its completion has been pushed by a week in order to allow for the time to finish the bio and other information required for the showcase website. Considering the fact that the final submission has been pushed by a week, this would not delay the work needed to complete the project on time. It will, however, allow more time focussed on

the actual code towards the end that was supposed to be used for the project showcase information.

Now What?

The things that are to be completed in the following month are:

- Finalise all the planned scenarios with logistic regression – train and validate using different combinations of pre-processing and experiment with model-tuning – along with adding this information to the final document
- Finalise everything related to the showcase – bio, project description, images, technologies, poster
- Theory background and training / validation using Naïve Bayes with the documentation
- Theory background on Support Vector Machine and starting training / validation

Month: February

What?

Early February was used for completing all the project showcase profile details – from writing a bio, project description to selecting and resizing / cropping images. Towards the end of that work, the feedback was provided for the work done during the first semester and it transpired that some additional input would be needed. That means that additional datasets needed to be sourced in order to continue with the machine learning stages of the project.

So What?

Due to this fact, the rest of the work completed in February revolved around sourcing additional datasets. The two sources added were reviews scraped from TripAdvisor using Selenium and Yelp's open dataset. Each of these datasets has its own specifics – TripAdvisor is not really clear on their scraping policy so the outcome on that dataset will be known once the Ethics form submission is reviewed. As for Yelp, this is a massive dataset, with over 8 million reviews of restaurants. This has two knock-on effects on the overall project. The first one is that all the background research on the machine learning approaches to be used is pushed before actual work on the data. This way, all the preparation for all the machine learning approaches will be done while the new datasets are being evaluated by the college. The second effect is that some of the previously planned machine learning techniques might need to be dropped as they work well with smaller datasets only. This means that a move towards more deep learning techniques will probably happen.

Now What?

The things that are to be completed in the following month are:

- Find more effective ways of loading large datasets faster (like the Yelp dataset)
- Explore batch training options for machine learning models – the textual reviews need to be vectorized and this requires a lot of memory so batch training would be a good workaround for memory issues

- Explore deep learning and write the generic code snippets that can be used to loop through different models and train / validate data using different parameters
- Complete the exploratory data analysis, pre-processing and transformation phases of the new datasets and update the final document with that data
- Finally complete the Logistic Regression part of the project with the new datasets included

Month: March

What?

Before the mid-term break, there was a meeting with Siobhan from the Careers Office where feedback was provided on what needs to be changed in the project description. In addition to this, the supervisor proposed having a Kanban board with all the active task so it is easier to track the overall progress. Finally, the Easter break is approaching and the first results obtained from machine learning models were not encouraging in terms of achieving high accuracy (over 80%) so there was more pressure added to get better results.

So What?

The first changes were made to the project description to include more technical details on the technologies used. Afterwards, the decision was made to also include a second target column to the dataset – this one focussing on whether the review was good, bad or neutral and the three machine learning techniques used so far (Logistic Regression, Support Vector Machines, Random Forest) showed much better results. The hyper parametrization using SearchGrid CV additionally improved the results. Finally, a Kanban board was set up through Monday.com for the supervisor to be able to have a better overview which items are being worked on.

Now What?

The things that are to be completed in the following month are:

- Finish all the machine learning techniques with the train and validate sets
- Compile plots and charts using data visualization tools to show how models with the best parameters were found
- Update the documentation reflecting all the work that has been done
- Regularly keep updated the board containing active tasks so that the supervisor can see the progress before the weekly meetings

Month: April

What?

A lot of the time this month was spent on either finishing continuous assessments or working on terminal assessments, which were all pretty huge. However, the work on the project continued in the background, with it getting the full focus after the 22/04. As most of the work with the first three algorithms was done, it was important to get more done with the last two approaches used (naïve bayes classifier and neural nets).

So What?

All the algorithms were put through all the phases initially planned – training, hyper-parametrization and validation of the results on the validation set. All this did lead to an improvement in performance, but some additional tweaking can be done in the last two weeks. Additional literature has been consulted in order to find more ideas on how to improve the performance. Finally, the documentation has been updated with the changes made.

Now What?

The things that are to be completed in the final phase of the project:

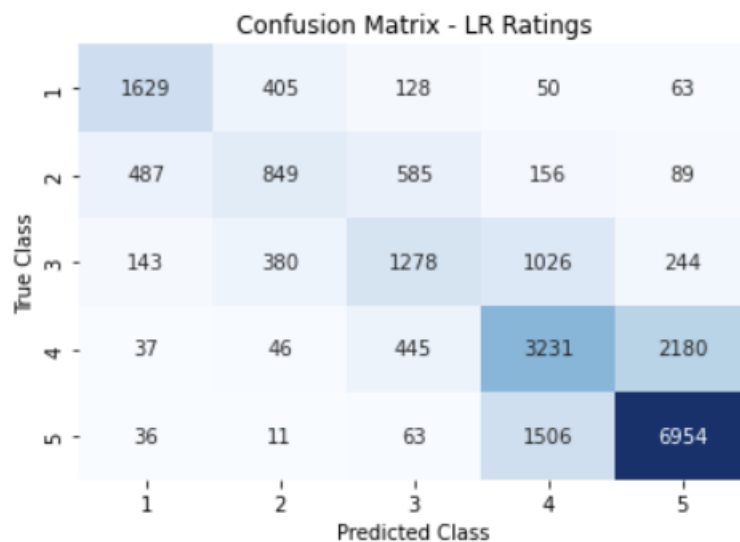
- Tweak the vocabulary that goes into the Tfidf and CountVectorizer and train the model using that data – this should account for the data being skewed towards positive reviews
- Use VotingClassifier to try to get better classification results using several algorithms at the same time on the dataset
- Run the most successful configurations on the test set
- Update the documentation reflecting all the work that has been done
- Create the project poster as part of the submission
- Record the final presentation video

9.3. Outputs of models' performance on test set

1. Logistic Regression ratings accuracy

```
round(accuracy_score(y_true=test_reviews["rating"], y_pred=p_final_lr_ratings), 4)
0.6331
```

2. Logistic Regression ratings - confusion matrix



3. Logistic Regression ratings – all metrics

```
print(classification_report(test_reviews["rating"], p_final_lr_ratings, target_names=target_names))
```

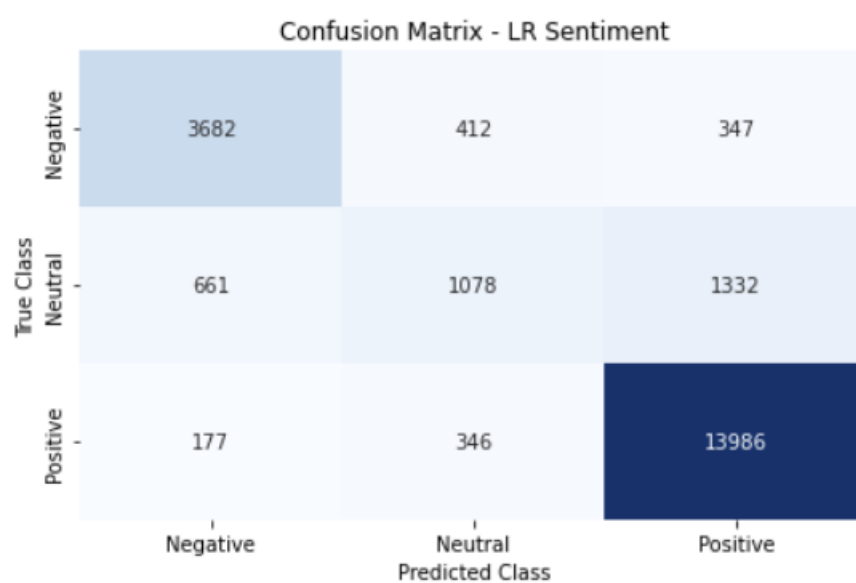
	precision	recall	f1-score	support
1 star	0.70	0.72	0.71	2275
2 stars	0.50	0.39	0.44	2166
3 stars	0.51	0.42	0.46	3071
4 stars	0.54	0.54	0.54	5939
5 stars	0.73	0.81	0.77	8570
accuracy			0.63	22021
macro avg	0.60	0.58	0.58	22021
weighted avg	0.62	0.63	0.63	22021

4. Logistic Regression sentiment accuracy

```
round(accuracy_score(y_true=test_reviews["sentiment"], y_pred=p_final_lr_sentiment), 4)
```

0.8513

5. Logistic Regression sentiment - confusion matrix



6. Logistic Regression sentiment – all metrics

```
print(classification_report(test_reviews["sentiment"], p_final_lr_sentiment, target_names=target_names))
```

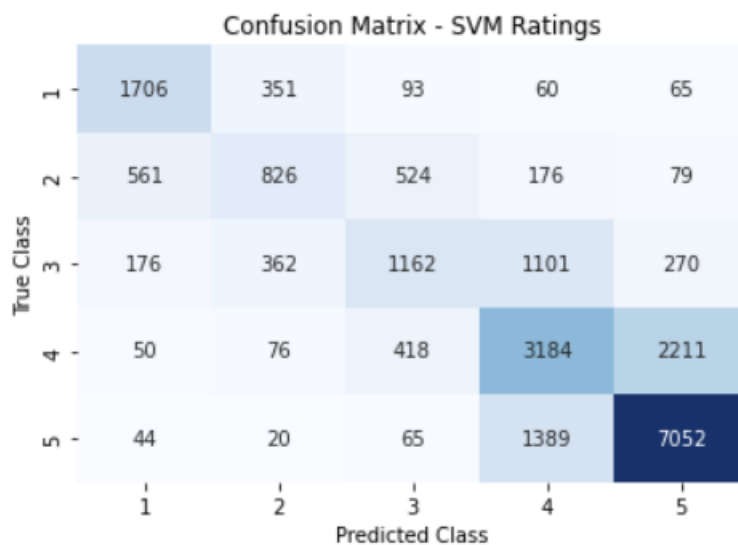
	precision	recall	f1-score	support
Negative	0.81	0.83	0.82	4441
Neutral	0.59	0.35	0.44	3071
Positive	0.89	0.96	0.93	14509
accuracy			0.85	22021
macro avg	0.76	0.71	0.73	22021
weighted avg	0.83	0.85	0.84	22021

7. Support Vector Machines ratings accuracy

```
round(accuracy_score(y_true=test_reviews["rating"], y_pred=p_final_svm_ratings), 4)
```

0.6326

8. Support Vector Machines ratings – confusion matrix



9. Support Vector Machines ratings – all metrics

```
print(classification_report(test_reviews["rating"], p_final_svm_ratings, target_names=target_names))
```

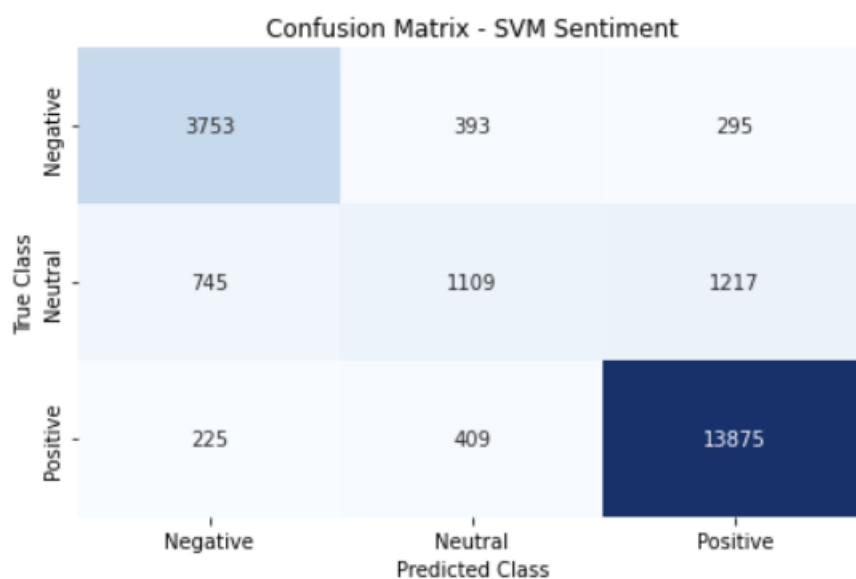
	precision	recall	f1-score	support
1 star	0.67	0.75	0.71	2275
2 stars	0.51	0.38	0.43	2166
3 stars	0.51	0.38	0.44	3071
4 stars	0.54	0.54	0.54	5939
5 stars	0.73	0.82	0.77	8570
accuracy			0.63	22021
macro avg	0.59	0.57	0.58	22021
weighted avg	0.62	0.63	0.62	22021

10. Support Vector Machines sentiment accuracy

```
round(accuracy_score(y_true=test_reviews["sentiment"], y_pred=p_final_svm_sentiment), 4)
```

0.8509

11. Support Vector Machines sentiment – confusion matrix



12. Support Vector Machines sentiment – all metrics

```
print(classification_report(test_reviews["sentiment"], p_final_svm_sentiment, target_names=target_names))
```

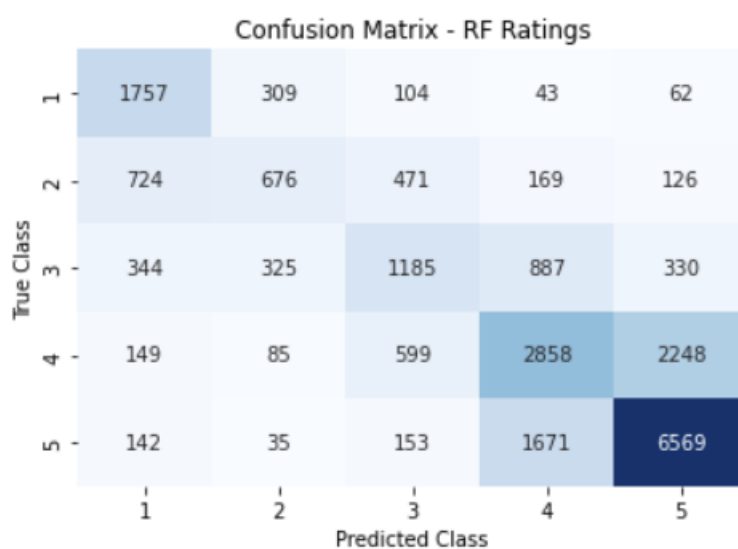
	precision	recall	f1-score	support
Negative	0.79	0.85	0.82	4441
Neutral	0.58	0.36	0.45	3071
Positive	0.90	0.96	0.93	14509
accuracy			0.85	22021
macro avg	0.76	0.72	0.73	22021
weighted avg	0.84	0.85	0.84	22021

13. Random Forest ratings accuracy

```
round(accuracy_score(y_true=test_reviews["rating"], y_pred=p_final_rf_ratings), 4)
```

0.5924

14. Random Forest ratings - confusion matrix



15. Random Forest ratings – all metrics

```
print(classification_report(test_reviews["rating"], p_final_rf_ratings, target_names=target_names))
```

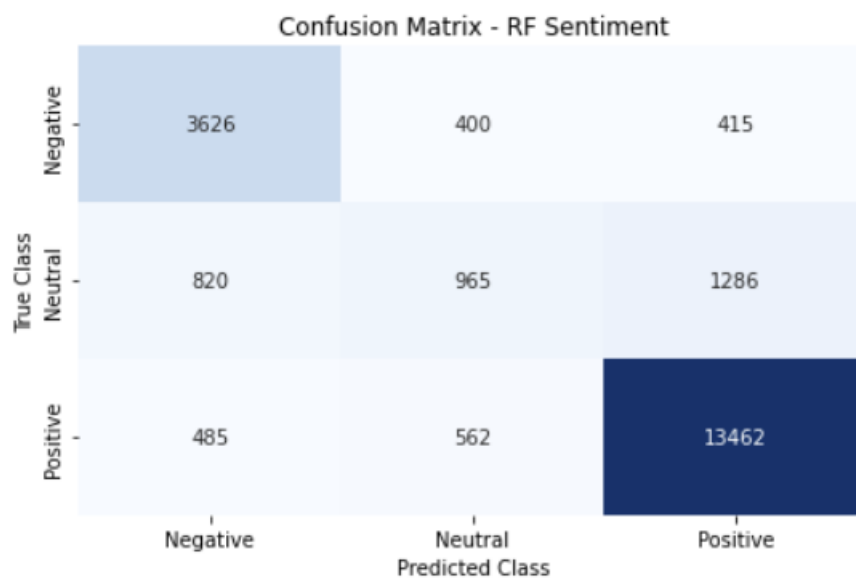
	precision	recall	f1-score	support
1 star	0.56	0.77	0.65	2275
2 stars	0.47	0.31	0.38	2166
3 stars	0.47	0.39	0.42	3071
4 stars	0.51	0.48	0.49	5939
5 stars	0.70	0.77	0.73	8570
accuracy			0.59	22021
macro avg	0.54	0.54	0.54	22021
weighted avg	0.58	0.59	0.58	22021

16. Random Forest sentiment accuracy

```
round(accuracy_score(y_true=test_reviews["sentiment"], y_pred=p_final_rf_sentiment), 4)
```

0.8198

17. Random Forest sentiment - confusion matrix



18. Random Forest sentiment – all metrics

```
print(classification_report(test_reviews["sentiment"], p_final_rf_sentiment, target_names=target_names))
```

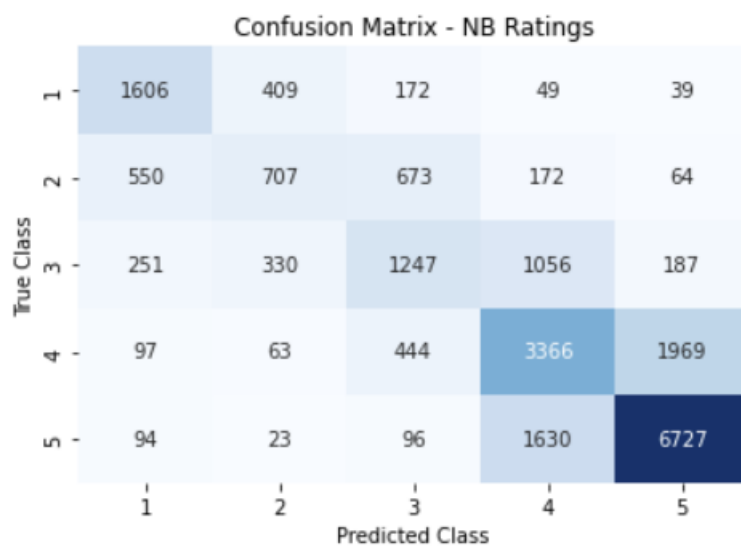
	precision	recall	f1-score	support
Negative	0.74	0.82	0.77	4441
Neutral	0.50	0.31	0.39	3071
Positive	0.89	0.93	0.91	14509
accuracy			0.82	22021
macro avg	0.71	0.69	0.69	22021
weighted avg	0.80	0.82	0.81	22021

19. Naïve Bayes ratings accuracy

```
round(accuracy_score(y_true=test_reviews["rating"], y_pred=p_final_mnb_ratings), 4)
```

0.62

20. Naïve Bayes ratings - confusion matrix



21. Naïve Bayes ratings – all metrics

```
print(classification_report(test_reviews["rating"], p_final_mnb_ratings, target_names=target_names))
```

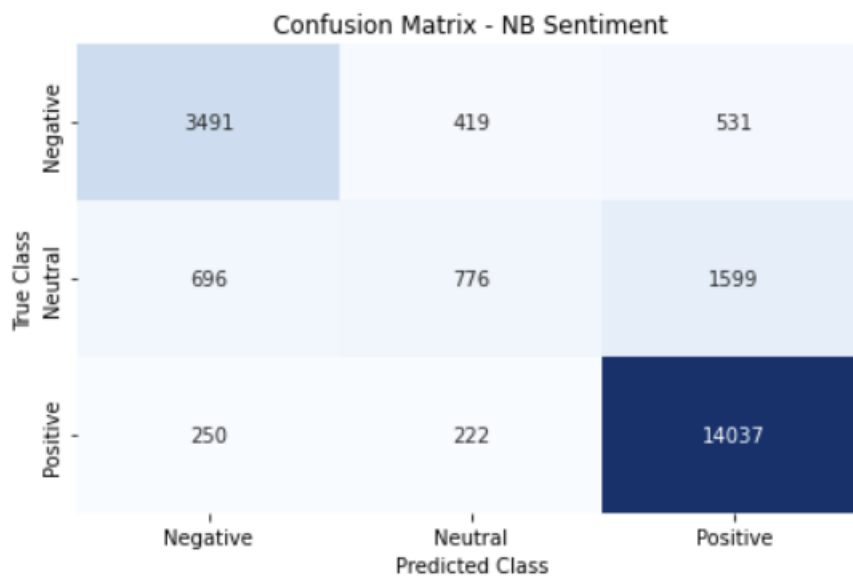
	precision	recall	f1-score	support
1 star	0.62	0.71	0.66	2275
2 stars	0.46	0.33	0.38	2166
3 stars	0.47	0.41	0.44	3071
4 stars	0.54	0.57	0.55	5939
5 stars	0.75	0.78	0.77	8570
accuracy			0.62	22021
macro avg	0.57	0.56	0.56	22021
weighted avg	0.61	0.62	0.61	22021

22. Naïve Bayes sentiment accuracy

```
round(accuracy_score(y_true=test_reviews["sentiment"], y_pred=p_final_mnb_sentiment), 4)
```

0.8312

23. Naïve Bayes sentiment - confusion matrix



24. Naïve Bayes sentiment – all metrics

```
print(classification_report(test_reviews["sentiment"], p_final_mnb_sentiment, target_names=target_names))
```

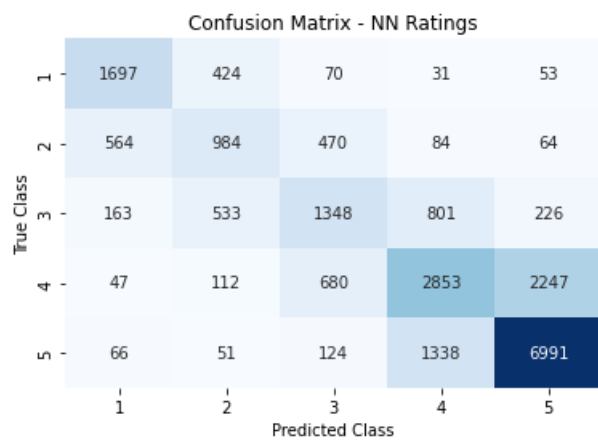
	precision	recall	f1-score	support
Negative	0.79	0.79	0.79	4441
Neutral	0.55	0.25	0.35	3071
Positive	0.87	0.97	0.92	14509
accuracy			0.83	22021
macro avg	0.73	0.67	0.68	22021
weighted avg	0.81	0.83	0.81	22021

25. Neural Networks ratings accuracy

```
round(accuracy_score(y_true=y_test, y_pred=p_final_nn_ratings), 4)
```

0.63

26. Neural Networks ratings - confusion matrix



27. Neural Networks ratings – all metrics

```
print(classification_report(y_test, p_final_nn_ratings, target_names=target_names))
```

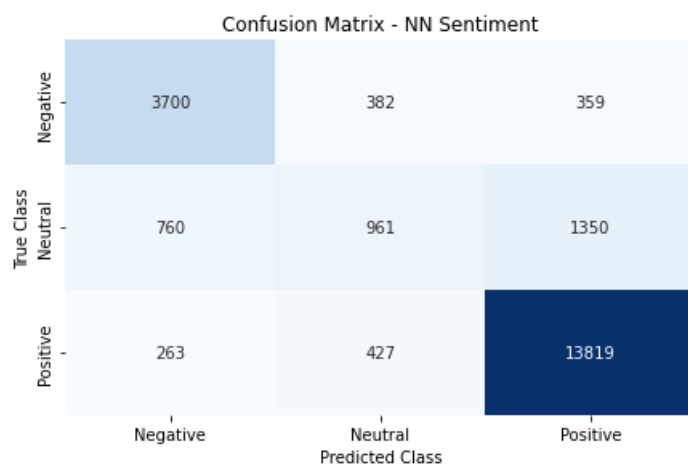
	precision	recall	f1-score	support
1 star	0.67	0.75	0.71	2275
2 stars	0.47	0.45	0.46	2166
3 stars	0.50	0.44	0.47	3071
4 stars	0.56	0.48	0.52	5939
5 stars	0.73	0.82	0.77	8570
accuracy			0.63	22021
macro avg	0.59	0.59	0.58	22021
weighted avg	0.62	0.63	0.62	22021

28. Neural Networks sentiment accuracy

```
round(accuracy_score(y_true=y_test, y_pred=p_final_nn_sentiment), 4)
```

0.8392

29. Neural Networks sentiment - confusion matrix



30. Neural Networks sentiment – all metrics

```
print(classification_report(y_test, p_final_nn_sentiment, target_names=target_names))
```

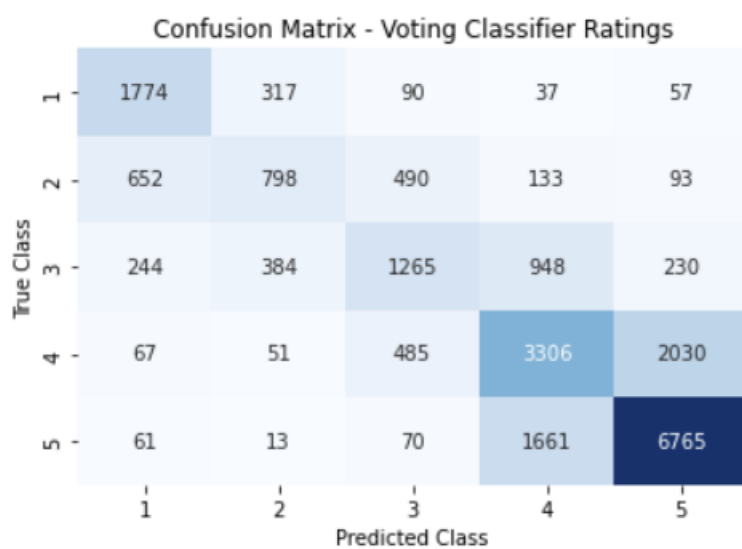
	precision	recall	f1-score	support
Negative	0.78	0.83	0.81	4441
Neutral	0.54	0.31	0.40	3071
Positive	0.89	0.95	0.92	14509
accuracy			0.84	22021
macro avg	0.74	0.70	0.71	22021
weighted avg	0.82	0.84	0.82	22021

31. Voting Classifier ratings accuracy

```
round(accuracy_score(y_true=test_reviews["rating"], y_pred=p_voting_rating), 4)
```

0.6316

32. Voting Classifier ratings - confusion matrix



33. Voting Classifier ratings – all metrics

```
print(classification_report(test_reviews["rating"], p_voting_rating, target_names=target_names))
```

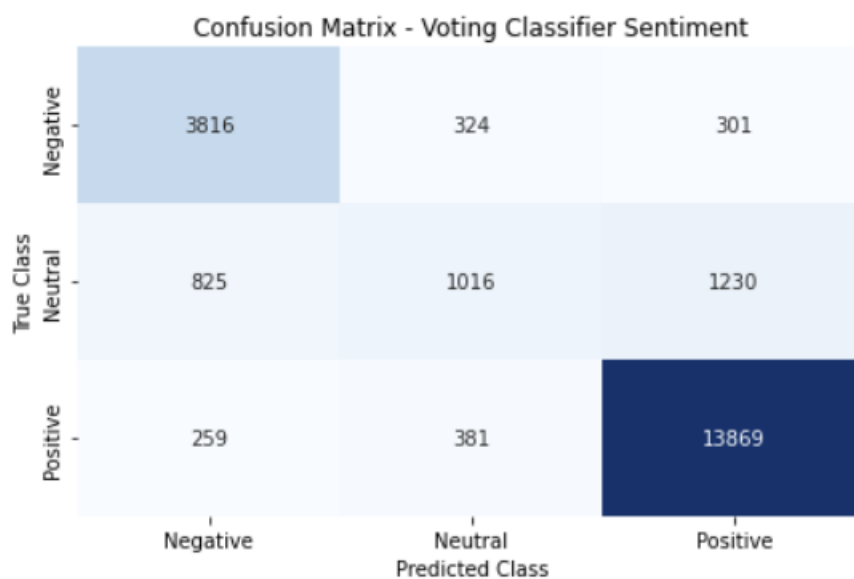
	precision	recall	f1-score	support
1 star	0.63	0.78	0.70	2275
2 stars	0.51	0.37	0.43	2166
3 stars	0.53	0.41	0.46	3071
4 stars	0.54	0.56	0.55	5939
5 stars	0.74	0.79	0.76	8570
accuracy			0.63	22021
macro avg	0.59	0.58	0.58	22021
weighted avg	0.62	0.63	0.62	22021

34. Voting Classifier sentiment accuracy

```
round(accuracy_score(y_true=test_reviews["sentiment"], y_pred=p_voting_sentiment), 4)
```

0.8492

35. Voting Classifier sentiment - confusion matrix



36. Voting Classifier sentiment – all metrics

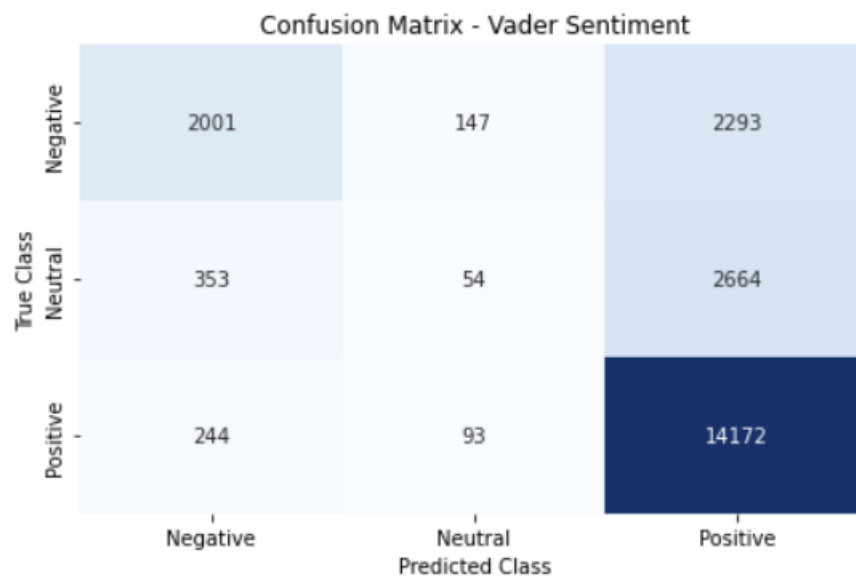
```
print(classification_report(test_reviews["sentiment"], p_voting_sentiment, target_names=target_names))
```

	precision	recall	f1-score	support
Negative	0.78	0.86	0.82	4441
Neutral	0.59	0.33	0.42	3071
Positive	0.90	0.96	0.93	14509
accuracy			0.85	22021
macro avg	0.76	0.72	0.72	22021
weighted avg	0.83	0.85	0.83	22021

37. Vader sentiment accuracy

```
round(accuracy_score(y_true=test_reviews_vader["sentiment"], y_pred=test_reviews_vader["vader_sentiment"]), 4)
0.7369
```

38. Vader sentiment – confusion matrix



39. Vader sentiment – all metrics

```
print(classification_report(test_reviews_vader["sentiment"], test_reviews_vader["vader_sentiment"], target_names=target_names))
```

	precision	recall	f1-score	support
Negative	0.77	0.45	0.57	4441
Neutral	0.18	0.02	0.03	3071
Positive	0.74	0.98	0.84	14509
accuracy			0.74	22021
macro avg	0.56	0.48	0.48	22021
weighted avg	0.67	0.74	0.67	22021