Department of Artificial Intelligence and Data Science

A PROJECT BASED LABORATORY REPORT

Title: Sentiment Analysis

Submitted By:

2100039004 - I. TABU CHOUDARY

2100039125 - M. VYSHNAVI

2100032073 - K. RENUKA RAJESWARI

2100032228 - K. SAI SRI MAHA LAKSHMI

**Content**

1. Abstract
2. Introduction
3. Project Description
4. Dataset Preprocessing
   1. Data Collection
   2. Data Exploration
   3. Text Preprocessing
   4. Model Training
5. Results & Analysis
6. References

**1 Abstract**

Natural Language Processing (NLP) techniques have gained significant traction in recent years in the realm of sentiment analysis. In this project, our group presents a sentiment analysis classifier for restaurant reviews, using NLP methodologies. The objective is to automatically categorize reviews into positive, negative, or neutral sentiments, aiding consumers in making informed dining decisions and assisting restaurant owners in gauging customer satisfaction.

We use a dataset comprising a diverse range of reviews, ensuring the classifier's robustness across different user demographics. Our approach integrates pre-processing techniques such as tokenization, stop-word removal to enhance the quality of textual data.

Feature engineering encompasses methods like tokenization, Bag-of-Words, which facilitate effective representation of review semantics. We evaluate the classifier's performance using metrics such as accuracy to ensure reliable generalization. Furthermore, we explore the impact of different machine learning algorithms such as Multinomial Naive Bayes Classifier, and deep learning architectures on sentiment classification accuracy.

In a nutshell, our project contributes to the advancement of NLP-driven sentiment analysis techniques in the domain of restaurant reviews, fostering improved user experiences and business insights in the culinary industry.

**2 Introduction**

Natural Language Processing (NLP) is a field of artificial intelligence (AI) dedicated to enabling computers to comprehend, process and generate human speech, similar to human perception. It involves many tasks ranging from word tokenization to complex tasks such as sentiment analysis and machine translation. This field aims to bridge the gap between human language and machine capabilities which can be used in the future for seamless interaction between both entities.

There are many applications of NLP and in most aspects of life. NLP techniques enable the extraction of valuable insights from large corpuses and facilitate decision-making processes in fields such as business intelligence and healthcare. This field of AI is also used most frequently in voice-controlled devices, virtual assistants and chatbots which enable users to interact with technology in a natural manner.

NLP can help to uncover underlying principles of language, evolution, etc by analysing linguistic patterns, semantic structures and coherence in large datasets of text. This serves as a tool for both practical utility and an “understanding” into the complexities of human language and thought process.

In essence, Natural Language Processing is multidisciplinary in nature and has practical benefits. The combination of linguistics, data science and machine learning elements result in a thoughtful and systematic approach that helps with problem-solving skills.

**3 Project Description**

This project focuses on conducting sentiment analysis on a dataset comprising restaurant reviews. Sentiment analysis involves deciphering the sentiment expressed in textual data, in this case, restaurant reviews, to classify them as positive, negative, or neutral. By using Natural Language Processing (NLP) techniques and machine learning algorithms, the project aims to automate the process of understanding customer sentiment towards various aspects of dining experiences.

Our choice for this project was driven by two motivations that are rooted in how technology and customer experience are intertwined. Online reviews are considered as a means of shaping customer’s feedback. Platforms such as TripAdvisor leverage on the insights contained in such reviews and is essential for business to succeed in a competitive market. The project aims to fulfill practical needs and provide an application to interpret customers’ opinions using concepts of Natural Language Processing (NLP) and Machine Learning (ML).

Our project uses algorithms such as Multinomial Naive Bayes Classifier, which is a popular machine learning algorithm for text classification. It is useful for problems that involve extraction of information from unstructured text data, hence, providing the stakeholders of the restaurant with actionable intelligence. The appeal of this project lies in its ability to connect the dots between raw data and actionable insights, enabling companies to make informed decisions by having a better understanding of customer behaviour.

**4 Dataset Preprocessing**

**4.1 Data Collection**

Original Dataset Name: Restaurant\_Reviews.tsv

Converted Dataset Name: Restaurant\_Reviews.csv

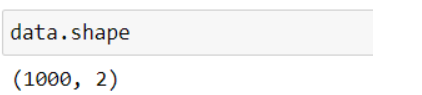
Link for dataset: <https://www.kaggle.com/datasets/vigneshwarsofficial/reviews>

We have converted the original dataset that was in TSV format to CSV to make it easier to work with in our project.

**4.2 Data Exploration**

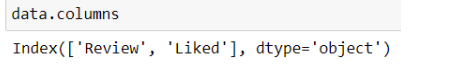
This section aims to explain what we have done with the dataset that we found in the previous section.

*Dataset Shape*

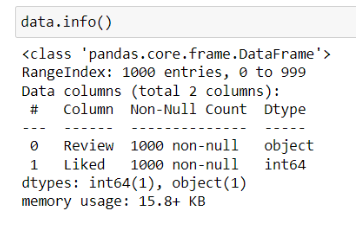


This shows that there are a total of 1000 rows of data and only 2 columns in the entire reviews dataset.

*Column Labels*



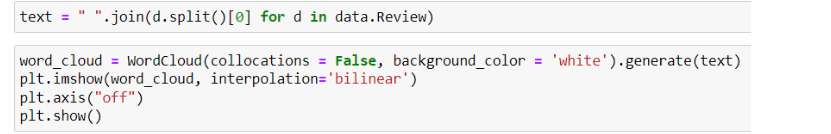
The labels (“Review” and “Liked”) of the columns are of object data type. This means that a fixed-size block of memory corresponds to the array item that should be interpreted.

*Summarised information*  


This *info()* method is used to show the summarised information/quick overview of the data frame. As seen from the above code, there are no null values present in both columns. This means that there is no need to clean the dataset to remove null values.

To depict the most used words in the dataset, we split the text present in each row, into singular words (also known as tokens) and use a Word Cloud to depict them accordingly.

Word cloud is a data visualisation technique that helps to communicate important information at a glance. The more a word appears in the dataset/corpus, the larger and bolder the word is depicted in the word cloud. Hence, word clouds are used when you want to quickly identify a theme in a large body of text, when you want to understand the overall sentiment of the text, explore patterns in the dataset and communicate key ideas in a visually engaging way.





As shown from the above screenshot of the code, the most frequently used word is “Great”, followed by “Food” and “Service”.

The code selects the text of the first review from a dataframe and then creates a word cloud object to generate an image from the given text using the *generate()* function. The collocations attribute in *WordCloud()* function is used to eliminate words that are frequently grouped together in the text.

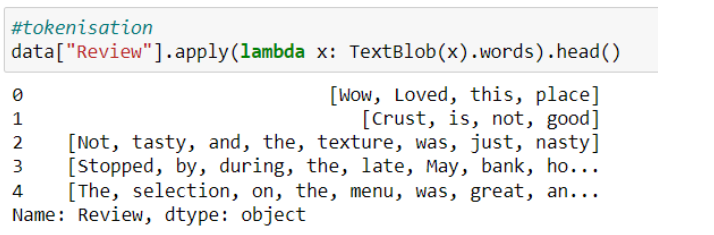
The interpolation attribute in *imshow()* function specifies the method of displaying the image which is ‘bilinear’. This method calculates the colour of each pixel by calculating the weighted average of the nearest four pixels in the original image.

**4.3 Text Preprocessing**

This section aims to explain the different methods used in the project to preprocess the text in the dataset that we have obtained.

**4.3.1 Tokenization**

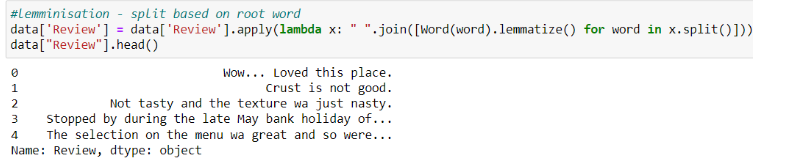
Tokenization is the process of breaking down complex words into smaller tokens which can be words, sentences, etc. It is considered as the first step since it sets the foundation for the remaining processes that take place in Sentiment Analysis.



Here, we use the “TextBlob” package, which is a simple and convenient API that can be used in tasks such as tokenization. As seen, the output of the code are tokens (words) that have been broken down individually from their original sentence.

**4.3.2 Lemmatization**

Lemmatization is a pre-processing technique used to break a word down into its root meaning to identify similarities. It is a reliable form of preprocessing as the word will return to its valid base form, compared to Stemming which only removes the prefix and suffix of the text.



As seen from the code above, we used the *split()* function to split the text in “x” (the input values) into individual words. A word object is created using the *Word()* function and the *lemmatize()* function is applied to the word to reduce it to its base form.

**4.3.3 Term Frequencies**

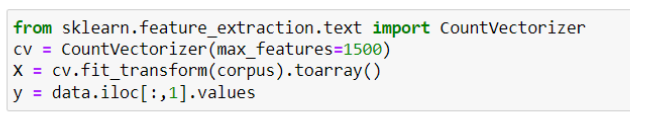
Term frequencies are used to represent the importance of individual words within a large corpus of documents. It does this by measuring how frequently a term (word) occurs in a document relative to the total number of words in that document.



Here, we access the “Review” column in the dataset. The *pd.value\_counts()* method returns a pandas Series where the index is the unique number of words in the document and the values are their corresponding frequencies.  
By using *reset\_index()*, we are able to reset the index of the resulting Dataframe. By default, the index would be the unique words from the reviews and hence, resetting would result in a regular column and new default integer index.

**4.4 Model Training**

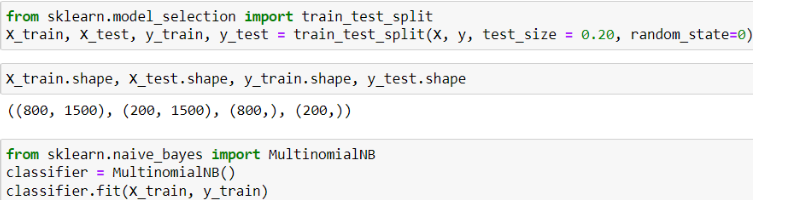
Firstly, we extracted features from a text and the CountVectorizer() tool is used to convert a collection of text documents into a matrix of token counts.



In the above code, the “max\_features” attribute is used to specify the maximum number of features/words to be extracted and included in the resultant matrix.

Secondly, we split the dataset into “training” and “testing” data We split the dataseet using to avoid overfitting, using the *rain\_test\_split()* function in the sci-kit learn package. Overfitting happens when the model fits its training data too well and fails to work on unseen and new data.

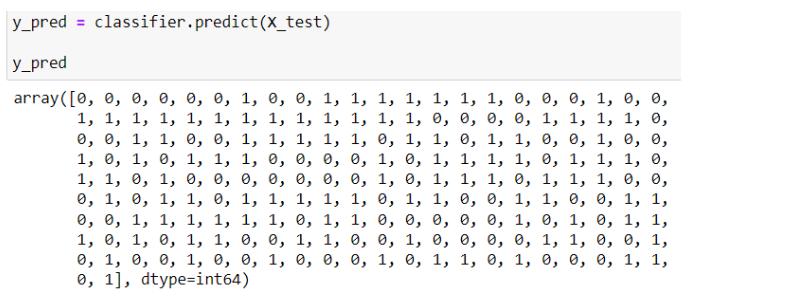
We fit the previously split data into the classifier for it to train using it on the input values (x) to get output values (y).



In our project, we used Multinomial Naive Bayes Classifier (Multinomial NB) as it is a well-suited algorithm for text classification tasks, especially in scenarios where tokens are discrete and can only be figured out based on frequency counts.

It can also be used to handle multiple categories such as POSITIVE, NEUTRAL or NEGATIVE sentiments in these reviews. In addition, the algorithm is computationally efficient when handling large corpuses/datasets.

Thirdly, we use the *predict()* function to get predictions based on the trained model. As seen, we use the “x\_test” data to create these predictions. The 0s and 1s represent the output of the predicted data where 0 stands for false and 1 stands for true.



As seen from the above image, the output is an array of predicted binary class labels (0 and 1) generated by the classifier and each element in the array corresponds to a prediction for a particular data point in the test set.

In this case, the output is either “Did not like” (0) or “Like” (1) the food and the restaurant.

**5 Results & Analysis**

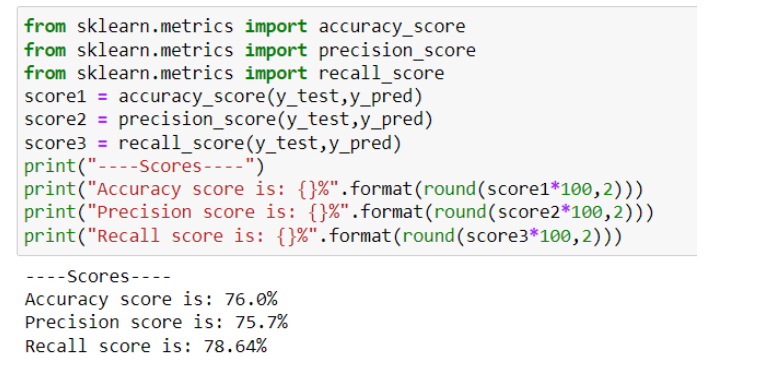
This section aims to delve into the results of the model that we have trained and the inferences/analysis we have made based on the said results.

Based on the predicted values in the above code, we use common evaluation metrics such as accuracy score, precision score, recall score, confusion matrix on our model to see how well it was able to train and work on unseen data.

Accuracy score measures the proportion of correct predictions among the total number of predictions made by the model. Accuracy = (TP + TN)/(TP + TN + FP + FN).

Precision score measures the proportion of true positive predictions among the total number of instances predicted as positive by the model. Precision = TP / (TP + FP)

Recall score measures the true positive predictions among the actual positive instances of the model. Recall = TP / (TP + FN)

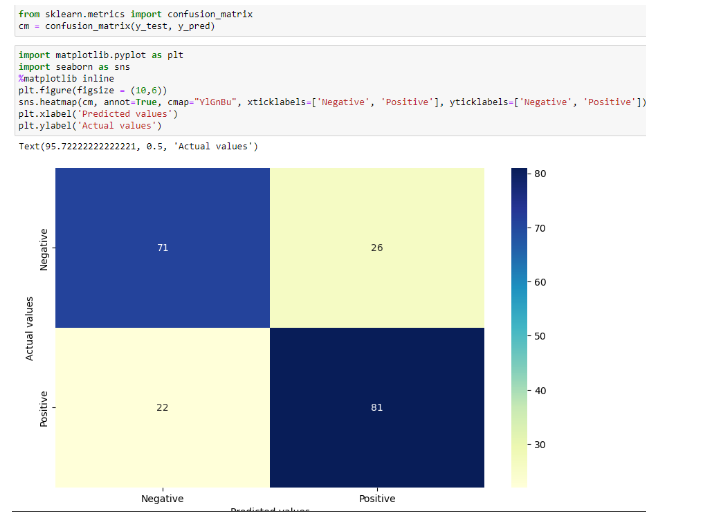


As seen from the above image, we can see that the model has performed fairly well in making the correct predictions, with an accuracy score of 76.0%.

This means that the model is able to fairly predict the sentiment of the review such as “Positive”, “Neutral”,  “Negative”.

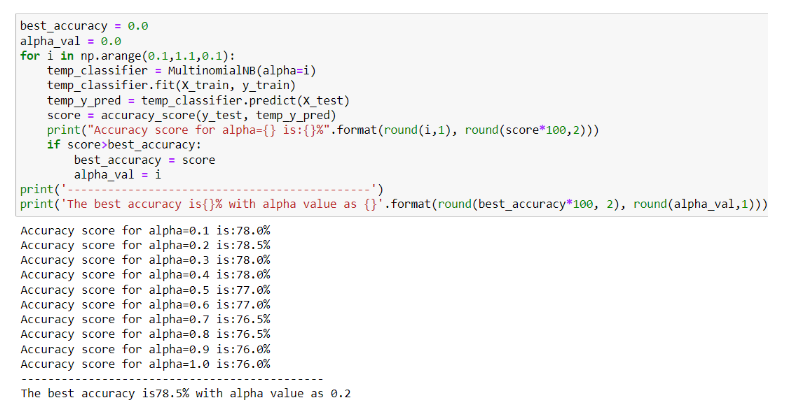
In the image below, we created a confusion matrix to assess the performance of the Multinomial NB classifier by summarising the predictions it makes compared to the actual labels of the data.

The confusion matrix is created from the sci-kit learn package’s metrics module. In this case, the labels are “Negative” and “Positive” with data from the y\_test and y\_pred dataframes. This means that the values that are being compared are the predicted and the actual values.



Based on the output above, we can see that 71 values are true positives, meaning that the positive predicted values correctly match the negative actual values. 81 values are true negatives, meaning that the negative predicted values correctly match the negative actual values.

The code below performs a grid search over different values of the “alpha” hyperparameter for the Multinomial NB classifier. It eventually prints out the accuracy score for each value of alpha and identifies the best accuracy achieved.



The hyperparameter is a smoothing parameter that helps to handle unseen words in the test set. Tuning that parameter can impact the performance of the classifier especially when the distribution of words differs between the training and test data. Hence, the code is trying to find the optimal value that maximises the accuracy on the test set.

The code calculates and prints the accuracy score for each value of “alpha” allowing the practitioner to evaluate how well the classifier performs at different smoothing levels.