Sentiment Analysis of Amazon Product Database

**Group Members**

Akshay Phapale: 001316563

Kaushik Patil: 001056490

Sachin: 001054734

INDEX

1. Introduction…………………………………………………………………………….3
2. Objective…………………………………………………………………………………4
3. Dataset……………………………………………………………………………………5
4. Importing Data………………………………………………………………………6
5. Data preprocessing………………………………………………………………….7
   1. NaN value Identification and Removal………………………. 9
   2. Duplicate value Identification and Removal.……………….9
   3. Data Balancing…………………….……………………………………10
6. TF-IDF Vectorizer…………………….…………………………………………….11
7. Algorithms & Implementation……………………………………………….12
   1. Multinomial Naïve Bayes………………………………………….13
   2. Random Forest…………………………………………………………14
   3. Decision Tree……………………………………………………………17
   4. Logistic Regression…………………………………………………..18
   5. SVM…………………………………………………………………………19
   6. KNN………………………………………………………………………….20
8. Conclusion…………………………………………………………………………….21
9. References…………………………………………………………………………….22

1. Introduction

One of the most important elements for businesses is being in touch with their customer base. It is vital for these firms to know exactly what consumers or clients think of new and established products or services, recent initiatives, and customer service offerings.

**Sentiment analysis is one way to accomplish this necessary task.**

Sentiment Analysis is a field of Natural Language Processing (NLP) that builds models that try to identify and classify attributes of the expression e.g.:

* **Polarity**: if the speaker expresses a positive or negative opinion,
* **Subject**: the thing that is being talked about,
* **Opinion holder**: the person, or entity that expresses the opinion.

In a world where we generate quintillion bytes of data every day, sentiment analysis has become a key tool for making sense of that data. This has allowed companies to get key insights and automate all kinds of processes.

Sentiment Analysis can help to automatically transform the unstructured information into structured data of public opinions about products, services, brands, politics or any other topic that people can express opinions about. This data can be very useful for commercial applications like **marketing analysis, public relations, product reviews, net promoter scoring, product feedback, and customer service.**

In this project, we will show you how to implement various Supervised Learning algorithms that can classify Amazon product reviews as positive or negative. The model will take a whole review as an input and provide percentage ratings for checking whether the review conveys a positive or negative sentiment.

1. Objectives

* Analyzing and performing feature engineering on the Amazon product review dataset.
* Analyze sentiment on the dataset from the feature level.
* Classification of sentiments into positive or negative using NLP and various classification algorithms.
* Finding the correlation between the Amazon product reviews and the rating of the products given by the customers.

1. Dataset

**Link:** <https://nijianmo.github.io/amazon/index.html>

**Description:**

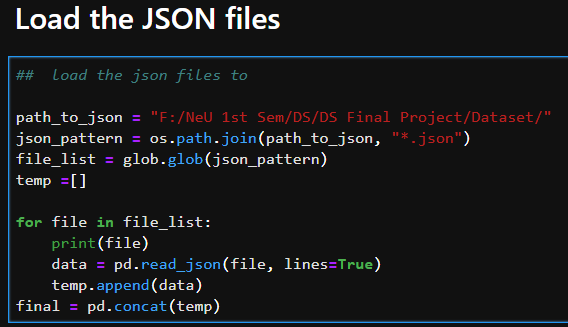
This Dataset is an updated version of the [Amazon review dataset](http://jmcauley.ucsd.edu/data/amazon/index_2014.html) released in 2014. Current data includes reviews in the range of May 1996 - Oct 2018. We will be using “Amazon Fashion”, “Appliances” and “Prime Pantry” dataset for our project.

**Columns:**

* reviewerID - ID of the reviewer
* asin - ID of the product
* reviewerName - the name of the reviewer
* vote - helpful votes of the review
* style - a dictionary of the product metadata, e.g., "Format" is "Hardcover"
* reviewText - text of the review
* overall - the rating of the product
* summary - summary of the review
* unixReviewTime - the time of the review (Unix time)
* reviewTime - the time of the review (raw)

1. Importing Data

The raw files which we used for this project were in JSON format. We downloaded 3 JSON files from the source and stored them in a local path. Then, we used the “*read\_json*” feature of pandas to import these files in DataFrame.



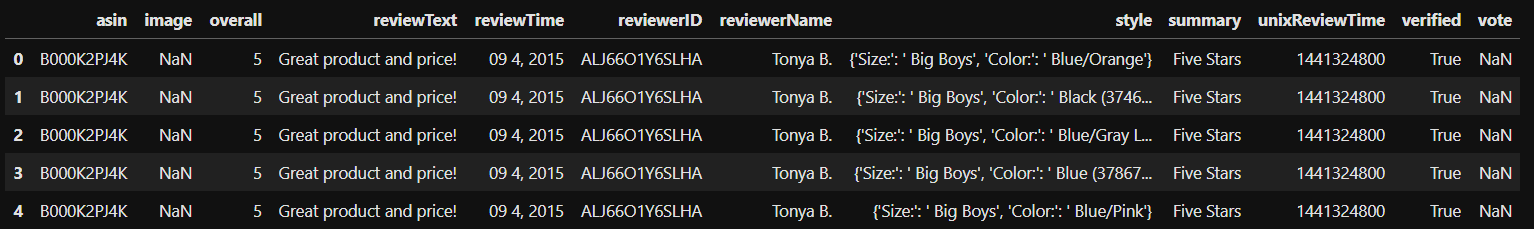
After successful data loading, we analyzed the data and then we came to the conclusion that there is a need to do data preprocessing to remove redundant data.

1. Data Pre-processing

Viewing the general information:

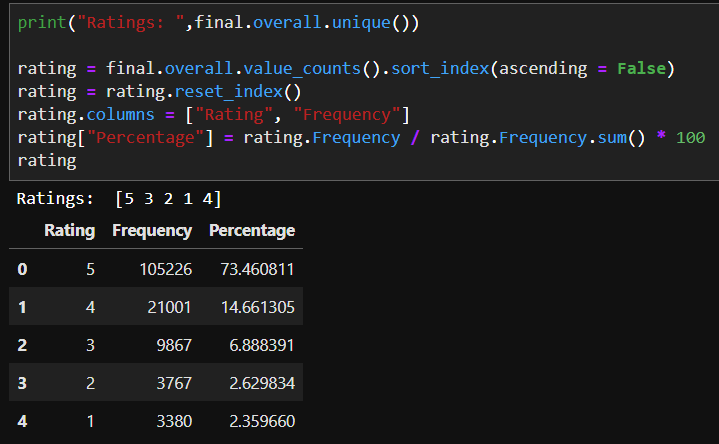
After viewing the general information, we observed below points:

1. The columns **"reviewText" & "summary"** appears to be the sentiments that we will be using for this analysis project
2. There isn’t any other column to determine if the review is positive or negative. We will have to use the column **"overall"** as a road map.

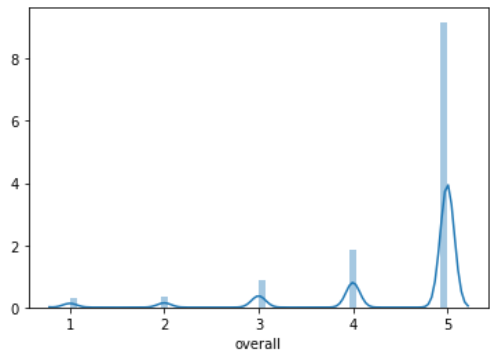
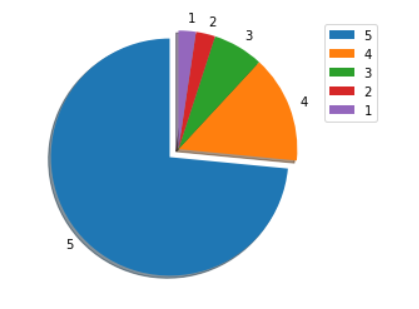


### Ratings:

Let's determine the ratings by breaking it down to the type of ratings the data contains and the distribution of the type of those ratings. Doing so will determine what will be considered to be positive or negative and whether we will have an imbalanced dataset.



Representation of the different types of rating frequency:



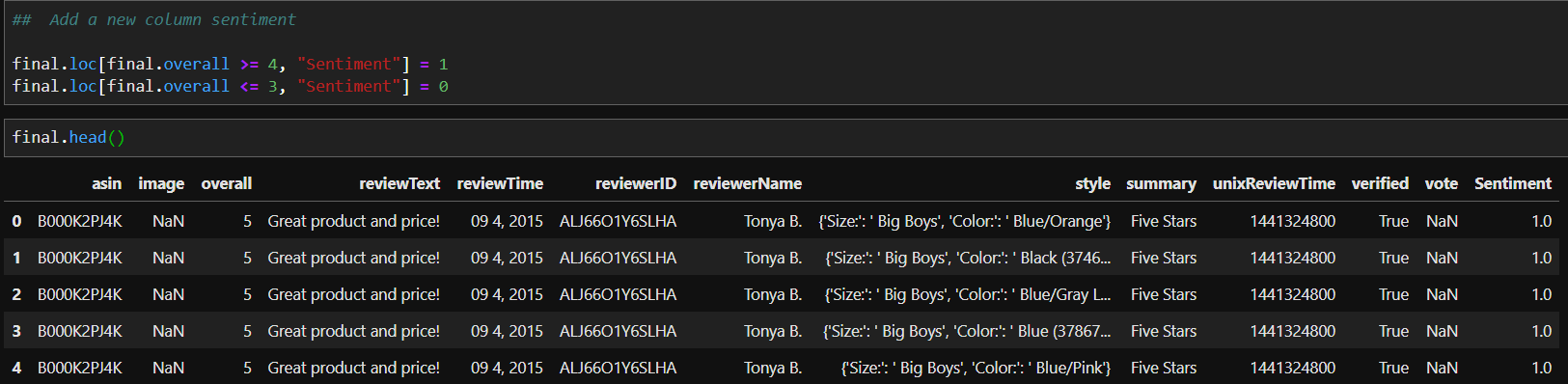
Pie-chart plot Frequency distribution plot

For this project, we will use binary classification for Sentiment analysis.

So, we created a new column “Sentiments” based on the ratings of the product.

We used the below approach:

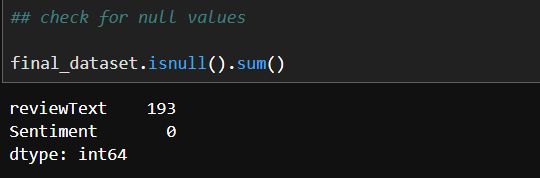
* If rating >=4, sentiment = 1 (positive review)
* If rating <=3, sentiment = 0 (negative review)



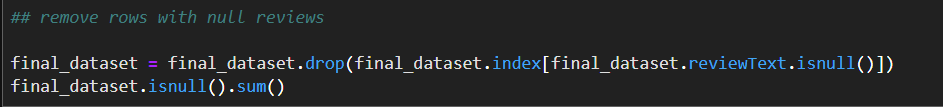
Dataset consists of many duplicates and NaN entries which will affect the prediction model that will produce wrong outcomes.

**5.1 NaN value identification and removal:**

* Identifying the NaN values in the dataset using numpy.isull() function.

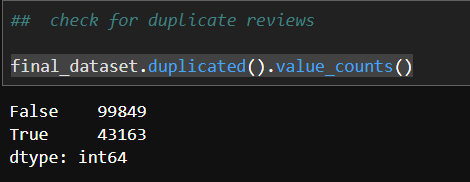


* Dropping these values from the Dataframe using numpy.drop() function.

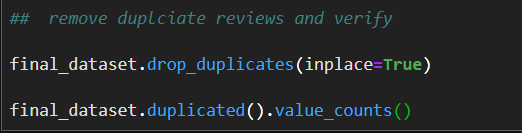


**5.2 Duplicate value identification and removal**

* Identifying the duplicate values in the dataset using numpy.duplicated() function.



* Removing the duplicate values from the dataset using drop\_duplicates() function.

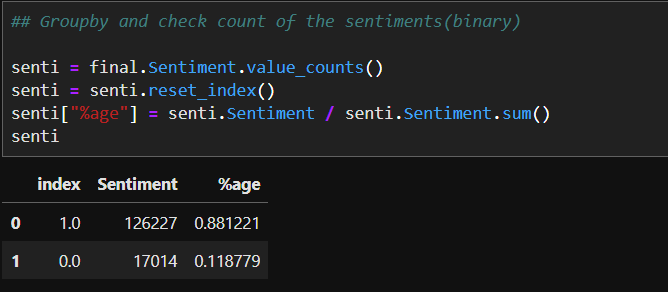


After removing these values, we are ended up getting unbalanced data i.e. we have 80% of records that have positive feedback and only 20% of records have negative feedback. This highly unbalanced data might prone to providing false predictions. Hence, we are balancing the data in the next stage.

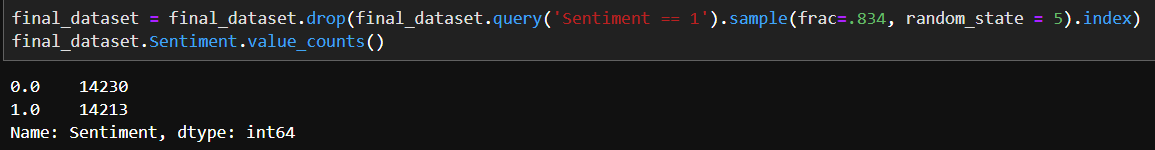
**5.3 Data Balancing**

* Since we cannot generate negative review data. Therefore, we took the fraction of positive review data and then done the analysis.

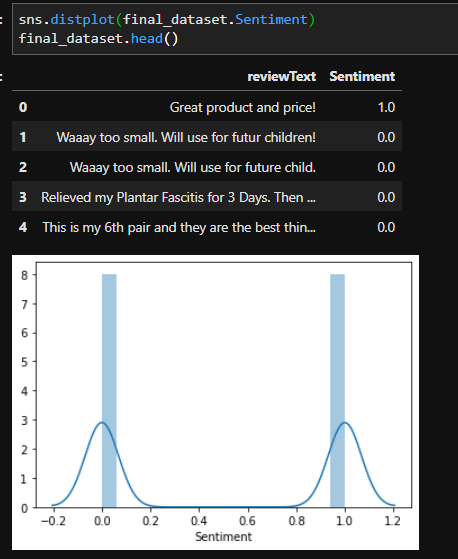
Unbalanced Data:



Balanced Data:

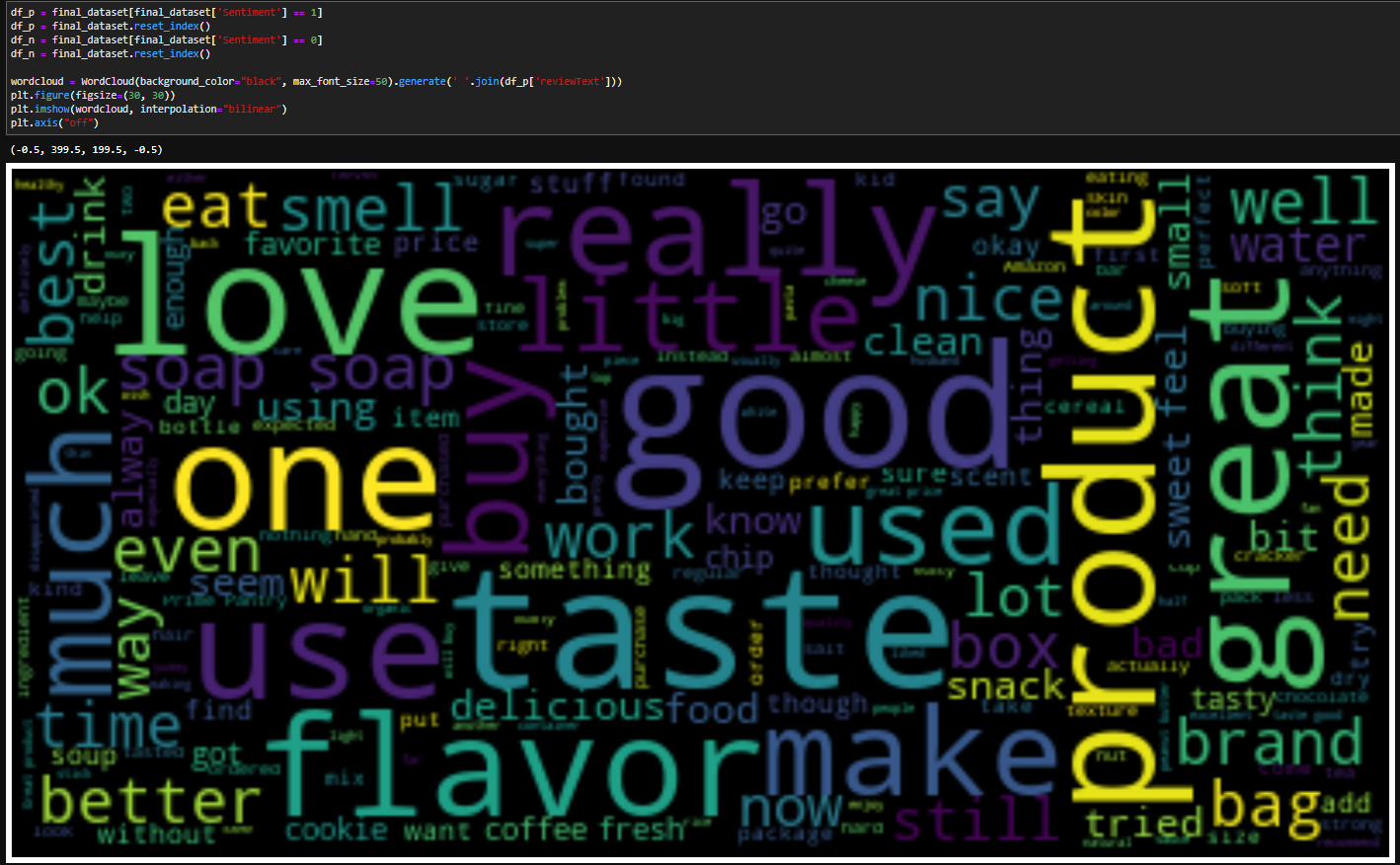


After this step, our dataset contains relevant information that might play an important role in the prediction of sentiment.

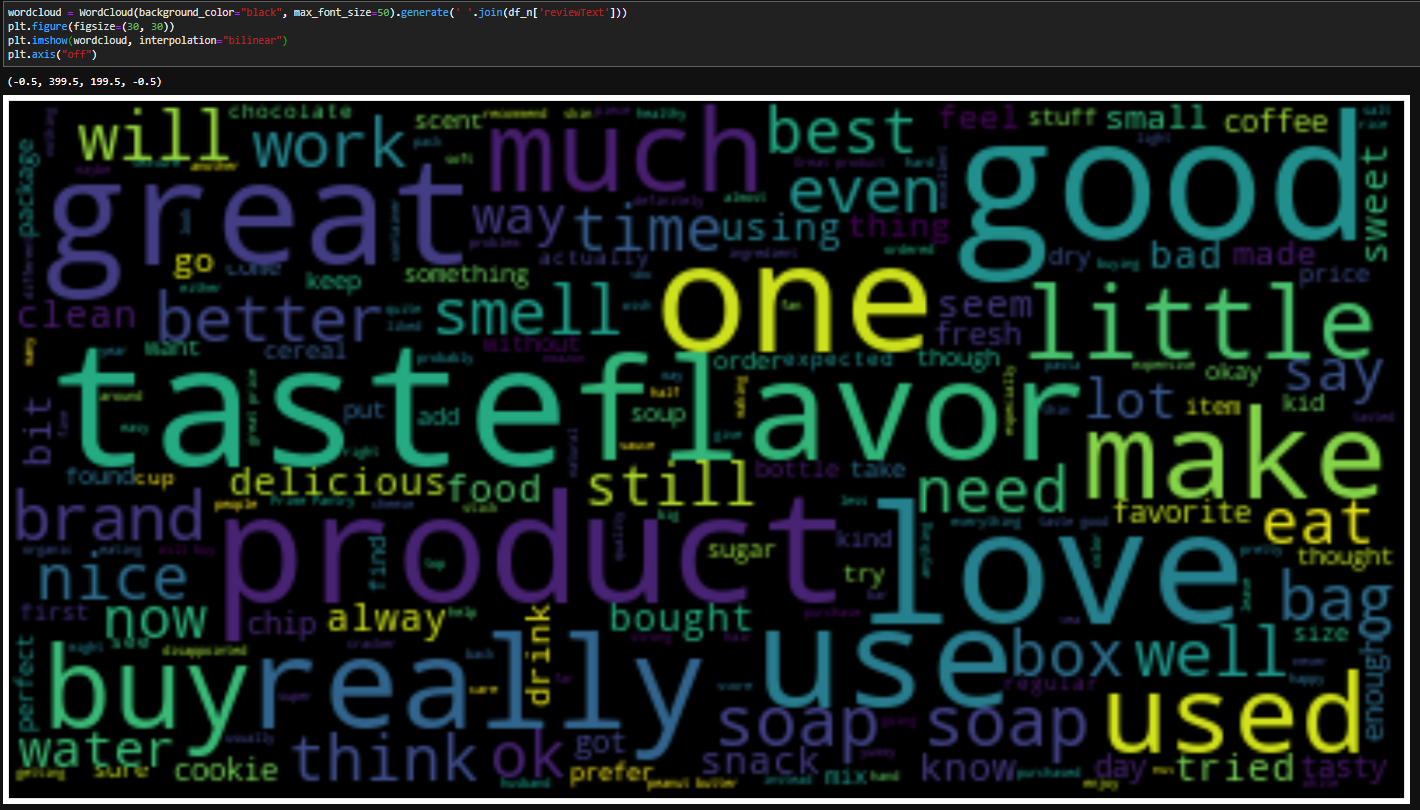


Visualizing positive and negative words with the help of word map.

Positive Word Map



Negative Word map



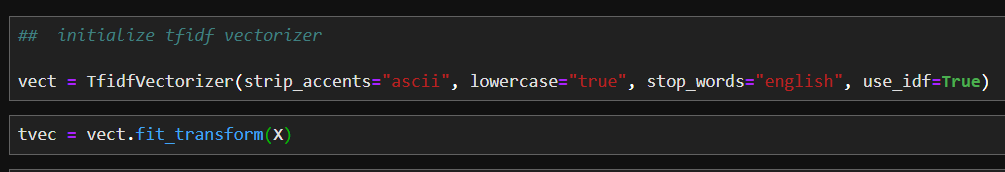
1. TF-IDF Vectorizer

Text data requires special preparation before you can start using it for predictive modeling. The text must be parsed to remove words, called tokenization. Then the words need to be encoded as integers or floating-point values for use as input to a machine learning algorithm, called feature extraction (or vectorization). We can use Count Vectorizer for feature extraction but the issue is that some words like “the” will appear many times and their large counts will not be very meaningful in the encoded vectors. An alternative is to calculate word frequencies, and by far the most popular method is called TF-IDF. This is an acronym that stands for “Term Frequency – Inverse Document” Frequency which are the components of the resulting scores assigned to each word.

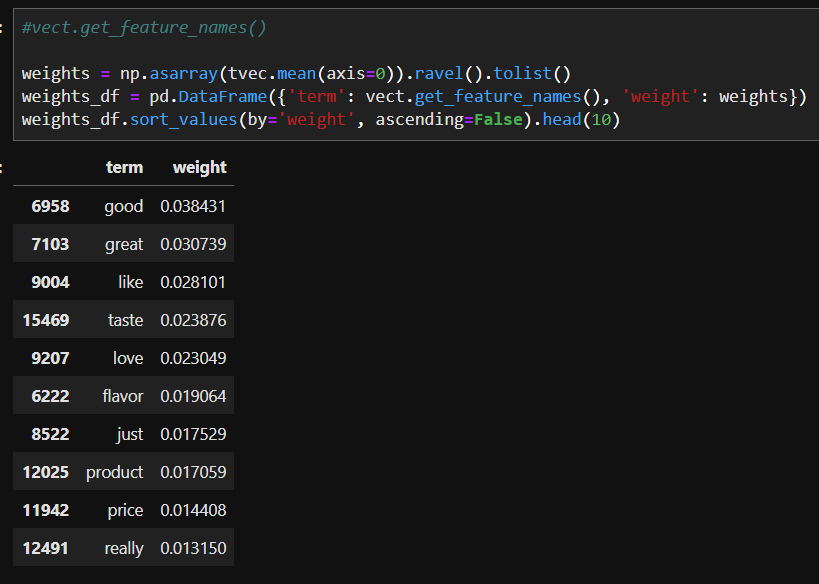
**Term Frequency:** This summarizes how often a given word appears within a document.

**Inverse Document Frequency:** This downscales words that appear a lot across documents.

The TF-IDF Vectorizer will tokenize documents, learn the vocabulary and inverse document frequency weightings, and allow you to encode new documents.



We can see the polarity of the features using the below code:

After this stage, we need to split the data into training and test set using below code sklearn.model\_selection. train\_test\_split. Now, we are going to train supervised learning models using the above dataset.

1. Algorithms & Implementation

Algorithms:

* Multinomial Naïve Bayes
* Random Forest
* Decision Tree
* Logistic Regression
* SVM
* KNN

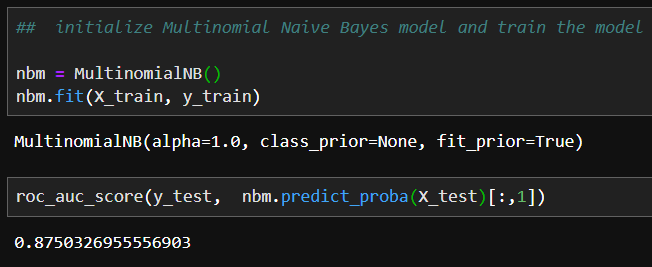
We have successfully implemented the above algorithms. A detailed explanation is provided in the upcoming sections.

7.1 Multinomial Naïve Bayes

Naive Bayes is a statistical classification technique based on Bayes Theorem. It is one of the simplest supervised learning algorithms. Naive Bayes classifier is the fast, accurate and reliable algorithm having high accuracy and speed on large datasets. Naive Bayes is the most straightforward and most potent algorithm. In spite of the significant advances in Machine Learning in the last couple of years, it has proved its worth. It has been successfully deployed in many applications from text analytics to recommendation engines.

Naive Bayes classifier assumes that the effect of a particular feature in a class is independent of other features.

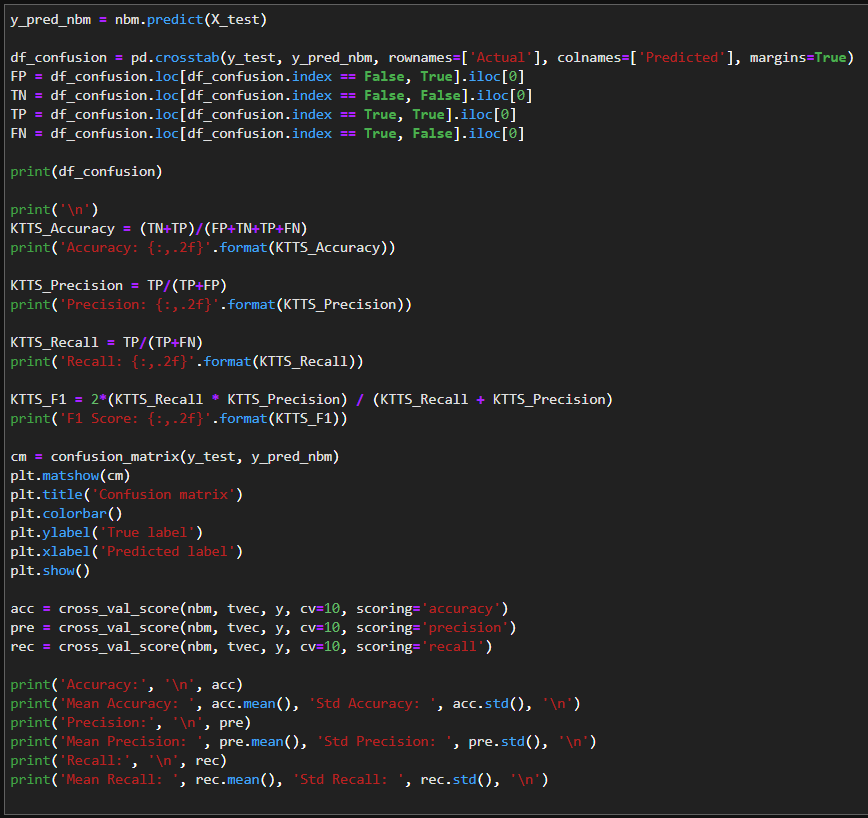
We have used the Multinomial Naïve Bayes model to train and predict our dataset.

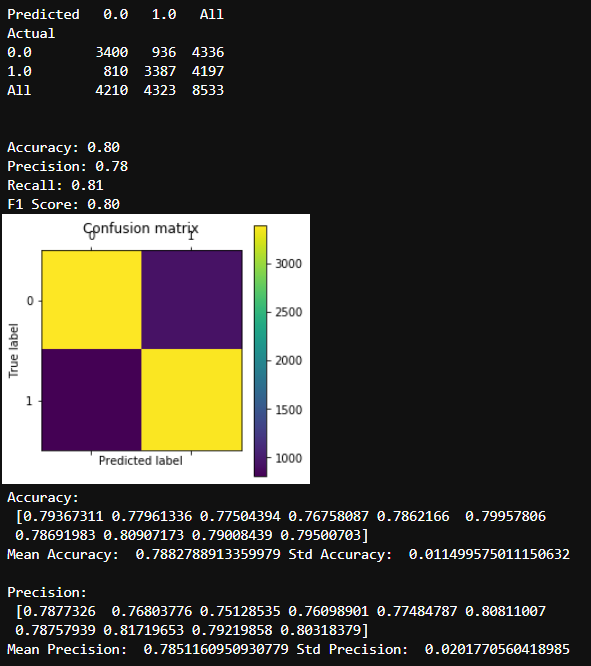


We are using roc\_auc\_score for measuring the accuracy of our model. The roc\_auc\_score always runs from 0 to 1 and is sorting predictive possibilities. 0.5 is the baseline for random guessing, so you want to always get above 0.5.

We were able to attain 87.5% roc\_auc\_score for our dataset.

Confusion Matrix of Multinomial Naïve Bayes:

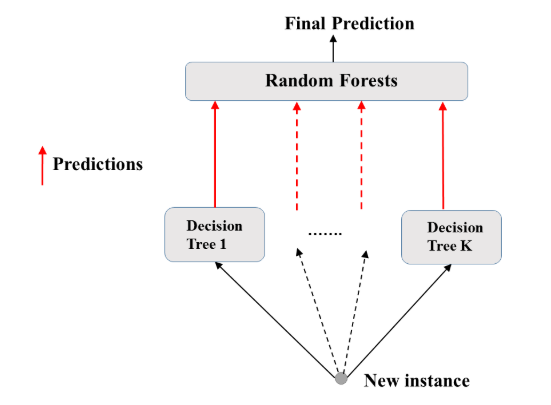




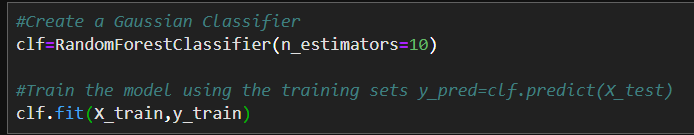
7.2 Random forest

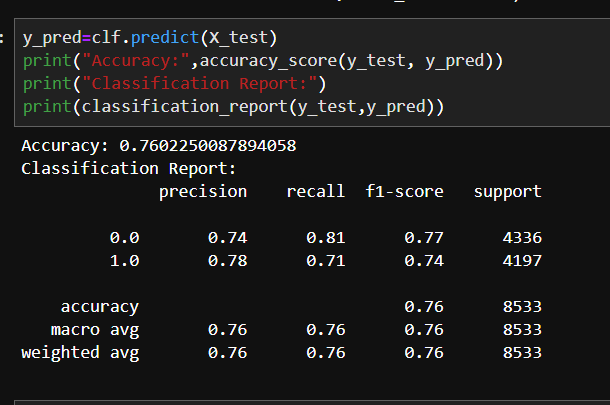
Random forests are a supervised learning algorithm. It can be used both for classification and regression. It is also the most flexible and easy to use. A forest is comprised of trees. It is said that the more trees it has, the more robust a forest is. Random forests create decision trees on randomly selected data samples, get a prediction from each tree and selects the best solution by means of voting. It also provides a pretty good indicator of the feature importance.

Random forests have a variety of applications, such as recommendation engines, image classification, and feature selection. It can be used to classify loyal loan applicants, identify fraudulent activity and predict diseases. The below figure explains the actual working of the Random forests.



Implementation:



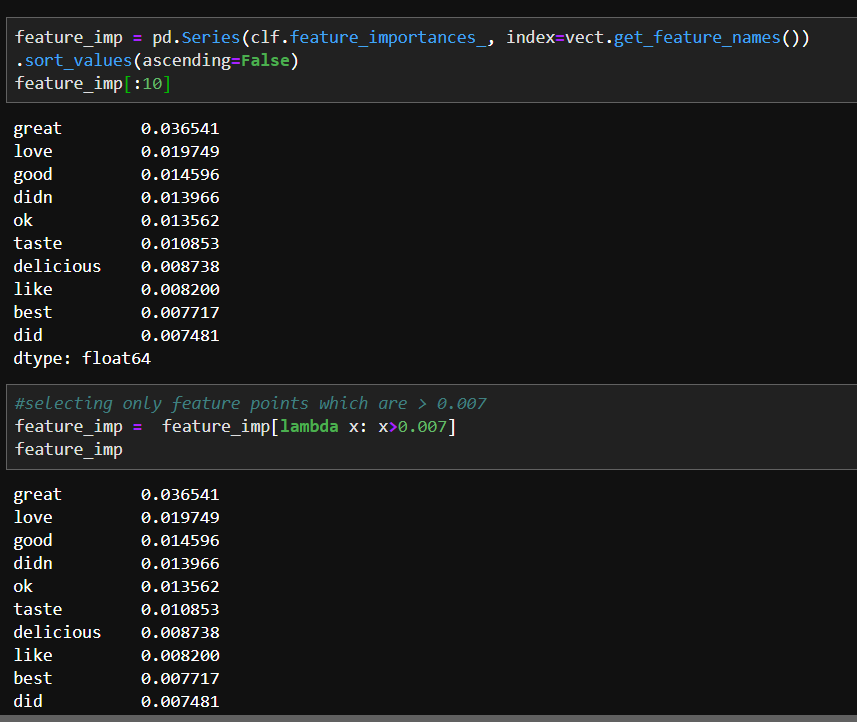


We were able to attain 76% accuracy using random forests i.e. greater than single decision tree prediction.

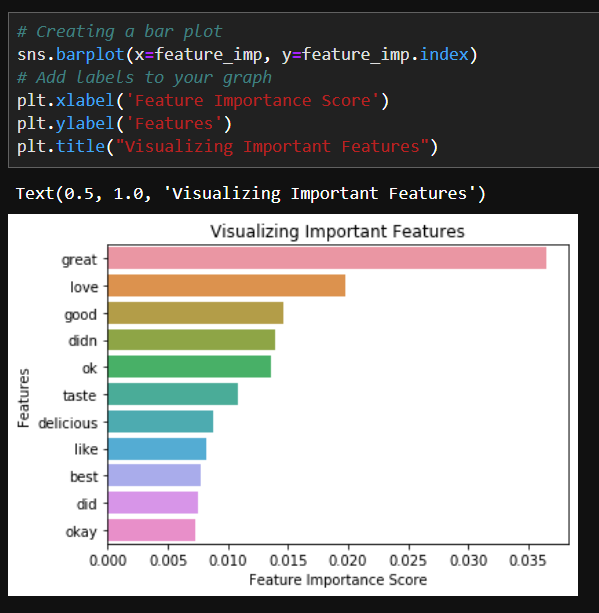
We can increase the accuracy of the decision tree using the following ways:

1. **Feature Importance:**

We need to select only important features that drive the prediction of the review text. By selecting important features, we are removing the redundant data from the calculation. Please find the below screen where we are selecting the features with greater than 0.007 importance.



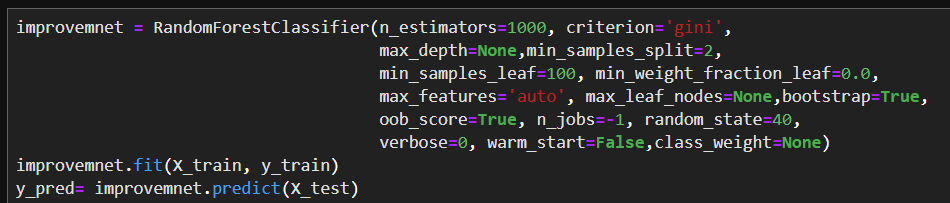
The visual representation of feature importance is as follows:



1. **Algorithm Tuning:**

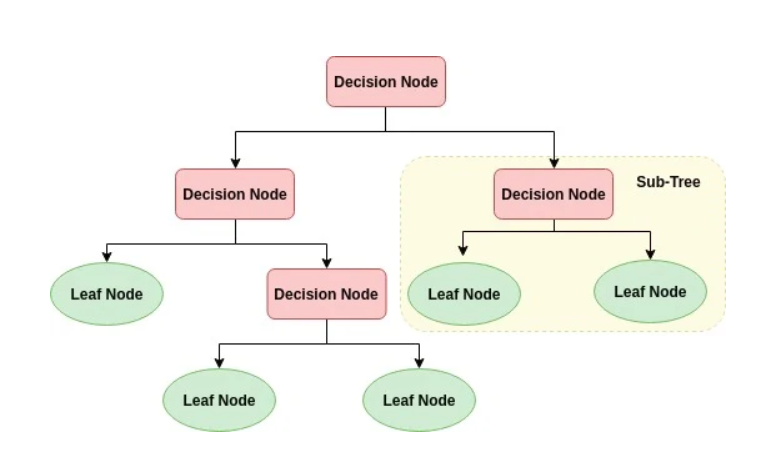
The objective of parameter tuning is to find the optimum value for each parameter to improve the accuracy of the model.

Implementation:

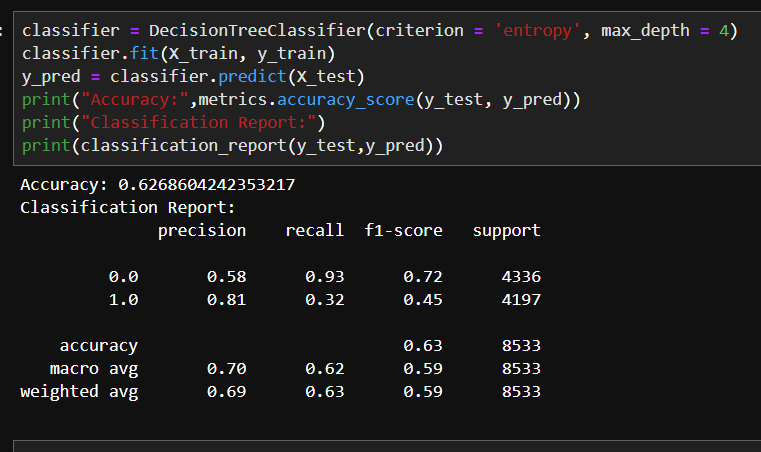


7.3 Decision Tree

A decision tree is a flowchart-like tree structure where an internal node represents feature (or attribute), the branch represents a decision rule, and each leaf node represents the outcome. The topmost node in a decision tree is known as the root node. It learns to partition on the basis of the attribute value. It partitions the tree in a recursive manner called recursive partitioning. This flowchart-like structure helps you in decision making. It's visualization like a flowchart diagram that easily mimics human-level thinking. That is why decision trees are easy to understand and interpret.



Implementation:



We were able to attain 62% accuracy using the Decision Tree model.

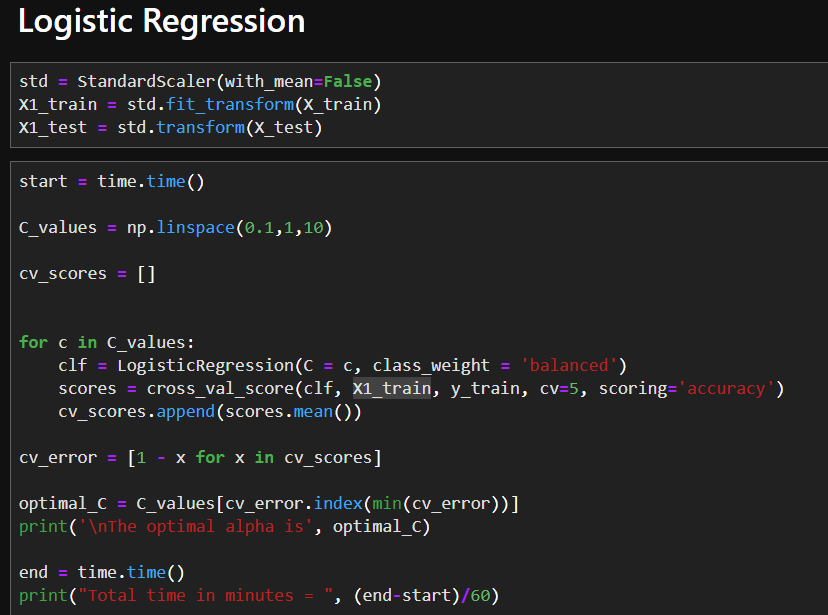
7.4 Logistic Regression

Logistic Regression is one of the simplest and commonly used Machine Learning algorithms for two-class classification. It is easy to implement and can be used as the baseline for any binary classification problem. Its basic fundamental concepts are also constructive in deep learning. Logistic regression describes and estimates the relationship between one dependent binary variable and independent variables.

Logistic regression is a statistical method for predicting binary classes. The outcome or target variable is dichotomous in nature. Dichotomous means there are only two possible classes. For example, it can be used for cancer detection problems. It computes the probability of an event occurrence.

It is a special case of linear regression where the target variable is categorical in nature. It uses a log of odds as the dependent variable. Logistic Regression predicts the probability of occurrence of a binary event utilizing a logit function.

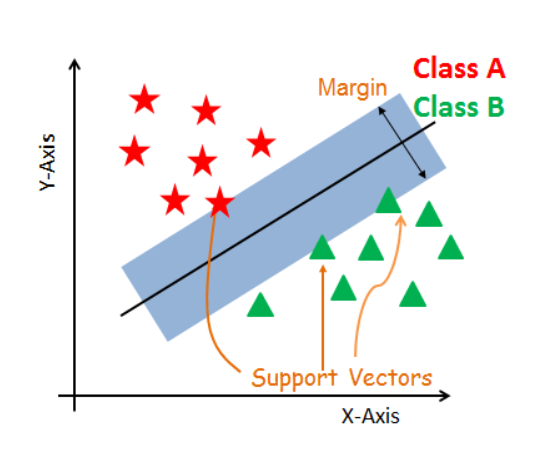
Implementation:



We were able to attain 72% accuracy using the Logistic Regression model.

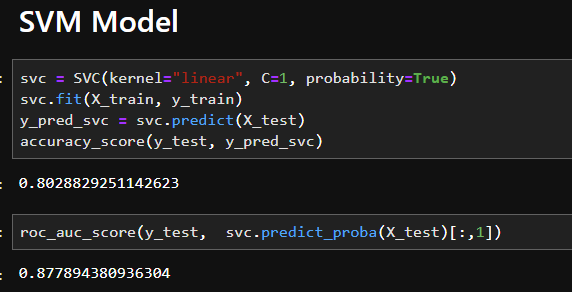
7.5 SVM

SVM offers very high accuracy compared to other classifiers such as logistic regression, and decision trees. It is known for its kernel trick to handle nonlinear input spaces. It is used in a variety of applications such as face detection, intrusion detection, classification of emails, news articles, and web pages, classification of genes, and handwriting recognition.

SVM is an exciting algorithm and the concepts are relatively simple. The classifier separates data points using a hyperplane with the largest amount of margin. That's why an SVM classifier is also known as a discriminative classifier. SVM finds an optimal hyperplane which helps in classifying new data points.

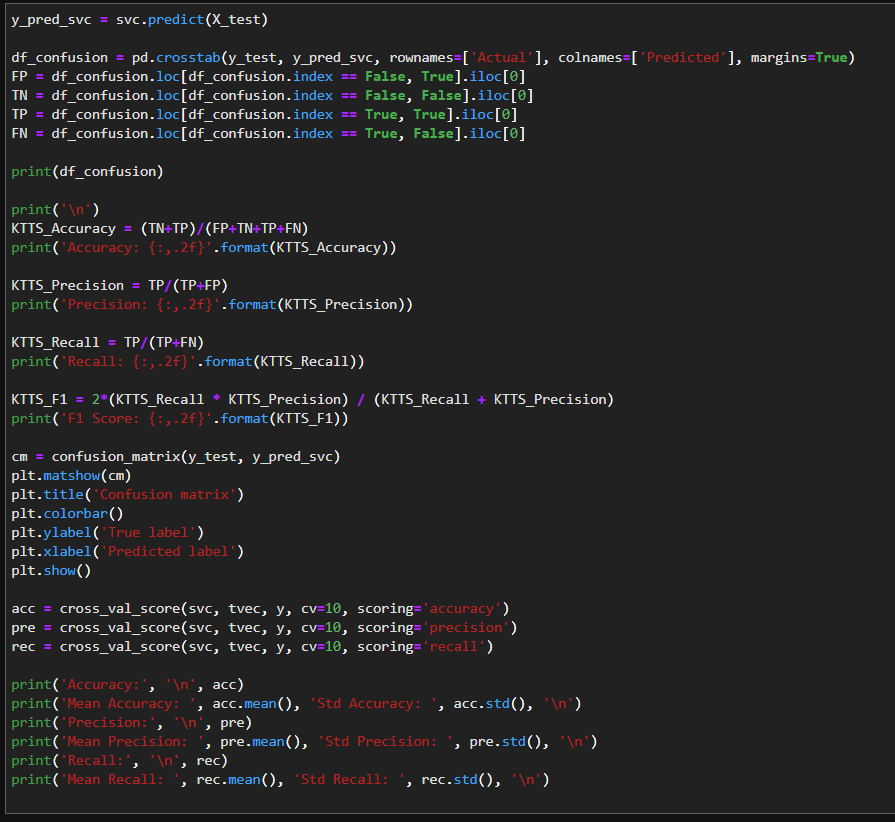
We trained our SVM model with different values of C and kernel.

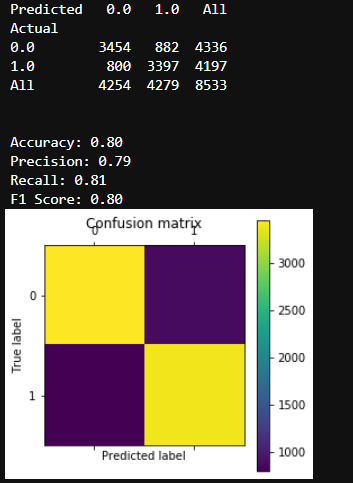
We found that maximum accuracy can be attained when kernel=”linear” and C=1.



We were able to attain 87% accuracy using the SVM model.

Confusion Matrix of SVM model:

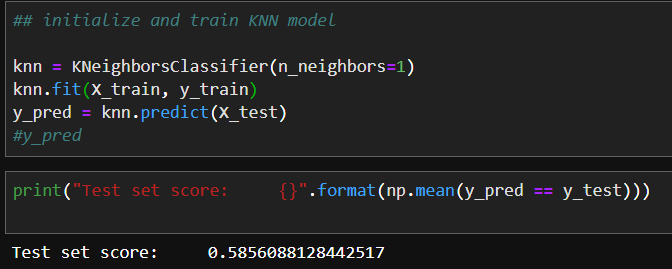


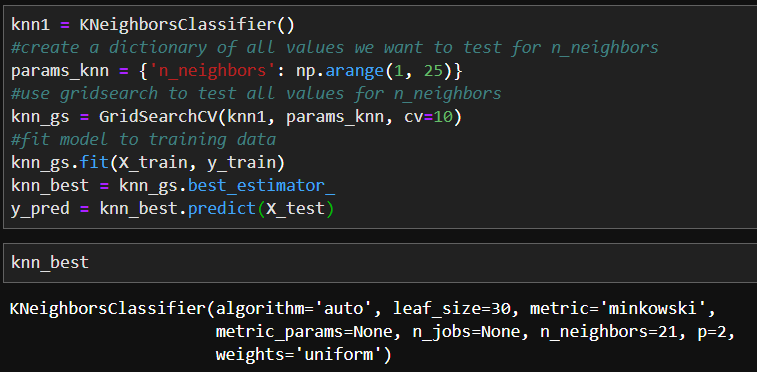


7.6 KNN

K-Nearest Neighbors, or KNN for short, is one of the simplest machine learning algorithms and is used in a wide array of institutions. KNN is a non-parametric, lazy learning algorithm. When we say a technique is non-parametric, it means that it does not make any assumptions about the underlying data. In other words, it makes its selection based on the proximity to other data points regardless of what feature the numerical values represent. Being a lazy learning algorithm implies that there is little to no training phase. Therefore, we can immediately classify new data points as they present themselves.

We trained our model initially with n=1 and got an accuracy of 62%.

Then we performed parameter optimization:



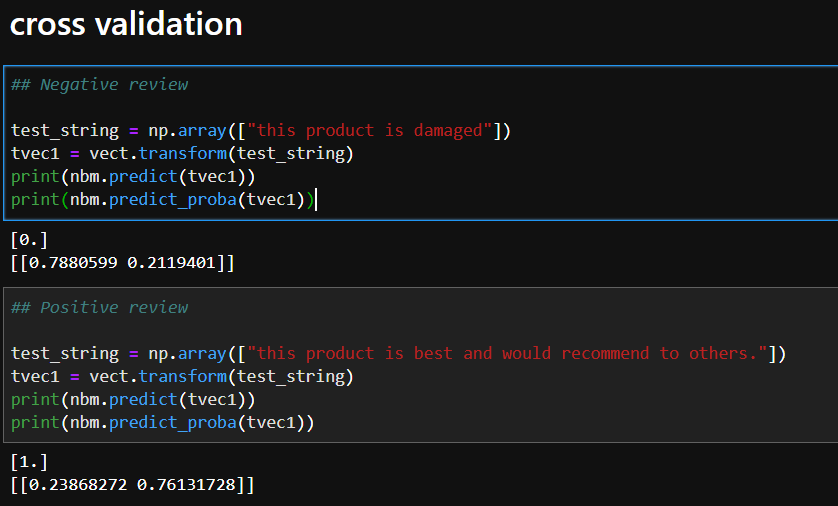
But this didn’t help much in increasing the accuracy of the model.

1. Conclusion

We performed data pre-processing and used NLTK WordNet Lemmatizer to lemmatize the sentences. We used TF-IDF Vectorizer to change the text of our review to numerical data. Post this, we implemented different Supervised Machine learning algorithms to check how well these models are able to learn from the training dataset and able to make correct predictions on the test dataset.

Out of all the 6 models which we used in this project, Multinomial Naïve Bayes turns out to be best among them with 87.4% predict probability score.

We cross-validated this by running the model for our own reviews and got the desired output.



1. References

* <https://www.analyticsvidhya.com/blog/2015/12/improve-machine-learning-results/>
* <https://towardsdatascience.com/improving-random-forest-in-python-part-1-893916666cd>
* <https://towardsdatascience.com/machine-learning-basics-with-the-k-nearest-neighbors-algorithm-6a6e71d01761>
* <https://medium.com/capital-one-tech/k-nearest-neighbors-knn-algorithm-for-machine-learning-e883219c8f26>
* <https://medium.com/@penggongting/implementing-decision-tree-from-scratch-in-python-c732e7c69aea>
* <https://www.datacamp.com/community/tutorials/text-analytics-beginners-nltk>