

**RATINGS PREDICTION**

Submitted by:

Harsha Rajan

**ACKNOWLEDGMENT**

I have taken efforts in this project. However it would not have been possible without the kind support and help of many individuals. I would like to express my sincere thanks to all of them.

I am highly indebted to Datatrained Education and Flip Robo technologies for their guidance as well as for providing necessary information regarding the project also for their support in completing the project.

I would like to express my special thanks of gratitude to my mentor(Sajid Choudhary) for the guidance.

I would like to express my gratitude towards my parents for their kind of co-operation and encouragement which help me in completion of this project.

I would like to express my gratitude and thanks to the industry persons for giving me such attention and time.

**INTRODUCTION**

**Problem Statement:**

We have a client who has a website where people write different reviews for technical products. Now they are adding a new feature to their website i.e. The reviewer will have to add stars(rating) as well with the review. The rating is out 5 stars and it only has 5 options available 1 star, 2 stars, 3 stars, 4 stars, 5 stars. Now they want to predict ratings for the reviews which were written in the past and they don’t have a rating. So, we have to build an application which can predict the rating by seeing the review.

**Data Collection Phase**

You have to scrape at least 20000 rows of data. You can scrape more data as well, it’s up to you. more the data better the model

In this section you need to scrape the reviews of different laptops, Phones, Headphones, smart watches, Professional Cameras, Printers, Monitors, Home theater, Router from different e-commerce websites.

Basically, we need these columns-

1) reviews of the product.

2) rating of the product.

You can fetch other data as well, if you think data can be useful or can help in the project. It completely depends on your imagination or assumption.

**Hint:**

• Try to fetch data from different websites. If data is from different websites, it will help our model to remove the effect of over fitting.

• Try to fetch an equal number of reviews for each rating, for example if you are fetching 10000 reviews then all ratings 1,2,3,4,5 should be 2000. It will balance our data set.

• Convert all the ratings to their round number, as there are only 5 options for rating i.e., 1,2,3,4,5. If a rating is 4.5 convert it 5.

**Model Building Phase**

After collecting the data, you need to build a machine learning model. Before model building do all data preprocessing steps involving NLP. Try different models with different hyper parameters and select the best model.

Follow the complete life cycle of data science. Include all the steps like-

1. Data Cleaning

2. Exploratory Data Analysis

3. Data Preprocessing

4. Model Building

5. Model Evaluation

6. Selecting the best mode

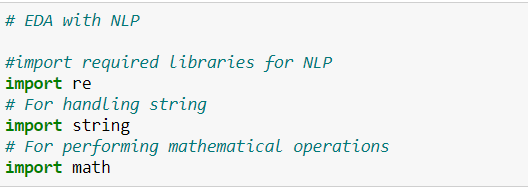
Load the data

The dataset has about 10000+ rows, each containing review text, and ratings,

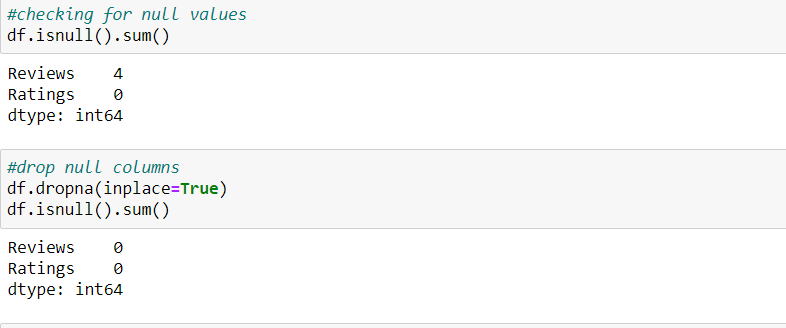
****

**Basic Text Data Pre-processing**

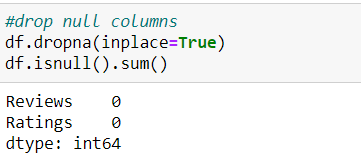
Before jumping to the exploration stage, we need to perform basic data pre-processing steps like null value imputation and removal of unwanted data. So, let’s start by importing libraries and reading our dataset:

****

Let’s see if there are any null values present in our dataset:

****

There are a few null values in the dataset. So, let’s drop these null values and proceed further:

****

Cleaning Text Data in Python

Generally, text data contains a lot of noise either in the form of symbols or in the form of punctuations and stopwords. Therefore, it becomes necessary to clean the text, not just for making it more understandable but also for getting better insights.

Lowercase the reviews

In NLP, models treat words like Goat and goat differently, even if they are the same. Therefore, to overcome this problem, we lowercase the words. Here, I am using the lower() function available in Python for converting text to lowercase:



### Remove digits and words containing digits

Next, we need to remove numbers and words containing digits from the reviews. I am doing this because digits and words containing digits do not give much importance to the main words. To do this, I am using regular expressions with lambda functions.



### Remove Punctuations

Punctuations are the marks in English like commas, hyphens, full stops, etc. These are important for English grammar but not for text analysis. Therefore, they need to be removed:



Here, **string.punctuations** function contains all the punctuations and we use regular expressions to search them in the text and remove them. Finally, we still have some extra spaces present in the data. Let’s remove them

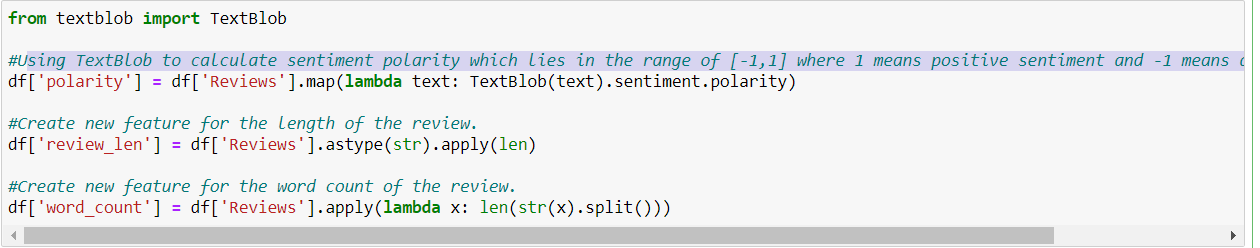
Let’s how our text looks after cleaning:



Using **[TextBlob](https://textblob.readthedocs.io/en/dev/)** to calculate sentiment polarity which lies in the range of [-1,1] where 1 means positive sentiment and -1 means a negative sentiment.

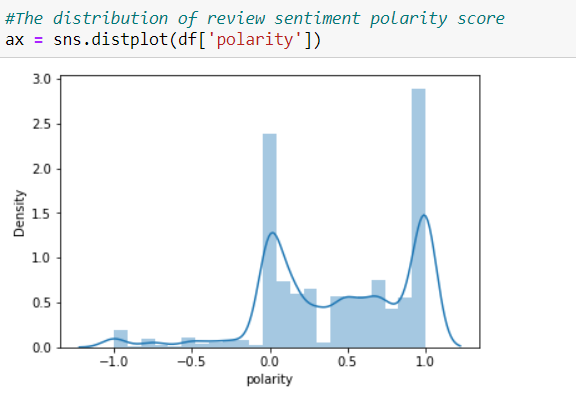
Create new feature for the length of the review.

Create new feature for the word count of the review.

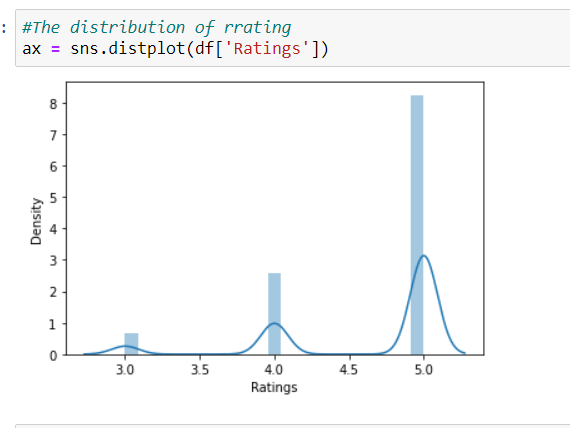


Univariate visualization with distplot

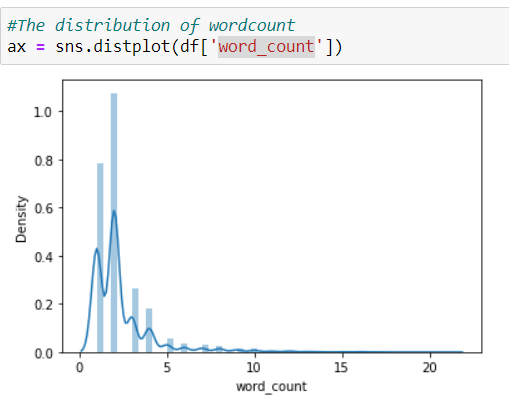
**The distribution of review sentiment polarity score**

****

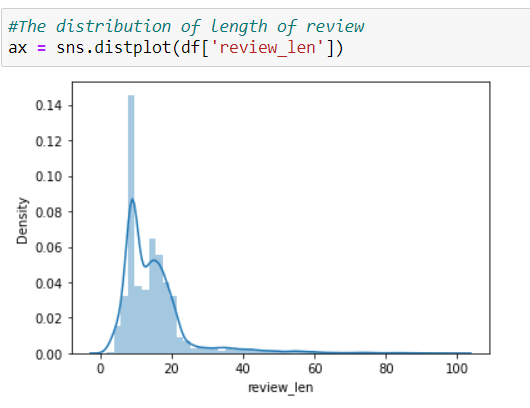
**Distribution of rating**

****

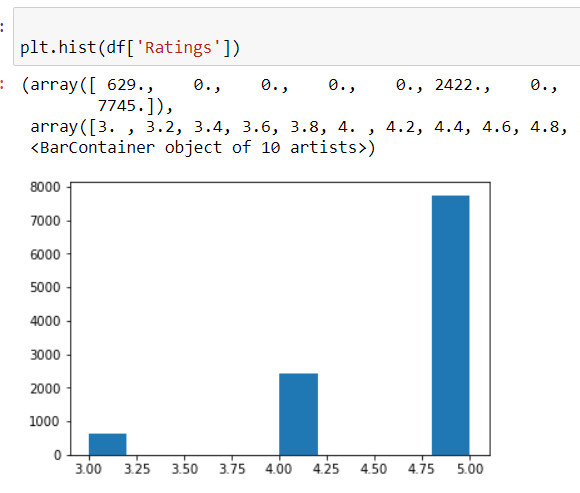
**Distribution of word count**

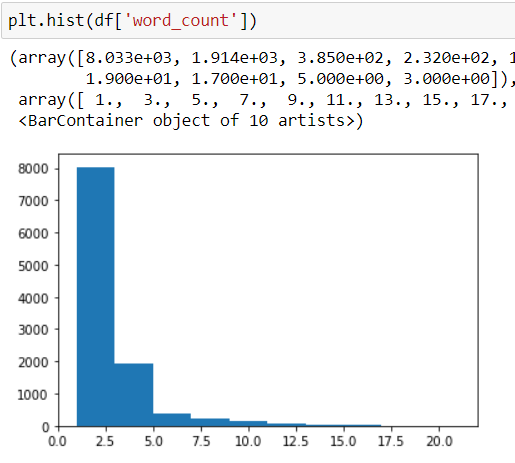
****

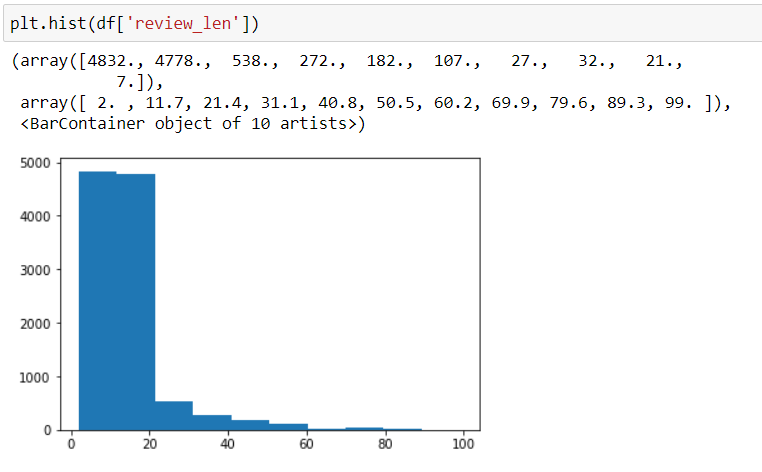
**Distribution of review length**

****

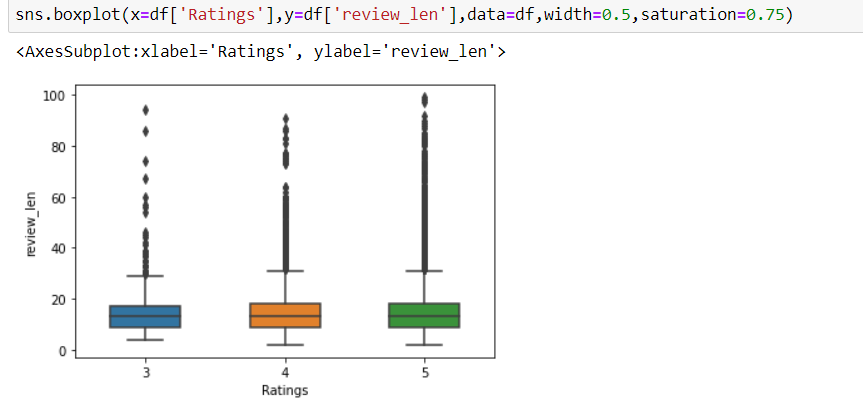
**Univariate analysis using histogram**



****

****

**Bi variate analysis using boxplot**

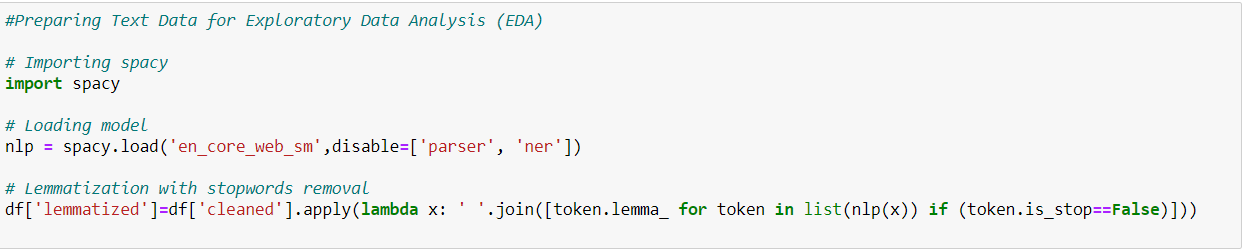
****

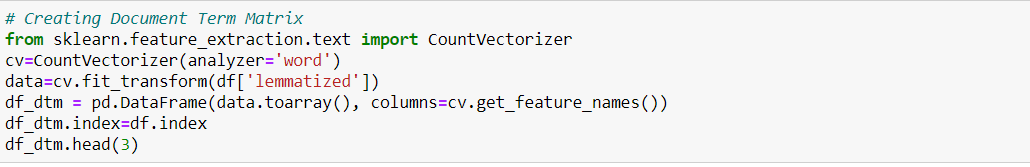
**Preparing Text Data for Exploratory Data Analysis (EDA)**

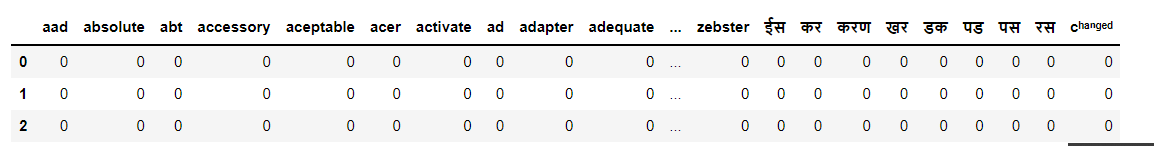
We have already cleaned our data and have our corpus ready, but there are still some steps left to be done before EDA. In this section, we’ll create a Document Term Matrix that we’ll later use in our analysis.

Now, you might be wondering what is a Document Term Matrix and why do we have to create one?

A Document Term Matrix provides the frequency of a word in a corpus (collection of documents), which in this case are reviews. It helps in analyzing the occurrence of words in different documents in a corpus. The following figure is an example of a document term matrix:

****

****

****

**Model Building**

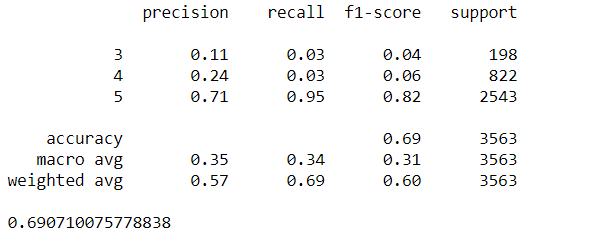
The first model I use is the Naive Bayes Model. In machine learning, naive Bayes classifiers are a family of simple “probabilistic classifiers” based on applying Bayes’ theorem with strong (naive) independence assumptions between the features.

All naive Bayes classifiers assume that the value of a particular feature is independent of the value of any other feature, given the class variable. In our model, the naive Bayes algorithm looks at particular keywords of a review to describe whether it is positive or negative, depending on the output set.

In the code below, I imported the GaussianNB class, which assumes that our data is normally distributed (with a Gaussian Bell Curve).

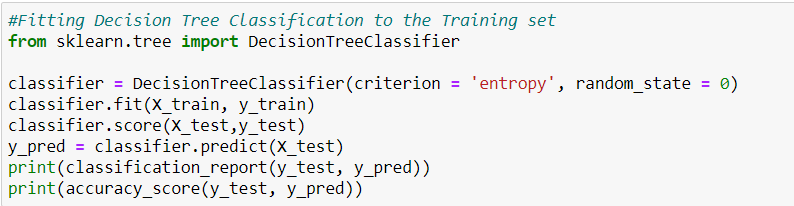
****

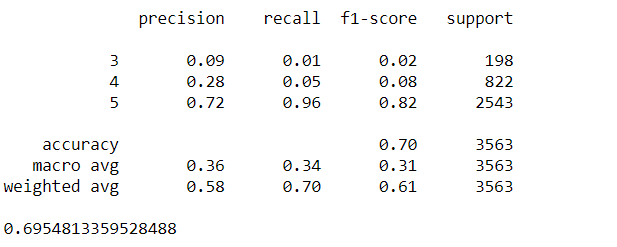
Accuracy is as the name goes. It measures the accuracy by adding True predictions and dividing them by the total number of predictions.

****

**Decision Tree**

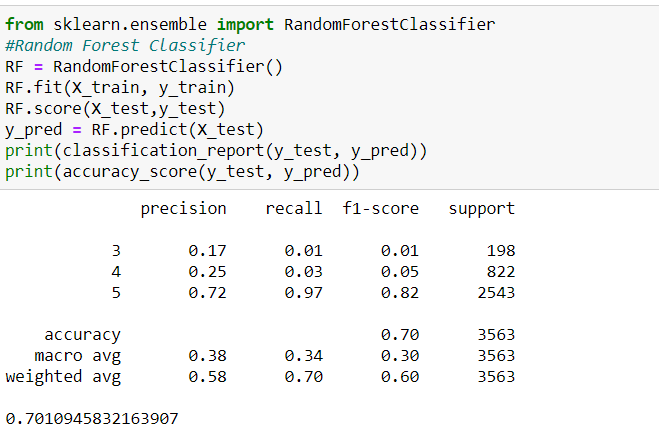
The next algorithm I used is the decision tree. Decision trees allow you to develop classification systems that predict or classify future observations based on a set of decision rules.

****

****

**Random Forest**

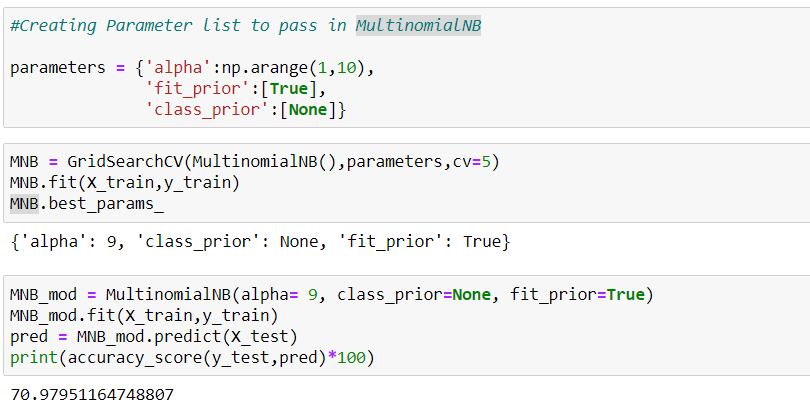
Finally, I used the Random Forest algorithm, which is just a combination of a number of decision trees. In my example, I chose to use 300 trees, but I could change that number depending on the kind of accuracy I want from the model.

****

**Hyper parameter tuning**

****

****

****

**CONCLUSION**

On average our models are about 71.3% accurate. While this may mean that the machine cannot predict every review with accuracy, it also shows us evidence that our models are not overfitting the data. Overfitting is a modeling error that occurs when a function is too closely fit to a limited set of data points. Overfitting the model generally occurs when using an overly complex model to explain idiosyncrasies in the data. However, since our model is being trained to think like a human brain, it is fair to assume that even humans may not be able to predict 100% of the time whether a review is positive or negative. It can really depend on the data and how it has been processed.

So which was the best algorithm for this model?  
Out of the 3 used in this project, the most accurate and precise was the**Naive Bayes** algorithm.