FLIP ROBO

RATING PREDICTION MODEL

Submitted by:

BHAVNA PIPLANI

ACKNOWLEDGMENT

This includes mentioning of all the references, research papers, data sources, professionals and other resources that helped and guided me in completion of the project.

INTRODUCTION

A. Problem Framing

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 Cleaning, Data Visualization using different plotting methods like distplot, countplot, Data pre-processing using NLP, Tfidf vectorizer, and Model Training using different algorithms.

B. Analytical Problem Framing

a) Mathematical/ Analytical Modelling of the Problem

Statistical modelling is the process of applying statistical analysis to a dataset. A statistical model is a mathematical representation (or mathematical model) of observed data.

When data analysts apply various statistical models to the data they are investigating, they are able to understand and interpret the information more strategically. Rather than sifting through the raw data, this practice allows them to identify relationships between variables, make predictions about future sets of data, and visualize that data so that non-analysts and stakeholders can consume and leverage it. Most common techniques will fall into the following two groups:

Supervised learning, including regression and classification models.

Unsupervised learning, including clustering algorithms and association rules.

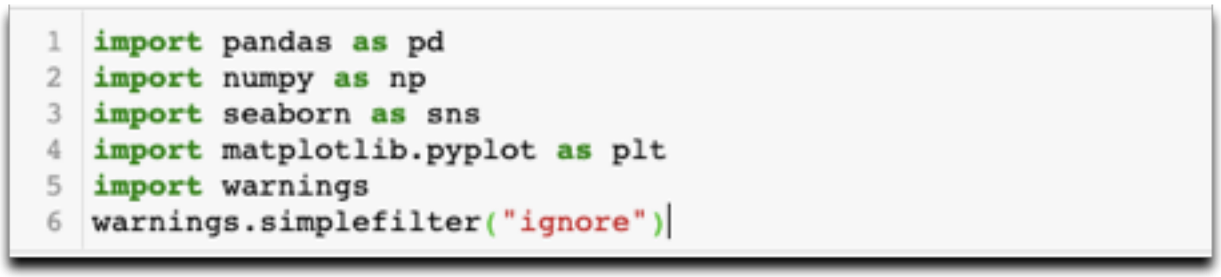
Some of the most common classifiers models include **MultinomialNB(), LinearSVC(), DecisionTreeClassifier(), RandomForestClassifier(), KNeighborsClassifier()**

b) Data Sources and their formats

Data Source is by fetching data from different e-commerce websites for further Data Cleaning, Data pre-processing and Model Training. Columns for the same to be used are Reviews and Ratings.

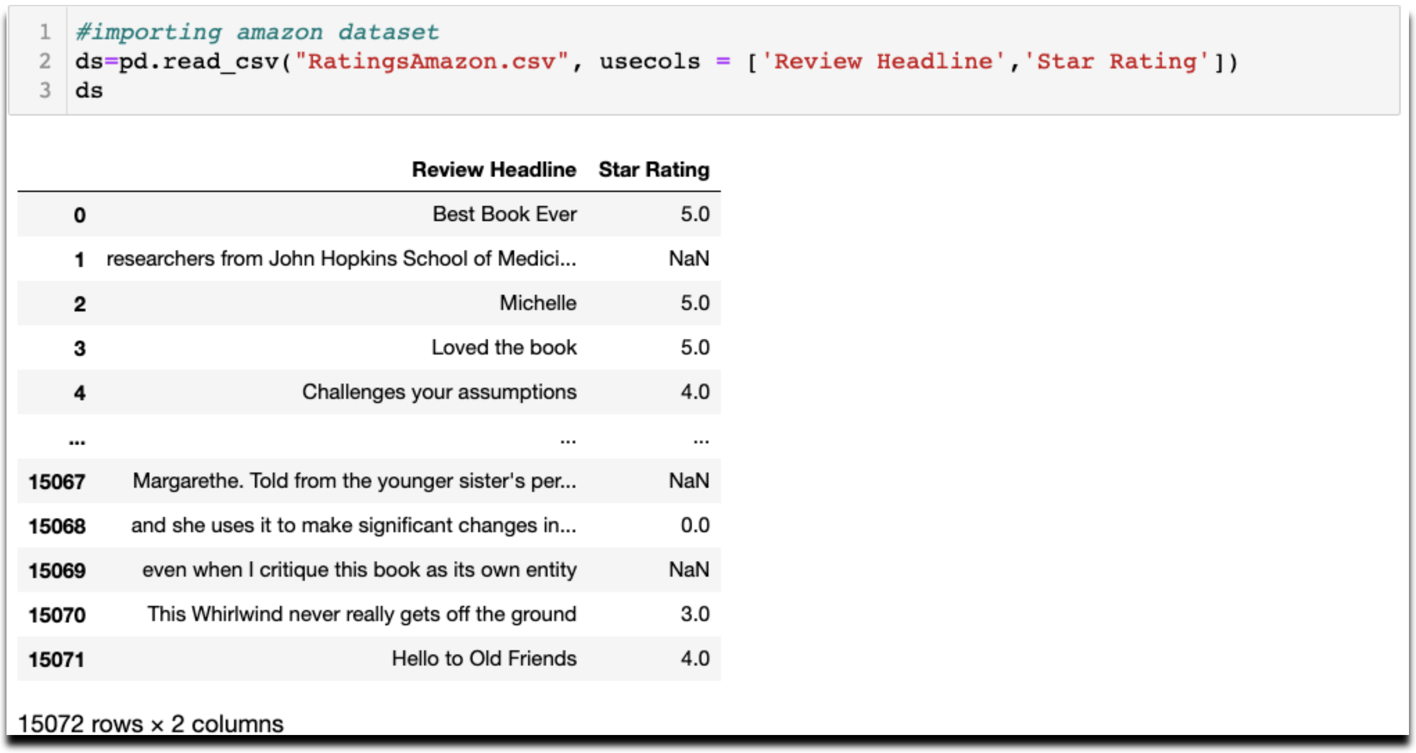
C. Explanatory Data Analysis

Importing the basic libraries to start any machine learning model.



```
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
import warnings
warnings.simplefilter("ignore")
```

Importing the dataset from the csv file scraped from amazon.in.

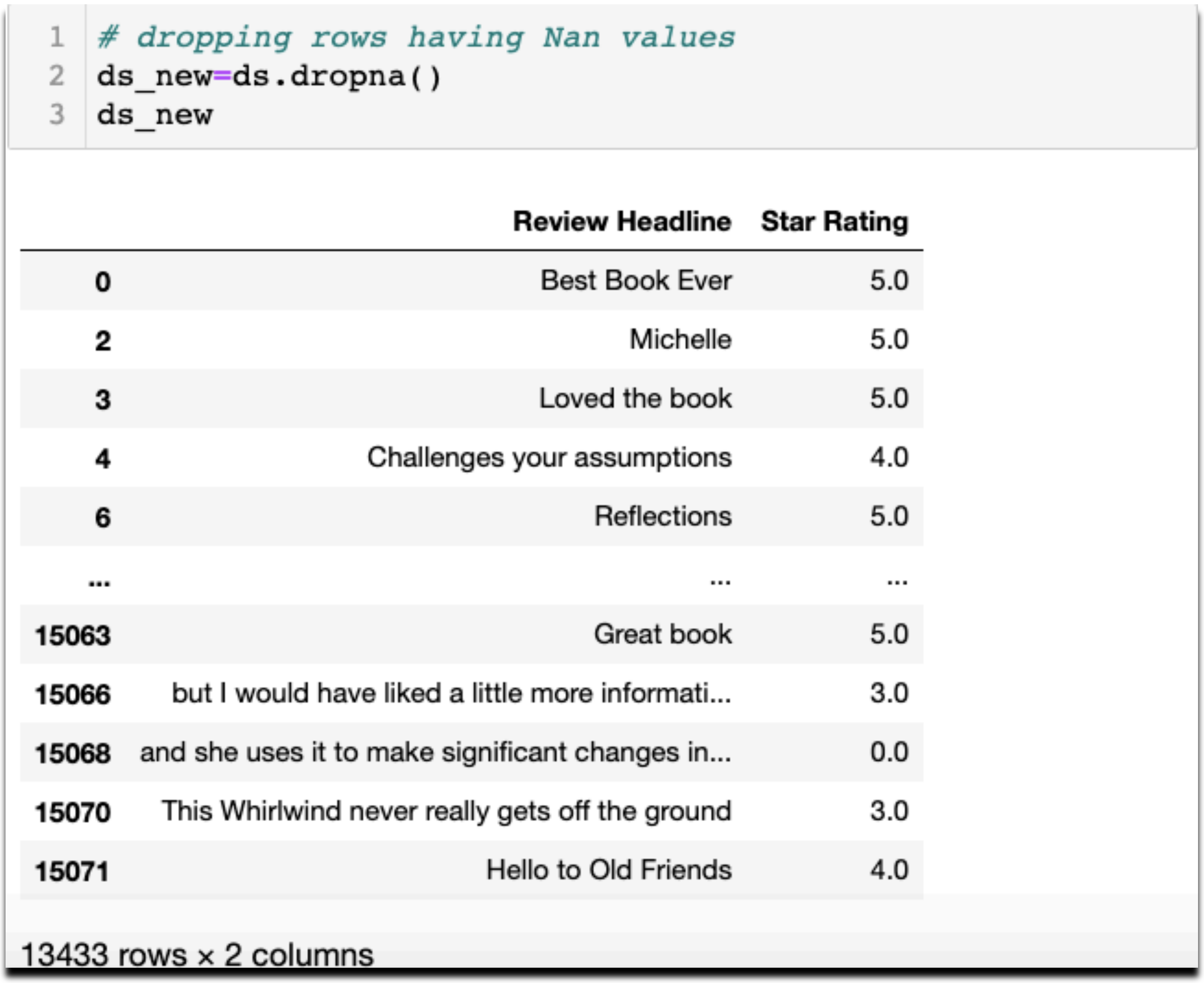


```
#importing amazon dataset
ds=pd.read_csv("RatingsAmazon.csv", usecols = ['Review Headline','Star Rating'])
ds
```

	Review Headline	Star Rating
0	Best Book Ever	5.0
1	researchers from John Hopkins School of Medici...	NaN
2	Michelle	5.0
3	Loved the book	5.0
4	Challenges your assumptions	4.0
...	...	...
15067	Margarethe. Told from the younger sister's per...	NaN
15068	and she uses it to make significant changes in...	0.0
15069	even when I critique this book as its own entity	NaN
15070	This Whirlwind never really gets off the ground	3.0
15071	Hello to Old Friends	4.0

15072 rows × 2 columns

After importing the dataset, it can be observed that it has Nan values. Also, it has ratings below 1 and above 5 too which is not required. For that, further data cleaning will be required.

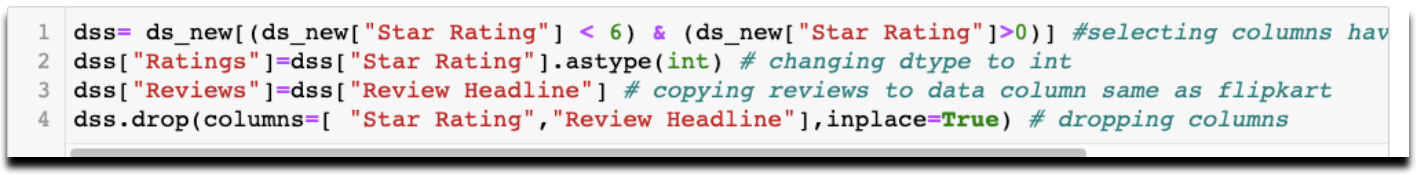


```
# dropping rows having Nan values
ds_new=ds.dropna()
ds_new
```

	Review Headline	Star Rating
0	Best Book Ever	5.0
2	Michelle	5.0
3	Loved the book	5.0
4	Challenges your assumptions	4.0
6	Reflections	5.0
...	...	...
15063	Great book	5.0
15066	but I would have liked a little more informati...	3.0
15068	and she uses it to make significant changes in...	0.0
15070	This Whirlwind never really gets off the ground	3.0
15071	Hello to Old Friends	4.0

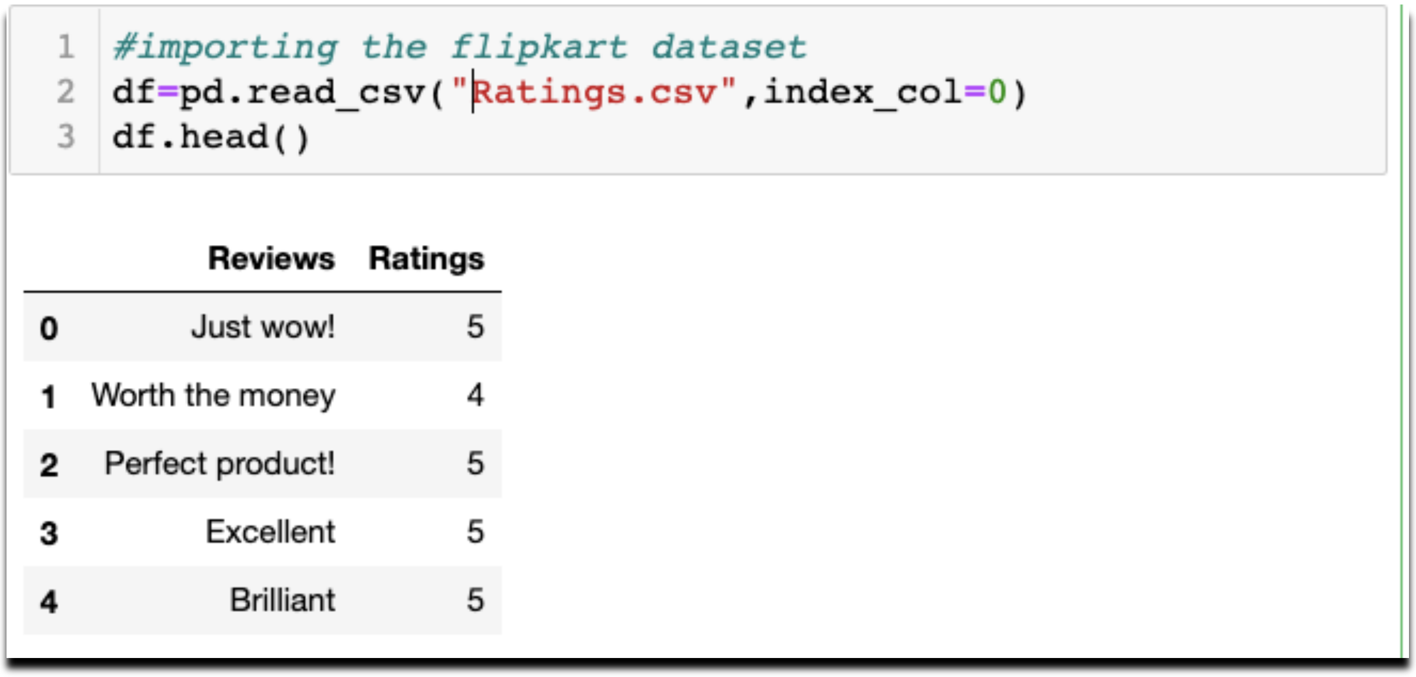
13433 rows × 2 columns

After removing null values, we will remove rows with ratings having less than 1 and greater than 5. Also, will copy columns names same in other e-commerce website to concatenate the data frames and removing the older columns as well.



```
dss= ds_new[(ds_new["Star Rating"] < 6) & (ds_new["Star Rating"]>0)] #selecting columns hav
dss["Ratings"]=dss["Star Rating"].astype(int) # changing dtype to int
dss["Reviews"]=dss["Review Headline"] # copying reviews to data column same as flipkart
dss.drop(columns=[ "Star Rating","Review Headline"],inplace=True) # dropping columns
```

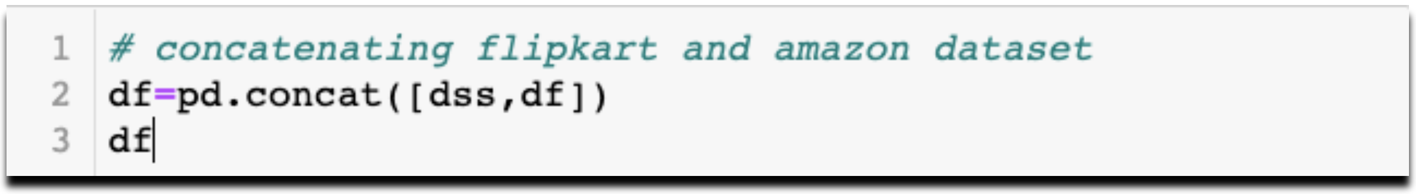
After the data cleaning from above dataset, we will move further to import other dataset to concatenate with this one.



```
#importing the flipkart dataset
df=pd.read_csv("Ratings.csv",index_col=0)
df.head()
```

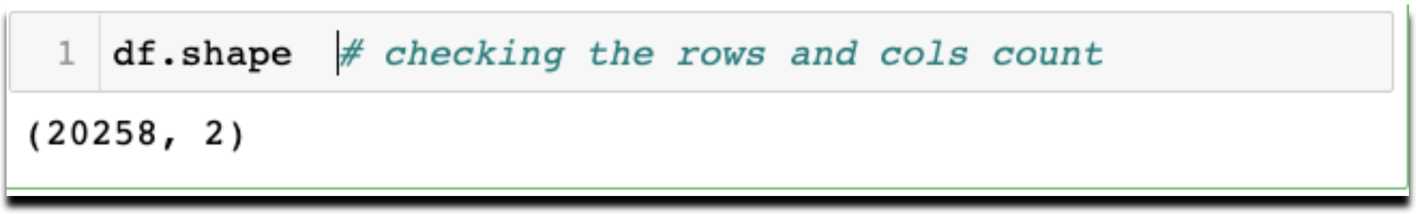
	Reviews	Ratings
0	Just wow!	5
1	Worth the money	4
2	Perfect product!	5
3	Excellent	5
4	Brilliant	5

Now, we will concatenate these datasets to further data training and modelling.



```
# concatenating flipkart and amazon dataset
df=pd.concat([dss,df])
df
```

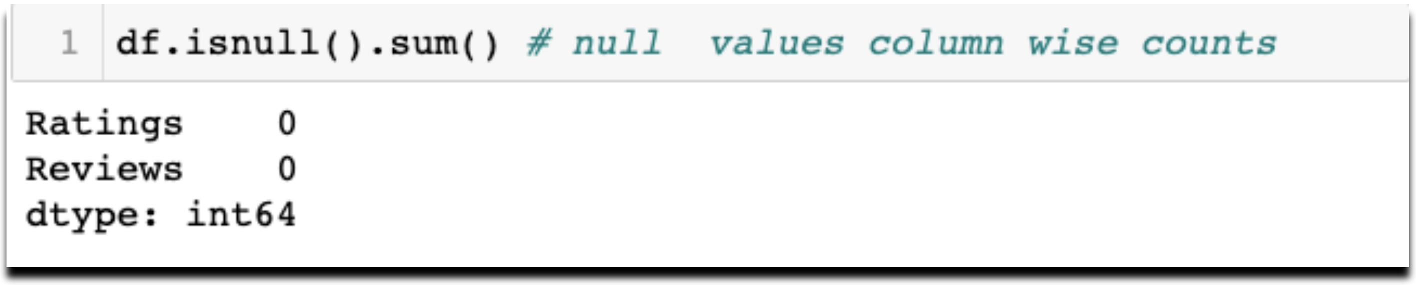
Train dataset has 20258 rows and 2 columns



```
df.shape  # checking the rows and cols count
```

```
(20258, 2)
```

After checking the no. of rows and columns count, we will check the null values if any, column count, datatypes of columns.

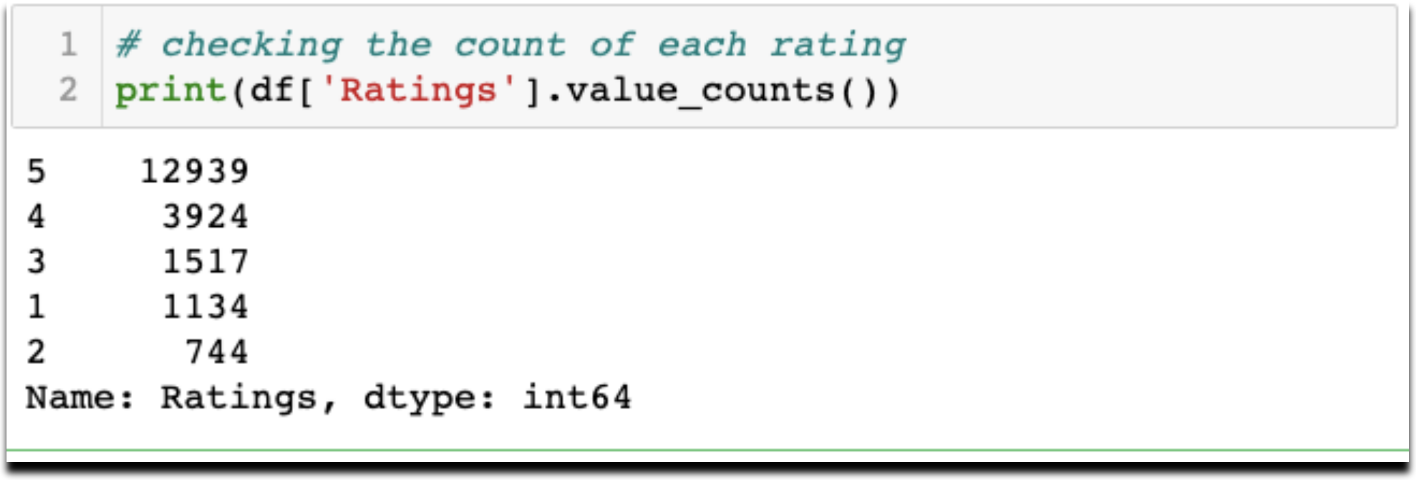


```
df.isnull().sum() # null  values column wise counts
```

```
Ratings    0
Reviews    0
dtype: int64
```

The dataset is pretty clean. There are no missing or null values in the dataset.

After that, we will check for the data if it is balanced or not.



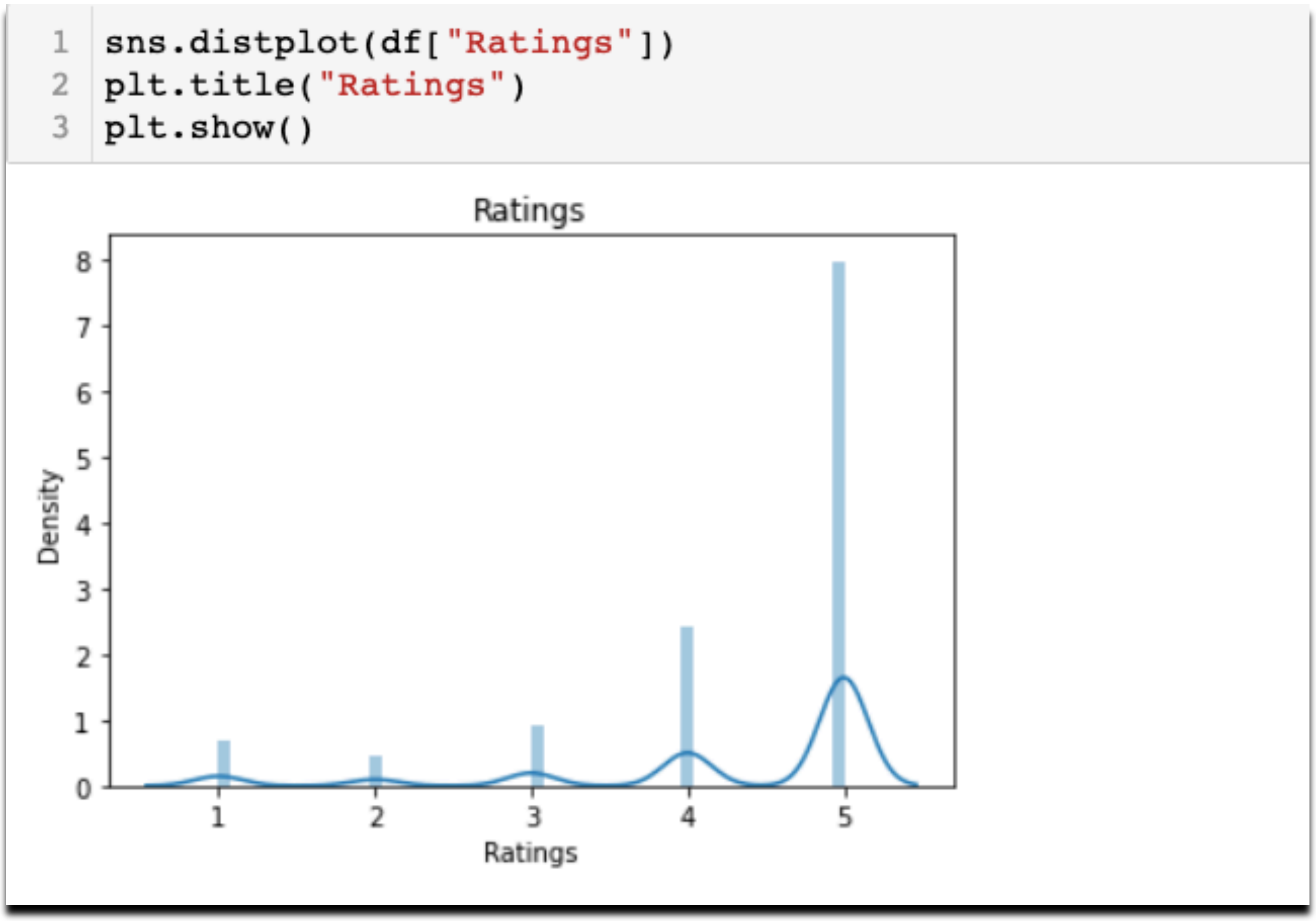
```
# checking the count of each rating
print(df['Ratings'].value_counts())
```

```
5    12939
4     3924
3     1517
1     1134
2      744
Name: Ratings, dtype: int64
```

Data is not balanced . Rows count for all the ratings are not same.

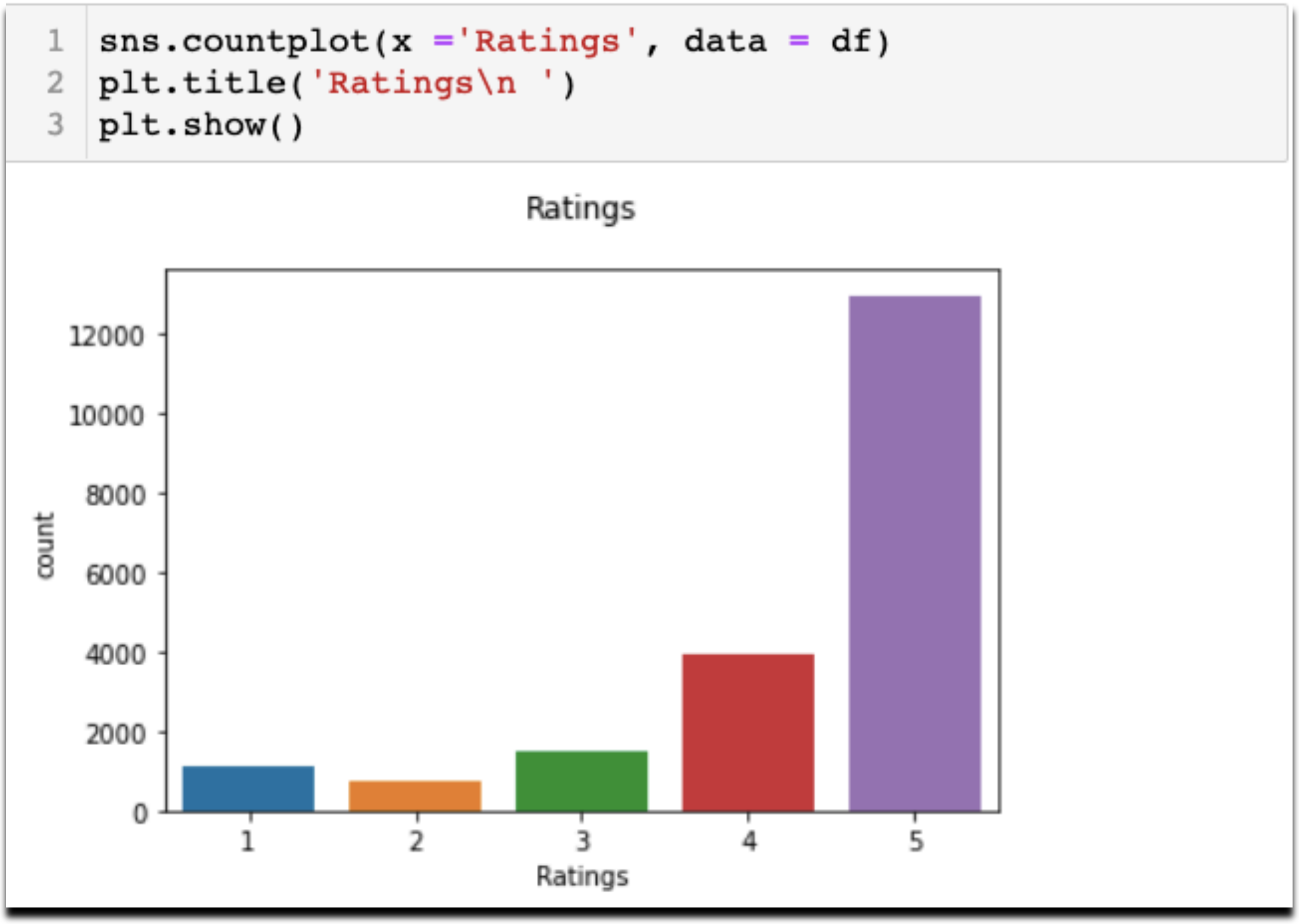
Data balancing techniques like oversampling will be used for data balancing.

D. Data Visualization



```
sns.distplot(df["Ratings"])
plt.title("Ratings")
plt.show()
```

Above distplot shows the max data is concentrated in 5 star rating



```
sns.countplot(x ='Ratings', data = df)
plt.title('Ratings\n ')
plt.show()
```

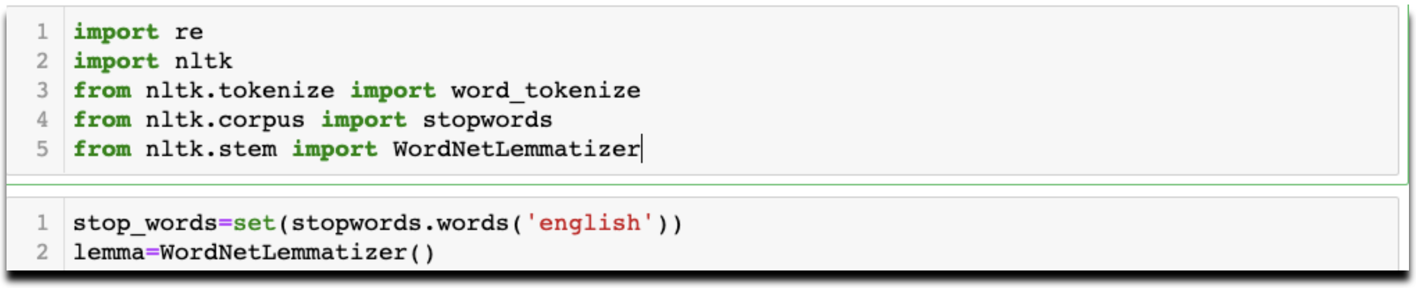
More than 12000 of records are concentrated in 5 star ratings which shows data is imbalanced.

E. Pre-Processing Pipeline

The Natural Language ToolKit is one of the best-known and most-used NLP libraries in the Python ecosystem, useful for all sorts of tasks from tokenization, to stemming, to part of speech tagging, and beyond.

Tokenization is a step which splits longer strings of text into smaller pieces, or tokens. Larger chunks of text can be tokenized into sentences, sentences can be tokenized into words, etc. Further processing is generally performed after a piece of text has been appropriately tokenized.

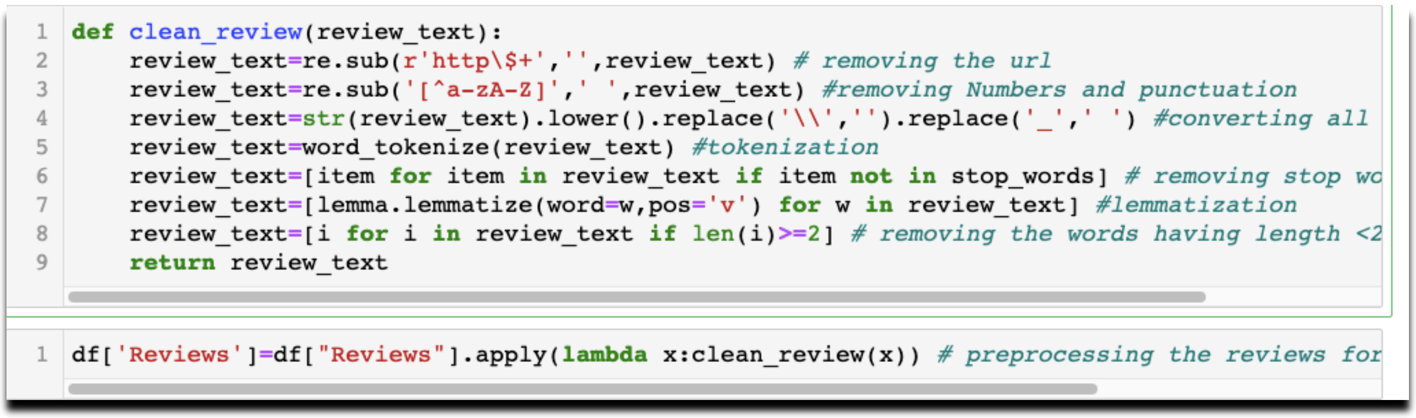
For our task, we will tokenize our sample text into a list of words. This is done using
NTLK's word_tokenize() function.



```python
import re
import nltk
from nltk.tokenize import word_tokenize
from nltk.corpus import stopwords
from nltk.stem import WordNetLemmatizer
```

```python
stop_words=set(stopwords.words('english'))
lemma=WordNetLemmatizer()
```

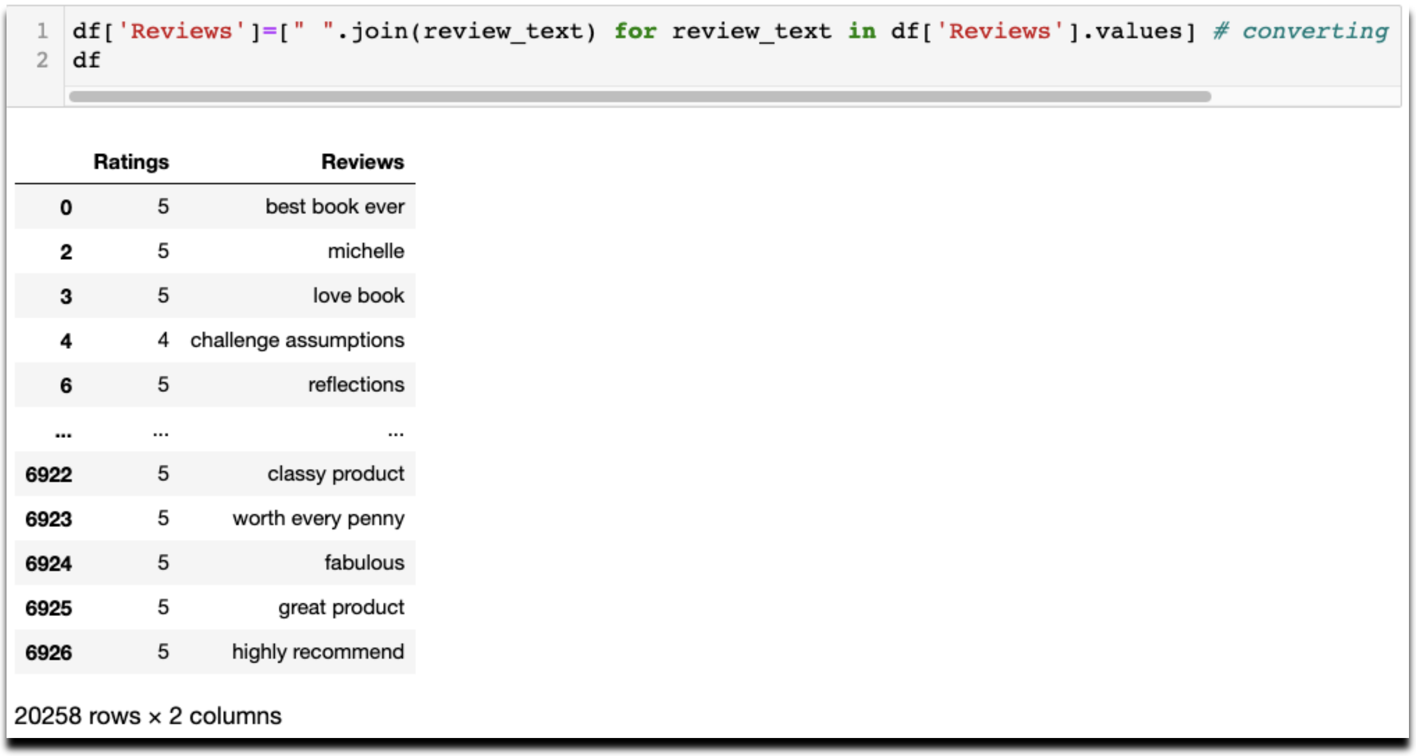
Using this below function we will clean the data like removing everything from the text other than alphabets , tokenization , lemmatization .



```python
def clean_review(review_text):
    review_text=re.sub(r'http\$+','',review_text) # removing the url
    review_text=re.sub('[^a-zA-Z]',' ',review_text) #removing Numbers and punctuation
    review_text=str(review_text).lower().replace('\\','').replace('_',' ') #converting all
    review_text=word_tokenize(review_text) #tokenization
    review_text=[item for item in review_text if item not in stop_words] # removing stop wo
    review_text=[lemma.lemmatize(word=w,pos='v') for w in review_text] #lemmatization
    review_text=[i for i in review_text if len(i)>=2] # removing the words having length <2
    return review_text
```

```python
df['Reviews']=df["Reviews"].apply(lambda x:clean_review(x)) # preprocessing the reviews for
```

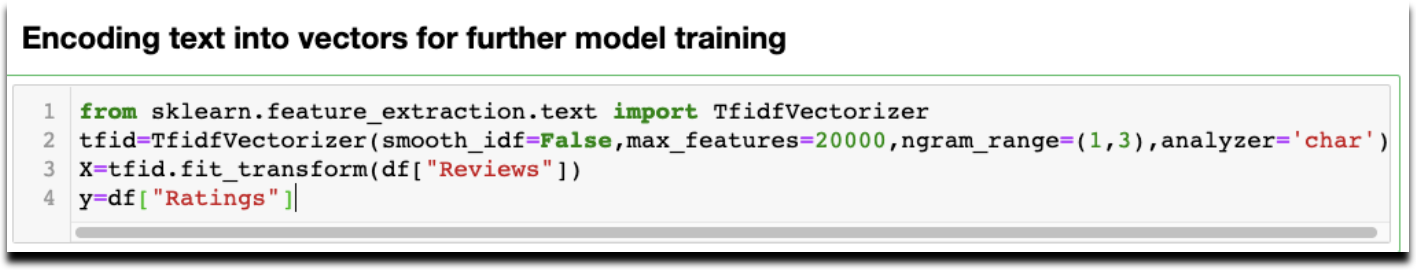
Below is the representation of cleaned dataset for model training.



```python
df['Reviews']=[" ".join(review_text) for review_text in df['Reviews'].values] # converting
df
```

	Ratings	Reviews
0	5	best book ever
2	5	michelle
3	5	love book
4	4	challenge assumptions
6	5	reflections
...	...	...
6922	5	classy product
6923	5	worth every penny
6924	5	fabulous
6925	5	great product
6926	5	highly recommend

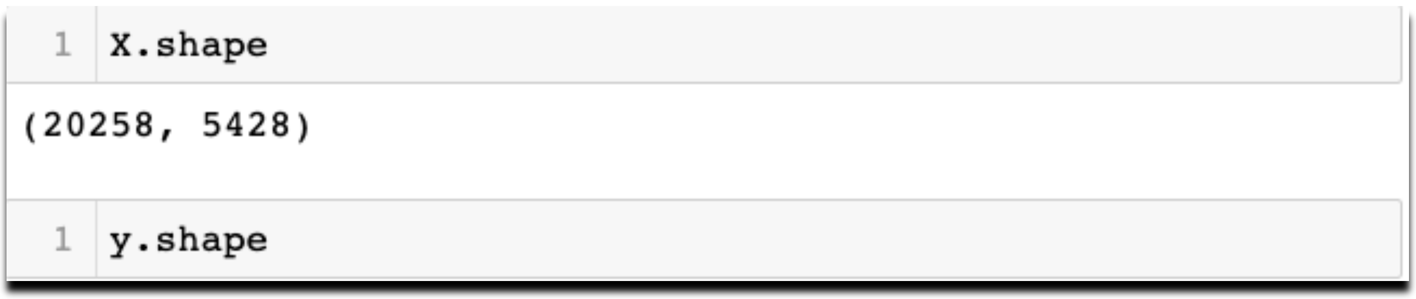
20258 rows × 2 columns



Encoding text into vectors for further model training

```
from sklearn.feature_extraction.text import TfidfVectorizer
tfid=TfidfVectorizer(smooth_idf=False,max_features=20000,ngram_range=(1,3),analyzer='char')
X=tfid.fit_transform(df["Reviews"])
y=df["Ratings"]
```

After all the data cleaning and data processing, X and y variables are processed for training the model.



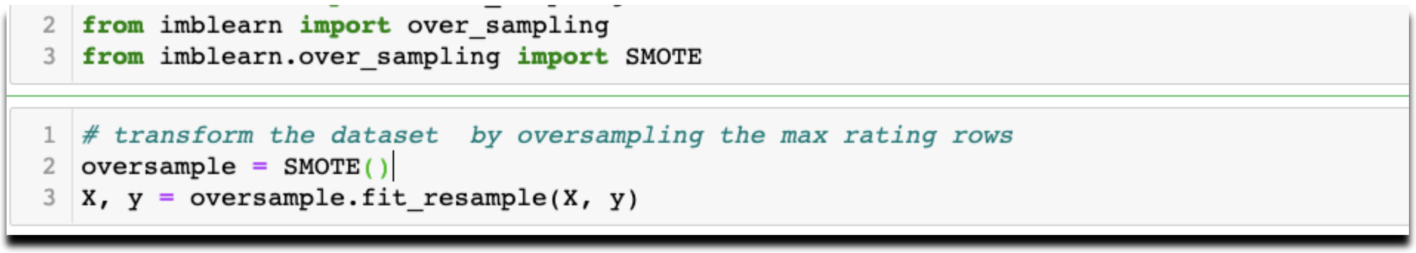
```
X.shape
```

```
(20258, 5428)
```

```
y.shape
```

F. Data Balancing

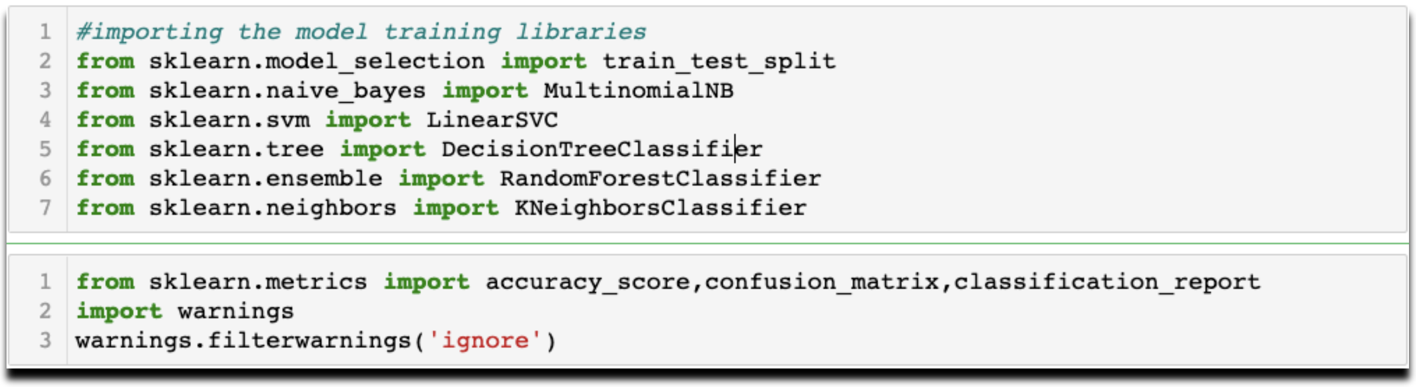
Imbalanced multiclass classification will give incorrect accuracy. because rows for 1,2,3 stars are very less as compared to 4,5 stars. For That will do oversampling to balance the data for data modelling.



```
from imblearn import over_sampling
from imblearn.over_sampling import SMOTE
```

```
# transform the dataset  by oversampling the max rating rows
oversample = SMOTE()
X, y = oversample.fit_resample(X, y)
```

Libraries and packages used for model training are listed below



```
#importing the model training libraries
from sklearn.model_selection import train_test_split
from sklearn.naive_bayes import MultinomialNB
from sklearn.svm import LinearSVC
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import RandomForestClassifier
from sklearn.neighbors import KNeighborsClassifier
```

```
from sklearn.metrics import accuracy_score,confusion_matrix,classification_report
import warnings
warnings.filterwarnings('ignore')
```

G. Model Development and Evaluation

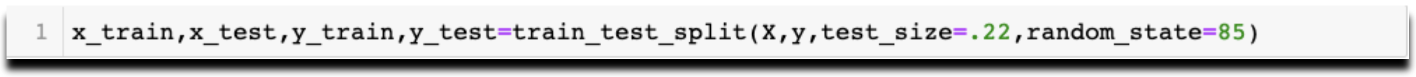
Identification of possible problem-solving approaches (methods). Most common techniques will fall into the following two groups:

Supervised learning, including regression and classification models.

Unsupervised learning, including clustering algorithms and association rules.Testing of Identified Approaches (Algorithms)

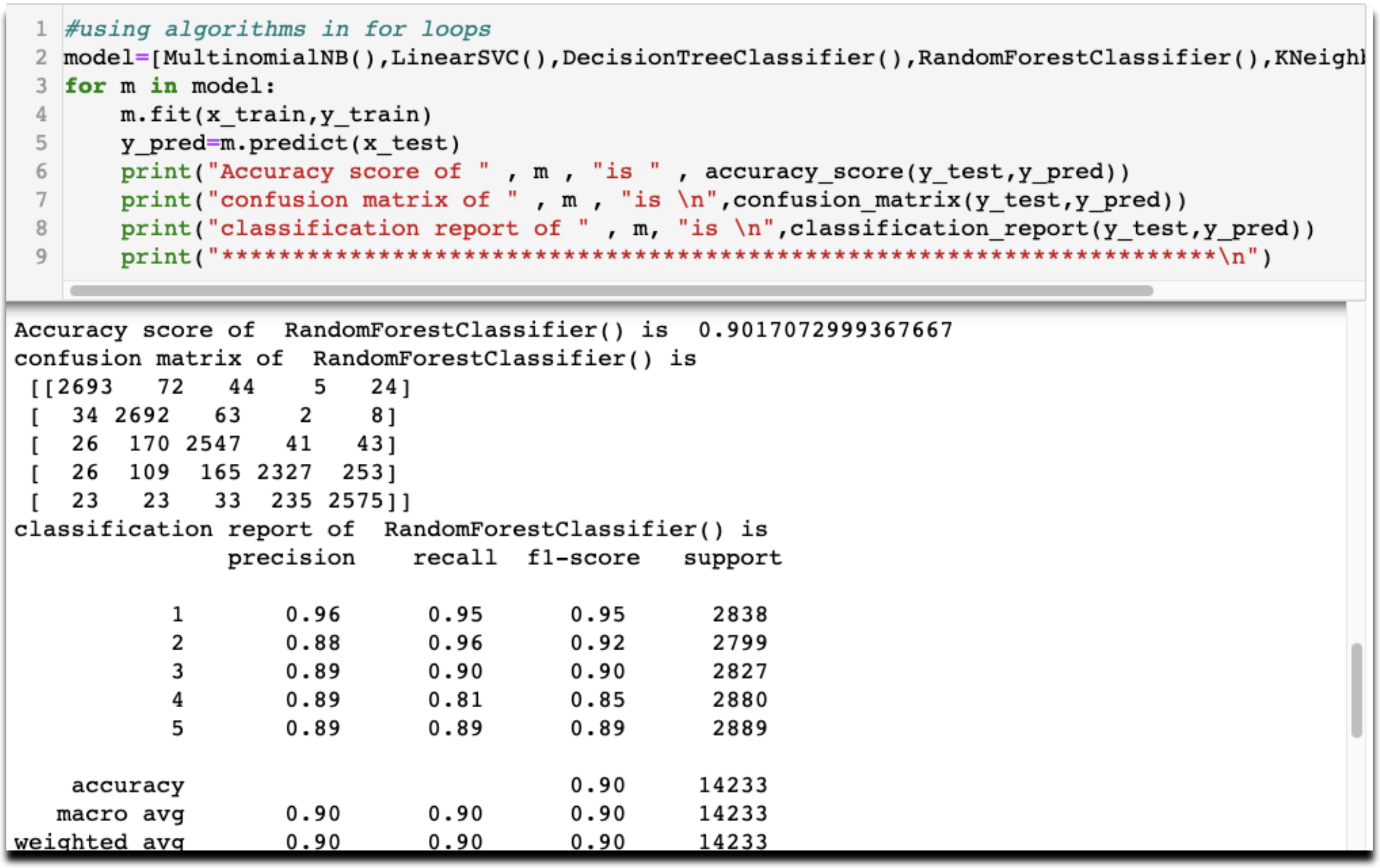
Here, In this project I will be using **MultinomialNB(), LinearSVC(), DecisionTreeClassifier(), RandomForestClassifier(), KNeighborsClassifier()** algorithms

First of all, Lets train the model. Here,I am using test_size=.22 that means 22% of data will go for testing purpose.



```
x_train,x_test,y_train,y_test=train_test_split(X,y,test_size=.22,random_state=85)
```

Using different algorithms, we will try to find the best model.

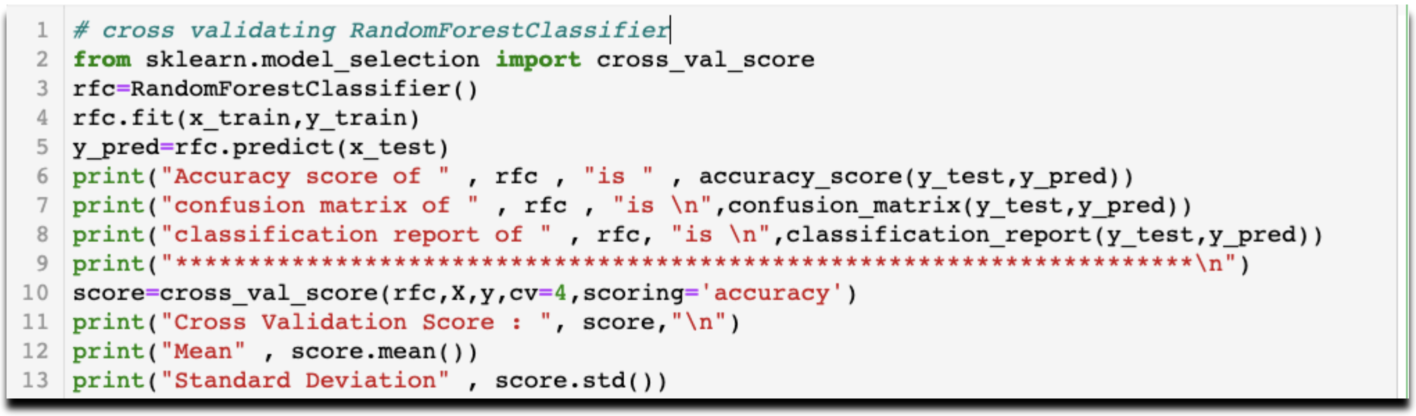


```
#using algorithms in for loops
model=[MultinomialNB(),LinearSVC(),DecisionTreeClassifier(),RandomForestClassifier(),KNeighb
for m in model:
    m.fit(x_train,y_train)
    y_pred=m.predict(x_test)
    print("Accuracy score of " , m , "is " , accuracy_score(y_test,y_pred))
    print("confusion matrix of " , m , "is \n",confusion_matrix(y_test,y_pred))
    print("classification report of " , m, "is \n",classification_report(y_test,y_pred))
    print("**********************************************************************\n")
```

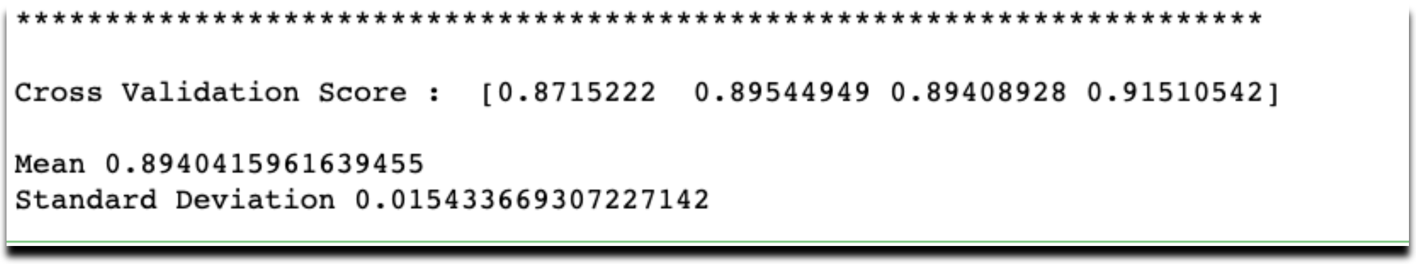
```
Accuracy score of  RandomForestClassifier() is  0.9017072999367667
confusion matrix of  RandomForestClassifier() is
 [[2693   72   44    5   24]
 [  34 2692   63    2    8]
 [  26  170 2547   41   43]
 [  26  109  165 2327  253]
 [  23   23   33  235 2575]]
classification report of  RandomForestClassifier() is
               precision    recall  f1-score   support

           1       0.96      0.95      0.95      2838
           2       0.88      0.96      0.92      2799
           3       0.89      0.90      0.90      2827
           4       0.89      0.81      0.85      2880
           5       0.89      0.89      0.89      2889

    accuracy                           0.90     14233
   macro avg       0.90      0.90      0.90     14233
weighted avg       0.90      0.90      0.90     14233
```

The best performing model among all being tested was RandomForestClassifier with more than 90% of accuracy. Also, precision and F1 score for all the ratings was pretty good.

Now, Lets check cross validate the RandomForestClassifier model.



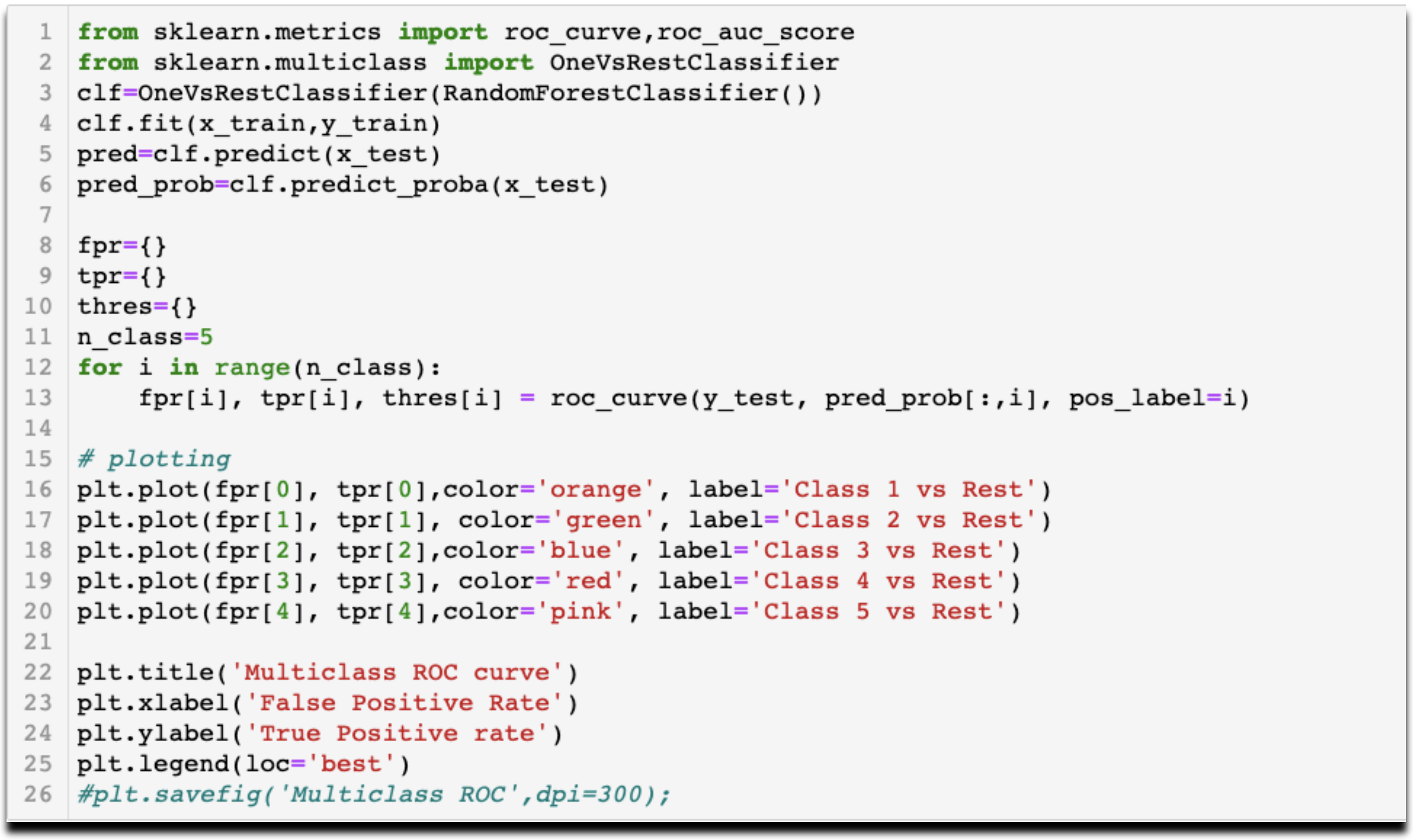
```python
# cross validating RandomForestClassifier
from sklearn.model_selection import cross_val_score
rfc=RandomForestClassifier()
rfc.fit(x_train,y_train)
y_pred=rfc.predict(x_test)
print("Accuracy score of " , rfc , "is " , accuracy_score(y_test,y_pred))
print("confusion matrix of " , rfc , "is \n",confusion_matrix(y_test,y_pred))
print("classification report of " , rfc, "is \n",classification_report(y_test,y_pred))
print("**********************************************************************\n")
score=cross_val_score(rfc,X,y,cv=4,scoring='accuracy')
print("Cross Validation Score : ", score,"\n")
print("Mean" , score.mean())
print("Standard Deviation" , score.std())
```



```
**********************************************************************

Cross Validation Score :  [0.8715222  0.89544949 0.89408928 0.91510542]

Mean 0.8940415961639455
Standard Deviation 0.015433669307227142
```

Accuracy for validation was also proved good. Mean was around .894.

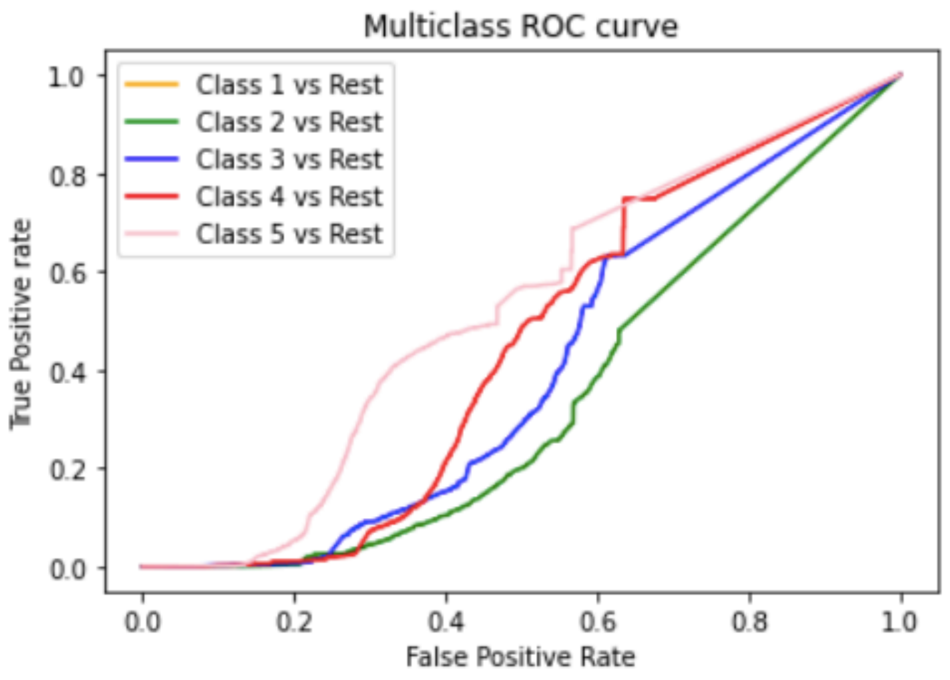
And Standard Deviation was very less. As, AUC_ROC curve is used for binary classification , OneVsRestClassifier used for representing the accuracy by auc_roc curve.



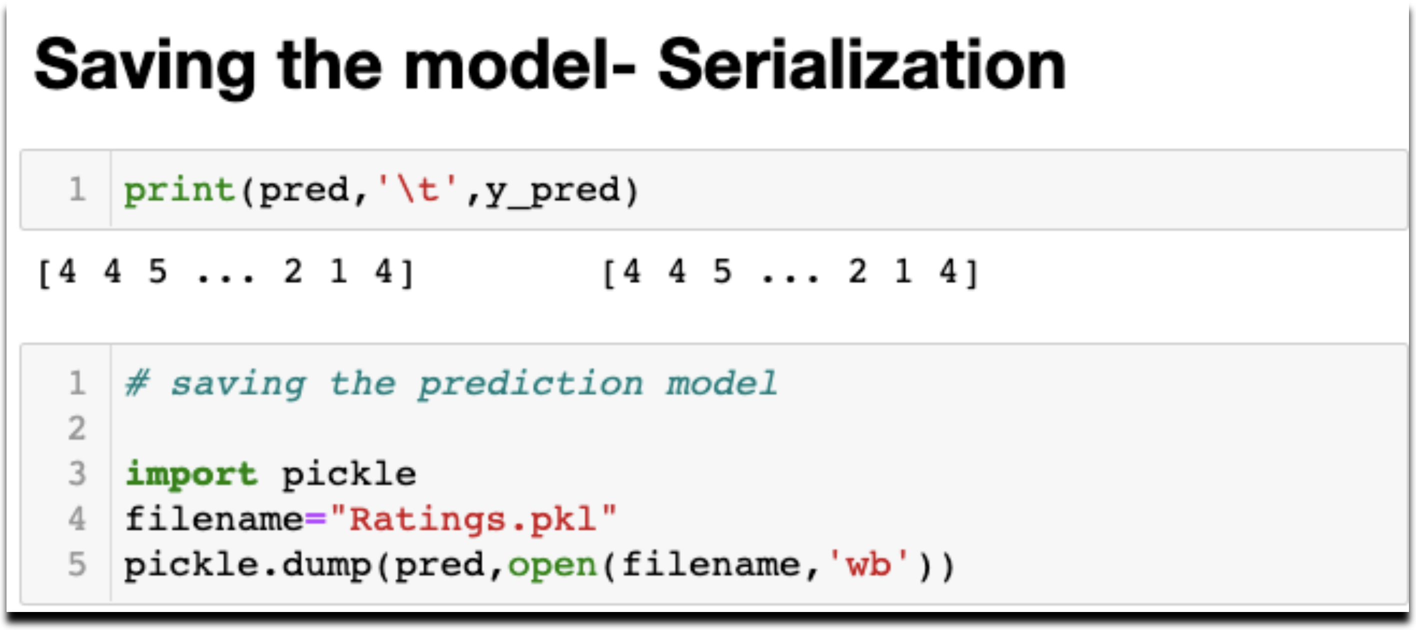
```python
from sklearn.metrics import roc_curve,roc_auc_score
from sklearn.multiclass import OneVsRestClassifier
clf=OneVsRestClassifier(RandomForestClassifier())
clf.fit(x_train,y_train)
pred=clf.predict(x_test)
pred_prob=clf.predict_proba(x_test)

fpr={}
tpr={}
thres={}
n_class=5
for i in range(n_class):
    fpr[i], tpr[i], thres[i] = roc_curve(y_test, pred_prob[:,i], pos_label=i)

# plotting
plt.plot(fpr[0], tpr[0],color='orange', label='Class 1 vs Rest')
plt.plot(fpr[1], tpr[1], color='green', label='Class 2 vs Rest')
plt.plot(fpr[2], tpr[2],color='blue', label='Class 3 vs Rest')
plt.plot(fpr[3], tpr[3], color='red', label='Class 4 vs Rest')
plt.plot(fpr[4], tpr[4],color='pink', label='Class 5 vs Rest')

plt.title('Multiclass ROC curve')
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive rate')
plt.legend(loc='best')
#plt.savefig('Multiclass ROC',dpi=300);
```

The results of the above classifier are shown below:

```
<matplotlib.legend.Legend at 0x7f90247f1520>
```



Now, as the model is performing good with the score of 90% , we will save the predicted_model .



Saving the model- Serialization

```
print(pred,'\t',y_pred)
```

```
[4 4 5 ... 2 1 4]          [4 4 5 ... 2 1 4]
```

```
# saving the prediction model

import pickle
filename="Ratings.pkl"
pickle.dump(pred,open(filename,'wb'))
```

CONCLUSION

Key Findings and Conclusions of the Study:

I. Data scraped from different e-commerce websites was uncleaned.
II. Data Preprocessing using NLP was done including tokenization, lemmatization, data cleaning by removing every symbol, special characters etc other than alphabets.
III. Data was imbalanced as the star ratings scraped from different websites were not same for 1 – 5 stars.
IV. SMOTE over_sampling was used to balance the class data
V. RandomForestClassifer was the best performing and fit model.