A PROJECT REPORT ON
“A STUDY ON CUSTOMERS BASED PERCEPTION IN LONDON HOTELS”

SUBMITTED BY

NAVEEN VARSHNEY M.SC. (DATA SCIENCE)

UNDER THE SUPERVISION OF

KALASH

ZEP ANALYTICS

ACKNOWLEDGMENT

It is our privilege to express our sincere thanks and a sense of deep gratitude to Satyajit Pattnaik, for all his help, painstaking efforts, and deep insight into the problem and thus improving the quality of work at all stages, providing us with the best knowledge to carry out our work.

We also like to express our gratefulness to Kalash, for his timely help, valuable advice, supervision, and immense patience due to which this work was able to take the shape in which it has been presented.

Our efforts could not be successful without our friends giving their valuable time to complete the questionnaires by expressing their opinions.

We also thank our parents and siblings for providing us with moral support and resources to finish our work.

INDEX

1.1 Scope of Study

Through this study, we hope to understand how customers view Hotel Systems and their services. Under what circumstances, customer perception about hotel varies. This study helps us to know about the “HOTEL Market”. We will know about the customer perception on Services provided by hotels in the LONDON area. We also get to know what variables are affecting their perception. These findings will help different service providers to work on the different parameters to improve their services. Also, it will help new hotels or service providers to know about customers and helps them compete in the market.

1.2 Objectives

- To study the level of awareness of consumers towards hotels.
- To study the preference towards hotel services.
- To study the satisfaction level of the consumers toward services provided by the hotels.
- To identify the problems faced by the consumer while using hoteling.
- To provide suggestions and recommendations based on the findings

2.1 Descriptive Analysis Methods

Frequency Table - Frequency is a measure of the number of occurrences of a particular score in a given set of data. A frequency table is a method of organizing raw data in a compact form by displaying a series of scores in ascending or descending order, together with their frequencies—the number of times each score occurs in the respective data set. Included in a frequency table are typically a column for the scores and a column showing the frequency of each score in the data set. However, more detailed tables may also contain relative frequencies (proportions) and percentages. Frequency tables may be computed for both discrete and continuous variables and may take either an ungrouped or a grouped format.

Bar chart - The pictorial representation of grouped data, in the form of vertical or horizontal rectangular bars, where the lengths of the bars are equivalent to the measure of data, are known as bar graphs or bar charts. The bars drawn are of uniform width, and the variable quantity is represented on one of the axes. Also, the measure of the variable is depicted on the other axes. The heights or the lengths of the bars denote the value of the variable, and these graphs are also used to compare certain quantities. The frequency distribution tables can be easily represented using bar charts which simplify the calculations and understanding of data.

2.2 Software Used

Python - Python is a computer programming language often used to build websites and software, automate tasks, and conduct data analysis. Python is a general-purpose language, meaning it can be used to create a variety of different programs and is not specialized for any specific problems.

Library Used:

NumPy: NumPy is a Python library used for working with arrays. It also has functions for working in the domain of linear algebra, Fourier transform, and matrices. NumPy was created in 2005 by Travis Oliphant. It is an open-source project and you can use it freely.

Pandas: Pandas is an open-source Python package that is most widely used for data science/data analysis and machine learning tasks. It is built on top of another package named NumPy, which provides support for multi-dimensional arrays

NLTK: NLTK is a toolkit built for working with NLP in Python. It provides us various text-processing libraries with a lot of test datasets. A variety of tasks can be performed using NLTK such as tokenizing, parse tree visualization, etc

Translator: Translators is a library which aims to bring free, multiple, translation to individuals and students in Python. It based on the translation interface of Google, Yandex, Microsoft (Bing), Baidu, Alibaba, Tencent, NetEase (Youdao), Sogou, Kingsoft (Iciba), Iflytek, Niutrans, Lingvanex, Naver(Papago), Deepl, Reverso, Itranslate, Caiyun, TranslateCom, Mglip, Utibet, Argos, etc.

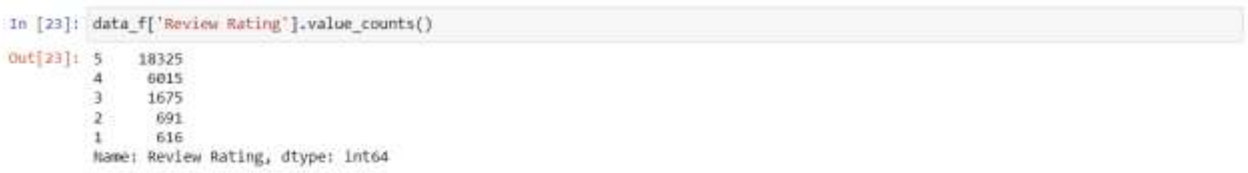
Seaborn: Seaborn: Seaborn, on the other hand, provides a variety of visualization patterns. It uses fewer syntax and has easily interesting default themes. It specializes in statistics visualization and is used if one has to summarize data in visualizations and also show the distribution in the data.

Scikit-Learn: Scikit-learn (Sklearn) is the most useful and robust library for machine learning in Python. It provides a selection of efficient tools for machine learning and statistical modelling including classification, regression, clustering, and dimensionality reduction via a consistence interface in Python.

3. Analysis of Results

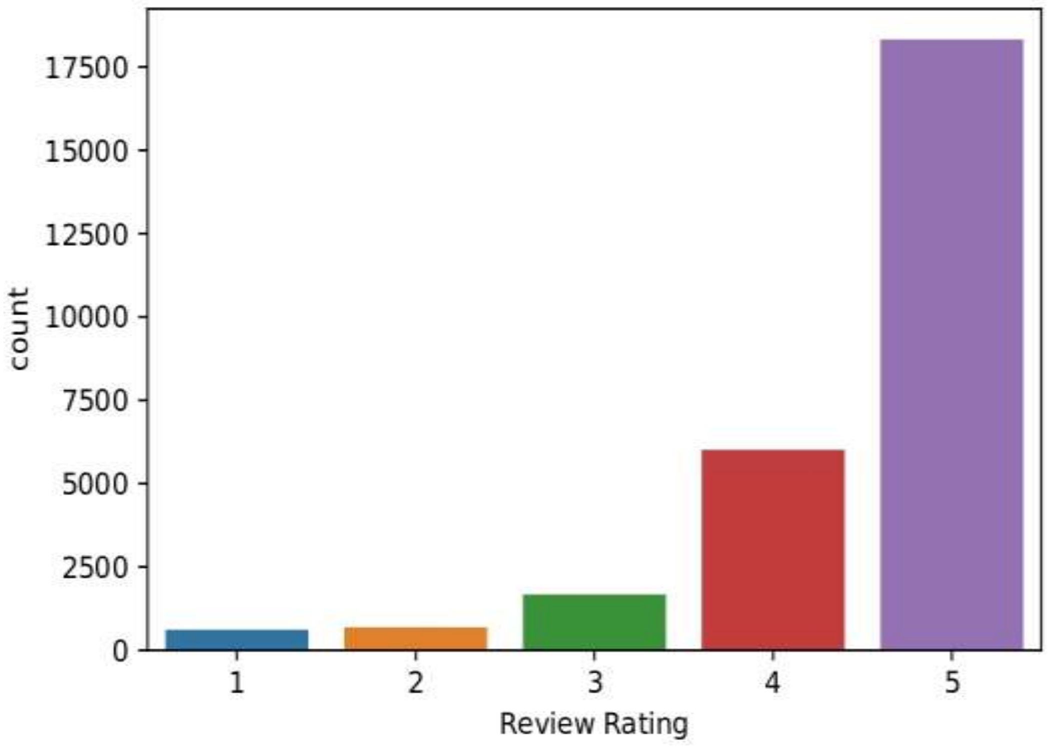
We have only one variable ‘Review Rating’ which is quantitative.

The frequency of each class in ‘Review Rating’ is

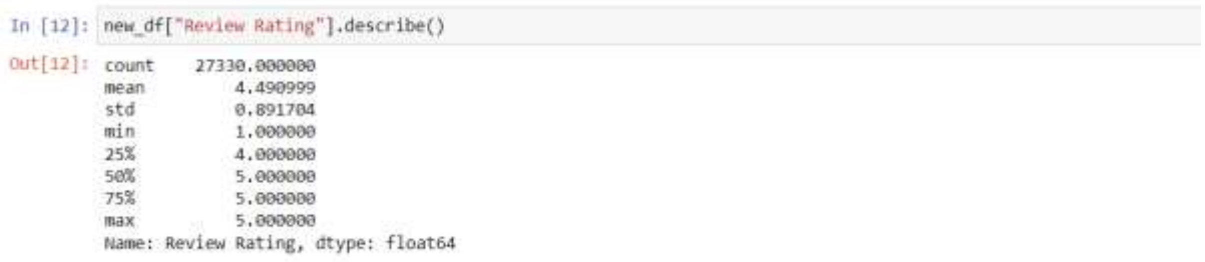


```
In [23]: data_f['Review Rating'].value_counts()

Out[23]: 5    18325
         4     6015
         3     1675
         2      691
         1      616
         Name: Review Rating, dtype: int64
```

Visually,



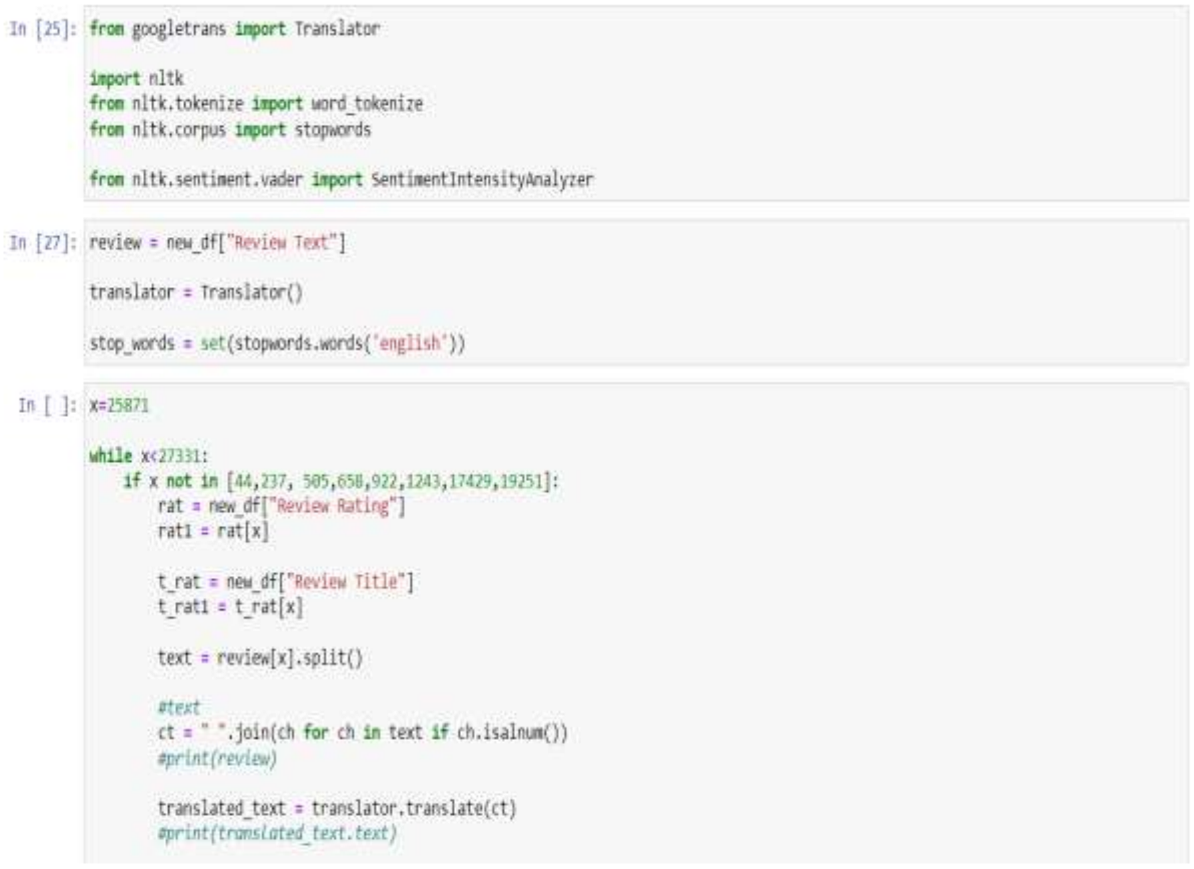
5-scale summary of the variable Review Title



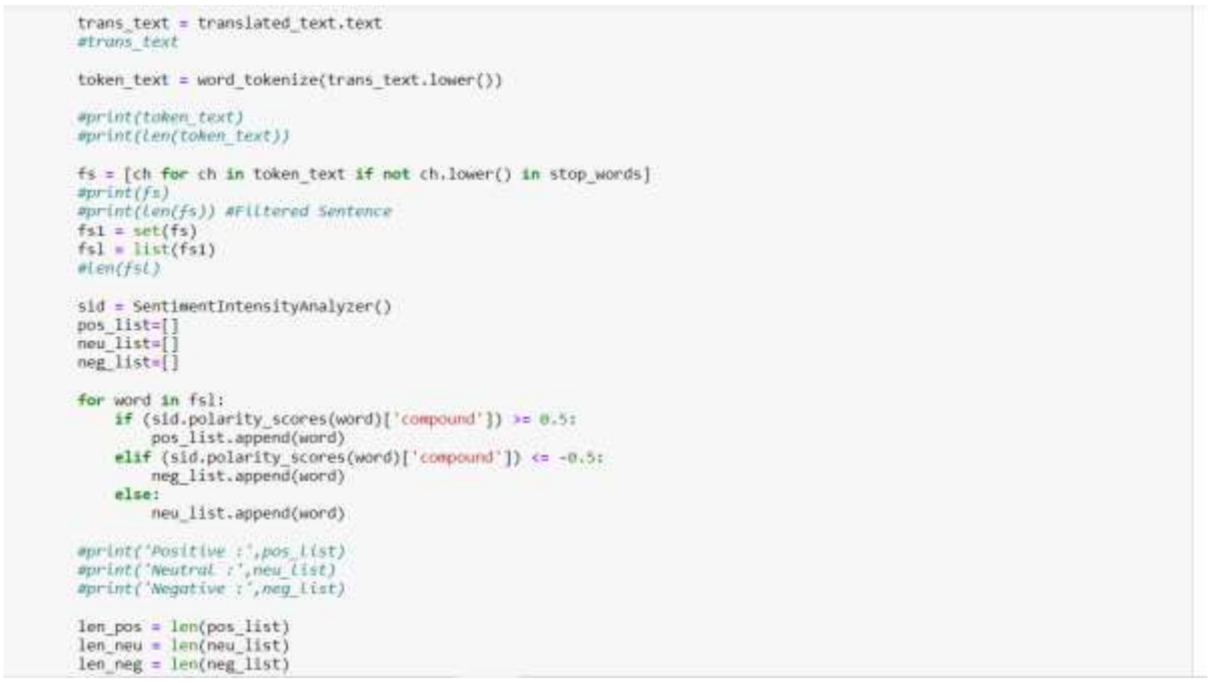
```
In [12]: new_df["Review Rating"].describe()

Out[12]: count    27330.000000
         mean         4.490999
         std          0.891704
         min          1.000000
         25%          4.000000
         50%          5.000000
         75%          5.000000
         max          5.000000
         Name: Review Rating, dtype: float64
```

Based on the Review Text, firstly we removed stop words from the text using Natural Language Toolkit (NLTK) then we have taken out the positive words, neutral words, and negative words from the text.

Based on the frequency of positive words, neutral words, and negative words in the given review text, we try to predict its review rating.

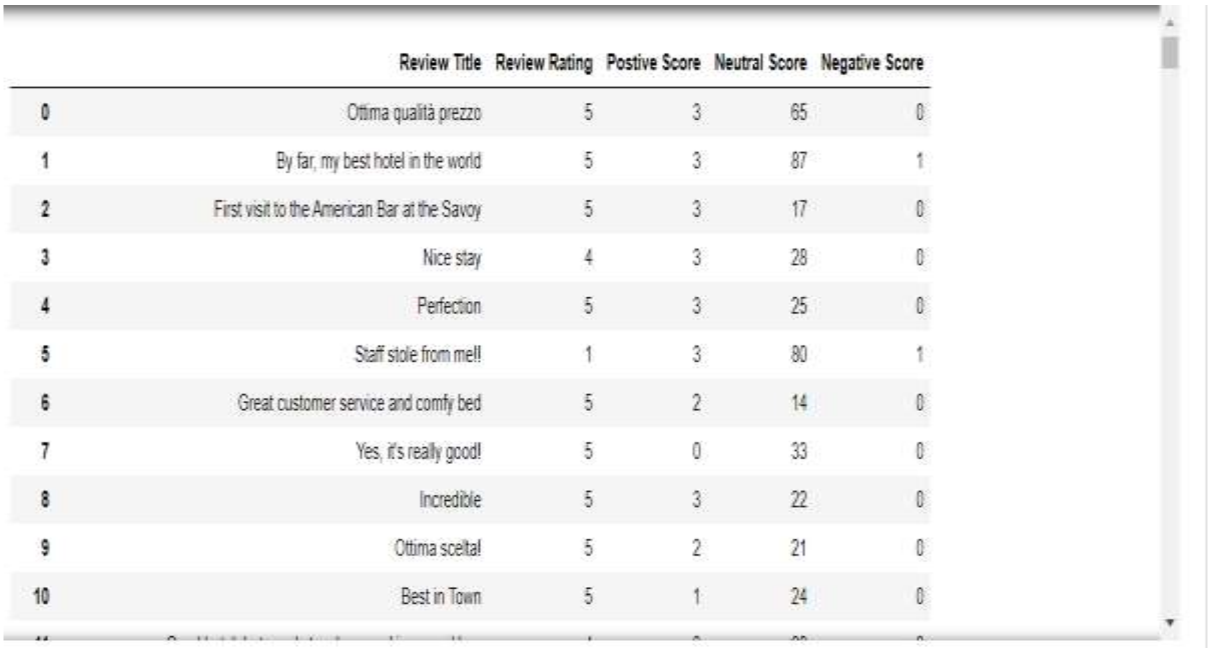


```
In [25]: from googletrans import Translator

         import nltk
         from nltk.tokenize import word_tokenize
         from nltk.corpus import stopwords

         from nltk.sentiment.vader import SentimentIntensityAnalyzer

In [27]: review = new_df["Review Text"]

         translator = Translator()

         stop_words = set(stopwords.words('english'))

In [ ]: x=25871

        while x<27331:
            if x not in [44,237, 505,658,922,1243,17429,19251]:
                rat = new_df["Review Rating"]
                rat1 = rat[x]

                t_rat = new_df["Review Title"]
                t_rat1 = t_rat[x]

                text = review[x].split()

                #text
                ct = " ".join(ch for ch in text if ch.isalnum())
                #print(review)

                translated_text = translator.translate(ct)
                #print(translated_text.text)
```



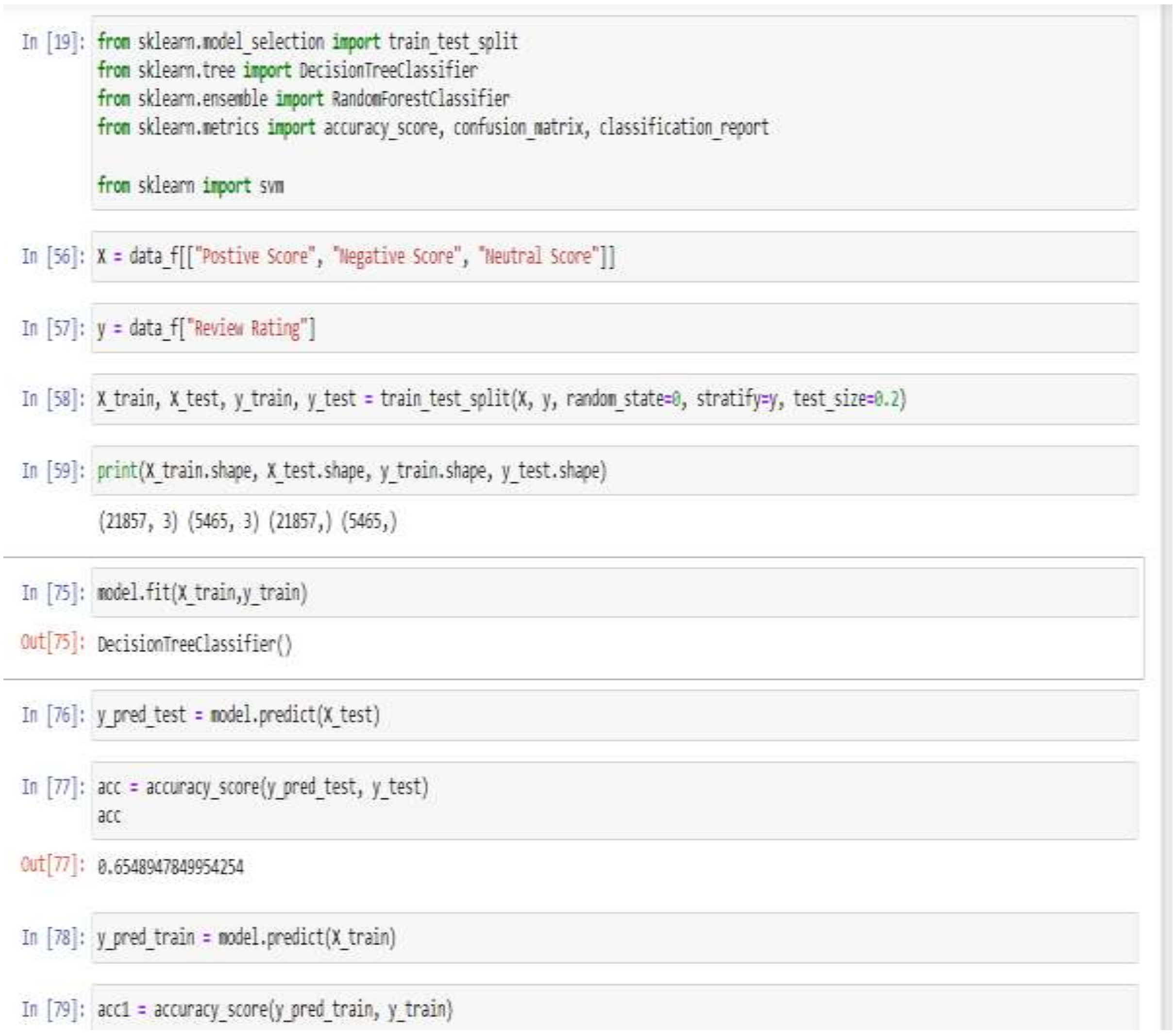
```
    trans_text = translated_text.text
    #trans_text

    token_text = word_tokenize(trans_text.lower())

    #print(token_text)
    #print(len(token_text))

    fs = [ch for ch in token_text if not ch.lower() in stop_words]
    #print(fs)
    #print(len(fs)) #Filtered Sentence
    fs1 = set(fs)
    fsl = list(fs1)
    #len(fsl)

    sid = SentimentIntensityAnalyzer()
    pos_list=[]
    neu_list=[]
    neg_list=[]

    for word in fsl:
        if (sid.polarity_scores(word)['compound']) >= 0.5:
            pos_list.append(word)
        elif (sid.polarity_scores(word)['compound']) <= -0.5:
            neg_list.append(word)
        else:
            neu_list.append(word)

    #print('Positive :',pos_list)
    #print('Neutral :',neu_list)
    #print('Negative :',neg_list)

    len_pos = len(pos_list)
    len_neu = len(neu_list)
    len_neg = len(neg_list)
```

We have created a new table like..

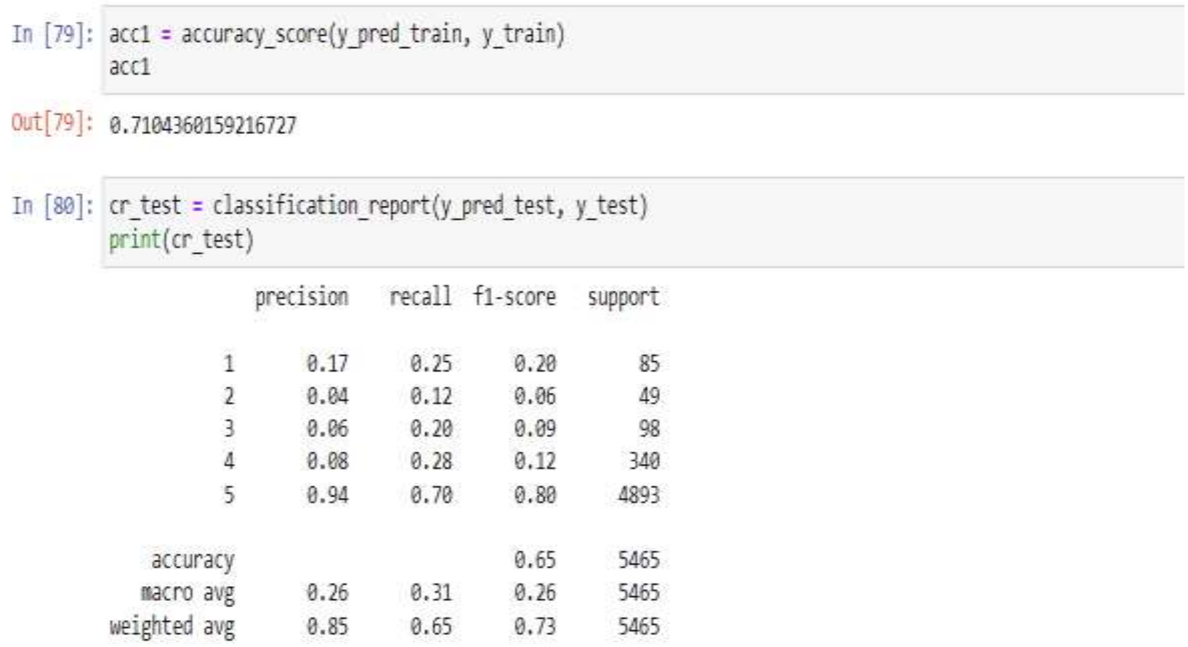


	Review Title	Review Rating	Postive Score	Neutral Score	Negative Score
0	Ottima qualità prezzo	5	3	65	0
1	By far, my best hotel in the world	5	3	87	1
2	First visit to the American Bar at the Savoy	5	3	17	0
3	Nice stay	4	3	28	0
4	Perfection	5	3	25	0
5	Staff stole from me!!	1	3	80	1
6	Great customer service and comfy bed	5	2	14	0
7	Yes, it's really good!	5	0	33	0
8	Incredible	5	3	22	0
9	Ottima scelta!	5	2	21	0
10	Best in Town	5	1	24	0

We have used Decision Tree algorithm to perform Classification



```
In [19]: from sklearn.model_selection import train_test_split
         from sklearn.tree import DecisionTreeClassifier
         from sklearn.ensemble import RandomForestClassifier
         from sklearn.metrics import accuracy_score, confusion_matrix, classification_report

         from sklearn import svm

In [56]: X = data_f[["Postive Score", "Negative Score", "Neutral Score"]]

In [57]: y = data_f["Review Rating"]

In [58]: X_train, X_test, y_train, y_test = train_test_split(X, y, random_state=0, stratify=y, test_size=0.2)

In [59]: print(X_train.shape, X_test.shape, y_train.shape, y_test.shape)

         (21857, 3) (5465, 3) (21857,) (5465,)

In [75]: model.fit(X_train,y_train)

Out[75]: DecisionTreeClassifier()

In [76]: y_pred_test = model.predict(X_test)

In [77]: acc = accuracy_score(y_pred_test, y_test)
         acc

Out[77]: 0.6548947849954254

In [78]: y_pred_train = model.predict(X_train)

In [79]: acc1 = accuracy_score(y_pred_train, y_train)
```



```
In [79]: acc1 = accuracy_score(y_pred_train, y_train)
         acc1

Out[79]: 0.7104360159216727

In [80]: cr_test = classification_report(y_pred_test, y_test)
         print(cr_test)

              precision    recall  f1-score   support

           1       0.17      0.25      0.20        85
           2       0.04      0.12      0.06        49
           3       0.06      0.20      0.09        98
           4       0.08      0.28      0.12       340
           5       0.94      0.70      0.80      4893

    accuracy                           0.65      5465
   macro avg       0.26      0.31      0.26      5465
weighted avg       0.85      0.65      0.73      5465
```

4. Conclusion

We have got 66% accuracy on testing data and 71% accuracy on the training data

As in the final table, we saw that precision for class 1 is 17% and for class 5 is 94%, implying our model can easily capture the good rating but find it difficult to classify class 1. This may be because people who give an almost similar proportion of positive and negative words in their review but give fewer ratings like 1 or 2