FLIP ROBO

Ratings Prediction

Submitted By-
Akanksha Amarnani

Acknowledgement

I would like to thank Almighty for giving me the confidence to pursue this project. Further, the concepts from DataTrained Academy guided me to complete the project.
In addition I would like to thank my mentor from Flip Robo Technology, Ms Khushboo Garg for clarifying my doubts and queries.

The references used for the completion of this project are-

- Predicting the ratings of reviews of a hotel using Machine Learning, B. Shiv Kumar, Analytics Vidhya, Feb 14, 2021
- 1 to 5 Star Ratings — Classification or Regression?, Sebastian Poliak, Towards Data Science, Nov 21, 2020

INTRODUCTION

Business 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

Conceptual Background of the Domain Problem-

The domain related concepts which help us in a better understanding are-

- Web Scraping- Web scraping to scrape reviews from different ecommerce websites
- Exploratory Data Analysis (EDA)- By conducting explanatory data analysis, we obtain a better understanding of our data. This yields insights that can be helpful later when building a model, as well as insights that are independently interesting.
- Splitting the data- The dataset is split into train and test sample using the train test split
- Downsampling the data- As the dataset is imbalanced, downsampling it is required
- Modeling- We apply Logistic Regression, KNeighbor Classifier, Decision Tree Classifier, Random Forest Classifier, Adaboost Classifier, Gradient Boosting Classifier and SVC to ckeck is the user ia a defaulter or not
- Regularization- Models are regularized and the parameters are hypertuned to enhance the efficiency of the models

Review of Literature-

Machine learning is a subfield of Artificial Intelligence (AI) that works with algorithms and technologies to extract useful information from data. Machine learning methods are appropriate in big data since attempting to manually process vast volumes of data would be impossible without the support of machines. Machine learning in computer science attempts to solve problems algorithmically rather than purely mathematically. Therefore, it is based on creating algorithms that permit the machine to learn. However, there are two general groups in machine learning which are supervised and unsupervised. Supervised is where the program gets trained on pre-determined set to be able to predict when a new data is given. Unsupervised is where the program tries to find the relationship and the hidden pattern between the data

The performance of the model build will be measured upon its accuracy to determine the rate of the product based on its review. The features will be reviews and rates of different appliances, in order to predict the rates of other reviews. Further the features can include product and brands to understand the rating scenario of different products and brands. We implement and evaluate various learning methods on the web-scraped dataset. However, proper EDA is to be kept in mind.The data used in the experiment will be handled by using a combination of pre-processing methods to improve the prediction accuracy

Motivation for the Problem Undertaken-

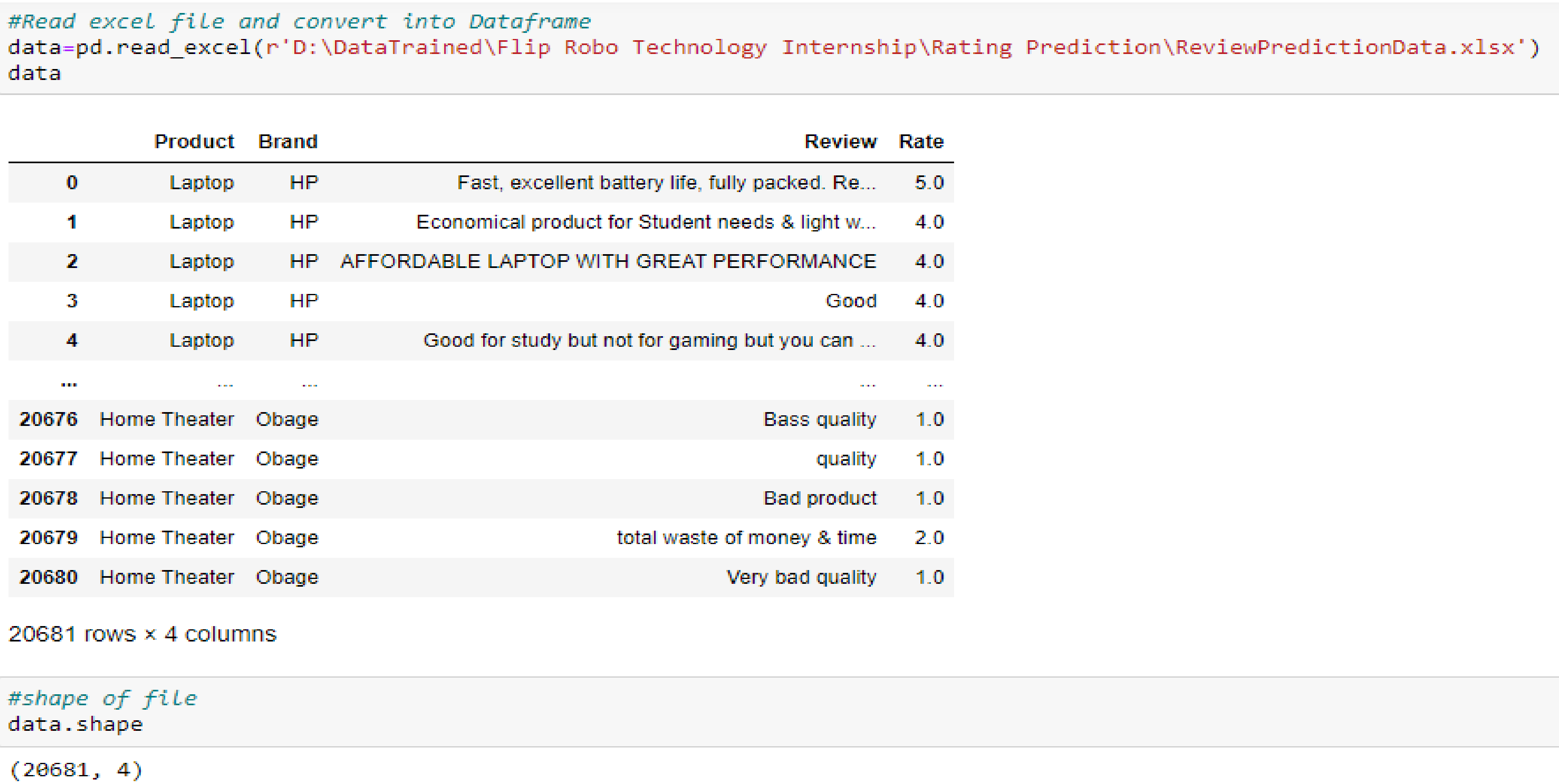
The prediction of the rating of a product based on only its review is considered an unpredictable, cumbersome process. Being a data analyst, it seems to be my responsibility to add the element of prediction in every unpredictable scenario, to solve the cumbersome unsure process into a more reliable, dependent matter. Therefore, the project motivated me to go further and predict the unpredictable. Further, every project has a lot to offer as well. The project and its attributes imparted a lot of knowledge about the technical sector, its dependable and the various criteria which varies the rating of the product

Data Collection

The data is scraped from 2 websites-

1. amazon.in
2. flipkart.com

20681 reviews are scraped from both the websites.



```
#Read excel file and convert into Dataframe
data=pd.read_excel(r'D:\DataTrained\Flip Robo Technology Internship\Rating Prediction\ReviewPredictionData.xlsx')
data
```

	Product	Brand	Review	Rate
0	Laptop	HP	Fast, excellent battery life, fully packed. Re...	5.0
1	Laptop	HP	Economical product for Student needs & light w...	4.0
2	Laptop	HP	AFFORDABLE LAPTOP WITH GREAT PERFORMANCE	4.0
3	Laptop	HP	Good	4.0
4	Laptop	HP	Good for study but not for gaming but you can ...	4.0
...	...	...	...	...
20676	Home Theater	Obage	Bass quality	1.0
20677	Home Theater	Obage	quality	1.0
20678	Home Theater	Obage	Bad product	1.0
20679	Home Theater	Obage	total waste of money & time	2.0
20680	Home Theater	Obage	Very bad quality	1.0

20681 rows × 4 columns

```
#shape of file
data.shape
```

```
(20681, 4)
```

ANALYTICAL PROBLEM FRAMING

EDA Steps and Visualization

- The web scraped data is saved onto an excel sheet
- The excel datasheet is extracted and saved in a dataframe
- The shape of the dataframe is checked-

  There are 20681 rows and 4 columns
- **The columns are as follows-**
  - **Product**
  - **Brand**
  - **Review**
  - **Rate**

The data type of each column is-

- **Product - object**
- **Brand - object**
- **Review - object**
- **Rate - float64**

The null values are checked. The whitespaces, and dashes ('—') are replaced by null values-

- **Product - 0**
- **Brand - 0**
- **Review - 5**
- **Rate - 0**

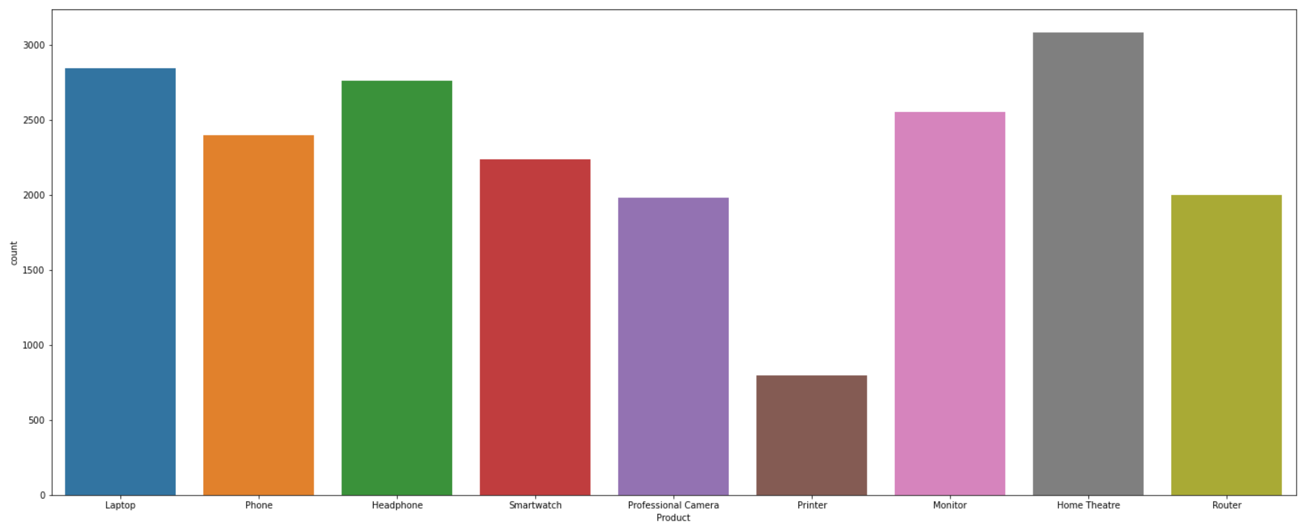
As 5 rows have null values, it is safe to delete these rows

- The shape of the dataframe now is-

There are 20676 rows and 4 columns

The data visualization, value counts and encoding object data for each column

Product

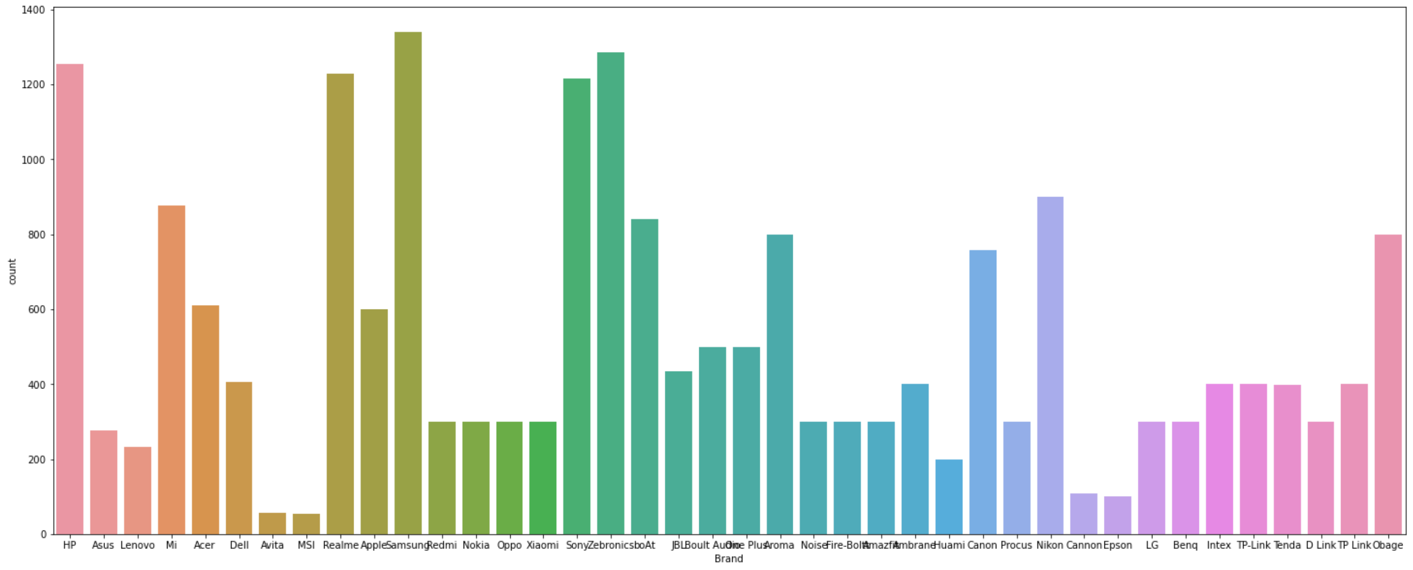


- 3085 are headphone reviews
- 2848 are laptop reviews
- 2557 are monitor reviews
- 2399 are phone reviews
- 2239 are smartwatch reviews
- 1999 are home theater reviews
- 1999 are router reviews
- 1985 are professional camera reviews
- 800 are printer reviews

↓

Encoding object data in numeric using Label Encoder

Brand



- Samsung -1340
- Zebronics -1286
- HP -1254
- Realme -1230
- Sony -1217
- Nikon -900
- Mi -878
- boAt- 840
- Aroma -800
- Obage- 800

- Canon -759
- Acer -610
- Apple -600
- Boult Audio- 500
- One Plus -500
- JBL- 434
- Dell -405
- Ambrane -400
- TP Link -400
- TP-Link -400

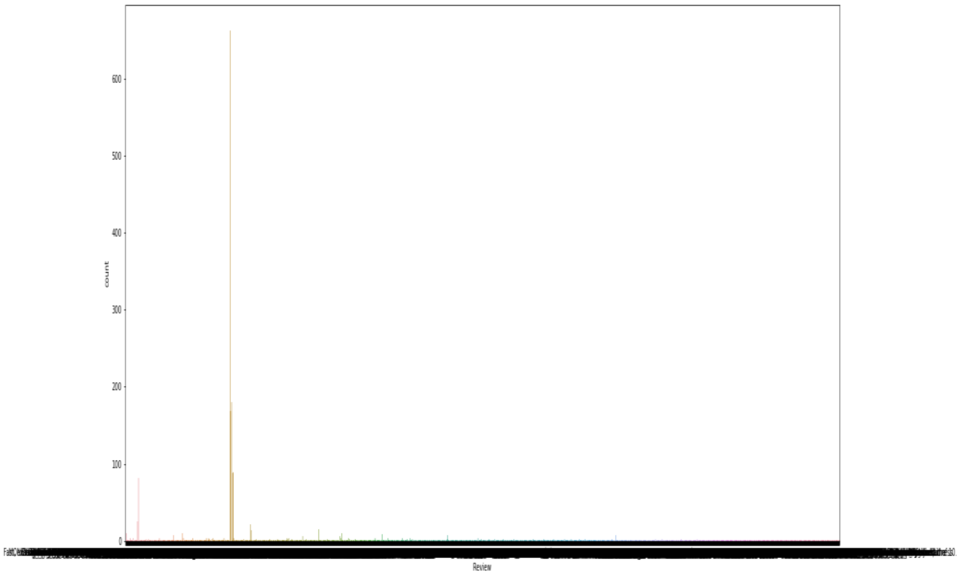
- Intex -400
- Tenda -399
- Procus -300
- D Link -300
- Benq -300
- LG -300
- Nokia -300
- Redmi -300
- Amazfit -300
- Fire-Boltt -300

- Oppo -300
- Xiaomi -300
- Noise -299
- Asus -277
- Lenovo -232
- Huami -200
- Cannon -107
- Epson- 100
- Avita -56
- MSI -53

$\downarrow$

Encoding object data in numeric using Label Encoder

Review

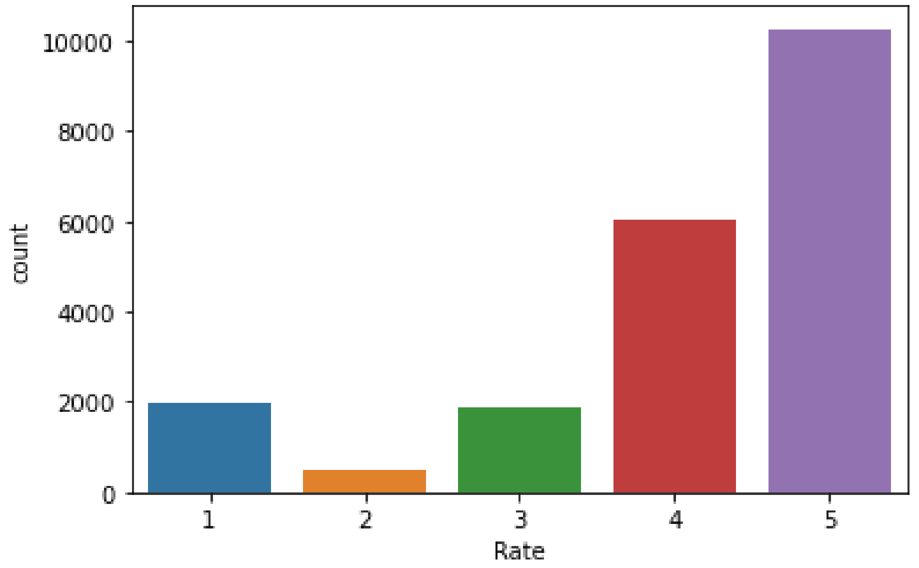


- Majority of the reviews were "Wonderful', 'Good', 'Excellent'.

↓

Encoding object data in numeric using Label Encoder

Rate

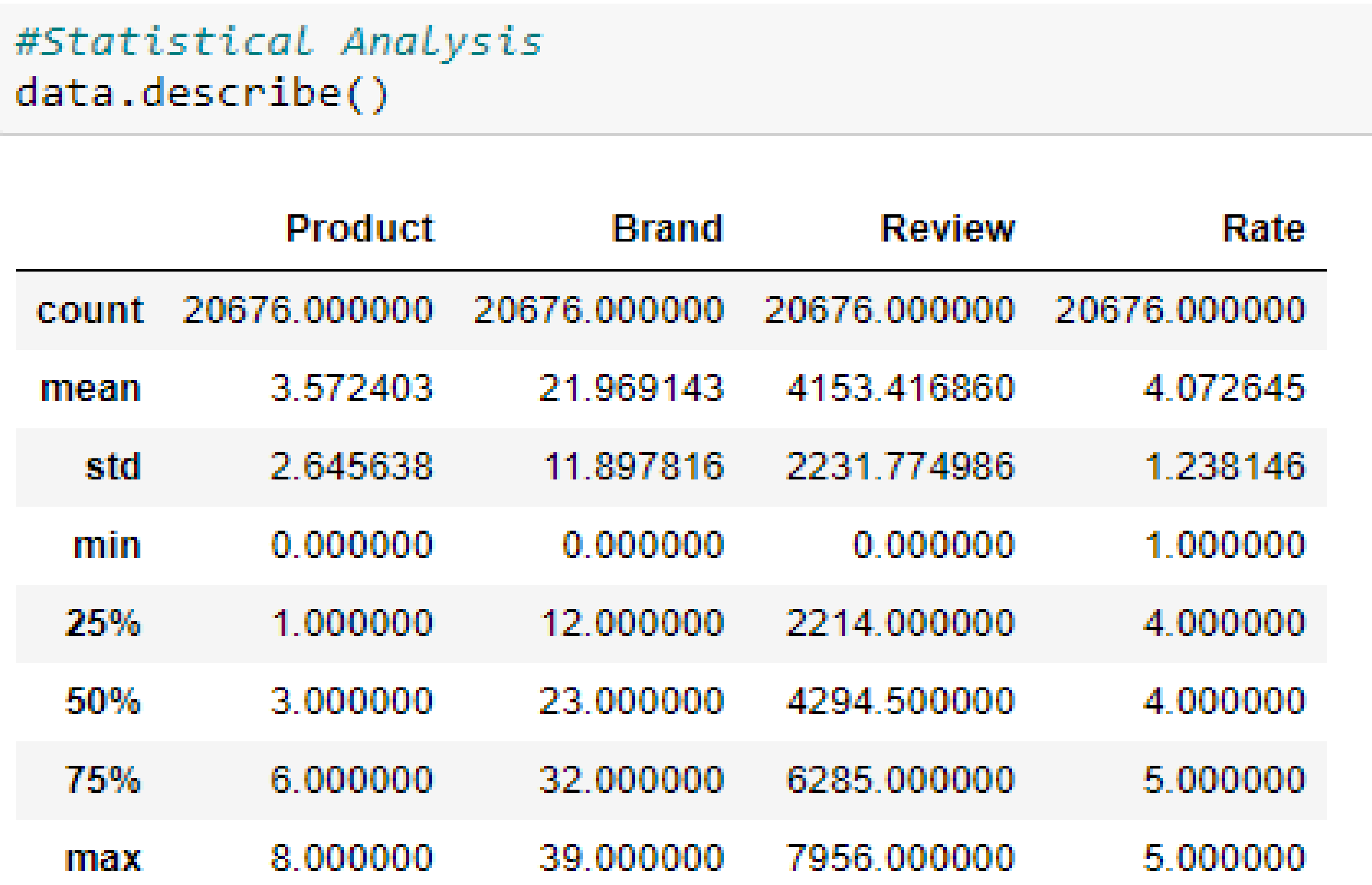


- 10270 products have 5 star rating
- 6058 products have 4 star rating
- 1963 products have 1 star rating
- 1891 products have 3 star rating
- 494 products have 2 stars rating

↓

Encoding object data in numeric using Label Encoder

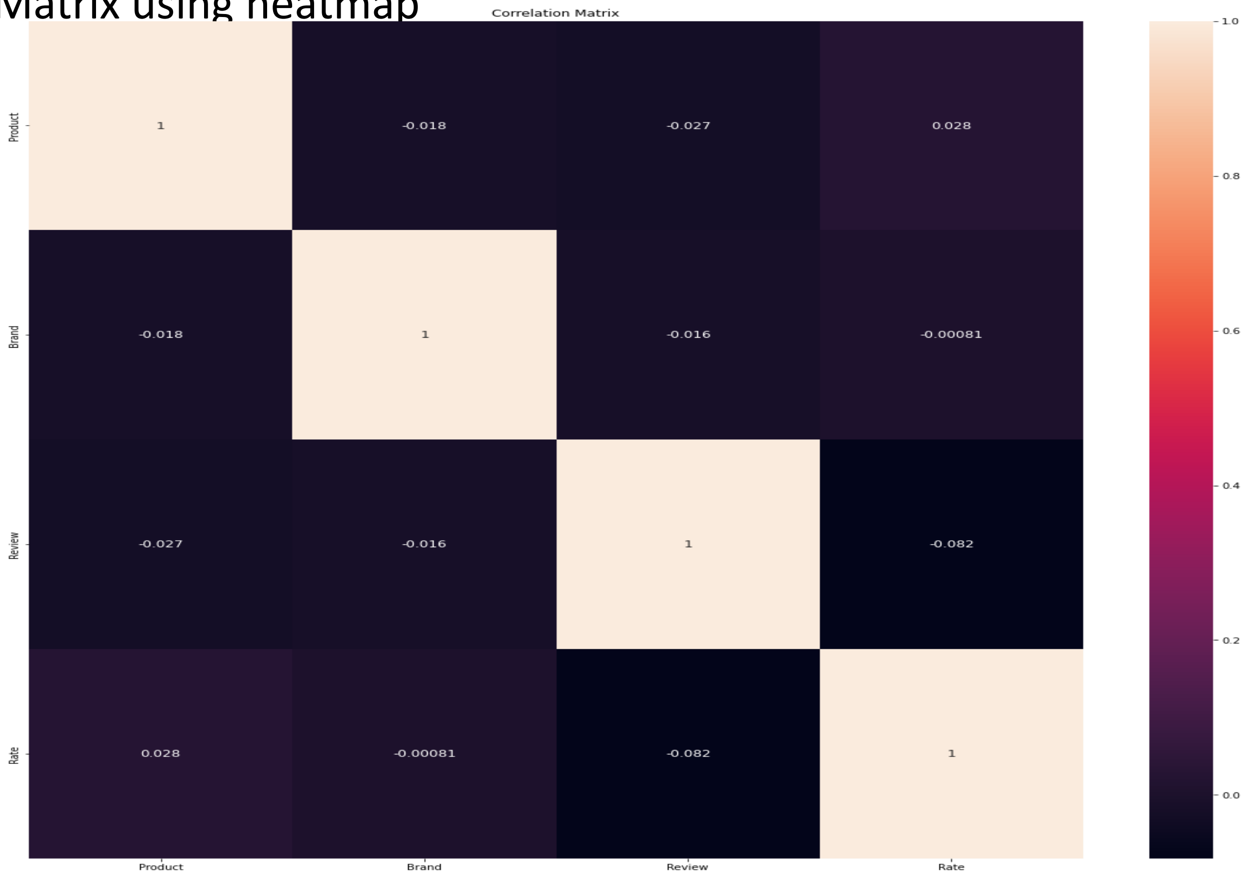
- Statistical analysis using describe method-



```
#Statistical Analysis
data.describe()
```

	Product	Brand	Review	Rate
count	20676.000000	20676.000000	20676.000000	20676.000000
mean	3.572403	21.969143	4153.416860	4.072645
std	2.645638	11.897816	2231.774986	1.238146
min	0.000000	0.000000	0.000000	1.000000
25%	1.000000	12.000000	2214.000000	4.000000
50%	3.000000	23.000000	4294.500000	4.000000
75%	6.000000	32.000000	6285.000000	5.000000
max	8.000000	39.000000	7956.000000	5.000000

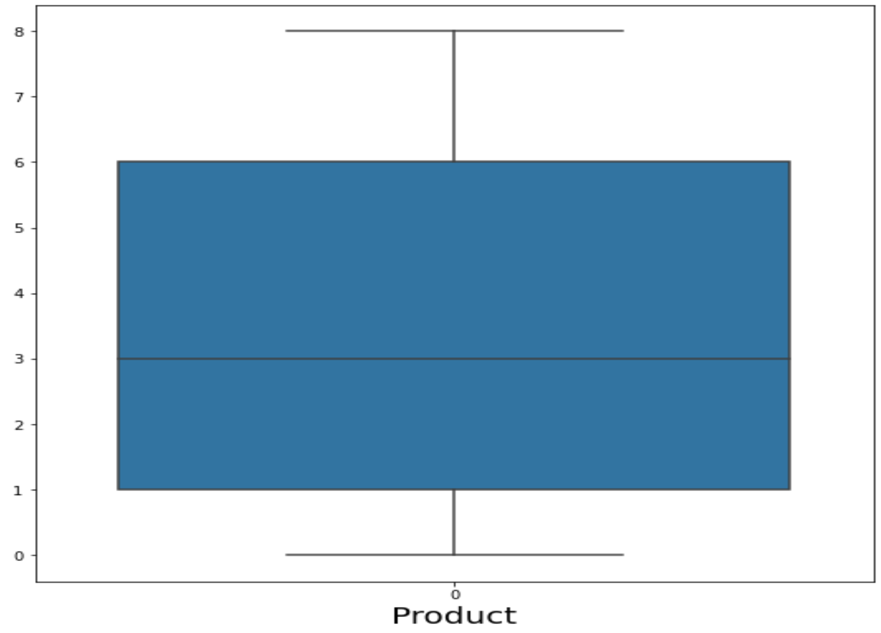
The Correlation Matrix using heatmap

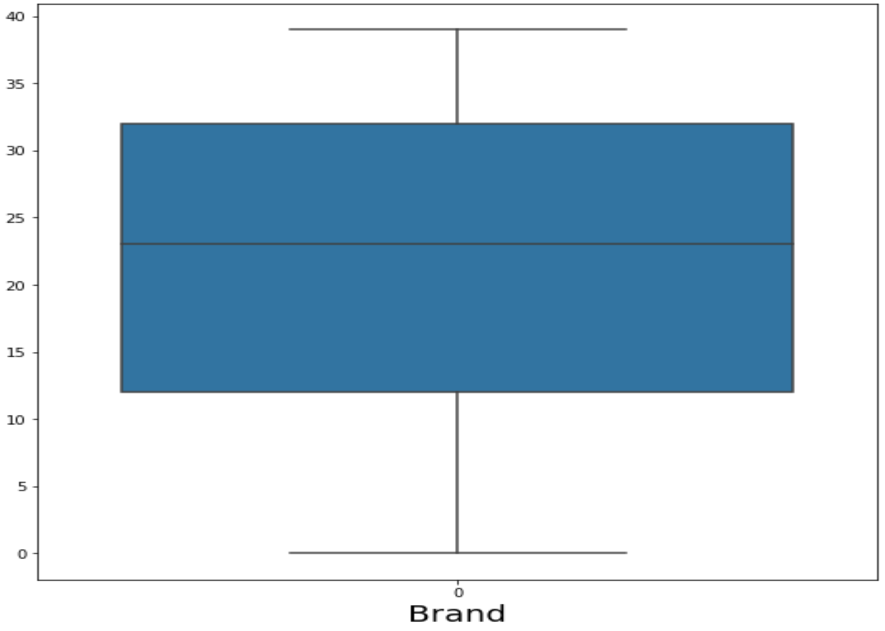


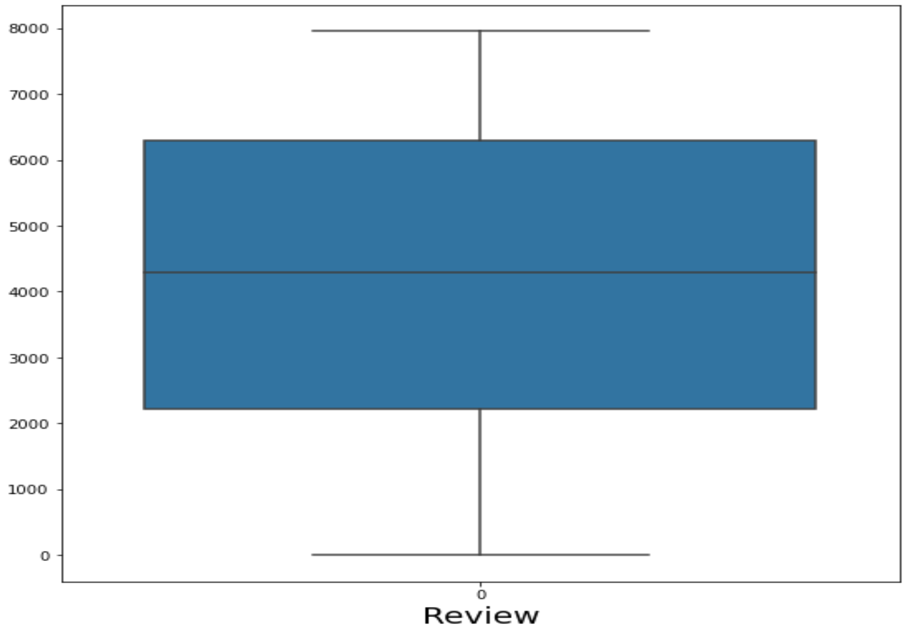
- Correlation between the columns and the label ‘Rate’ using corr method-
  - Product: 0.027645
  - Brand: -0.000810
  - Review: -0.081948

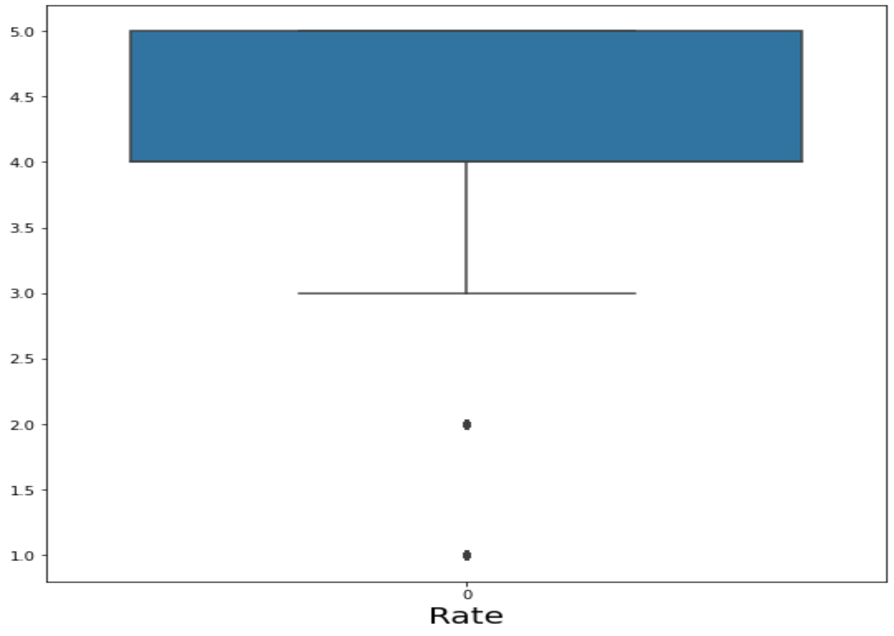
- Product is 2.72% positively correlated to ‘Rate’
- Review is 8.1% negatively correlated to ‘Rate’

- Visualizing outliers using boxplot method-









- Removing outliers using zscore method-

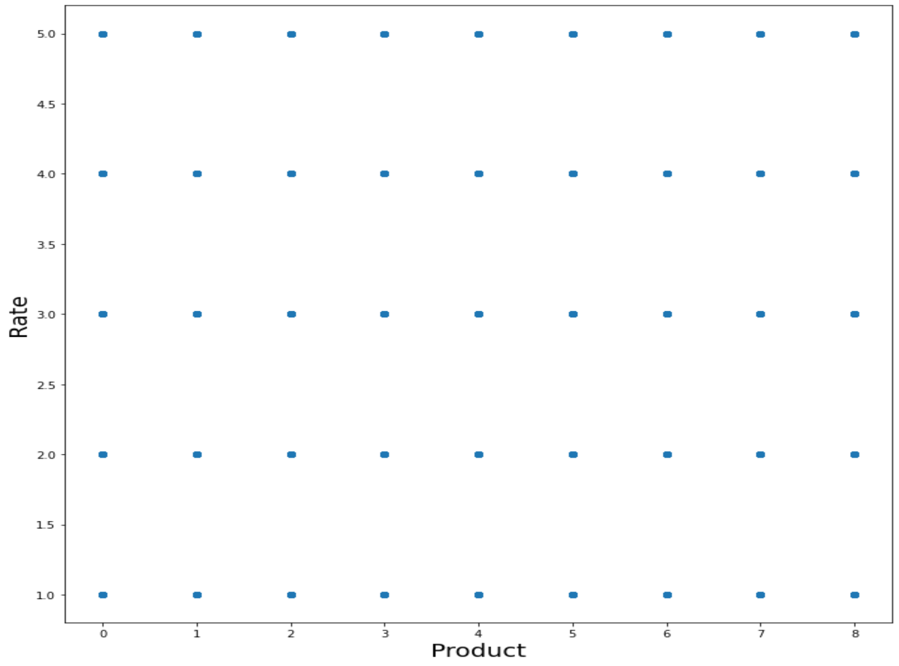
  On removing the outliers the data loss is 0%, which is acceptable, hence outliers are removed

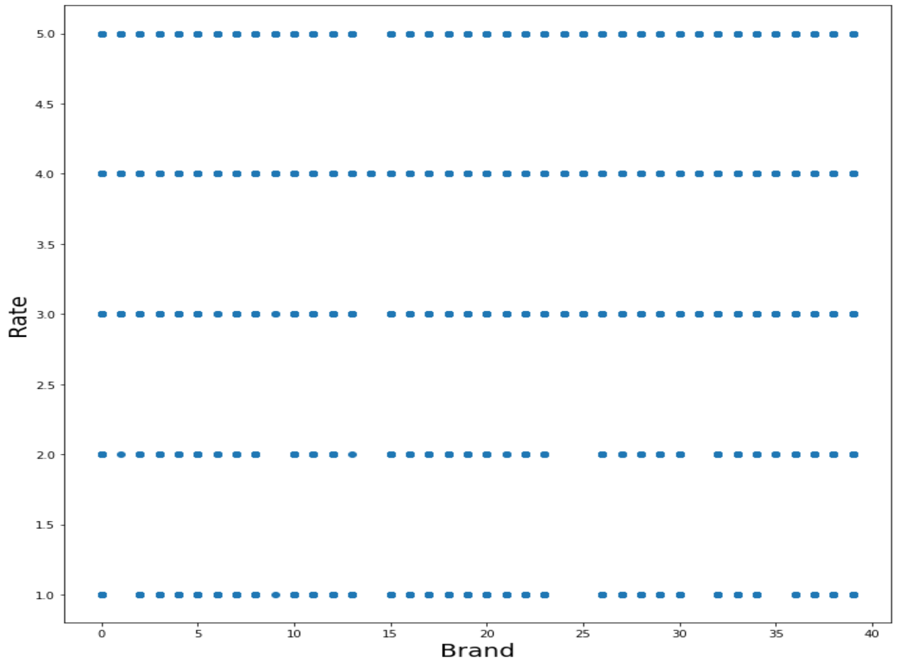
- The dataset is divided into x(features) and y (label)-

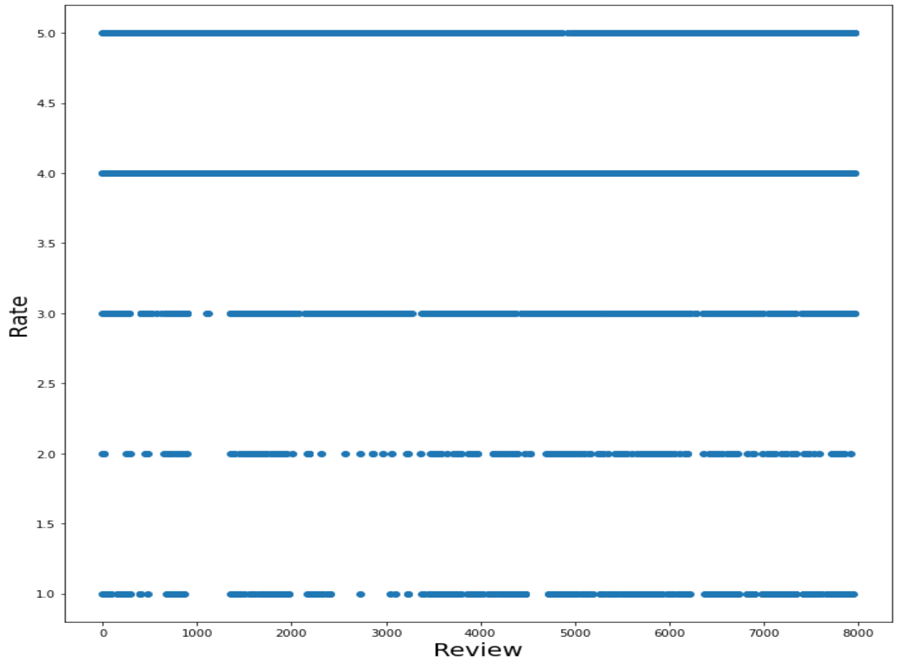
  The x contains all the features other than the label ‘Rate’

  The y contains only the label ‘Rate’

- Visualizing relationship between features and label-





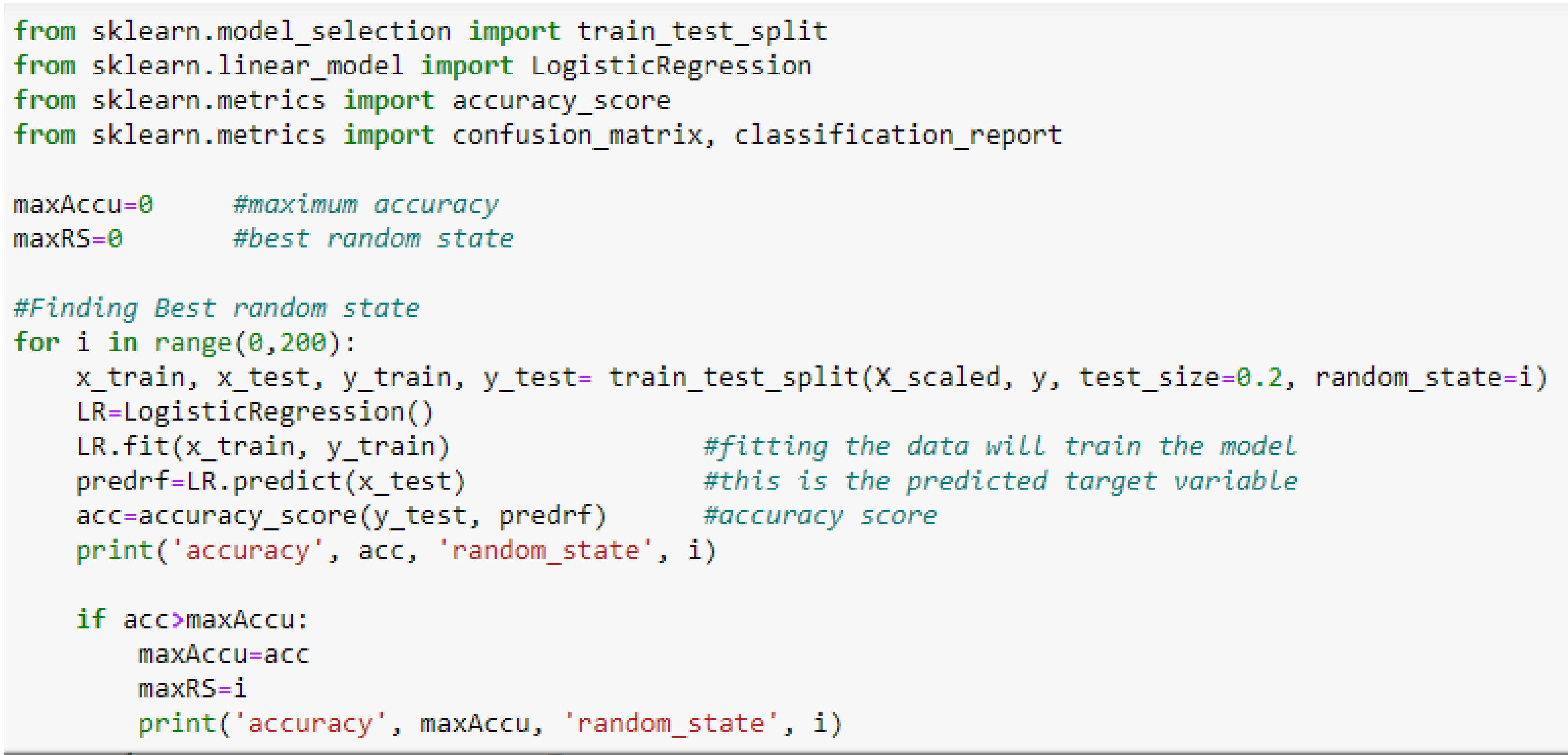


- The skewness observed in graphical analysis was confirmed by using the skew method-
  - Product: 0.302151
  - Review: -0.047409
  - Brand: -0.313749
- This skewness was removed using the power transformer
- The x(features) were scaled using the Standard Scaler
- The dataset was divided into train and test set using train test split and the best random state was found to be 3

Sofware Requirements-

- Jupyter Notebook – Interface for the program
- Pandas – for datafram working
- Numpy – to deal with null data
- matplotlib.pyplot – for data visualization
- Seaborn - for data visualization
- Warnings- to omit warnings
- sklearn.preprocessing – to import powertransform
- scipy.stats – to import zscore
- Zscore- to remove outliers
- power_transform- to remove the skewness in the data
- sklearn.model_selection- to import train_test_split
- train_test_split- to spit dataset into train and test samples
- sklearn.linear_model- to import Logistic Regression model,
- Logistic Regression – to use Logistic Regression model
- sklearn.metrics- to import accuracy, confusion matric and classification report, plot_roc_curve
- imblearn.under_sampling- to import NearMiss
- MearMiss- to undersample the data
- Collections- to import Counter
- Counter- to check the number of data under each classification in the label
- sklearn.neighbors- to import Kneighbours CLassifierModel
- Kneighbours-Classifier to use Kneighbours model
- sklearn.tree- to import DecisionTreeClassifier
- DecisionTreeClassifier- to use DecisionTreeClassifier Model
- sklearn.ensemble – to import RandomForestClassifier, AdaBoostClassifier, GradientBoostingClassifier
- RandomForestClassifier- to use RandomForestClassifier model
- AdaBoostClassifier- to use AdaBoostClassifier model
- GradientBoostingClassifier- to use GradientBoostingClassifier model
- sklearn.svm – to import SVC
- SVC- to use SVC model
- sklearn.model_selection- to import cross_val_score
- cross_val_score- to check for overfitting and underfitting
- sklearn.model_selection- to import GridSearchCV
- GridSearchCV to enhance the working of the model by manipulating the parameters
- plot_roc_curve- to plot ROC_AUC plot

Train Test Split

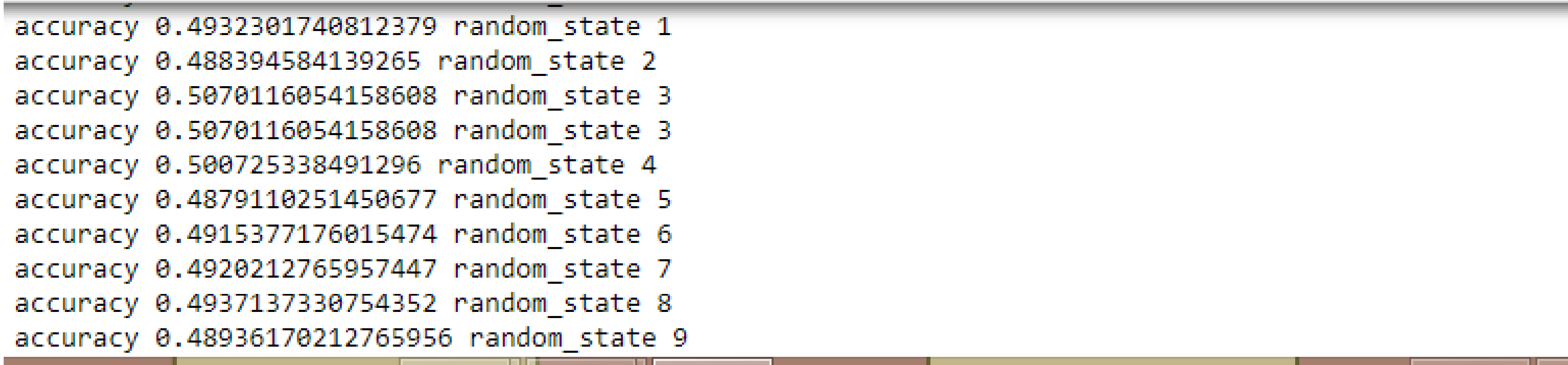


```python
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score
from sklearn.metrics import confusion_matrix, classification_report

maxAccu=0      #maximum accuracy
maxRS=0        #best random state

#Finding Best random state
for i in range(0,200):
    x_train, x_test, y_train, y_test= train_test_split(X_scaled, y, test_size=0.2, random_state=i)
    LR=LogisticRegression()
    LR.fit(x_train, y_train)                #fitting the data will train the model
    predrf=LR.predict(x_test)               #this is the predicted target variable
    acc=accuracy_score(y_test, predrf)      #accuracy score
    print('accuracy', acc, 'random_state', i)

    if acc>maxAccu:
        maxAccu=acc
        maxRS=i
        print('accuracy', maxAccu, 'random_state', i)
```

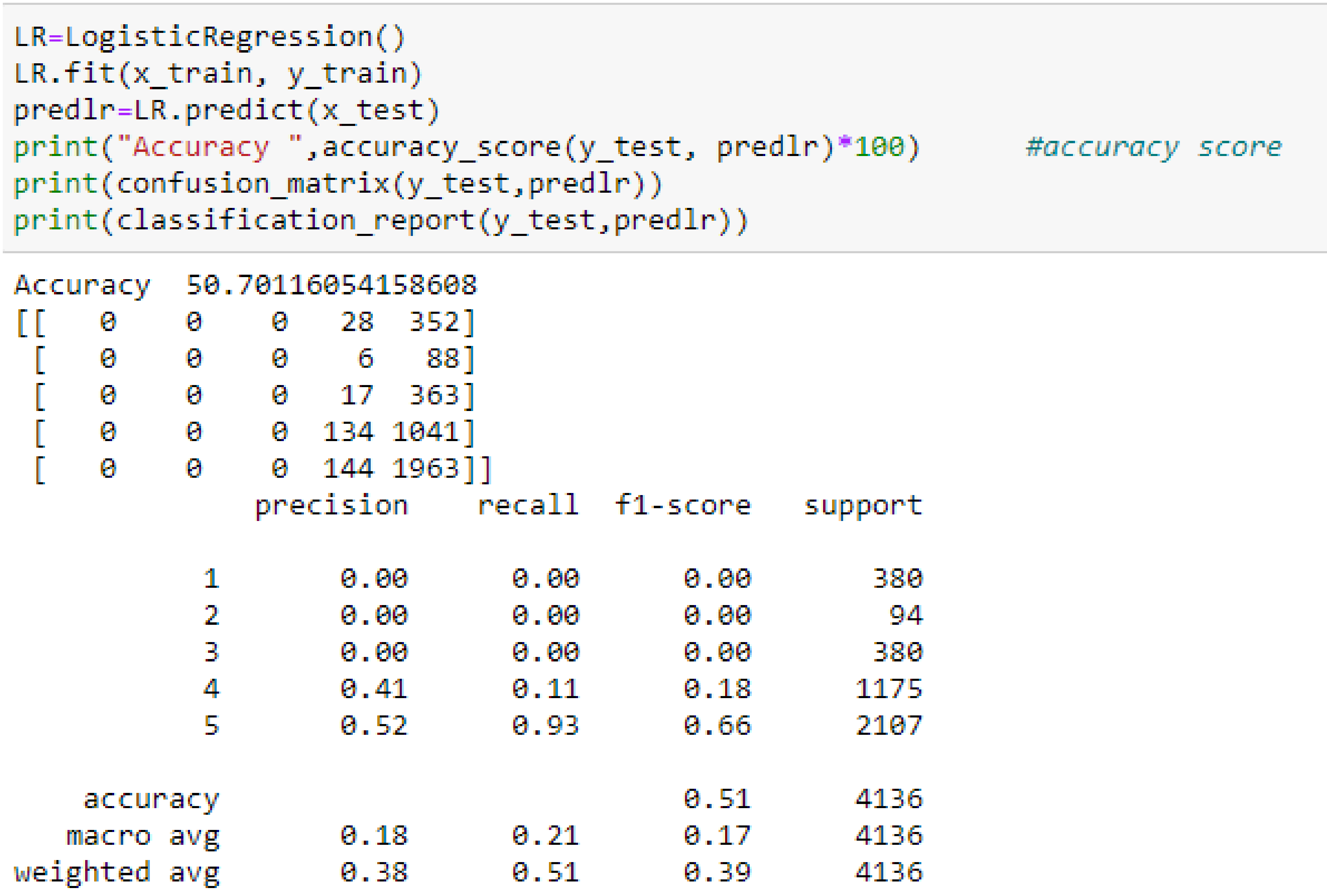


```
accuracy 0.4932301740812379 random_state 1
accuracy 0.488394584139265 random_state 2
accuracy 0.5070116054158608 random_state 3
accuracy 0.5070116054158608 random_state 3
accuracy 0.500725338491296 random_state 4
accuracy 0.4879110251450677 random_state 5
accuracy 0.4915377176015474 random_state 6
accuracy 0.4920212765957447 random_state 7
accuracy 0.4937137330754352 random_state 8
accuracy 0.48936170212765956 random_state 9
```

Model/s Development and Evaluation

The train and test data were applied on different models as follows

Logistic Regression Model



```
LR=LogisticRegression()
LR.fit(x_train, y_train)
predlr=LR.predict(x_test)
print("Accuracy ",accuracy_score(y_test, predlr)*100)      #accuracy score
print(confusion_matrix(y_test,predlr))
print(classification_report(y_test,predlr))
```

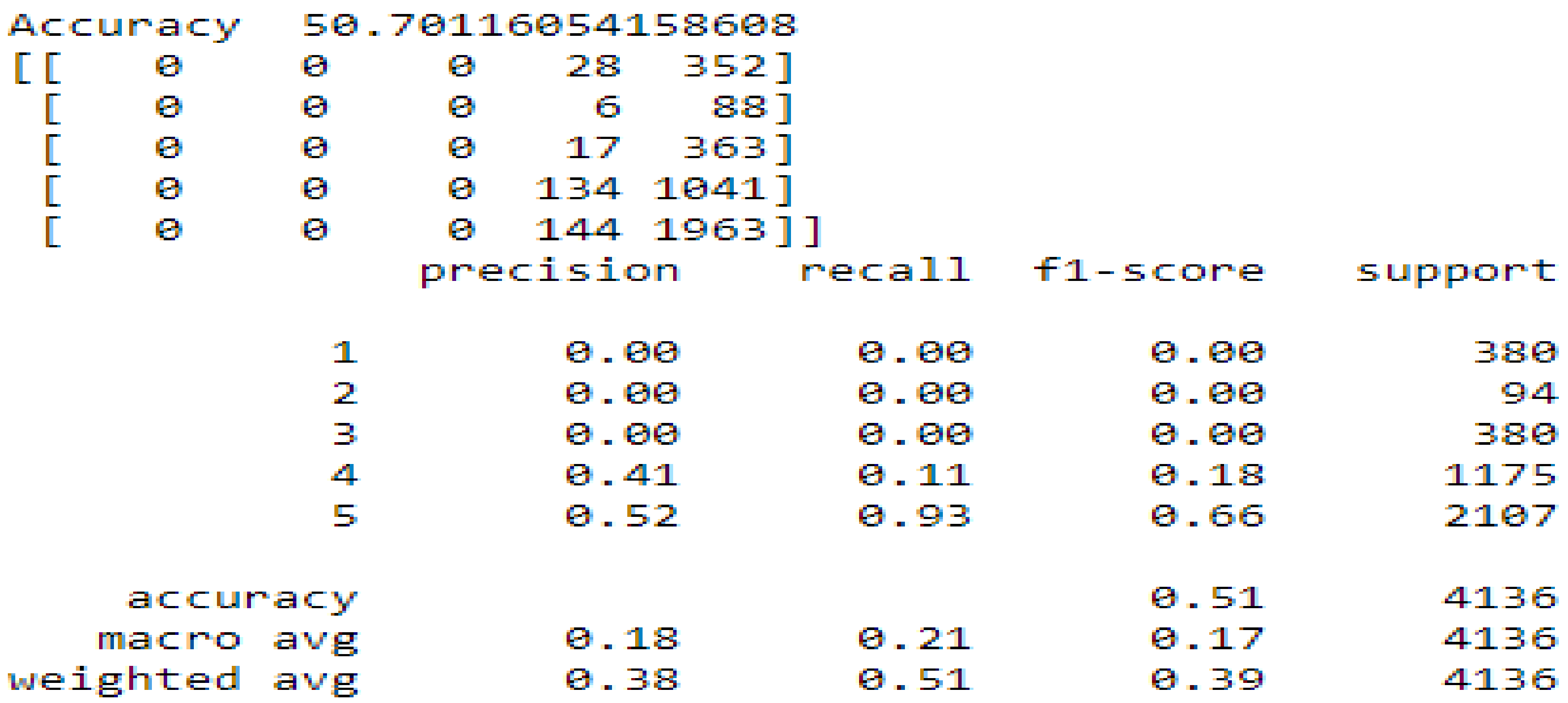
```
Accuracy  50.70116054158608
[[   0    0    0   28  352]
 [   0    0    0    6   88]
 [   0    0    0   17  363]
 [   0    0    0  134 1041]
 [   0    0    0  144 1963]]
              precision    recall  f1-score   support

           1       0.00      0.00      0.00       380
           2       0.00      0.00      0.00        94
           3       0.00      0.00      0.00       380
           4       0.41      0.11      0.18      1175
           5       0.52      0.93      0.66      2107

    accuracy                           0.51      4136
   macro avg       0.18      0.21      0.17      4136
weighted avg       0.38      0.51      0.39      4136
```

- The **Accuracy** for target test and pred_test(data predicted on features_test) is **50.70%**

- The **Confusion Matrix and Classification Report** for target test and pred_test(data predicted on features_test) is –

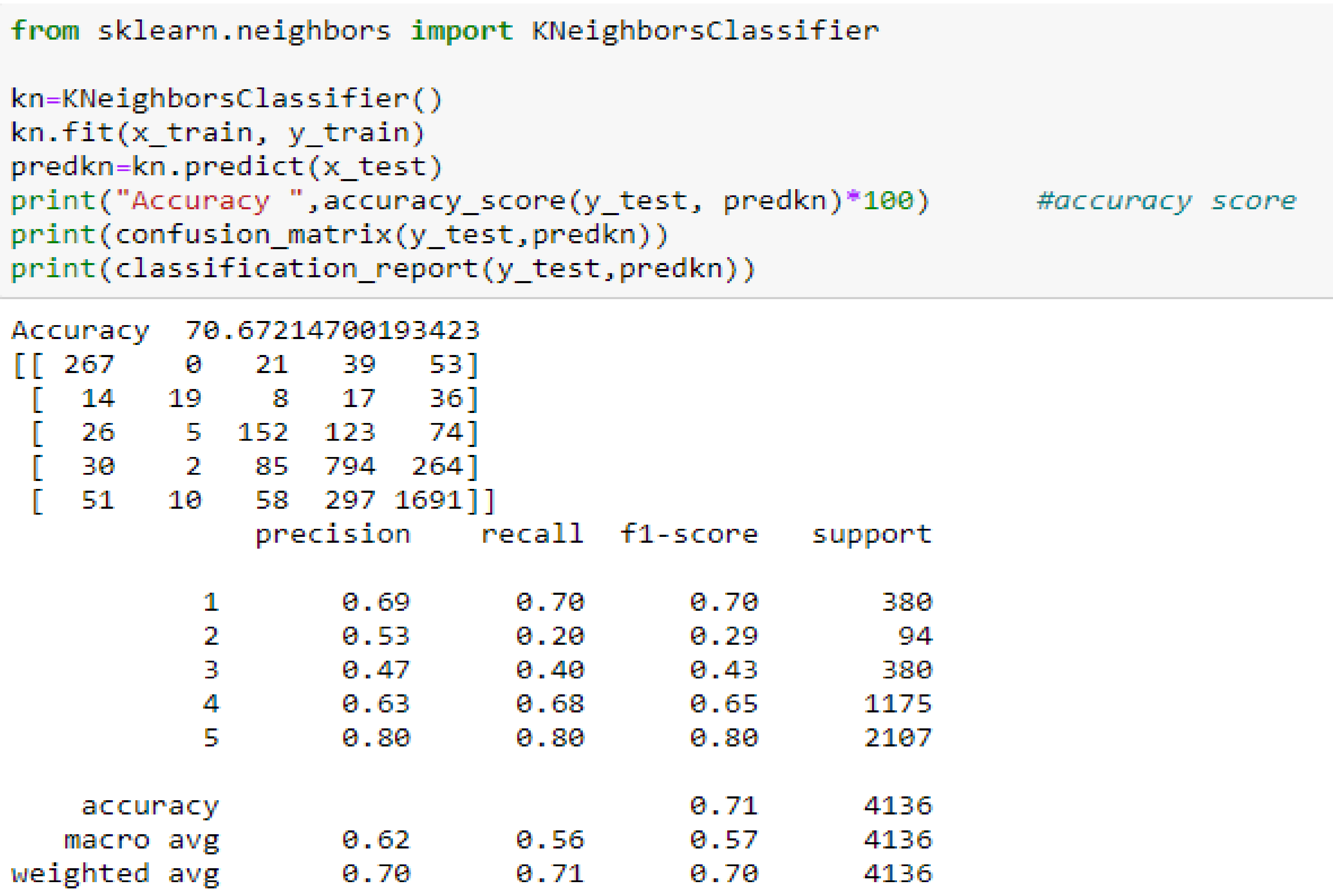


```
Accuracy  50.70116054158608
[[   0    0    0   28  352]
 [   0    0    0    6   88]
 [   0    0    0   17  363]
 [   0    0    0  134 1041]
 [   0    0    0  144 1963]]
              precision    recall  f1-score   support

           1       0.00      0.00      0.00       380
           2       0.00      0.00      0.00        94
           3       0.00      0.00      0.00       380
           4       0.41      0.11      0.18      1175
           5       0.52      0.93      0.66      2107

    accuracy                           0.51      4136
   macro avg       0.18      0.21      0.17      4136
weighted avg       0.38      0.51      0.39      4136
```

KNeighbors Classifier Model



```python
from sklearn.neighbors import KNeighborsClassifier

kn=KNeighborsClassifier()
kn.fit(x_train, y_train)
predkn=kn.predict(x_test)
print("Accuracy ",accuracy_score(y_test, predkn)*100)     #accuracy score
print(confusion_matrix(y_test,predkn))
print(classification_report(y_test,predkn))
```

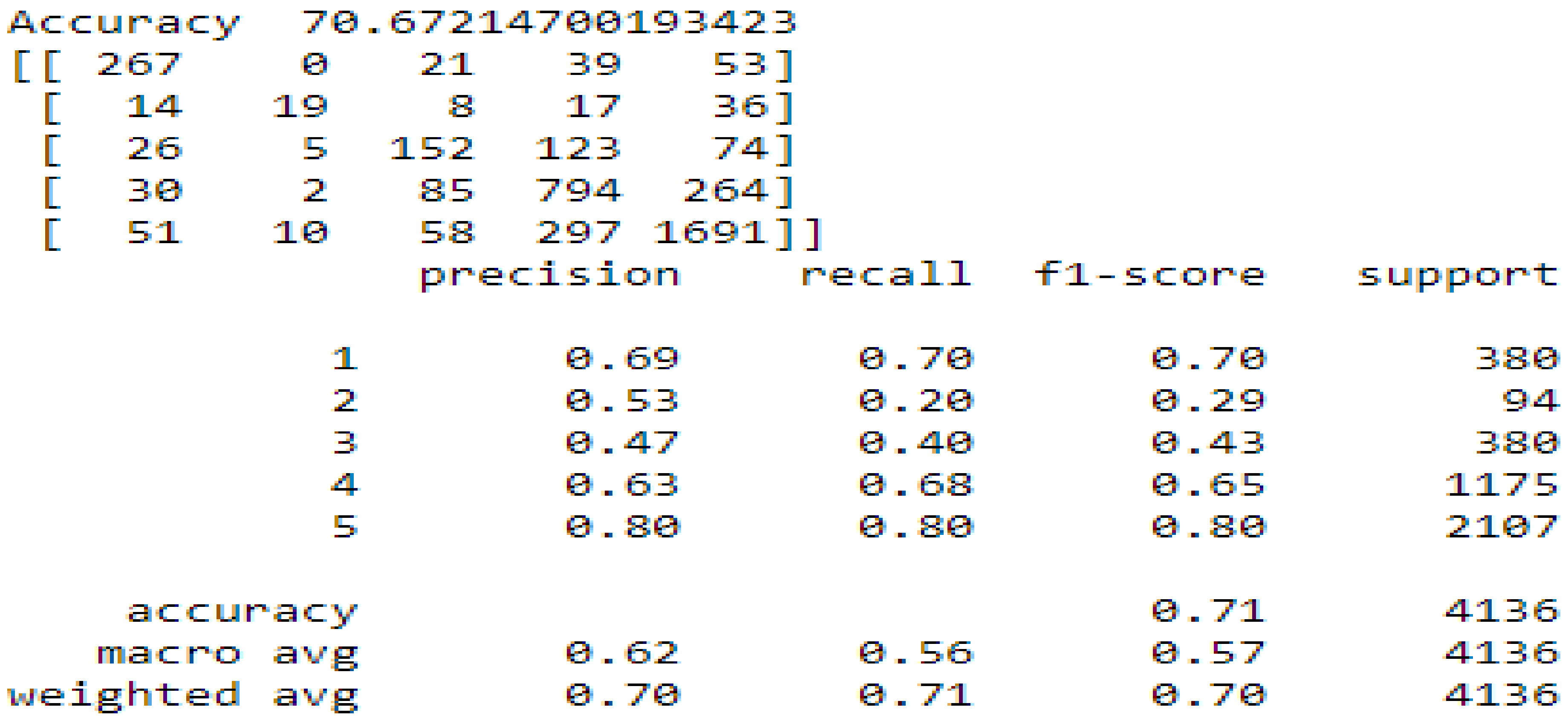
```
Accuracy  70.67214700193423
[[ 267    0   21   39   53]
 [  14   19    8   17   36]
 [  26    5  152  123   74]
 [  30    2   85  794  264]
 [  51   10   58  297 1691]]
              precision    recall  f1-score   support

           1       0.69      0.70      0.70       380
           2       0.53      0.20      0.29        94
           3       0.47      0.40      0.43       380
           4       0.63      0.68      0.65      1175
           5       0.80      0.80      0.80      2107

    accuracy                           0.71      4136
   macro avg       0.62      0.56      0.57      4136
weighted avg       0.70      0.71      0.70      4136
```

- The **Accuracy** for target test and pred_test(data predicted on features_test) is **70.67%**

- The **Confusion Matrix and Classification Report** for target test and pred_test(data predicted on features_test) is –

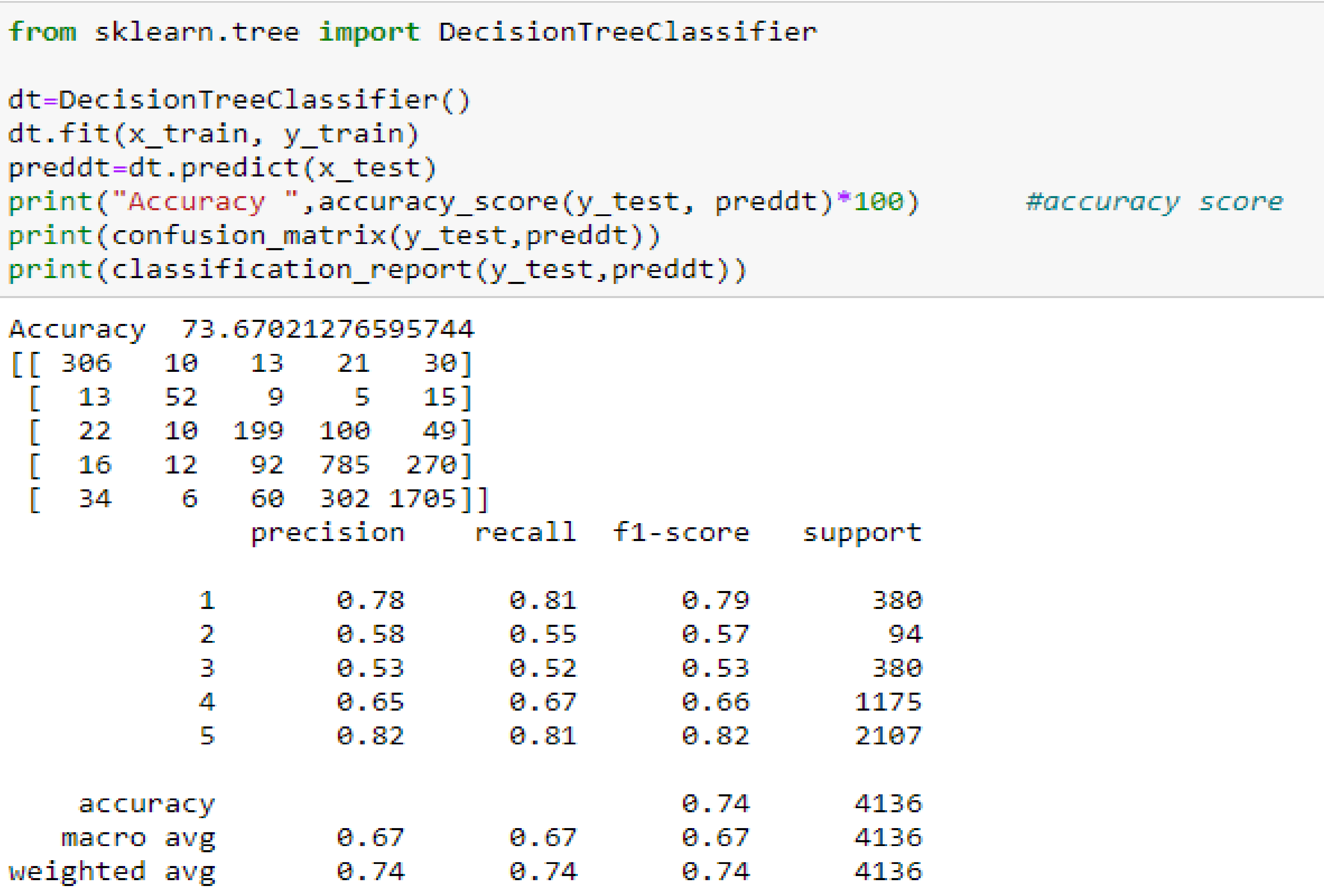


```
Accuracy  70.67214700193423
[[ 267    0   21   39   53]
 [   14   19    8   17   36]
 [   26    5  152  123   74]
 [   30    2   85  794  264]
 [   51   10   58  297 1691]]
              precision    recall  f1-score   support

           1       0.69      0.70      0.70       380
           2       0.53      0.20      0.29        94
           3       0.47      0.40      0.43       380
           4       0.63      0.68      0.65      1175
           5       0.80      0.80      0.80      2107

    accuracy                           0.71      4136
   macro avg       0.62      0.56      0.57      4136
weighted avg       0.70      0.71      0.70      4136
```

Decision Tree Classifier



```python
from sklearn.tree import DecisionTreeClassifier

dt=DecisionTreeClassifier()
dt.fit(x_train, y_train)
preddt=dt.predict(x_test)
print("Accuracy ",accuracy_score(y_test, preddt)*100)      #accuracy score
print(confusion_matrix(y_test,preddt))
print(classification_report(y_test,preddt))
```

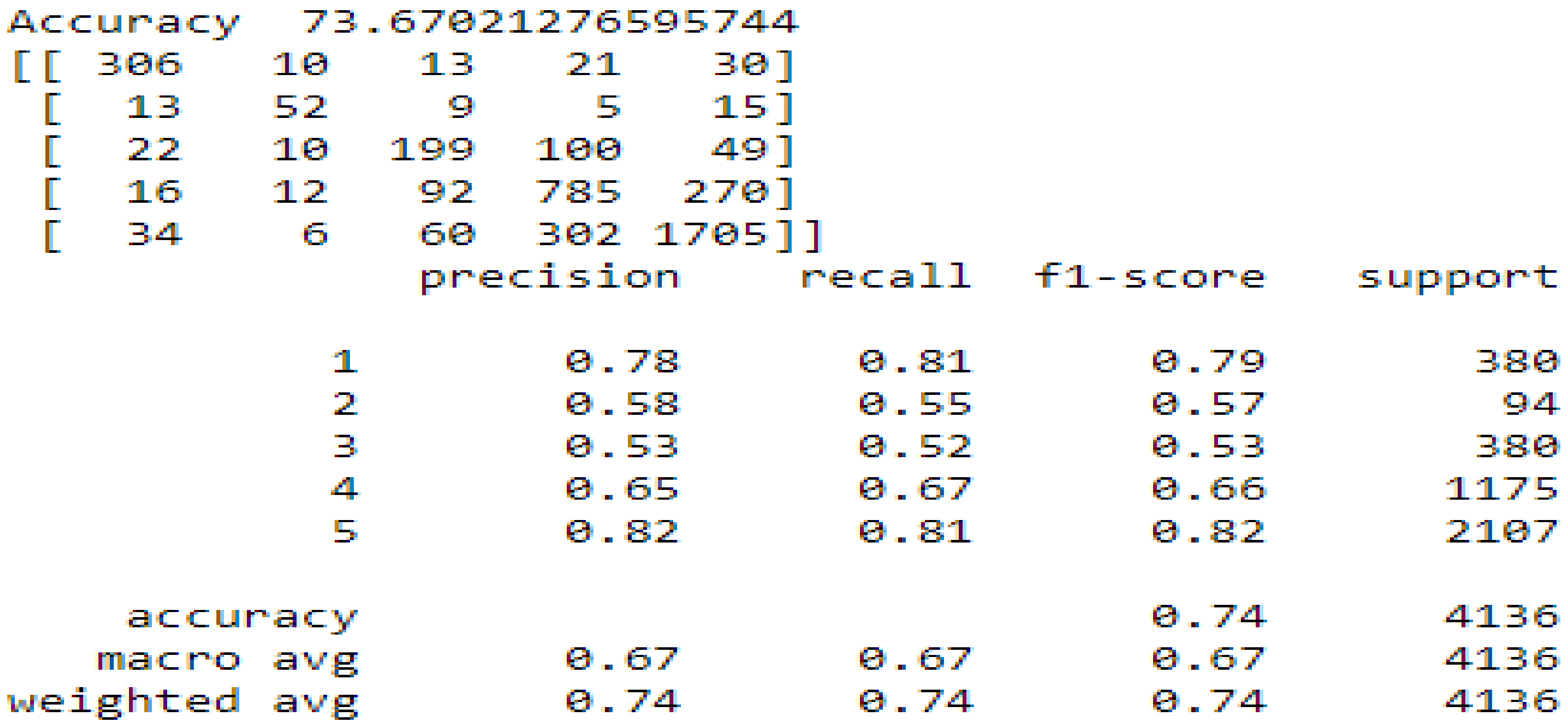
```
Accuracy  73.67021276595744
[[ 306   10   13   21   30]
 [  13   52    9    5   15]
 [  22   10  199  100   49]
 [  16   12   92  785  270]
 [  34    6   60  302 1705]]
              precision    recall  f1-score   support

           1       0.78      0.81      0.79       380
           2       0.58      0.55      0.57        94
           3       0.53      0.52      0.53       380
           4       0.65      0.67      0.66      1175
           5       0.82      0.81      0.82      2107

    accuracy                           0.74      4136
   macro avg       0.67      0.67      0.67      4136
weighted avg       0.74      0.74      0.74      4136
```

- The **Accuracy** for target test and pred_test(data predicted on features_test) is **73.67%**

- The **Confusion Matrix and Classification Report** for target test and pred_test(data predicted on features_test) is –

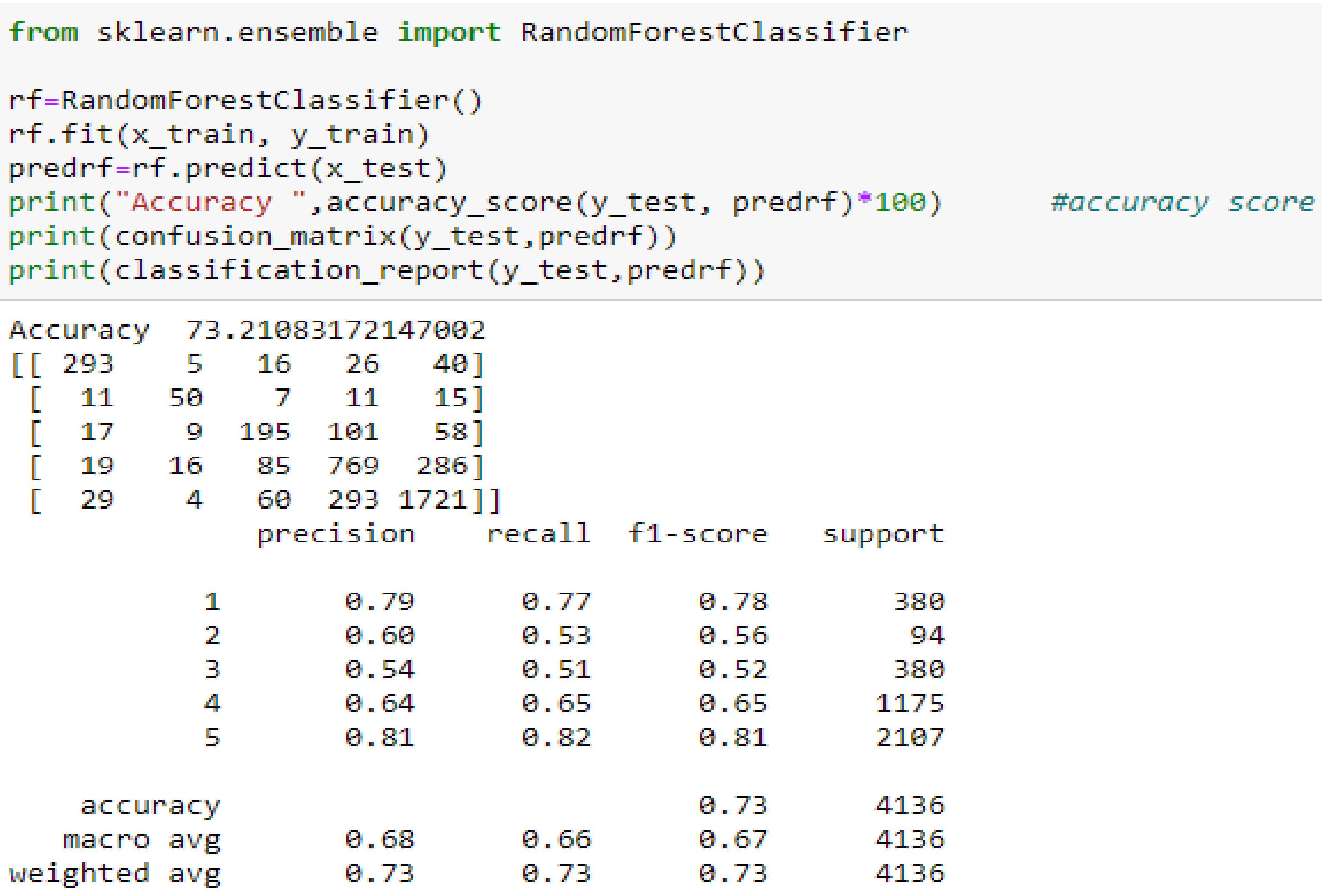


```
Accuracy  73.67021276595744
[[ 306   10   13   21   30]
 [  13   52    9    5   15]
 [  22   10  199  100   49]
 [  16   12   92  785  270]
 [  34    6   60  302 1705]]
              precision    recall  f1-score   support

           1       0.78      0.81      0.79       380
           2       0.58      0.55      0.57        94
           3       0.53      0.52      0.53       380
           4       0.65      0.67      0.66      1175
           5       0.82      0.81      0.82      2107

    accuracy                           0.74      4136
   macro avg       0.67      0.67      0.67      4136
weighted avg       0.74      0.74      0.74      4136
```

Random Forest Classifier



```
from sklearn.ensemble import RandomForestClassifier

rf=RandomForestClassifier()
rf.fit(x_train, y_train)
predrf=rf.predict(x_test)
print("Accuracy ",accuracy_score(y_test, predrf)*100)     #accuracy score
print(confusion_matrix(y_test,predrf))
print(classification_report(y_test,predrf))
```

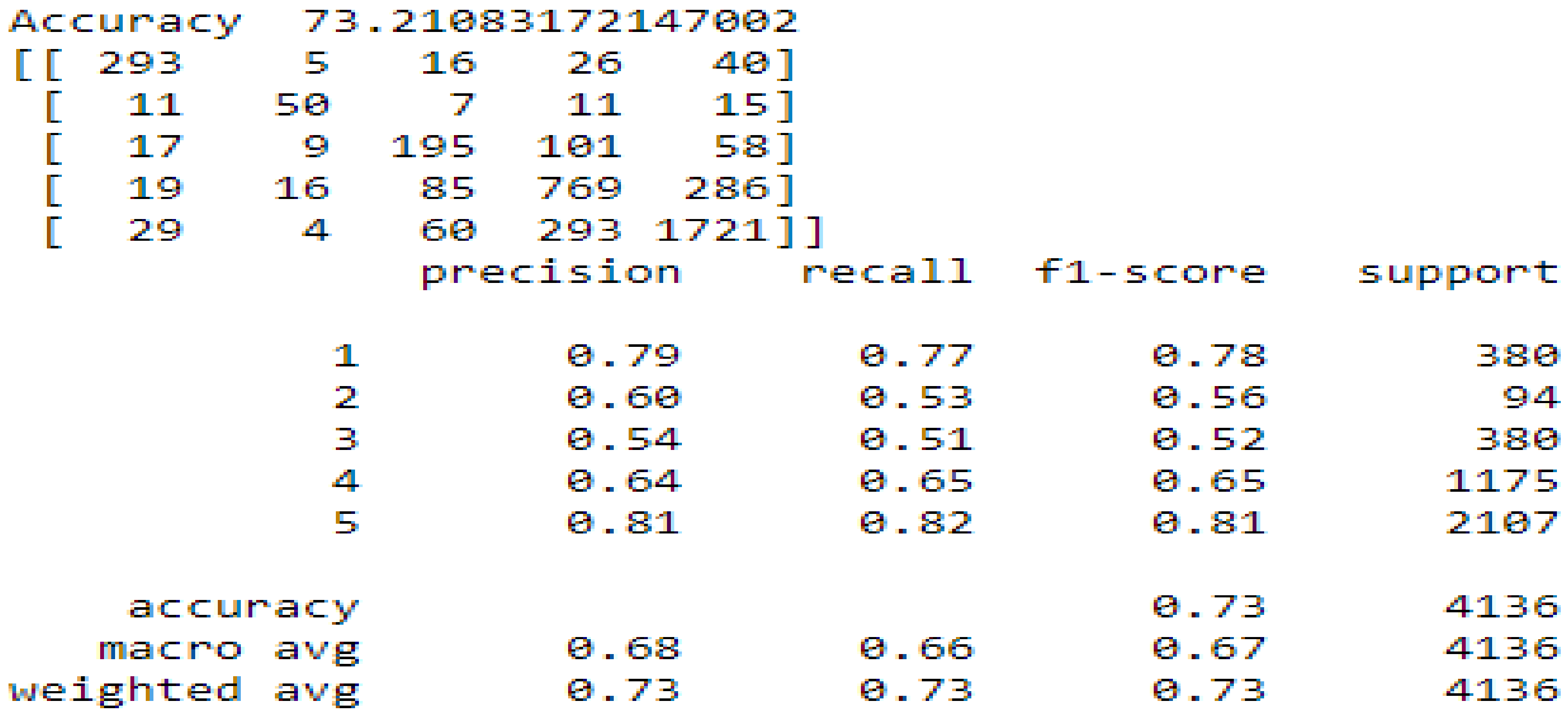
```
Accuracy  73.21083172147002
[[ 293    5   16   26   40]
 [  11   50    7   11   15]
 [  17    9  195  101   58]
 [  19   16   85  769  286]
 [  29    4   60  293 1721]]
              precision    recall  f1-score   support

           1       0.79      0.77      0.78       380
           2       0.60      0.53      0.56        94
           3       0.54      0.51      0.52       380
           4       0.64      0.65      0.65      1175
           5       0.81      0.82      0.81      2107

    accuracy                           0.73      4136
   macro avg       0.68      0.66      0.67      4136
weighted avg       0.73      0.73      0.73      4136
```

- The **Accuracy** for target test and pred_test(data predicted on features_test) is **73.21%**

- The **Confusion Matrix and Classification Report** for target test and pred_test(data predicted on features_test) is –

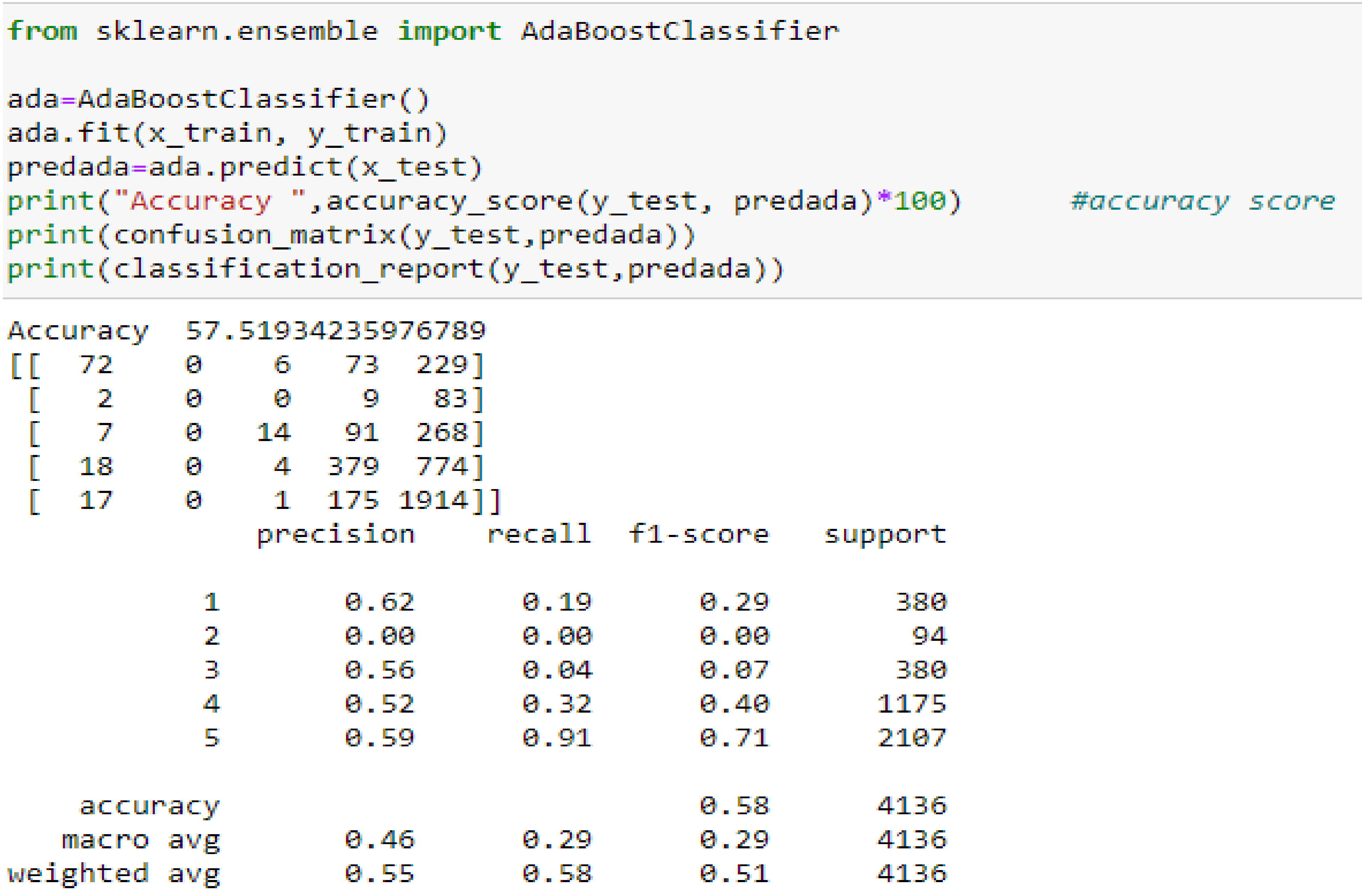


```
Accuracy  73.21083172147002
[[ 293    5   16   26   40]
 [  11   50    7   11   15]
 [  17    9  195  101   58]
 [  19   16   85  769  286]
 [  29    4   60  293 1721]]
              precision    recall  f1-score   support

           1       0.79      0.77      0.78       380
           2       0.60      0.53      0.56        94
           3       0.54      0.51      0.52       380
           4       0.64      0.65      0.65      1175
           5       0.81      0.82      0.81      2107

    accuracy                           0.73      4136
   macro avg       0.68      0.66      0.67      4136
weighted avg       0.73      0.73      0.73      4136
```

AdaBoost Classifier



```
from sklearn.ensemble import AdaBoostClassifier

ada=AdaBoostClassifier()
ada.fit(x_train, y_train)
predada=ada.predict(x_test)
print("Accuracy ",accuracy_score(y_test, predada)*100)      #accuracy score
print(confusion_matrix(y_test,predada))
print(classification_report(y_test,predada))
```

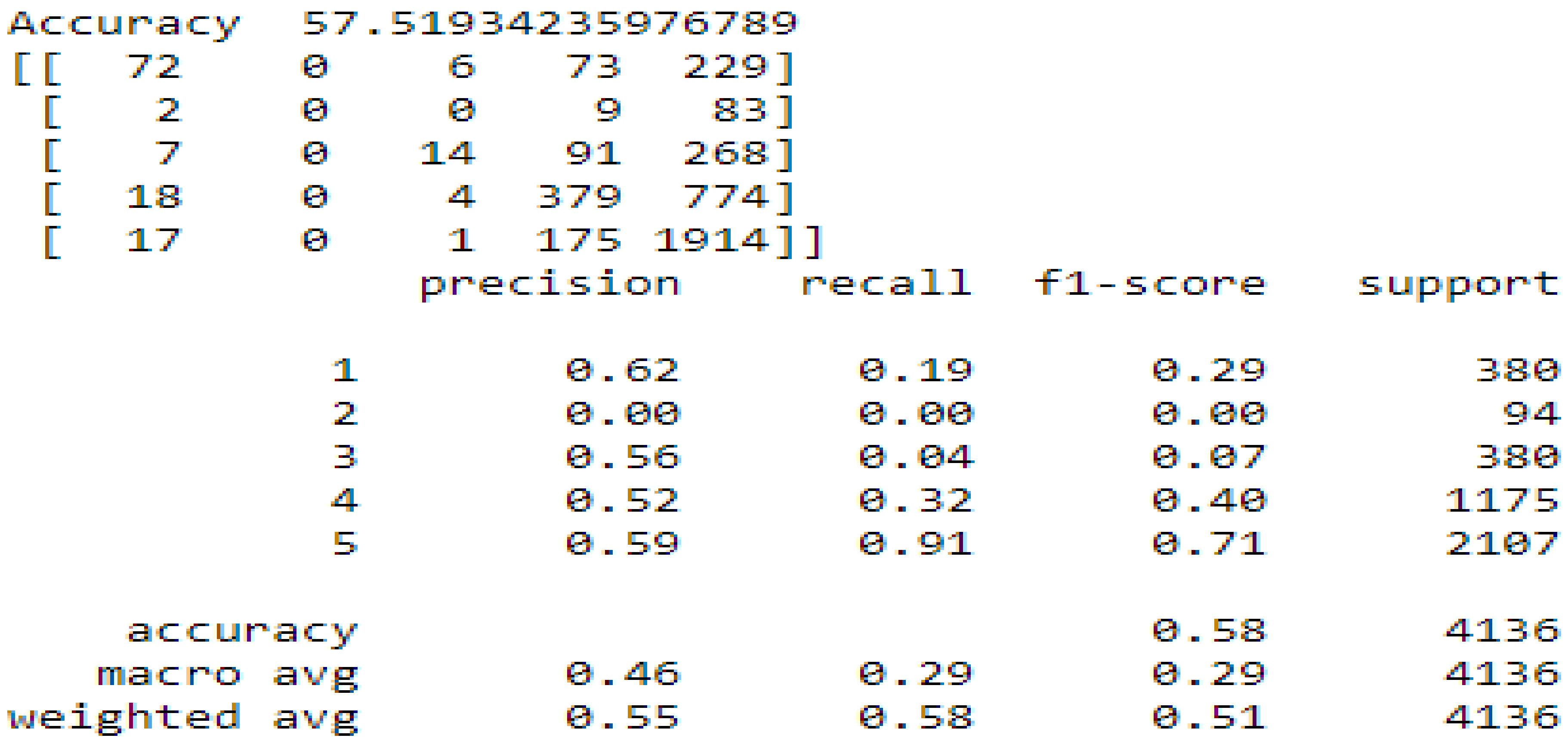
```
Accuracy  57.51934235976789
[[  72    0    6   73  229]
 [   2    0    0    9   83]
 [   7    0   14   91  268]
 [  18    0    4  379  774]
 [  17    0    1  175 1914]]
              precision    recall  f1-score   support

           1       0.62      0.19      0.29       380
           2       0.00      0.00      0.00        94
           3       0.56      0.04      0.07       380
           4       0.52      0.32      0.40      1175
           5       0.59      0.91      0.71      2107

    accuracy                           0.58      4136
   macro avg       0.46      0.29      0.29      4136
weighted avg       0.55      0.58      0.51      4136
```

- The **Accuracy** for target test and pred_test(data predicted on features_test) is **57.52%**

- The **Confusion Matrix and Classification Report** for target test and pred_test(data predicted on features_test) is –

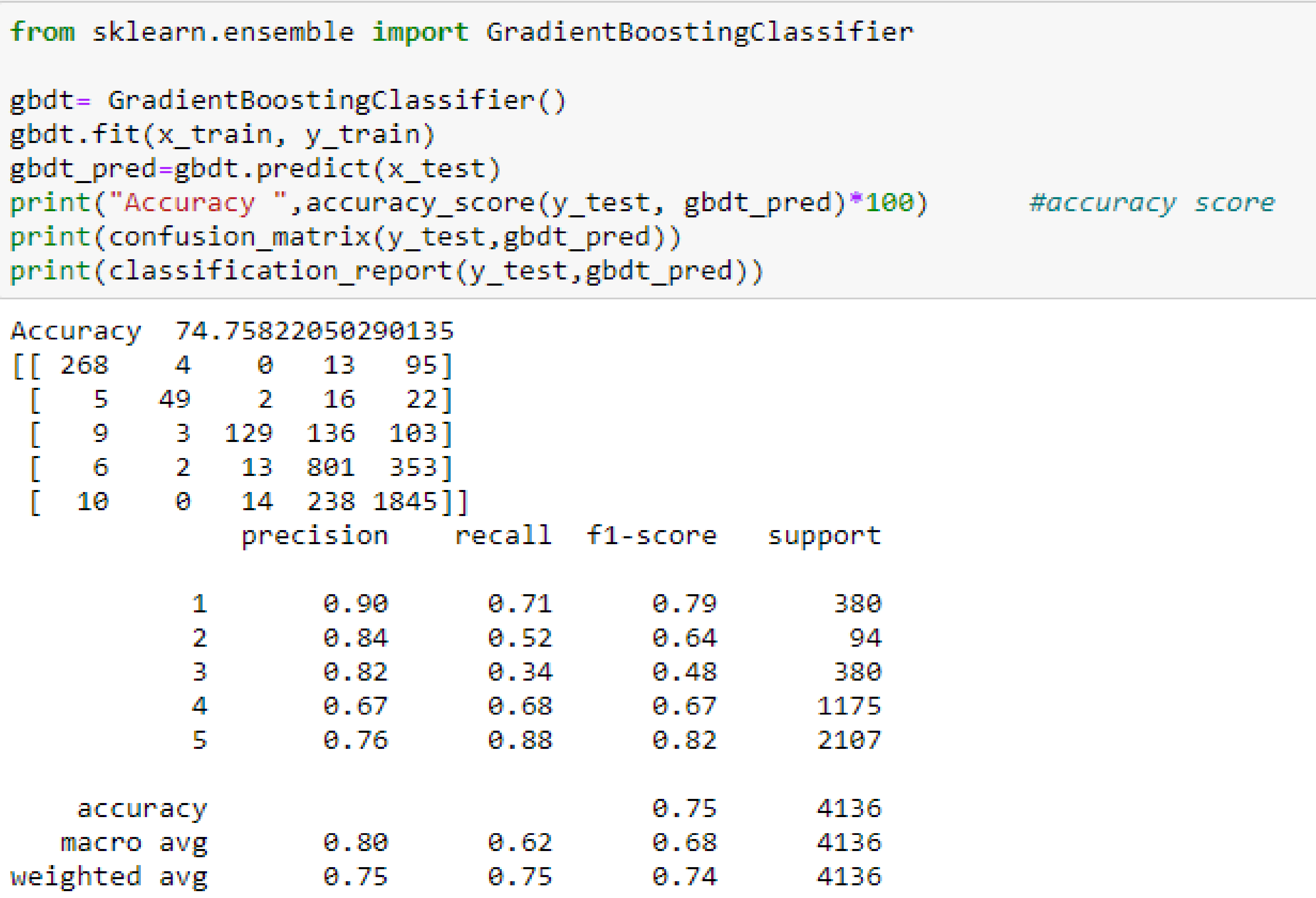


```
Accuracy  57.519342359767889
[[  72    0    6   73  229]
 [   2    0    0    9   83]
 [   7    0   14   91  268]
 [  18    0    4  379  774]
 [  17    0    1  175 1914]]
              precision    recall  f1-score   support

           1       0.62      0.19      0.29       380
           2       0.00      0.00      0.00        94
           3       0.56      0.04      0.07       380
           4       0.52      0.32      0.40      1175
           5       0.59      0.91      0.71      2107

    accuracy                           0.58      4136
   macro avg       0.46      0.29      0.29      4136
weighted avg       0.55      0.58      0.51      4136
```

Gradient Boosting Classifier



```
from sklearn.ensemble import GradientBoostingClassifier

gbdt= GradientBoostingClassifier()
gbdt.fit(x_train, y_train)
gbdt_pred=gbdt.predict(x_test)
print("Accuracy ",accuracy_score(y_test, gbdt_pred)*100)      #accuracy score
print(confusion_matrix(y_test,gbdt_pred))
print(classification_report(y_test,gbdt_pred))
```

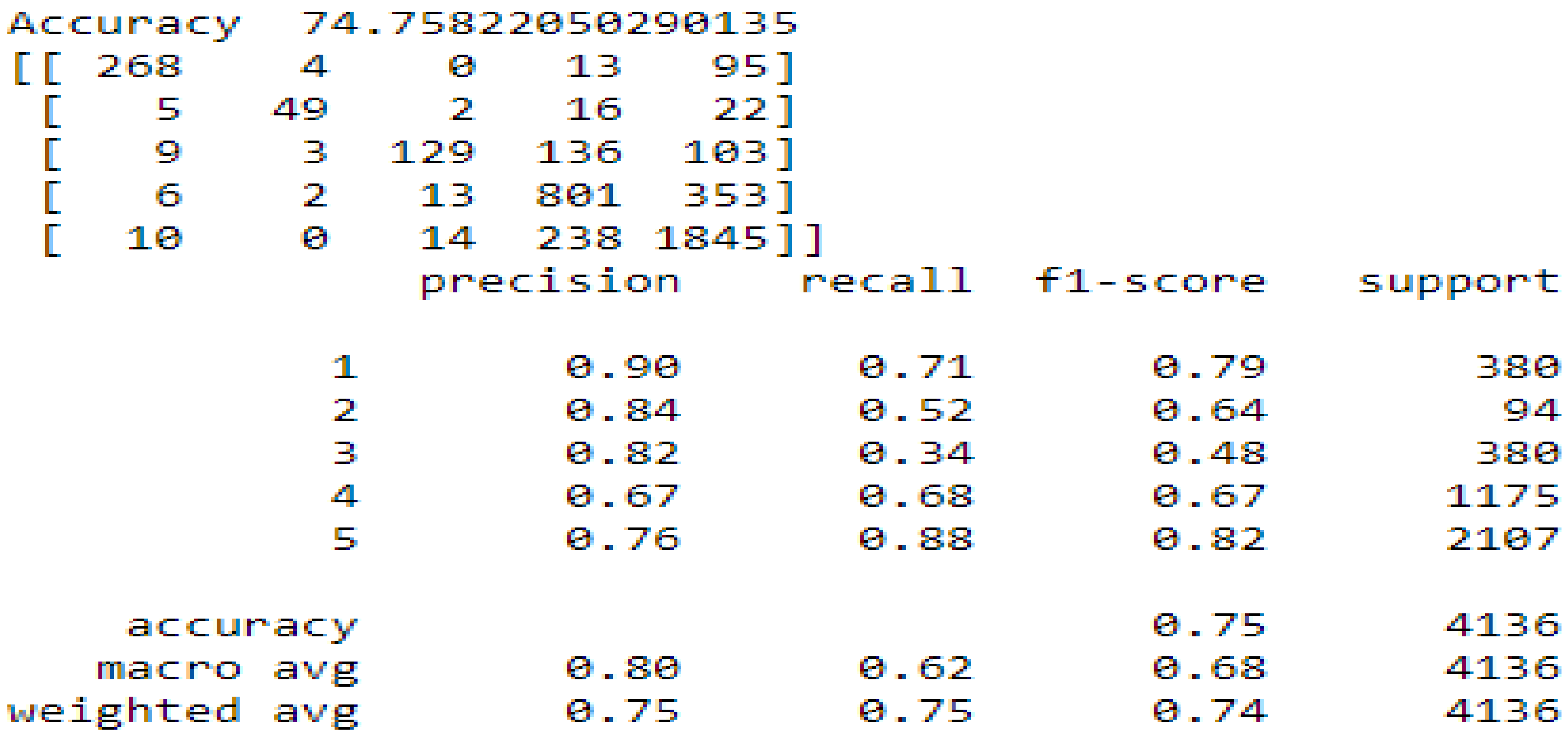
```
Accuracy  74.75822050290135
[[ 268    4    0   13   95]
 [    5   49    2   16   22]
 [    9    3  129  136  103]
 [    6    2   13  801  353]
 [   10    0   14  238 1845]]
              precision    recall  f1-score   support

           1       0.90      0.71      0.79       380
           2       0.84      0.52      0.64        94
           3       0.82      0.34      0.48       380
           4       0.67      0.68      0.67      1175
           5       0.76      0.88      0.82      2107

    accuracy                           0.75      4136
   macro avg       0.80      0.62      0.68      4136
weighted avg       0.75      0.75      0.74      4136
```

- The **Accuracy** for target test and pred_test(data predicted on features_test) is **74.76%**

- The **Confusion Matrix and Classification Report** for target test and pred_test(data predicted on features_test) is –

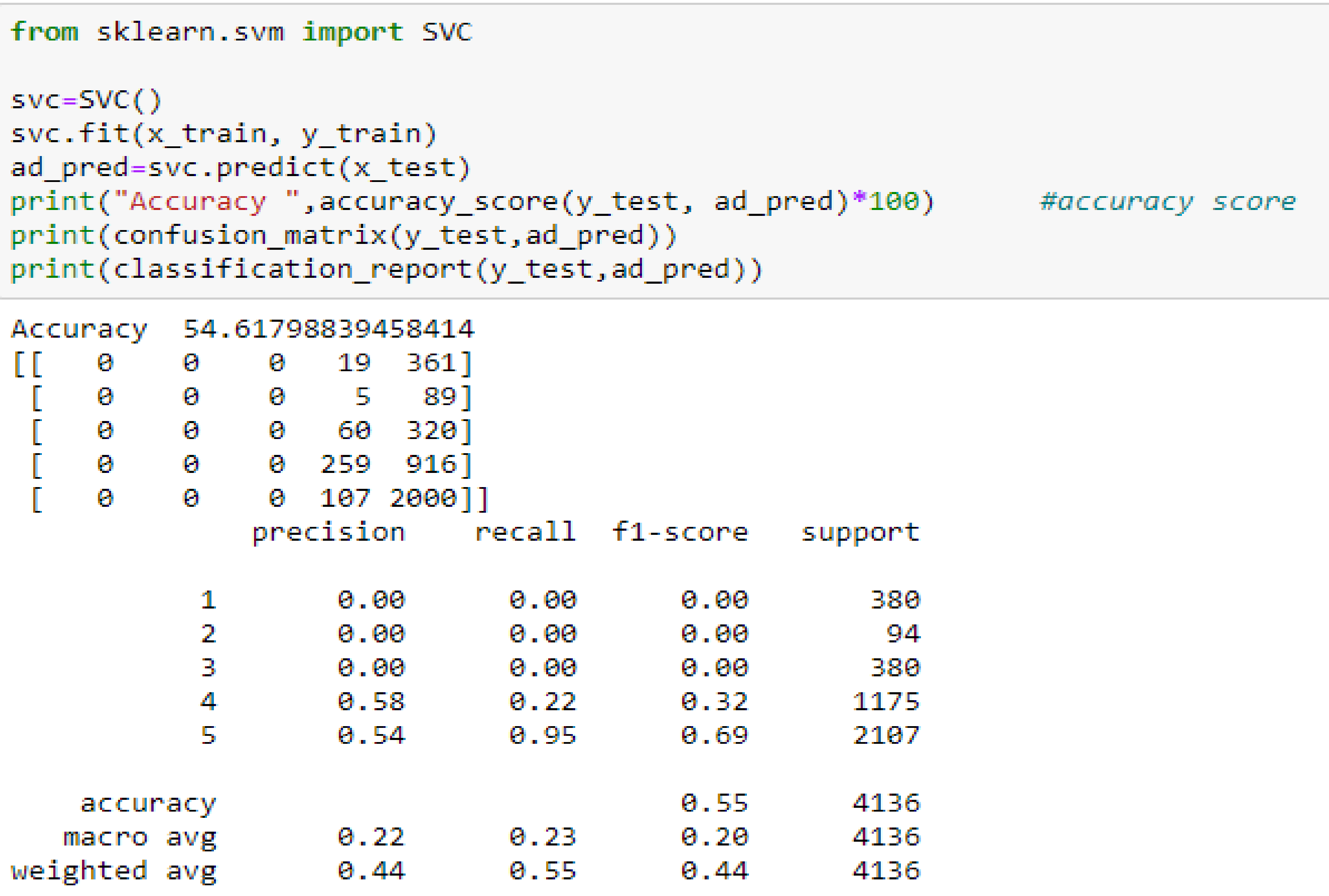


```
Accuracy  74.75822050290135
[[ 268    4    0   13   95]
 [   5   49    2   16   22]
 [   9    3  129  136  103]
 [   6    2   13  801  353]
 [  10    0   14  238 1845]]
              precision    recall  f1-score   support

           1       0.90      0.71      0.79       380
           2       0.84      0.52      0.64        94
           3       0.82      0.34      0.48       380
           4       0.67      0.68      0.67      1175
           5       0.76      0.88      0.82      2107

    accuracy                           0.75      4136
   macro avg       0.80      0.62      0.68      4136
weighted avg       0.75      0.75      0.74      4136
```

SVC



```python
from sklearn.svm import SVC

svc=SVC()
svc.fit(x_train, y_train)
ad_pred=svc.predict(x_test)
print("Accuracy ",accuracy_score(y_test, ad_pred)*100)       #accuracy score
print(confusion_matrix(y_test,ad_pred))
print(classification_report(y_test,ad_pred))
```

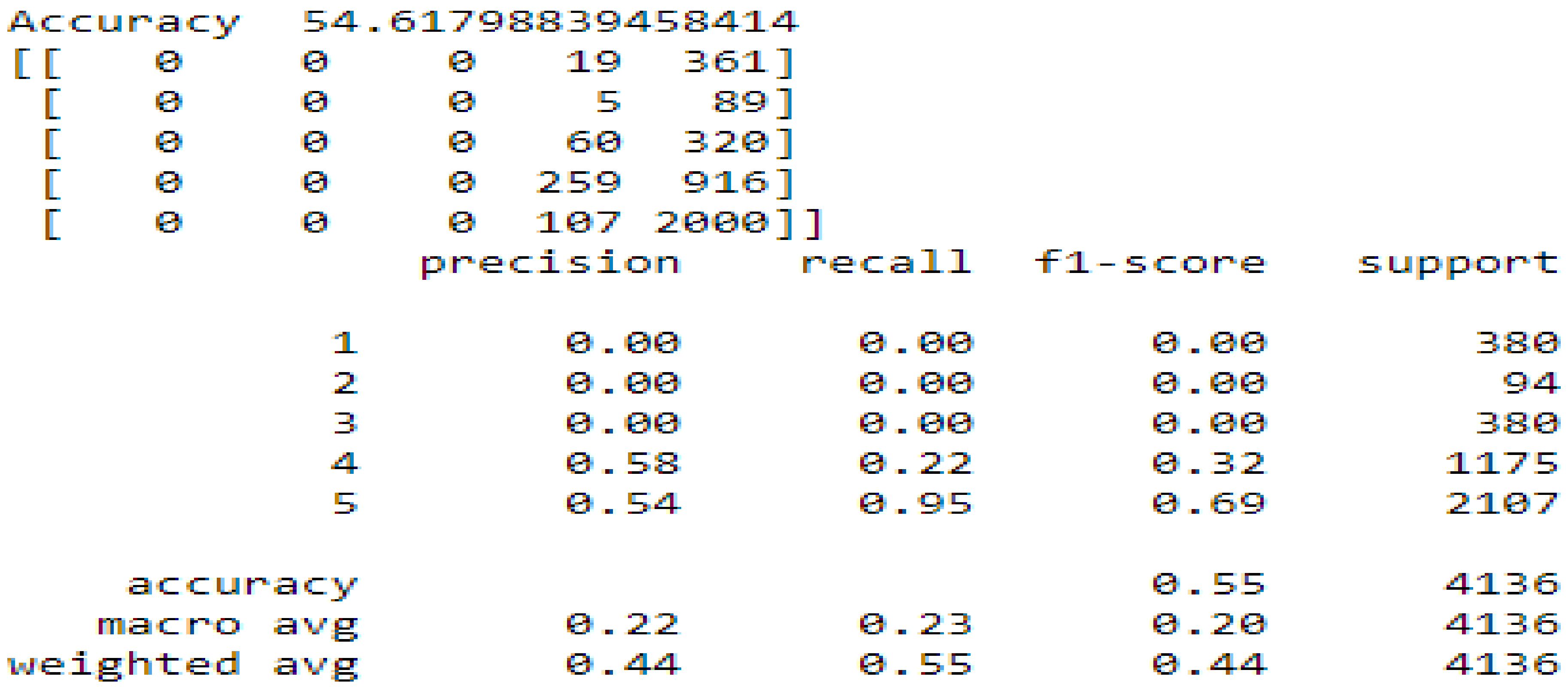
```
Accuracy  54.61798839458414
[[   0    0    0   19  361]
 [   0    0    0    5   89]
 [   0    0    0   60  320]
 [   0    0    0  259  916]
 [   0    0    0  107 2000]]
              precision    recall  f1-score   support

           1       0.00      0.00      0.00       380
           2       0.00      0.00      0.00        94
           3       0.00      0.00      0.00       380
           4       0.58      0.22      0.32      1175
           5       0.54      0.95      0.69      2107

    accuracy                           0.55      4136
   macro avg       0.22      0.23      0.20      4136
weighted avg       0.44      0.55      0.44      4136
```

- The **Accuracy** for target test and pred_test(data predicted on features_test) is **54.62%**

- The **Confusion Matrix and Classification Report** for target test and pred_test(data predicted on features_test) is –

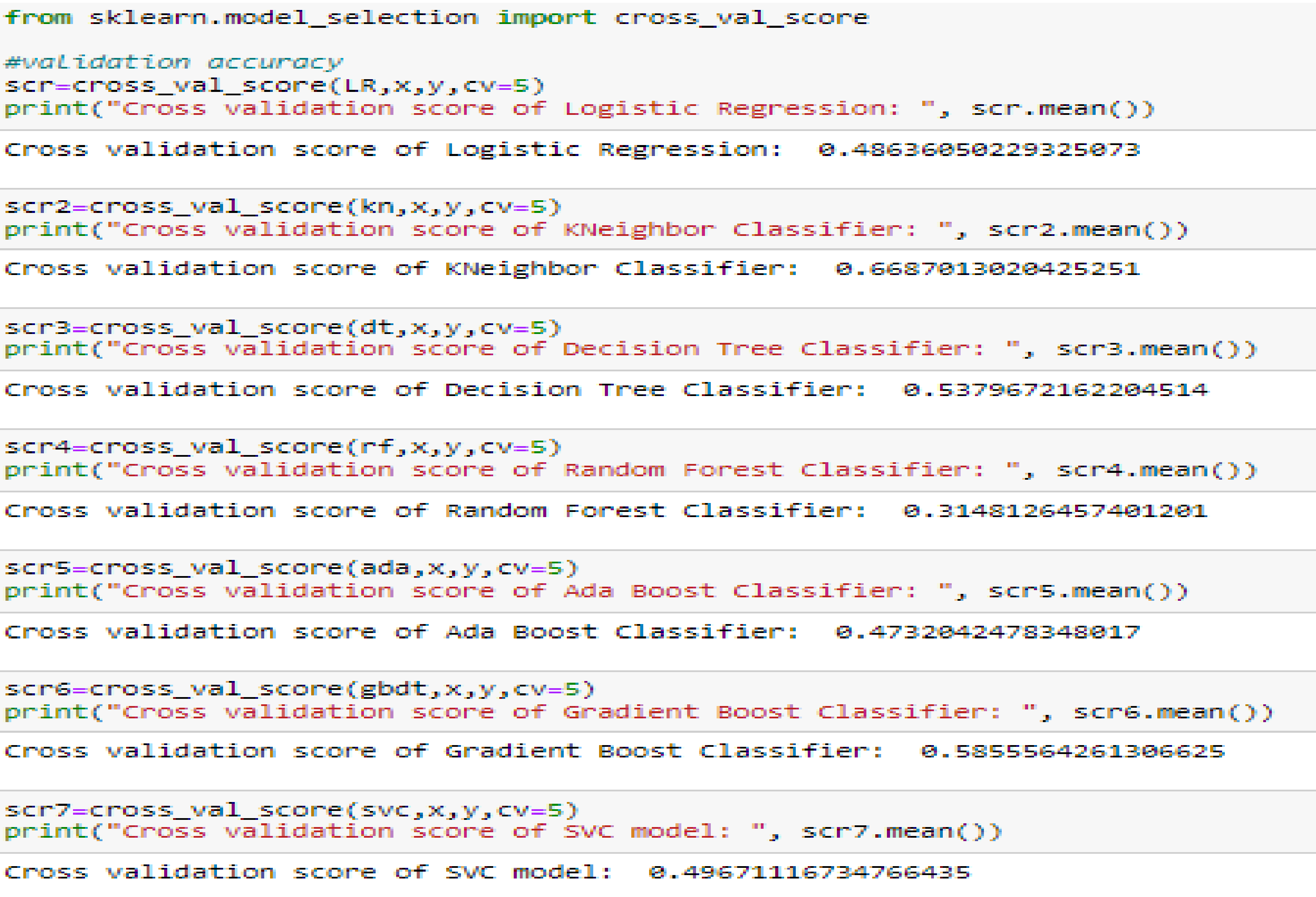


```
Accuracy  54.61798839458414
[[   0    0    0   19  361]
 [   0    0    0    5   89]
 [   0    0    0   60  320]
 [   0    0    0  259  916]
 [   0    0    0  107 2000]]
              precision    recall  f1-score   support

           1       0.00      0.00      0.00       380
           2       0.00      0.00      0.00        94
           3       0.00      0.00      0.00       380
           4       0.58      0.22      0.32      1175
           5       0.54      0.95      0.69      2107

    accuracy                           0.55      4136
   macro avg       0.22      0.23      0.20      4136
weighted avg       0.44      0.55      0.44      4136
```

Cross Validation



```
from sklearn.model_selection import cross_val_score

#validation accuracy
scr=cross_val_score(LR,x,y,cv=5)
print("Cross validation score of Logistic Regression: ", scr.mean())
```

```
Cross validation score of Logistic Regression:  0.4863605022932507З
```

```
scr2=cross_val_score(kn,x,y,cv=5)
print("Cross validation score of KNeighbor Classifier: ", scr2.mean())
```

```
Cross validation score of KNeighbor Classifier:  0.6687013020425251
```

```
scr3=cross_val_score(dt,x,y,cv=5)
print("Cross validation score of Decision Tree Classifier: ", scr3.mean())
```

```
Cross validation score of Decision Tree Classifier:  0.537967216220451Ӏ4
```

```
scr4=cross_val_score(rf,x,y,cv=5)
print("Cross validation score of Random Forest Classifier: ", scr4.mean())
```

```
Cross validation score of Random Forest Classifier:  0.314812645740120Ӏ
```

```
scr5=cross_val_score(ada,x,y,cv=5)
print("Cross validation score of Ada Boost Classifier: ", scr5.mean())
```

```
Cross validation score of Ada Boost Classifier:  0.47320424783480Ӏ7
```

```
scr6=cross_val_score(gbdt,x,y,cv=5)
print("Cross validation score of Gradient Boost Classifier: ", scr6.mean())
```

```
Cross validation score of Gradient Boost Classifier:  0.5855564261306625
```

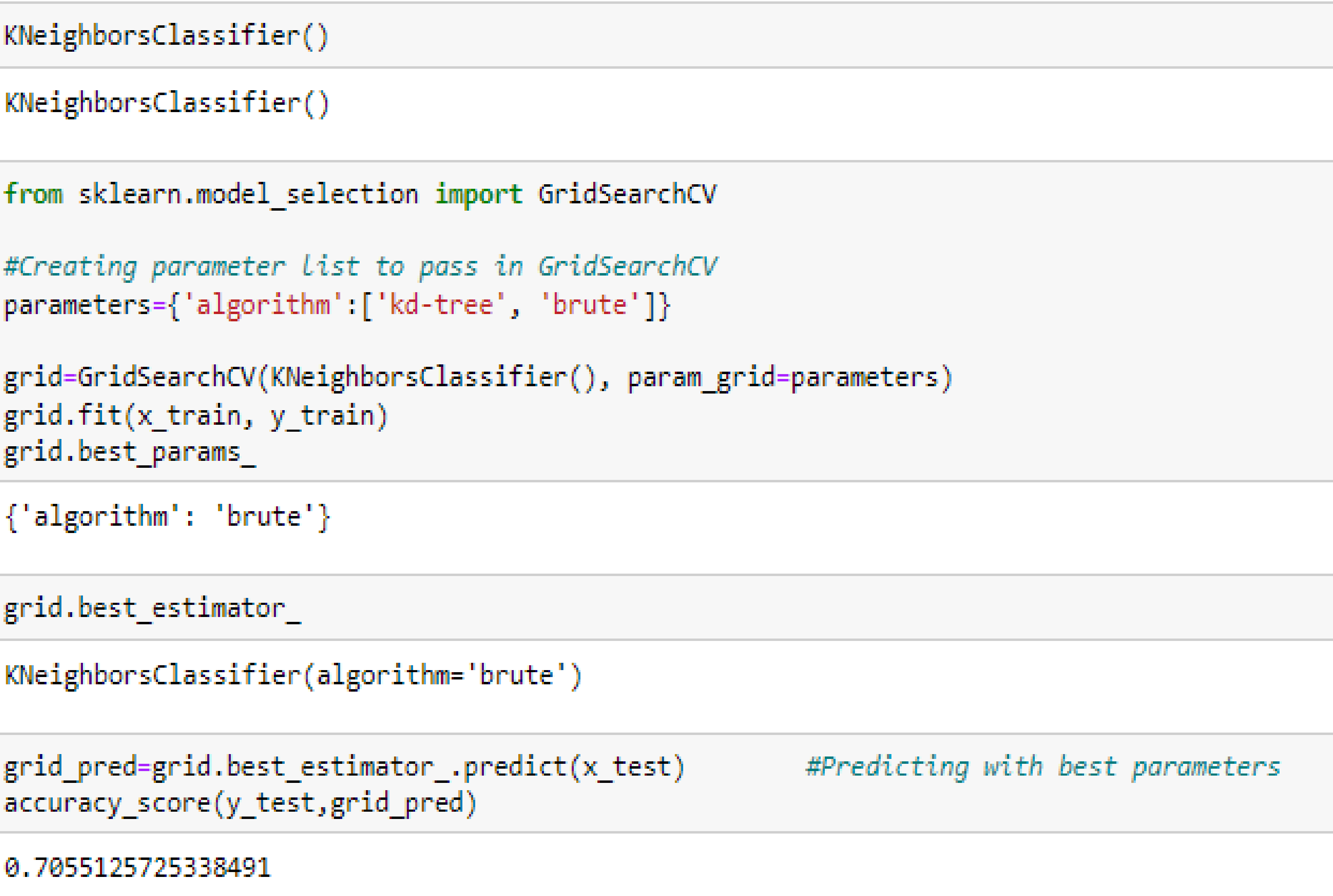
```
scr7=cross_val_score(svc,x,y,cv=5)
print("Cross validation score of SVC model: ", scr7.mean())
```

```
Cross validation score of SVC model:  0.4967111673476б435
```

Model	**Cross Validation Score**
Logistic Regression	0.4863
KNeighbor Classifier	0.6687
Decision Tree Classifier	0.5379
Random Forest Classifier	0.3148
Ada Boost Classifier	0.4732
Gradient Boost Classifier	0.5855
SVC	0.4967

- KNeighbor Classifier is performing better, hence it is carried forward

Hyperparameter tuned KNeighborsClassifier Model



```
KNeighborsClassifier()
```

```
KNeighborsClassifier()
```

```
from sklearn.model_selection import GridSearchCV

#Creating parameter list to pass in GridSearchCV
parameters={'algorithm':['kd-tree', 'brute']}

grid=GridSearchCV(KNeighborsClassifier(), param_grid=parameters)
grid.fit(x_train, y_train)
grid.best_params_
```

```
{'algorithm': 'brute'}
```

```
grid.best_estimator_
```

```
KNeighborsClassifier(algorithm='brute')
```

```
grid_pred=grid.best_estimator_.predict(x_test)        #Predicting with best parameters
accuracy_score(y_test,grid_pred)
```

```
0.7055125725338491
```

- KNeighborsClassifier Model is hyperparameter tuned using GridSearchCV

- The best parameters for algorithm is found as follows-
  - **algorithm: brute**

- Applying the above found best parameters on KNeighborsClassifier Model, the following was obtained-

  - The **Accuracy** for target test and pred_test(data predicted on features_test) is **70.55%**

Final Accuracy is 70.55% which depicts that our model is working well

CONCLUSION

- It is essential to scrape diverse data to allow the model to be applicable for versatile inputs
- The EDA analysis of the data is essential as it helps to understand the relationship between the target and features as well as omit out unwanted columns, thereby taking care of overfitting scenario
- The models should be used properly, as their regularization /hyperparamter tuning is highly advisable for the best outcome.
- The project imparted key knowledge about the e-commerce sites, the review and rating pattern and how essential it is to know the rating of any product before investing into it. Further the project will also help to identify the best rated brands under each product category.
- The limitation of the solution is that it predicts the rating of a review. However, reviews especially 2 stars and 3 stars are very rarely rewarded to products, as people either completely like or dislike the product. Hence the review for 2 star and 3 star are less.