CLASSIFICATION PROJECT

Mobile Price Range Prediction

Team Data Defenders

SHUBHAM SARTAPE
LOKESH TOKAS
SARASWAT MUKHERJEE
CHARAN

Content

1 Defining Problem Statement

2 Data Summary

3 EDA and Data Cleaning

4 Feature Engineering

5 Model Selection

5 Conclusion

Problem Statement

- To predict the price range of Mobile Phones based on the available features such as RAM, camera, battery, internal memory, cores, clock speed, etc...
- The Target Variables are classified into 4 types as below.

  0 - Low Cost Phones

  1 - Medium Cost Phones

  2 - High Cost Phones

  3 - Very High Cost Phones
- This will help mobile phone market companies to understand sales data of mobile phones and factors which drive the prices.
- The objective is to find out some relation between features of a mobile phone(eg:- RAM, Internal Memory, etc) and its selling price.

Data Summary

Independent Variables

Battery Power - Total energy a battery can store in one time measured in mAh

Blue - Has Bluetooth or not

Clock_speed - speed at which microprocessor executes instructions

Dual_sim - Has dual sim support or not

Fc - Front Camera mega pixels

Four_g - Has 4G or not

Int_memory - Internal Memory in Gigabytes

M_dep - Mobile Depth in cm

Independent Variables

Mobile_wt - Weight of mobile phone

N_cores - Number of cores of processor

Pc - Primary Camera mega pixels

Px_height - Pixel Resolution Height

Px_width - Pixel Resolution Width

Ram - Random Access Memory in Mega Bytes

Sc_h - Screen Height of mobile in cm

Sc_w - Screen Width of mobile in cm

Talk_time - longest time that a single battery charge will last over a call

Three_g - Has 3G or not

Data Summary

Independent Variables

Touch_screen - Has touch screen or not

Wifi - Has wifi or not

Dependent Variables

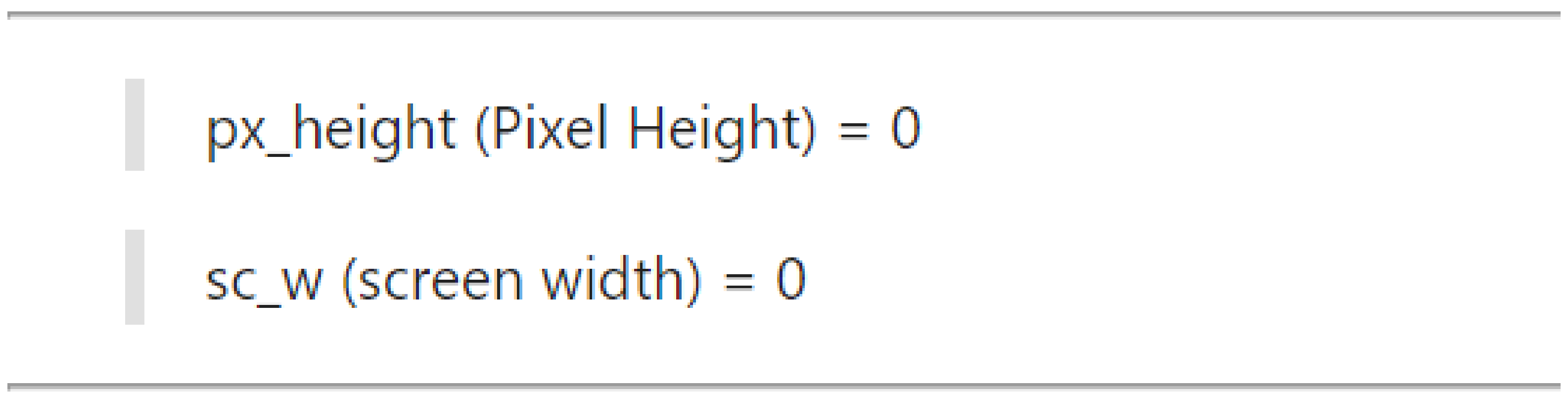
Price_range - This is the target variable with value of

0 (low cost), 1 (medium cost), 2 (high cost) and 3 (very high cost).

EDA - Data Cleaning

Detecting Data Anomaly

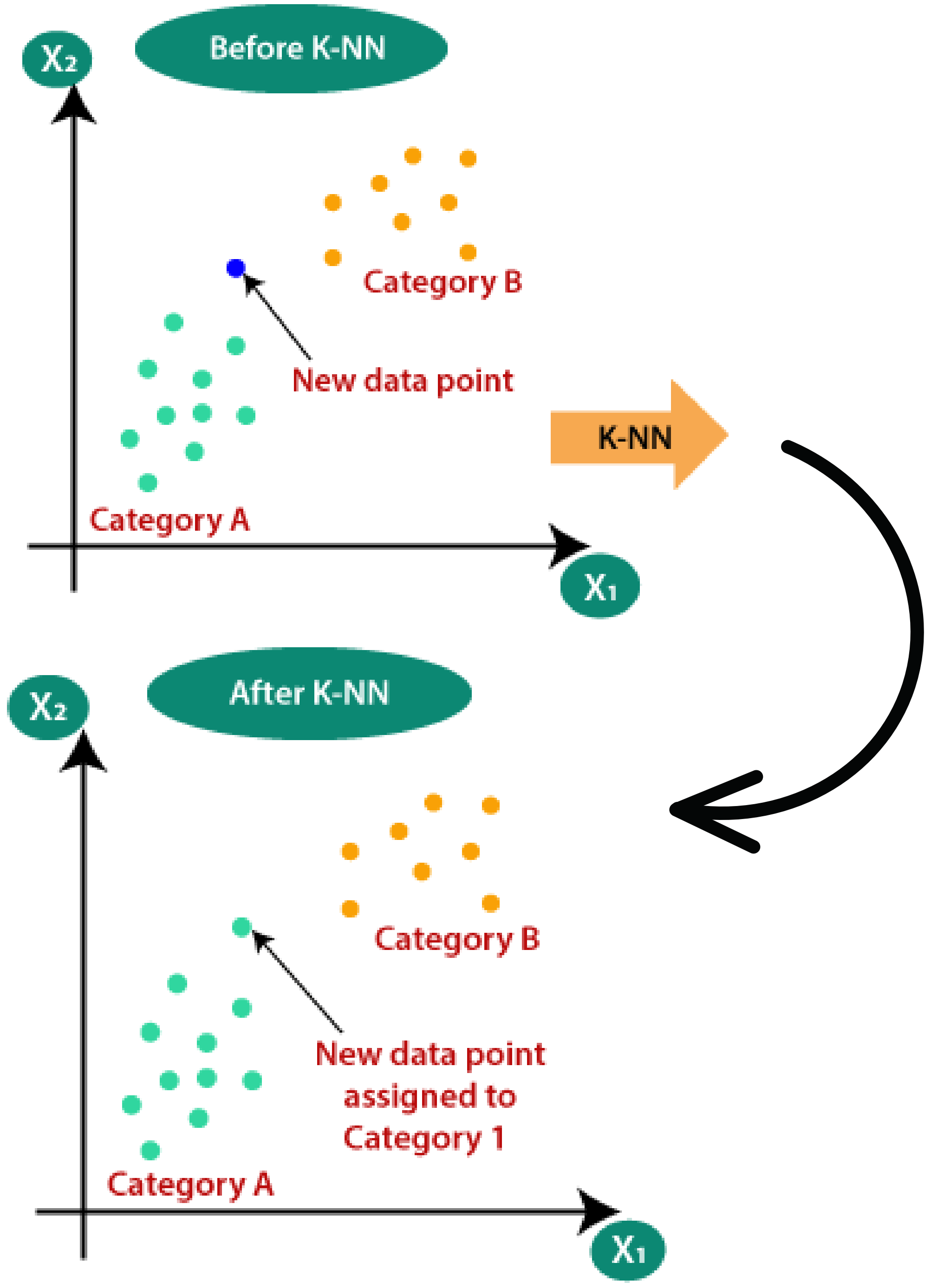
Following Anomalies were found



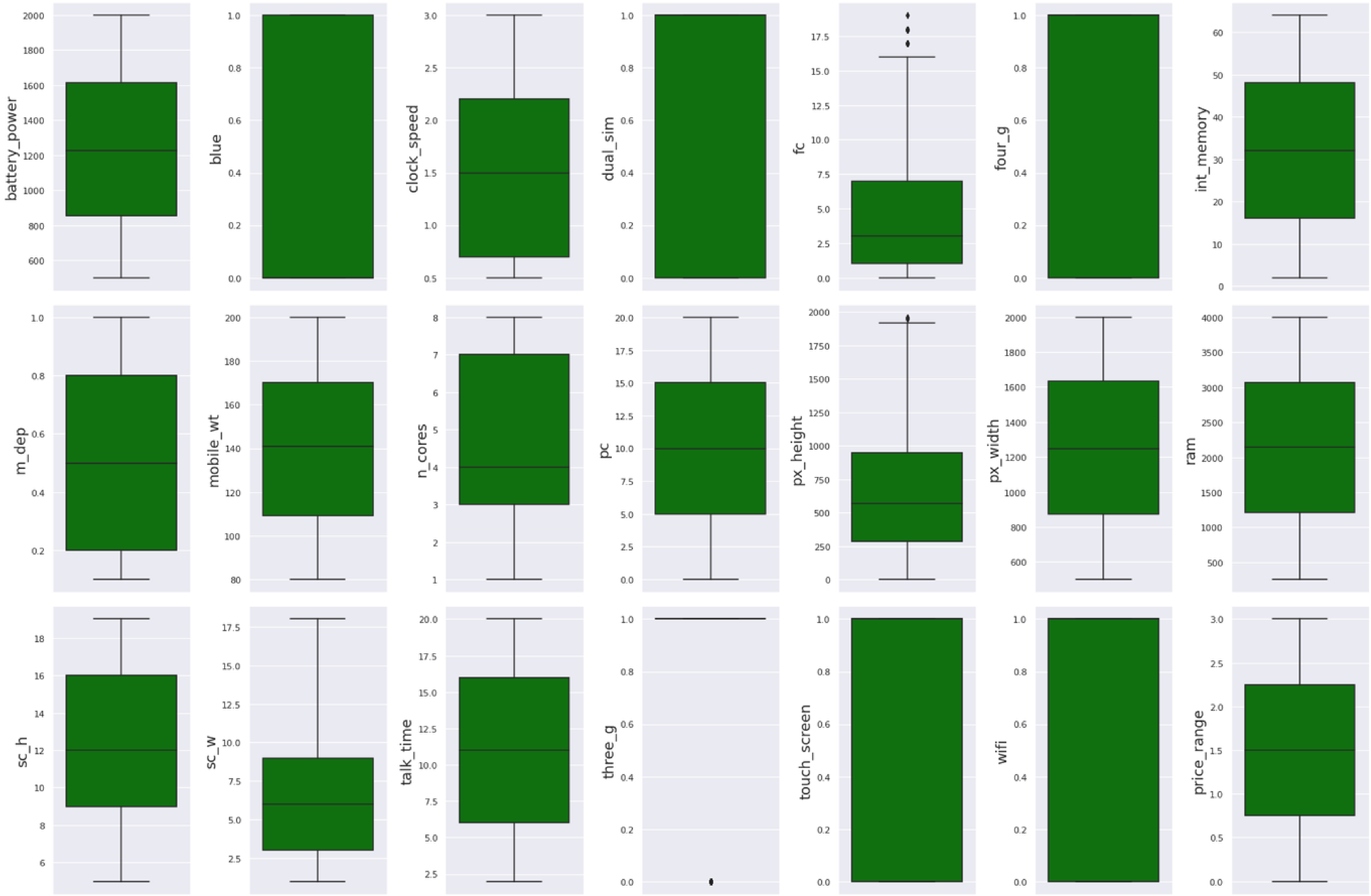
> px_height (Pixel Height) = 0

> sc_w (screen width) = 0

Replaced the anomalous values using KNN Imputer by assigning nearest possible value and not Mean/Avg Value.



EDA Outlier Detection

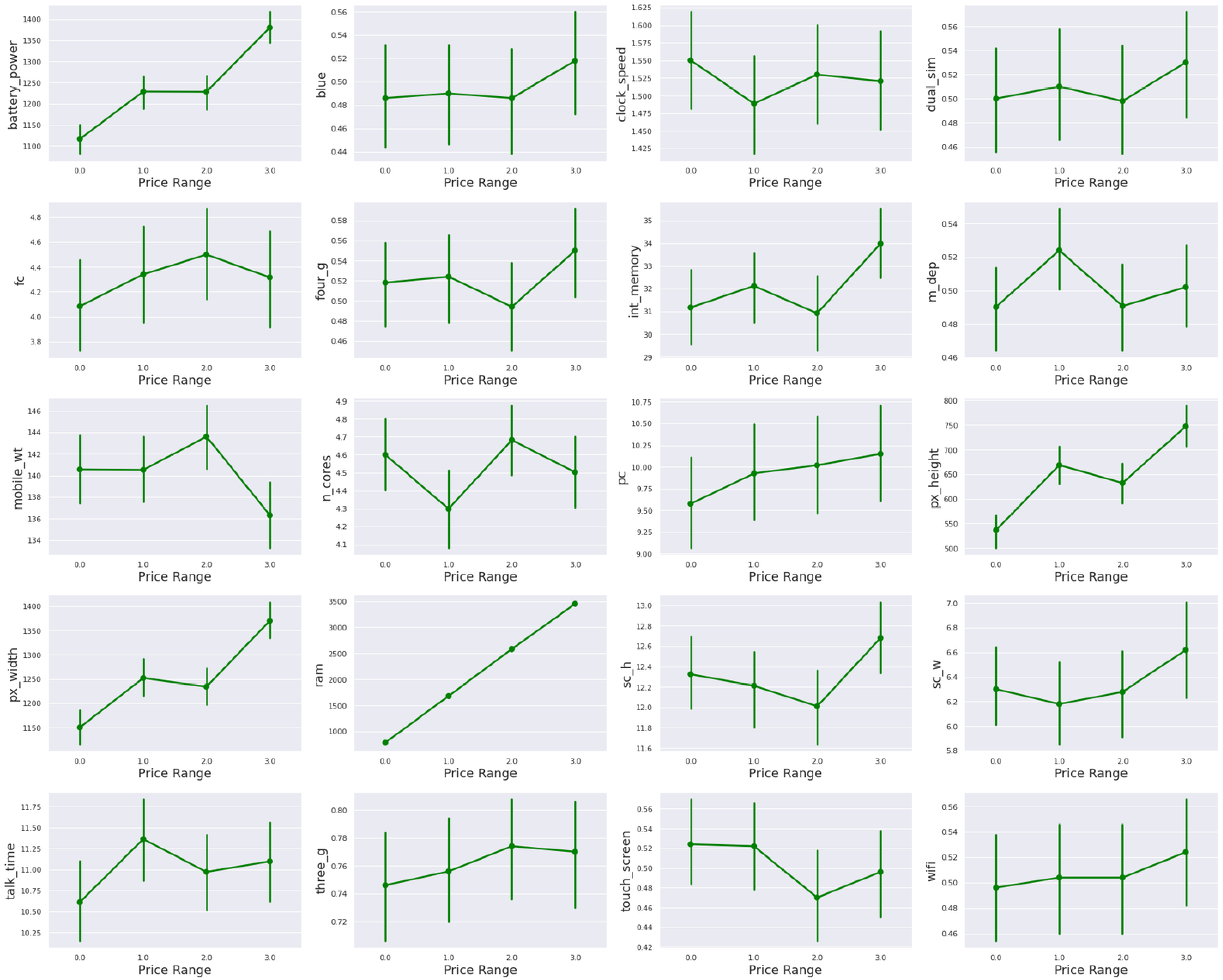


No Extreme Outliers detected.

- In 'px_height', 1 possible outlier was highlighted but after examination it was observed to be within reason.
- In 'fc' few observations were out of bound but normal to have high mp for experimental purpose or emerging technology.

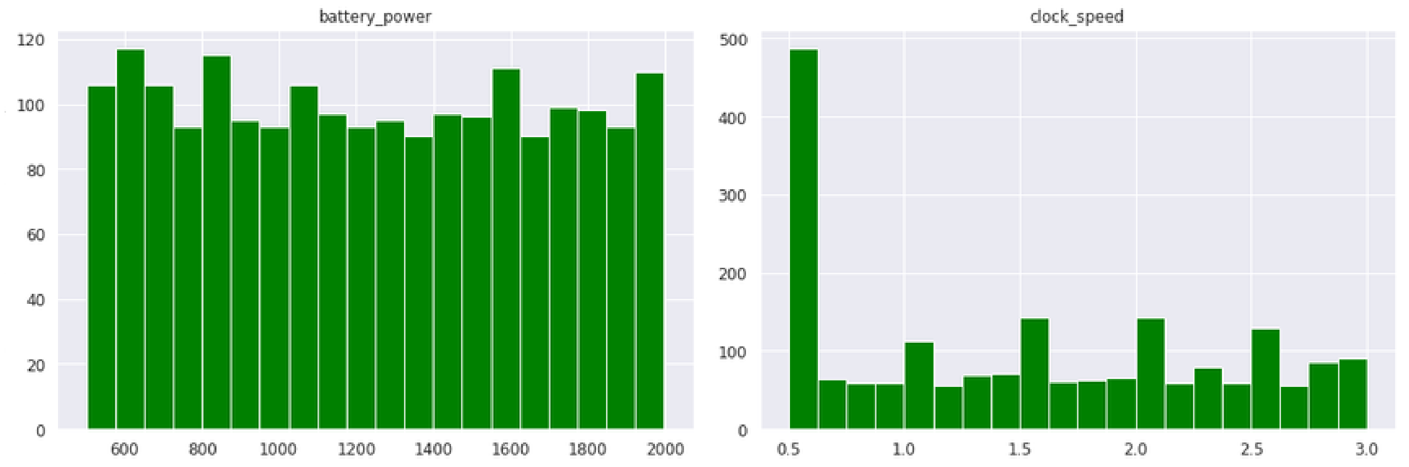
Feature Engineering

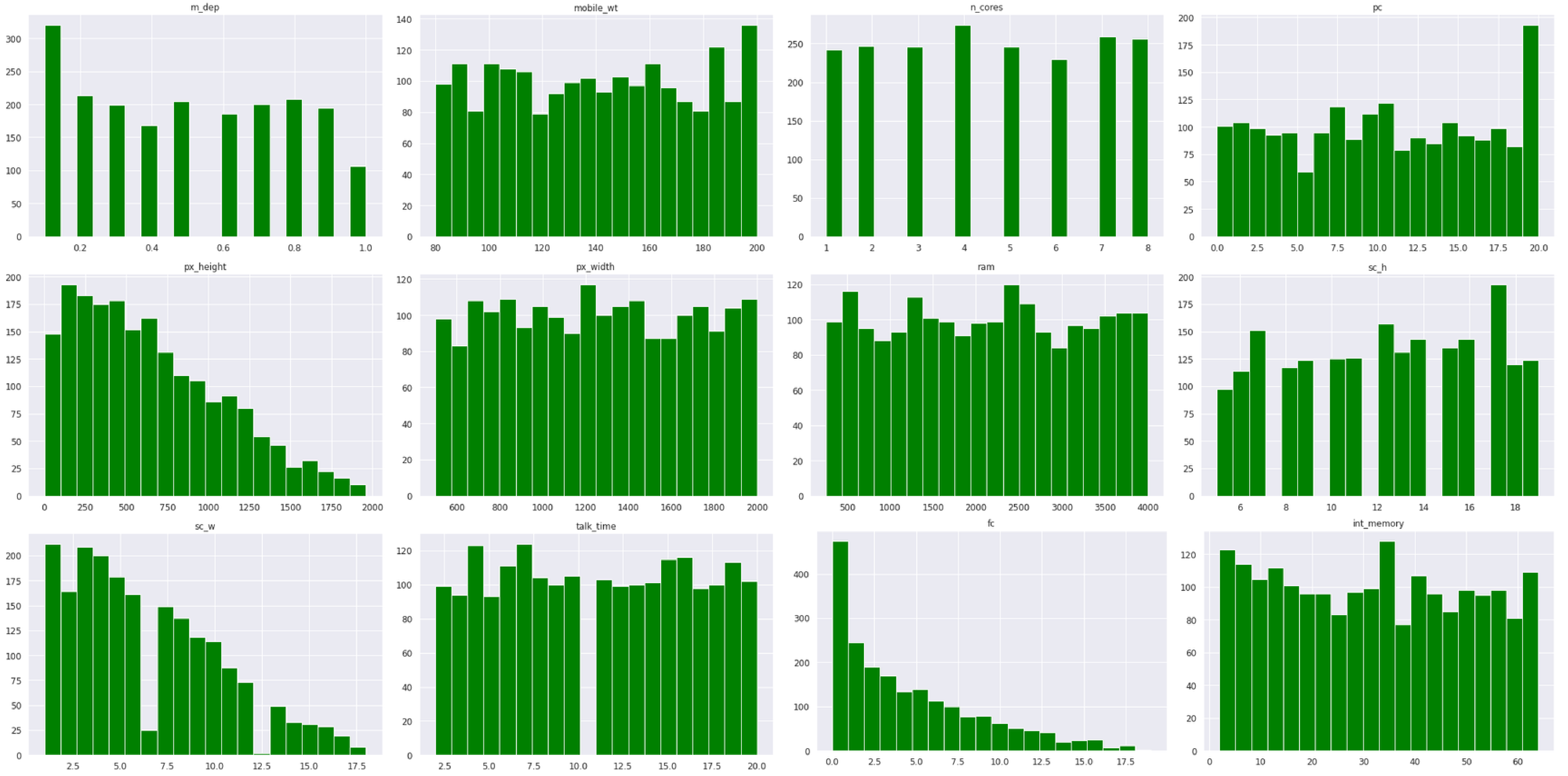
Point Plot of all features on 'price_range'



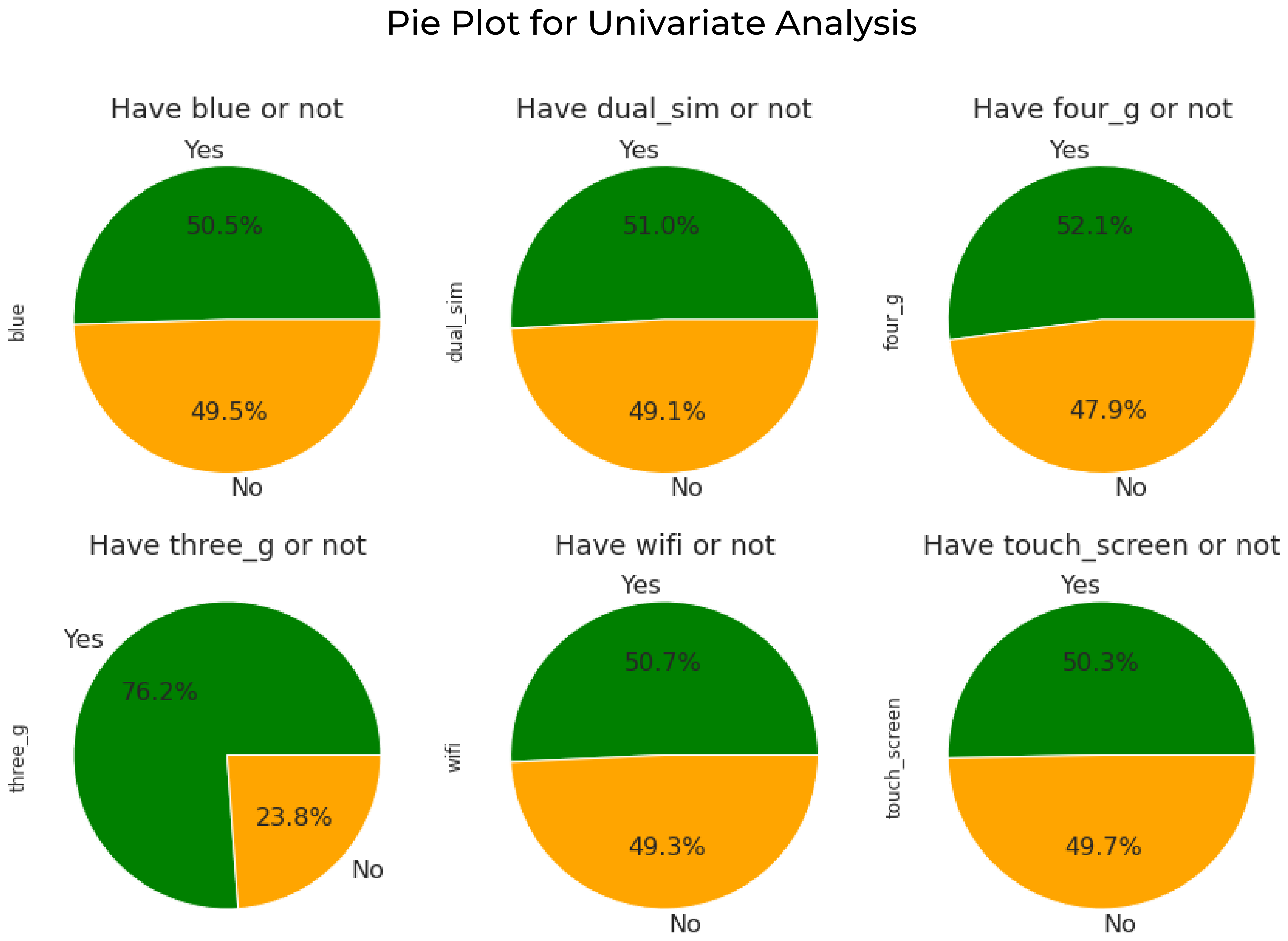
Feature Engineering

Bar Plot of all coloumns except univariate columns & 'price_range'





Feature Engineering

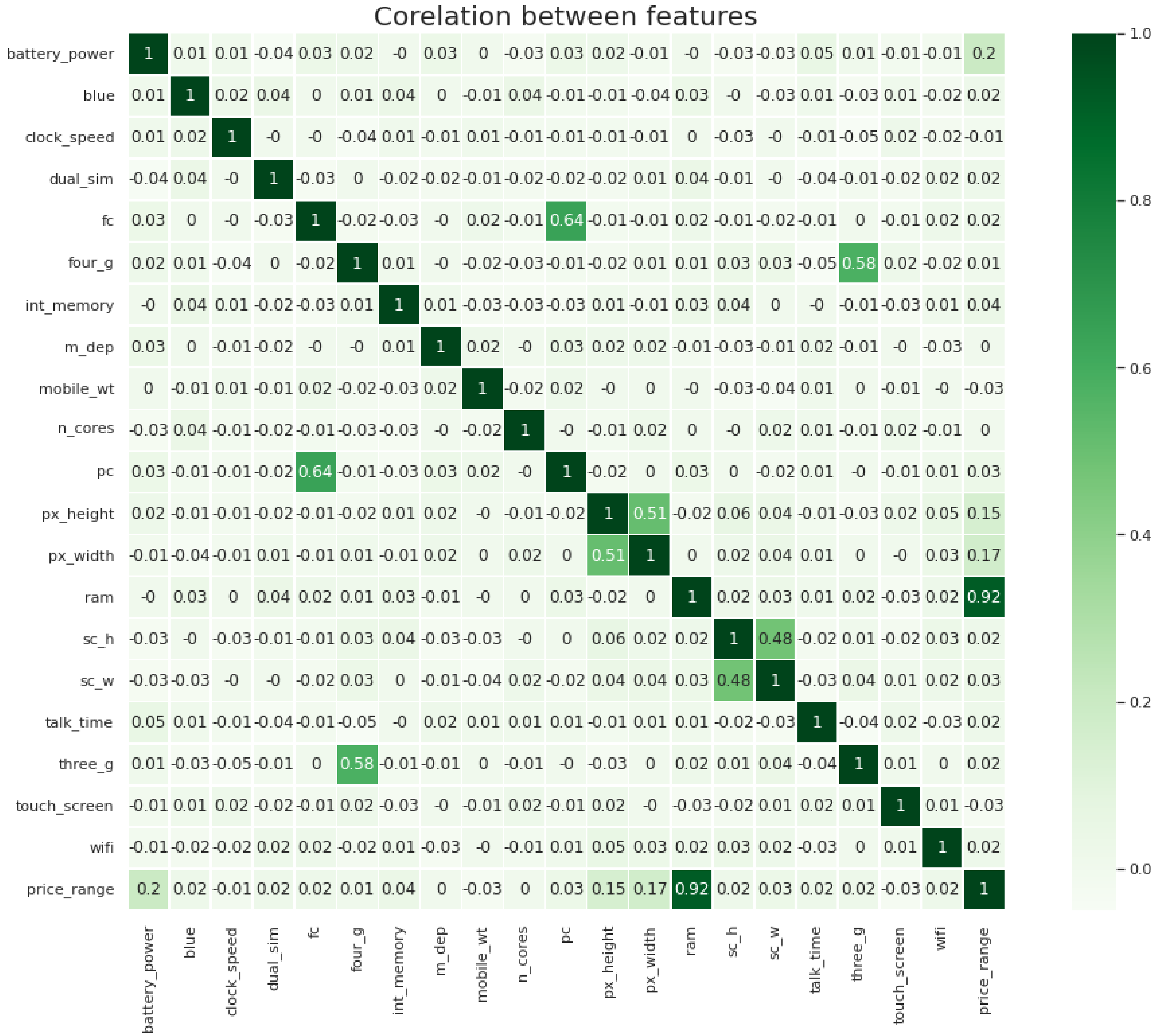


Feature Engineering

Heatmap to visualise - Corelation

- 'ram' & "price range"(target variable) is highly correlated.

  More ram = Higher Price
- 'three_g' & 'four_g' is moderately correlated.
- 'pc' (primary camera) & 'fc' (front camera) is moderately correlated.
- 'px_height' & 'px_width' is moderately correlated.
- 'sc_h' & 'sc_w' (screen height & screen width) is moderately correlated.



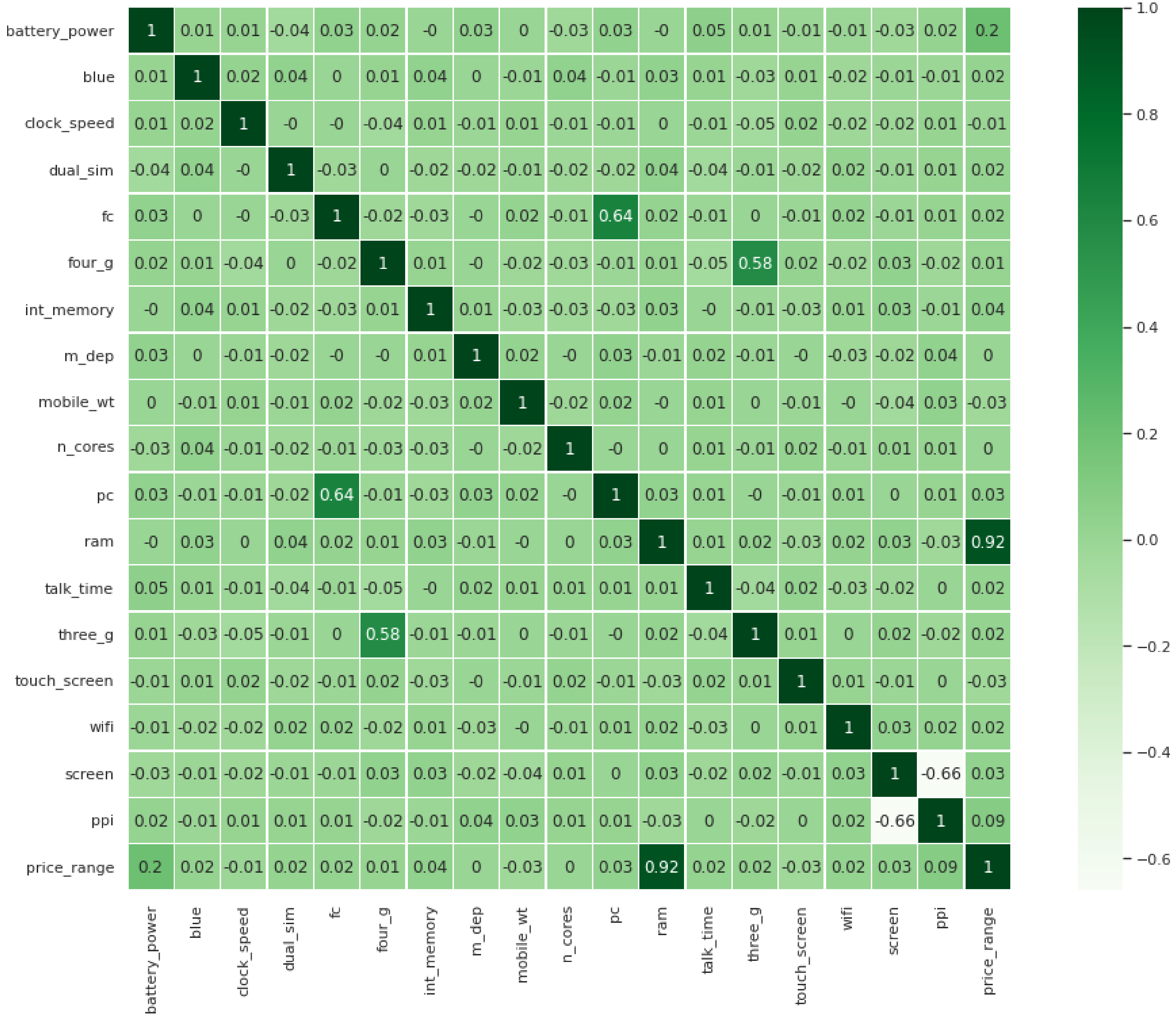
	battery_power	blue	clock_speed	dual_sim	fc	four_g	int_memory	m_dep	mobile_wt	n_cores	pc	px_height	px_width	ram	sc_h	sc_w	talk_time	three_g	touch_screen	wifi	price_range
battery_power	1	0.01	0.01	-0.04	0.03	0.02	-0	0.03	0	-0.03	0.03	0.02	-0.01	-0	-0.03	-0.03	0.05	0.01	-0.01	-0.01	0.2
blue	0.01	1	0.02	0.04	0	0.01	0.04	0	-0.01	0.04	-0.01	-0.01	-0.04	0.03	-0	-0.03	0.01	-0.03	0.01	-0.02	0.02
clock_speed	0.01	0.02	1	-0	-0	-0.04	0.01	-0.01	0.01	-0.01	-0.01	-0.01	-0.01	0	-0.03	-0	-0.01	-0.05	0.02	-0.02	-0.01
dual_sim	-0.04	0.04	-0	1	-0.03	0	-0.02	-0.02	-0.01	-0.02	-0.02	-0.02	0.01	0.04	-0.01	-0	-0.04	-0.01	-0.02	0.02	0.02
fc	0.03	0	-0	-0.03	1	-0.02	-0.03	-0	0.02	-0.01	0.64	-0.01	-0.01	0.02	-0.01	-0.02	-0.01	0	-0.01	0.02	0.02
four_g	0.02	0.01	-0.04	0	-0.02	1	0.01	-0	-0.02	-0.03	-0.01	-0.02	0.01	0.01	0.03	0.03	-0.05	0.58	0.02	-0.02	0.01
int_memory	-0	0.04	0.01	-0.02	-0.03	0.01	1	0.01	-0.03	-0.03	-0.03	0.01	-0.01	0.03	0.04	0	-0	-0.01	-0.03	0.01	0.04
m_dep	0.03	0	-0.01	-0.02	-0	-0	0.01	1	0.02	-0	0.03	0.02	0.02	-0.01	-0.03	-0.01	0.02	-0.01	-0	-0.03	0
mobile_wt	0	-0.01	0.01	-0.01	0.02	-0.02	-0.03	0.02	1	-0.02	0.02	-0	0	-0	-0.03	-0.04	0.01	0	-0.01	-0	-0.03
n_cores	-0.03	0.04	-0.01	-0.02	-0.01	-0.03	-0.03	-0	-0.02	1	-0	-0.01	0.02	0	-0	0.02	0.01	-0.01	0.02	-0.01	0
pc	0.03	-0.01	-0.01	-0.02	0.64	-0.01	-0.03	0.03	0.02	-0	1	-0.02	0	0.03	0	-0.02	0.01	-0	-0.01	0.01	0.03
px_height	0.02	-0.01	-0.01	-0.02	-0.01	-0.02	0.01	0.02	-0	-0.01	-0.02	1	0.51	-0.02	0.06	0.04	-0.01	-0.03	0.02	0.05	0.15
px_width	-0.01	-0.04	-0.01	0.01	-0.01	0.01	-0.01	0.02	0	0.02	0	0.51	1	0	0.02	0.04	0.01	0	-0	0.03	0.17
ram	-0	0.03	0	0.04	0.02	0.01	0.03	-0.01	-0	0	0.03	-0.02	0	1	0.02	0.03	0.01	0.02	-0.03	0.02	0.92
sc_h	-0.03	-0	-0.03	-0.01	-0.01	0.03	0.04	-0.03	-0.03	-0	0	0.06	0.02	0.02	1	0.48	-0.02	0.01	-0.02	0.03	0.02
sc_w	-0.03	-0.03	-0	-0	-0.02	0.03	0	-0.01	-0.04	0.02	-0.02	0.04	0.04	0.03	0.48	1	-0.03	0.04	0.01	0.02	0.03
talk_time	0.05	0.01	-0.01	-0.04	-0.01	-0.05	-0	0.02	0.01	0.01	0.01	-0.01	0.01	0.01	-0.02	-0.03	1	-0.04	0.02	-0.03	0.02
three_g	0.01	-0.03	-0.05	-0.01	0	0.58	-0.01	-0.01	0	-0.01	-0	-0.03	0	0.02	0.01	0.04	-0.04	1	0.01	0	0.02
touch_screen	-0.01	0.01	0.02	-0.02	-0.01	0.02	-0.03	-0	-0.01	0.02	-0.01	0.02	-0	-0.03	-0.02	0.01	0.02	0.01	1	0.01	-0.03
wifi	-0.01	-0.02	-0.02	0.02	0.02	-0.02	0.01	-0.03	-0	-0.01	0.01	0.05	0.03	0.02	0.03	0.02	-0.03	0	0.01	1	0.02
price_range	0.2	0.02	-0.01	0.02	0.02	0.01	0.04	0	-0.03	0	0.03	0.15	0.17	0.92	0.02	0.03	0.02	0.02	-0.03	0.02	1

Feature Engineering

- Converting 'sc_w' and 'sc_h' to a single variable named 'screen'
- Converting 'px_width' and 'px_height' to 'ppi' (pixel per inch).

Observation

- Now we also have negative corelation between 'screen' & 'ppi'



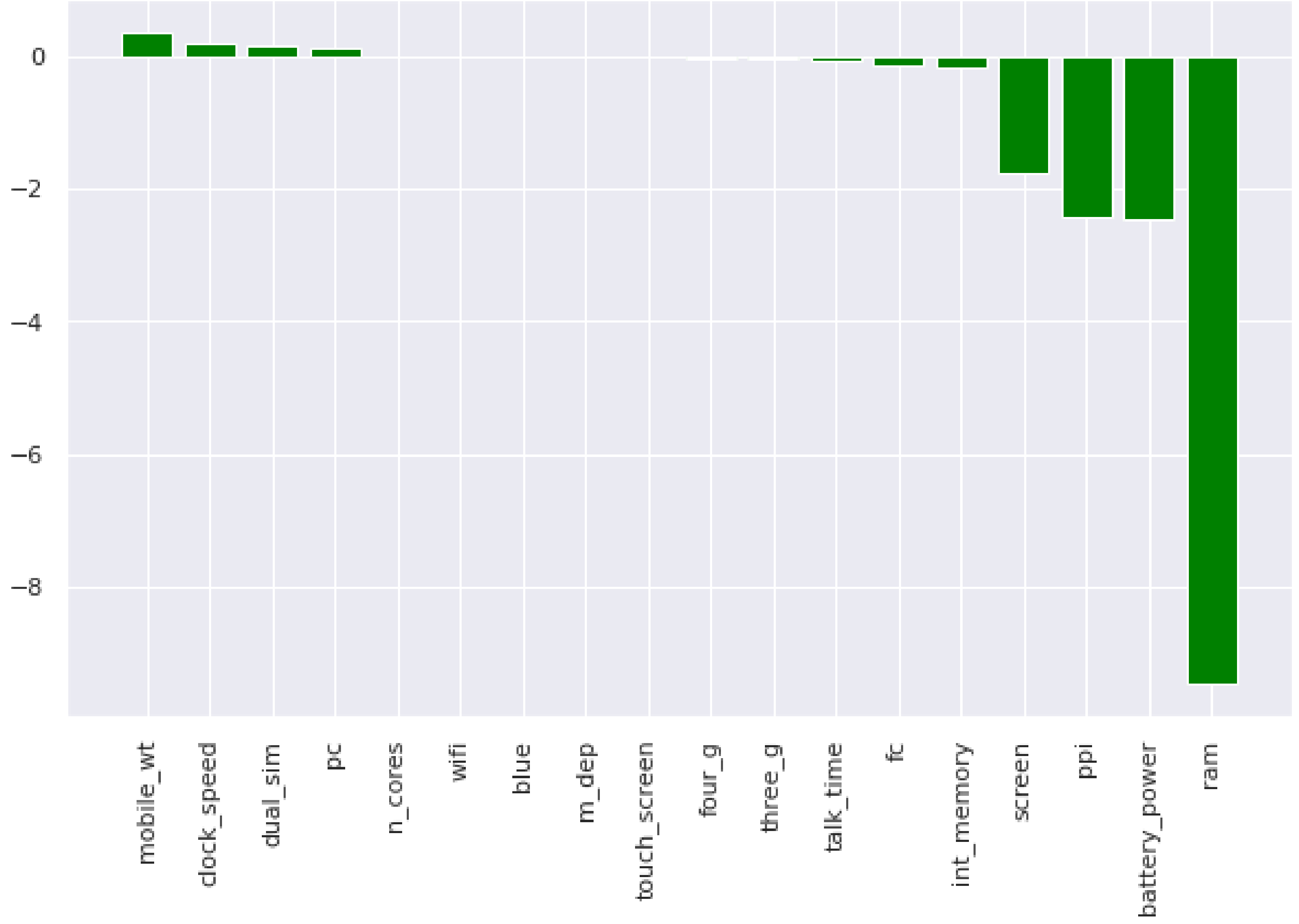
	battery_power	blue	clock_speed	dual_sim	fc	four_g	int_memory	m_dep	mobile_wt	n_cores	pc	ram	talk_time	three_g	touch_screen	wifi	screen	ppi	price_range
battery_power	1	0.01	0.01	-0.04	0.03	0.02	-0	0.03	0	-0.03	0.03	-0	0.05	0.01	-0.01	-0.01	-0.03	0.02	0.2
blue	0.01	1	0.02	0.04	0	0.01	0.04	0	-0.01	0.04	-0.01	0.03	0.01	-0.03	0.01	-0.02	-0.01	-0.01	0.02
clock_speed	0.01	0.02	1	-0	-0	-0.04	0.01	-0.01	0.01	-0.01	-0.01	0	-0.01	-0.05	0.02	-0.02	-0.02	0.01	-0.01
dual_sim	-0.04	0.04	-0	1	-0.03	0	-0.02	-0.02	-0.01	-0.02	-0.02	0.04	-0.04	-0.01	-0.02	0.02	-0.01	0.01	0.02
fc	0.03	0	-0	-0.03	1	-0.02	-0.03	-0	0.02	-0.01	0.64	0.02	-0.01	0	-0.01	0.02	-0.01	0.01	0.02
four_g	0.02	0.01	-0.04	0	-0.02	1	0.01	-0	-0.02	-0.03	-0.01	0.01	-0.05	0.58	0.02	-0.02	0.03	-0.02	0.01
int_memory	-0	0.04	0.01	-0.02	-0.03	0.01	1	0.01	-0.03	-0.03	-0.03	0.03	-0	-0.01	-0.03	0.01	0.03	-0.01	0.04
m_dep	0.03	0	-0.01	-0.02	-0	-0	0.01	1	0.02	-0	0.03	-0.01	0.02	-0.01	-0	-0.03	-0.02	0.04	0
mobile_wt	0	-0.01	0.01	-0.01	0.02	-0.02	-0.03	0.02	1	-0.02	0.02	-0	0.01	0	-0.01	-0	-0.04	0.03	-0.03
n_cores	-0.03	0.04	-0.01	-0.02	-0.01	-0.03	-0.03	-0	-0.02	1	-0	0	0.01	-0.01	0.02	-0.01	0.01	0.01	0
pc	0.03	-0.01	-0.01	-0.02	0.64	-0.01	-0.03	0.03	0.02	-0	1	0.03	0.01	-0	-0.01	0.01	0	0.01	0.03
ram	-0	0.03	0	0.04	0.02	0.01	0.03	-0.01	-0	0	0.03	1	0.01	0.02	-0.03	0.02	0.03	-0.03	0.92
talk_time	0.05	0.01	-0.01	-0.04	-0.01	-0.05	-0	0.02	0.01	0.01	0.01	0.01	1	-0.04	0.02	-0.03	-0.02	0	0.02
three_g	0.01	-0.03	-0.05	-0.01	0	0.58	-0.01	-0.01	0	-0.01	-0	0.02	-0.04	1	0.01	0	0.02	-0.02	0.02
touch_screen	-0.01	0.01	0.02	-0.02	-0.01	0.02	-0.03	-0	-0.01	0.02	-0.01	-0.03	0.02	0.01	1	0.01	-0.01	0	-0.03
wifi	-0.01	-0.02	-0.02	0.02	0.02	-0.02	0.01	-0.03	-0	-0.01	0.01	0.02	-0.03	0	0.01	1	0.03	0.02	0.02
screen	-0.03	-0.01	-0.02	-0.01	-0.01	0.03	0.03	-0.02	-0.04	0.01	0	0.03	-0.02	0.02	-0.01	0.03	1	-0.66	0.03
ppi	0.02	-0.01	0.01	0.01	0.01	-0.02	-0.01	0.04	0.03	0.01	0.01	-0.03	0	-0.02	0	0.02	-0.66	1	0.09
price_range	0.2	0.02	-0.01	0.02	0.02	0.01	0.04	0	-0.03	0	0.03	0.92	0.02	0.02	-0.03	0.02	0.03	0.09	1

Feature Engineering

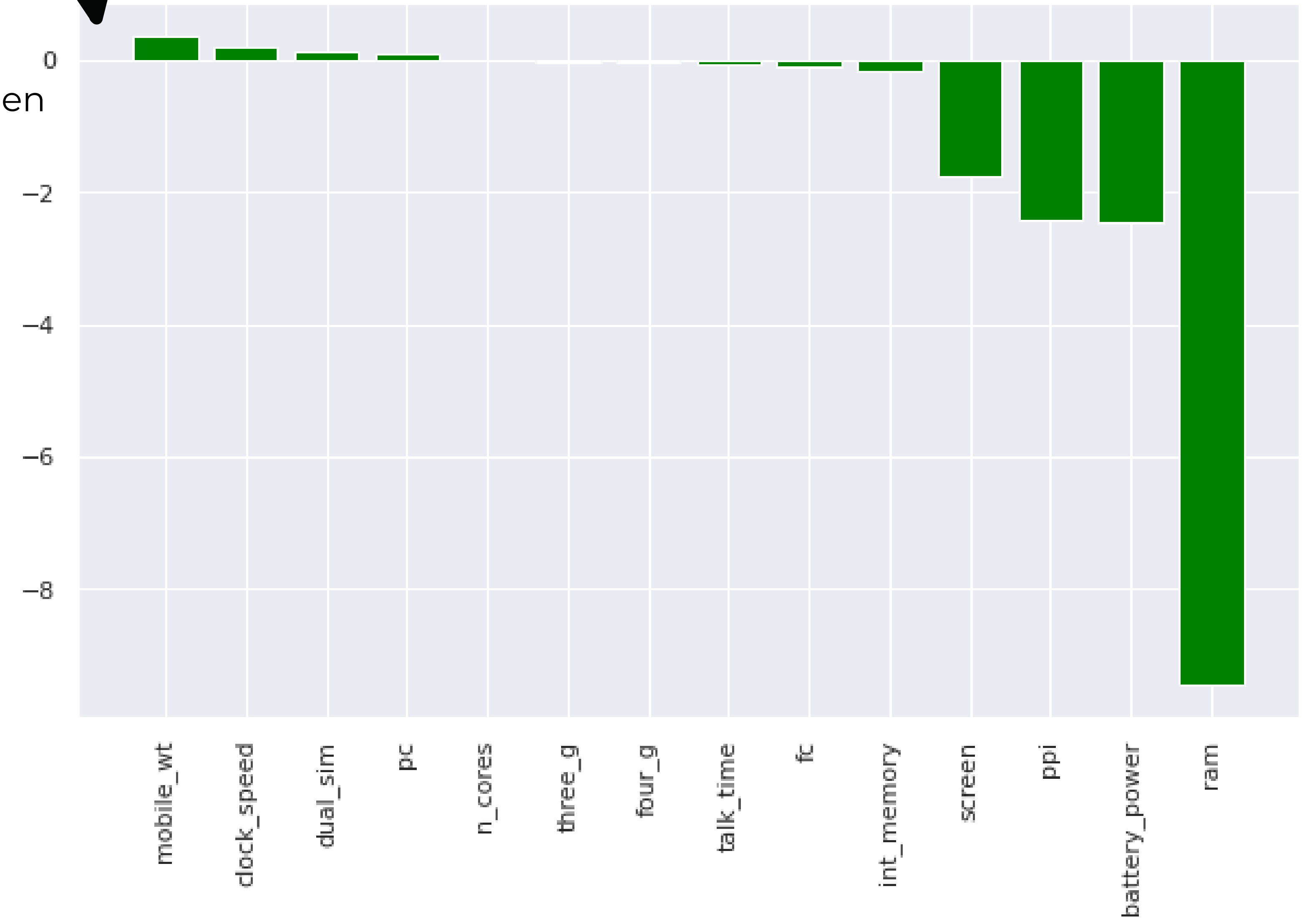
Dropping irrelevant columns

wifi,
blue,
m_dep
touch_screen

Feature importance obtained from coefficients



Feature importance obtained from coefficients



Model Selection

- According to the data, we need to select suitable classification models.
- We will be comparing between 4 models

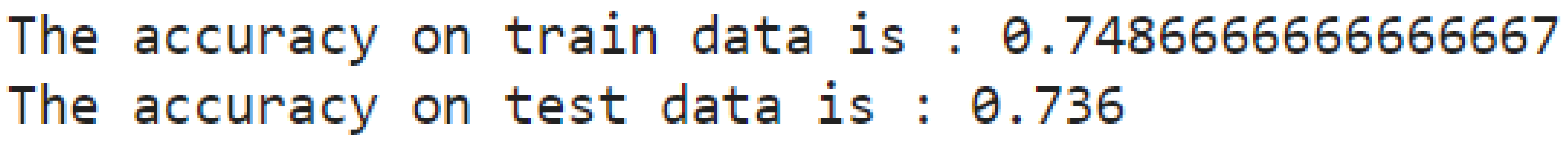
Logistic Regression

Random Forest

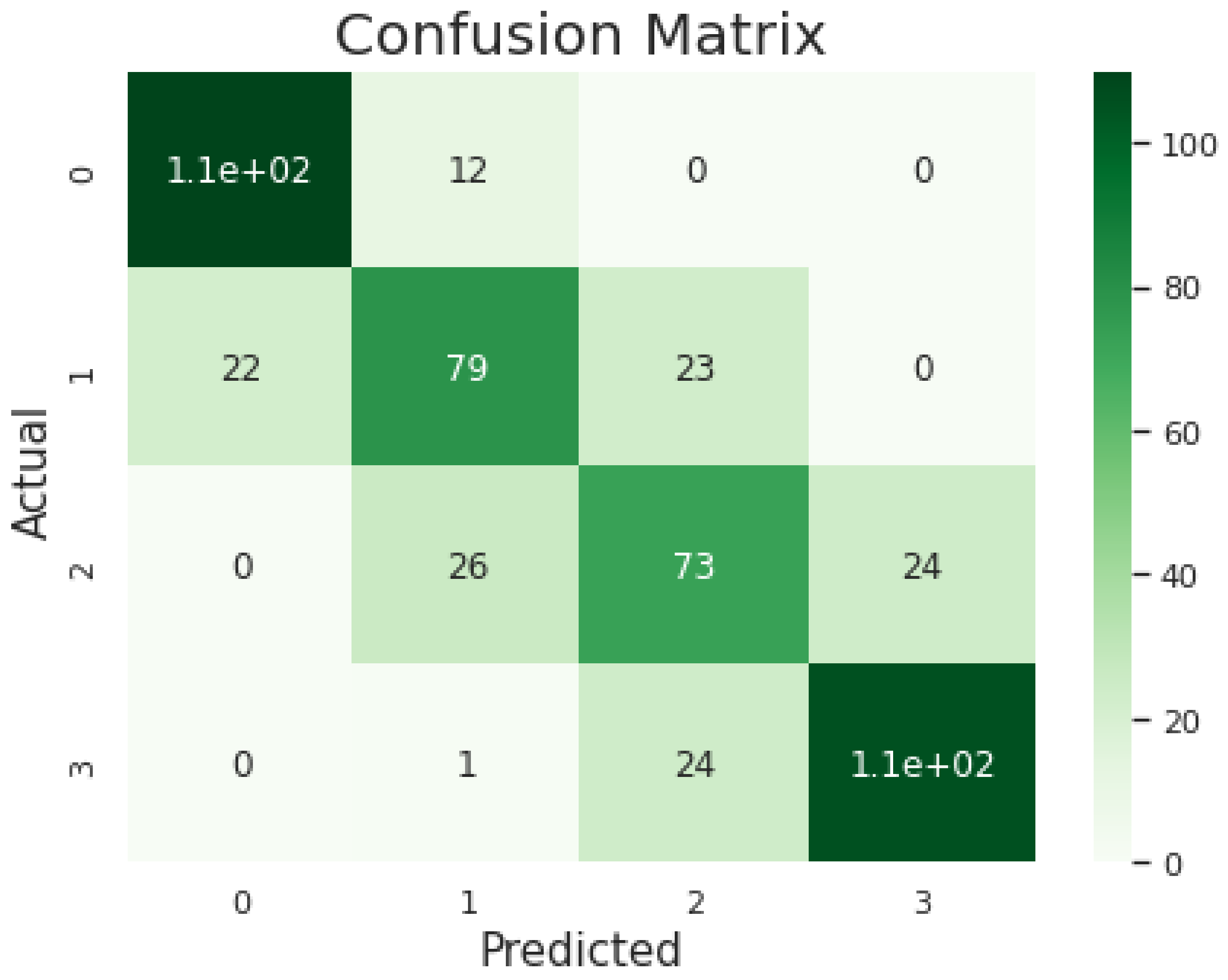
KNN

SVM

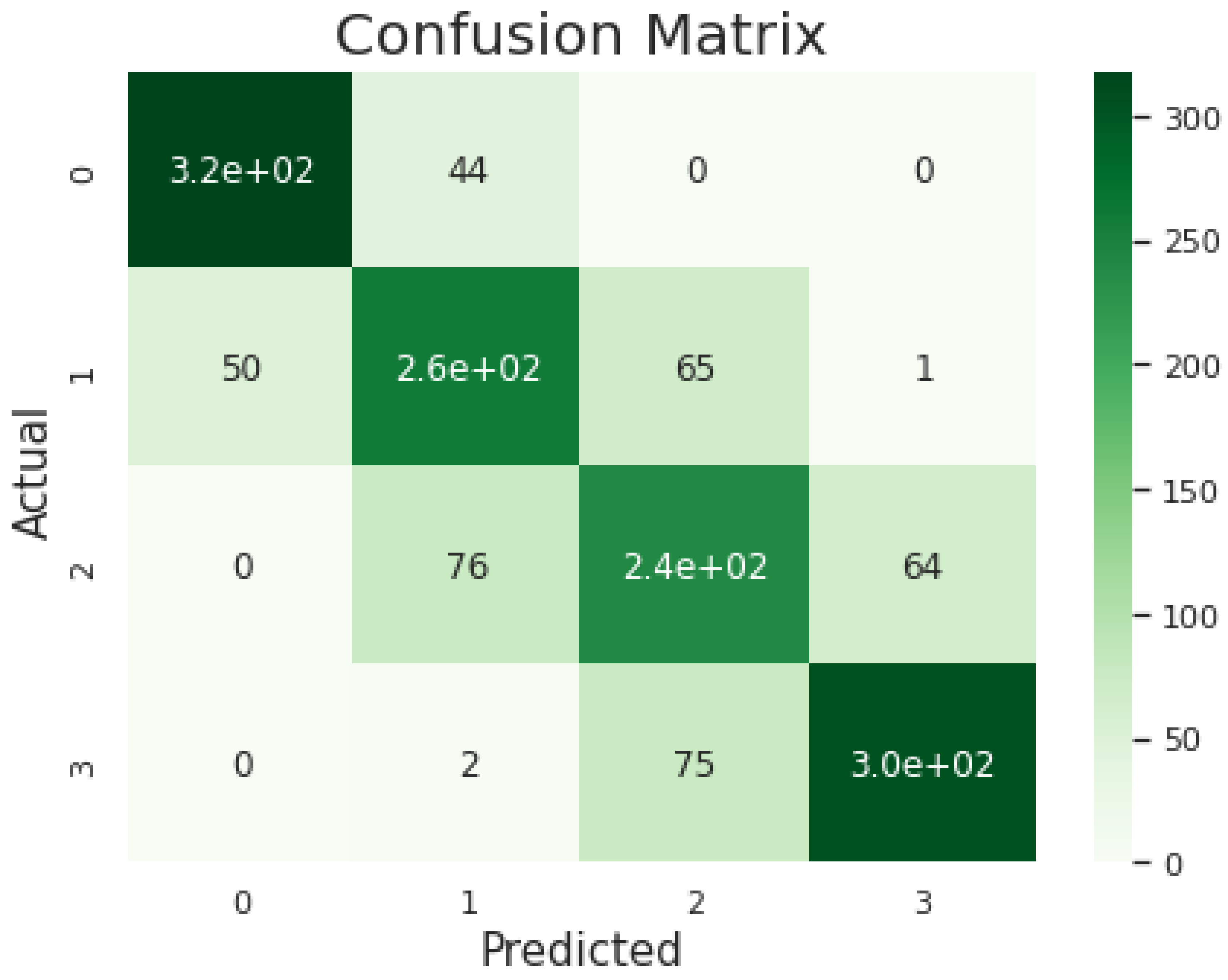
Logistic Regression



```
The accuracy on train data is : 0.7486666666666667
The accuracy on test data is : 0.736
```



Confusion Matrix of Test Set



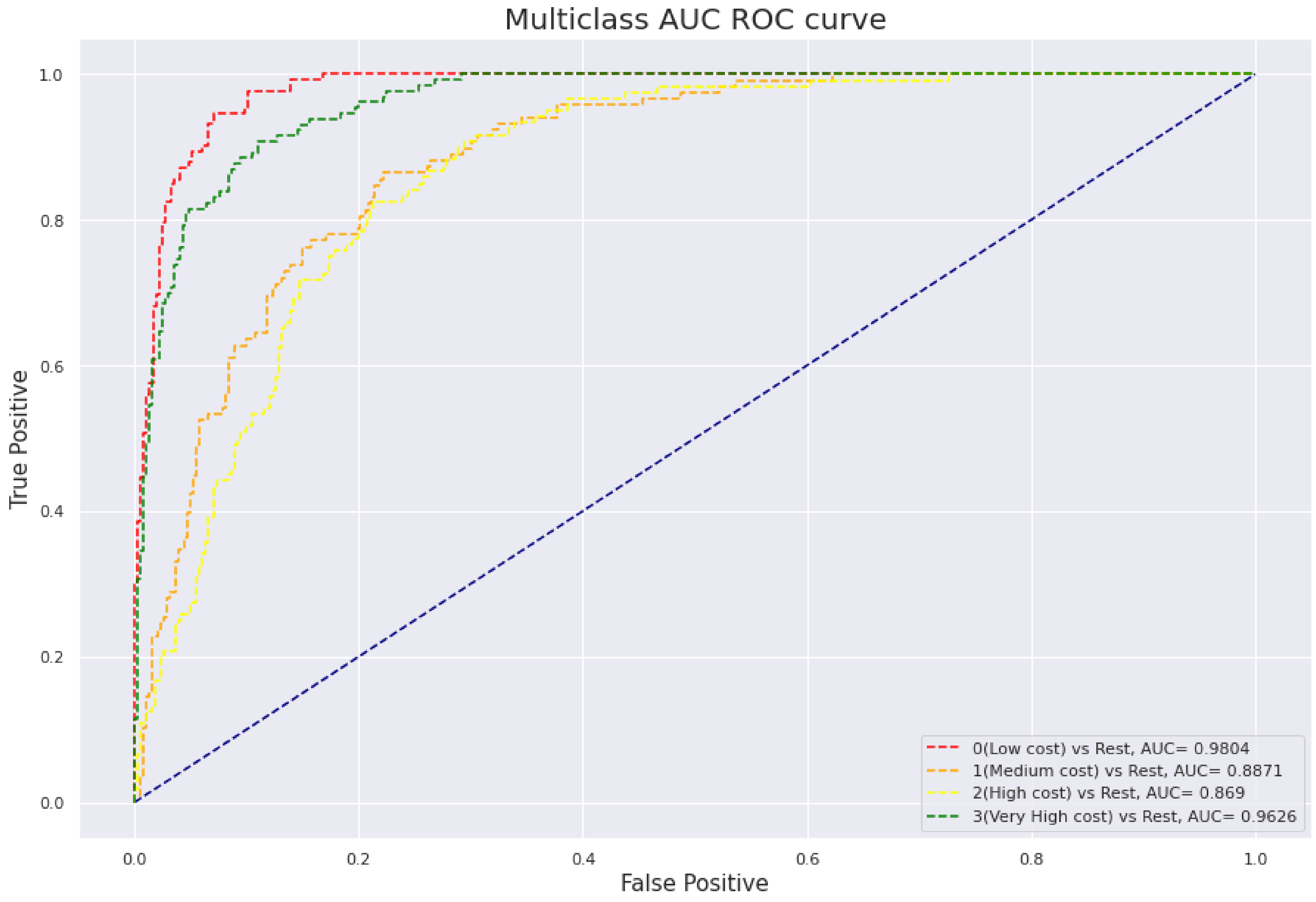
Confusion Matrix of Train Set

Logistic Regression

```
The ROC AUC score on the train data is:  0.9306323173995289


The ROC AUC score on the test data is:  0.9247836188284699
```

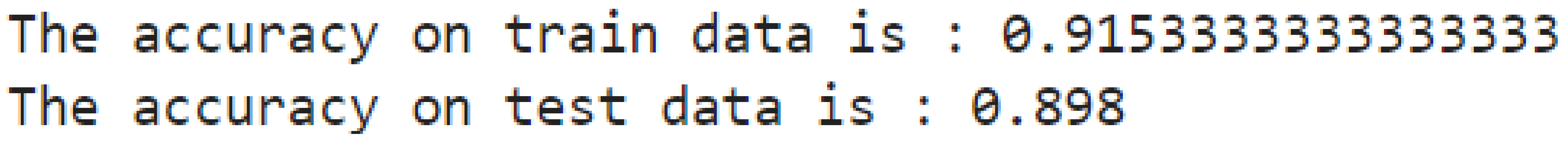
Observations

- Prediction of price range of Low & Very High Cost is excellent
- Prediction of price range of Medium and High Cost is good.
- Prediction accuracy is very good but not excellent, Not to optimise. bold text

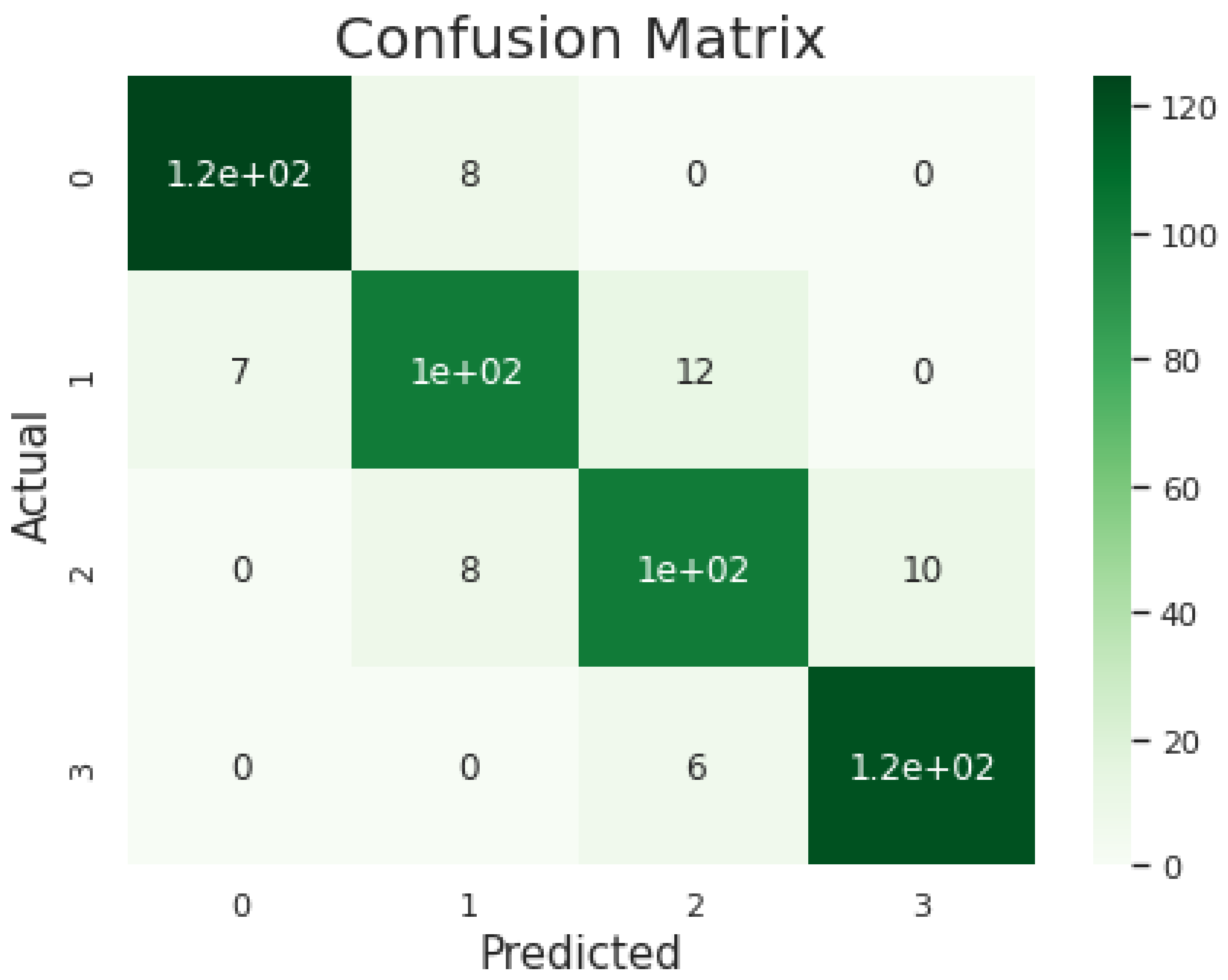


Logistic Regression

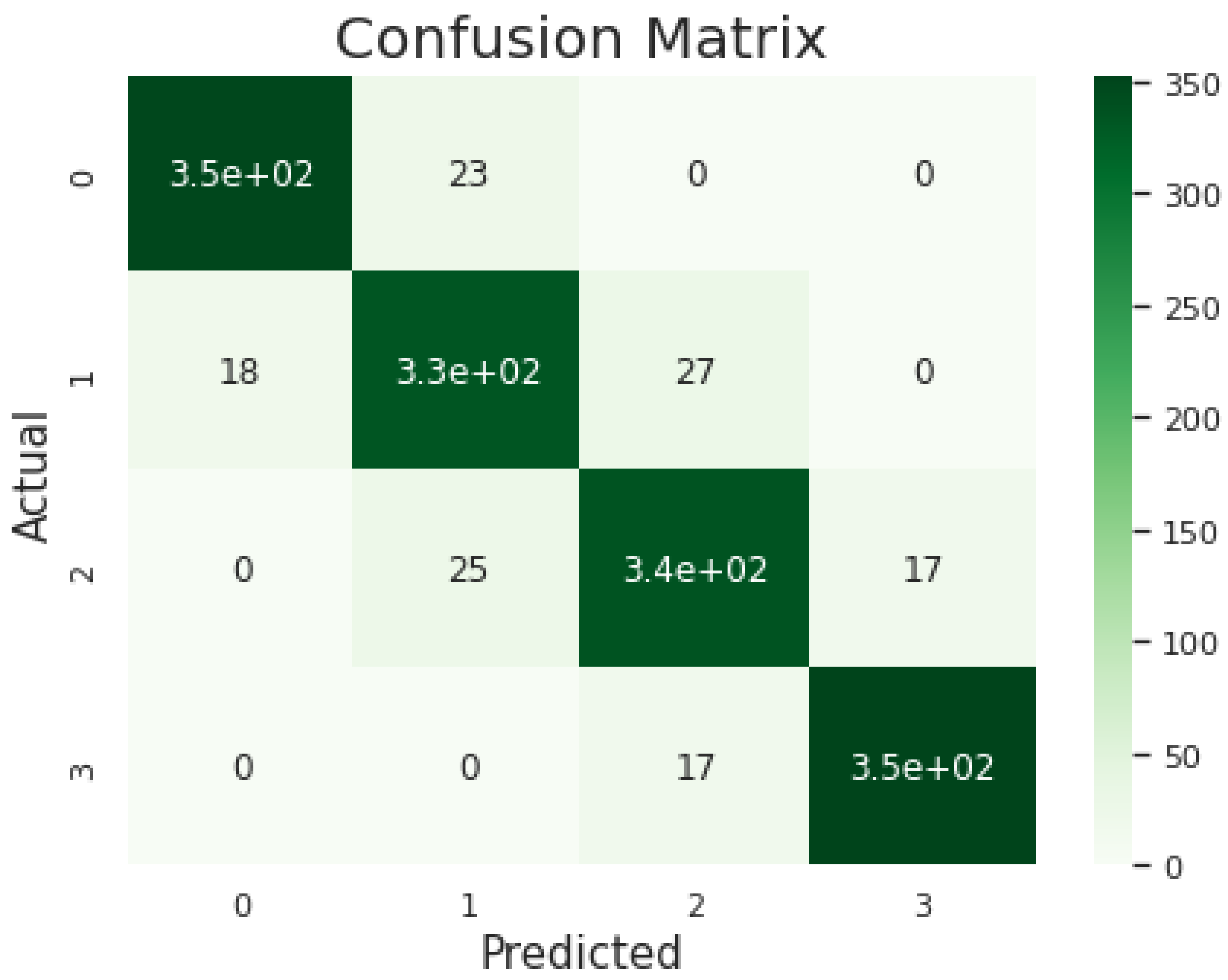
Hyper Parameter Tuning and Cross Validation of Logistic Regression Classification



```
The accuracy on train data is : 0.9153333333333333
The accuracy on test data is : 0.898
```

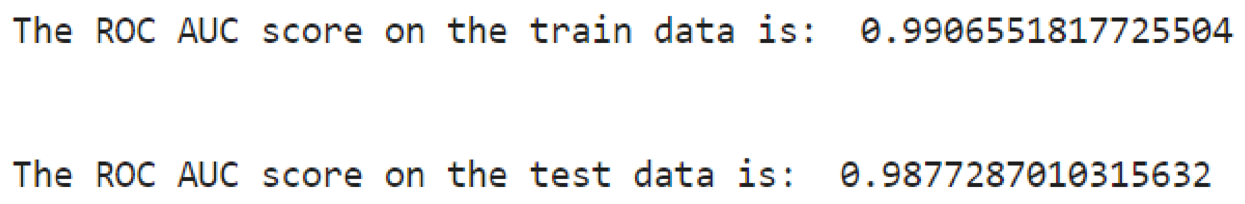


Confusion Matrix of Test Set



Confusion Matrix of Train Set

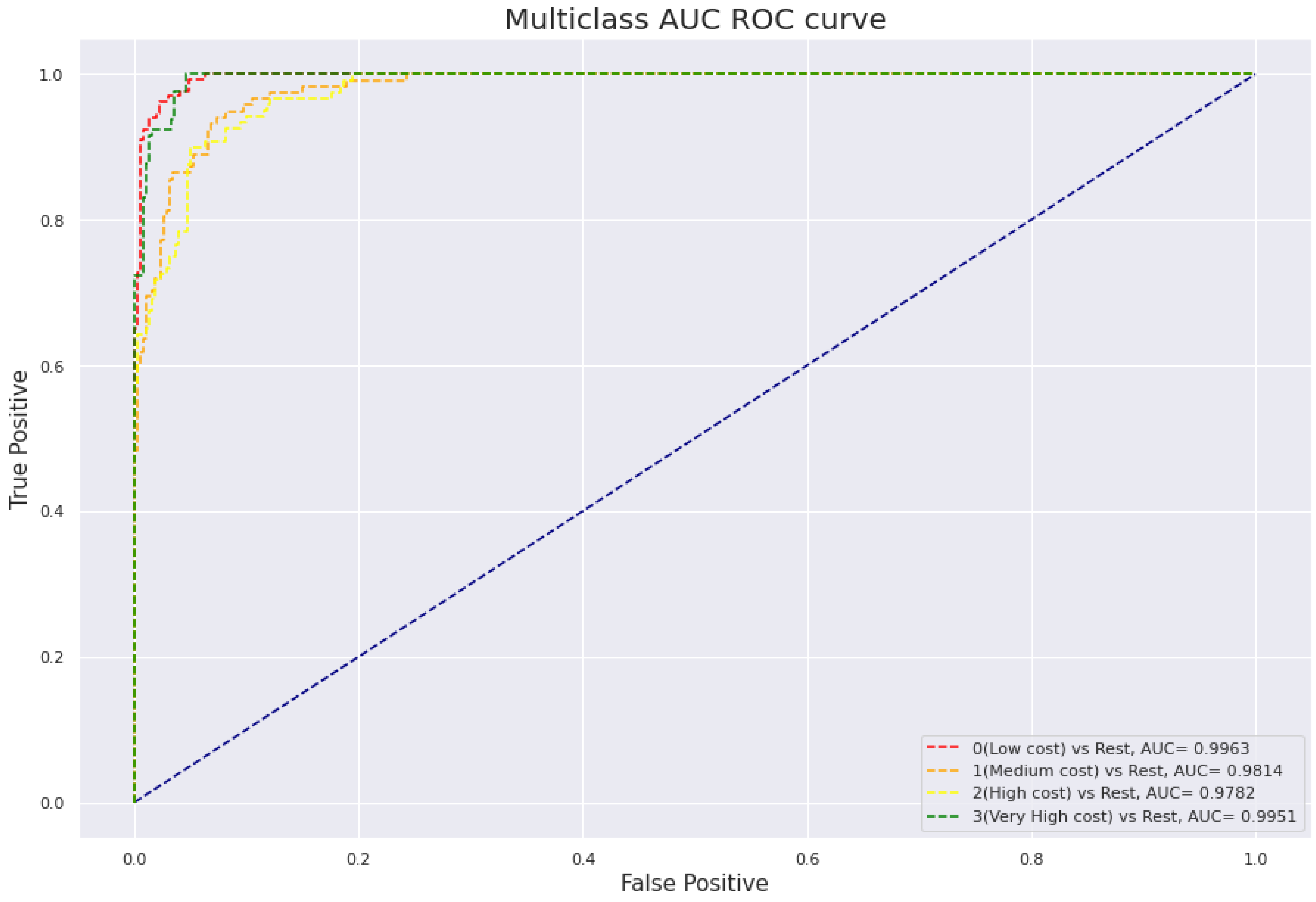
Logistic Regression



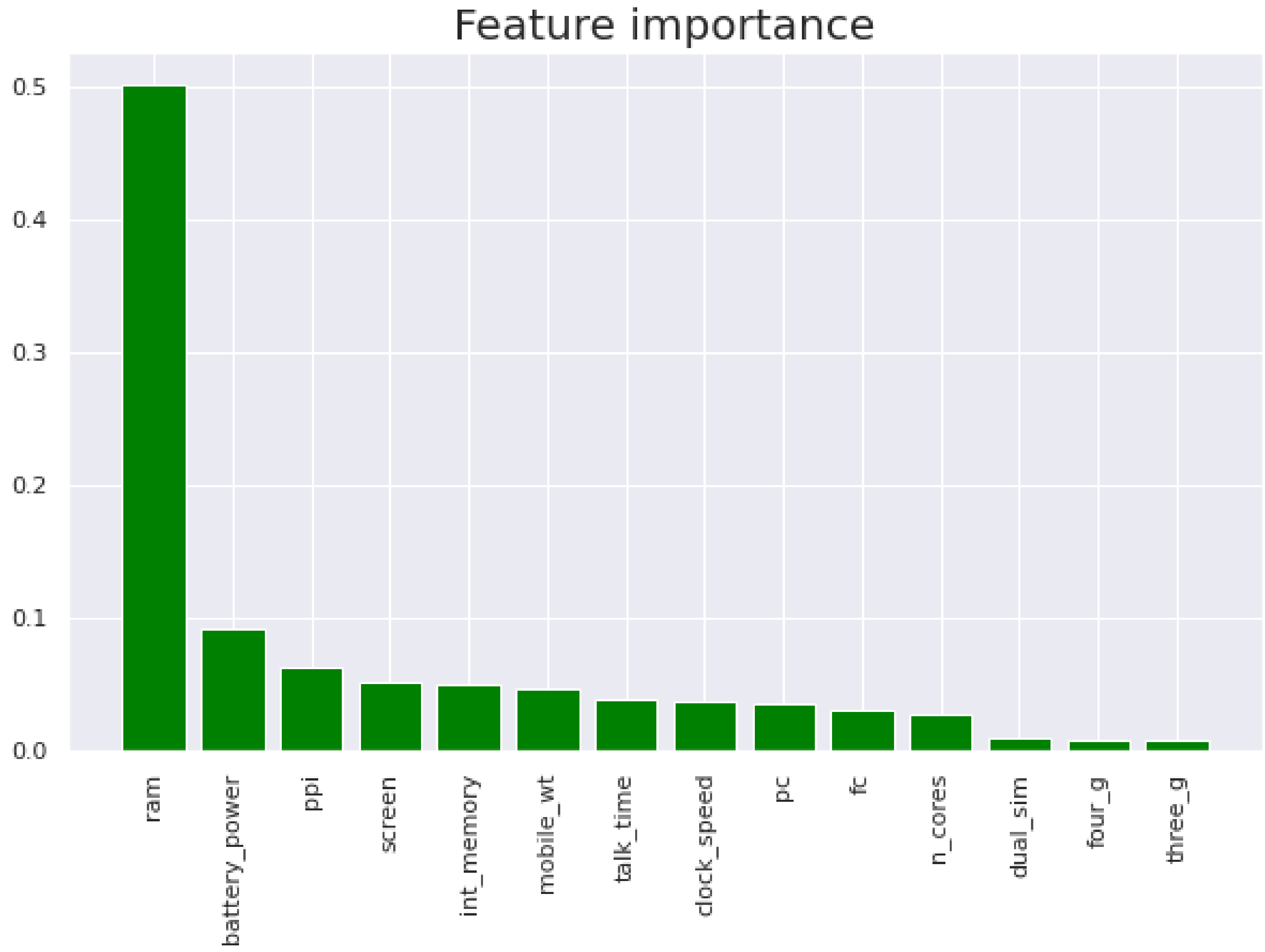
```
The ROC AUC score on the train data is:  0.9906551817725504


The ROC AUC score on the test data is:  0.9877287010315632
```

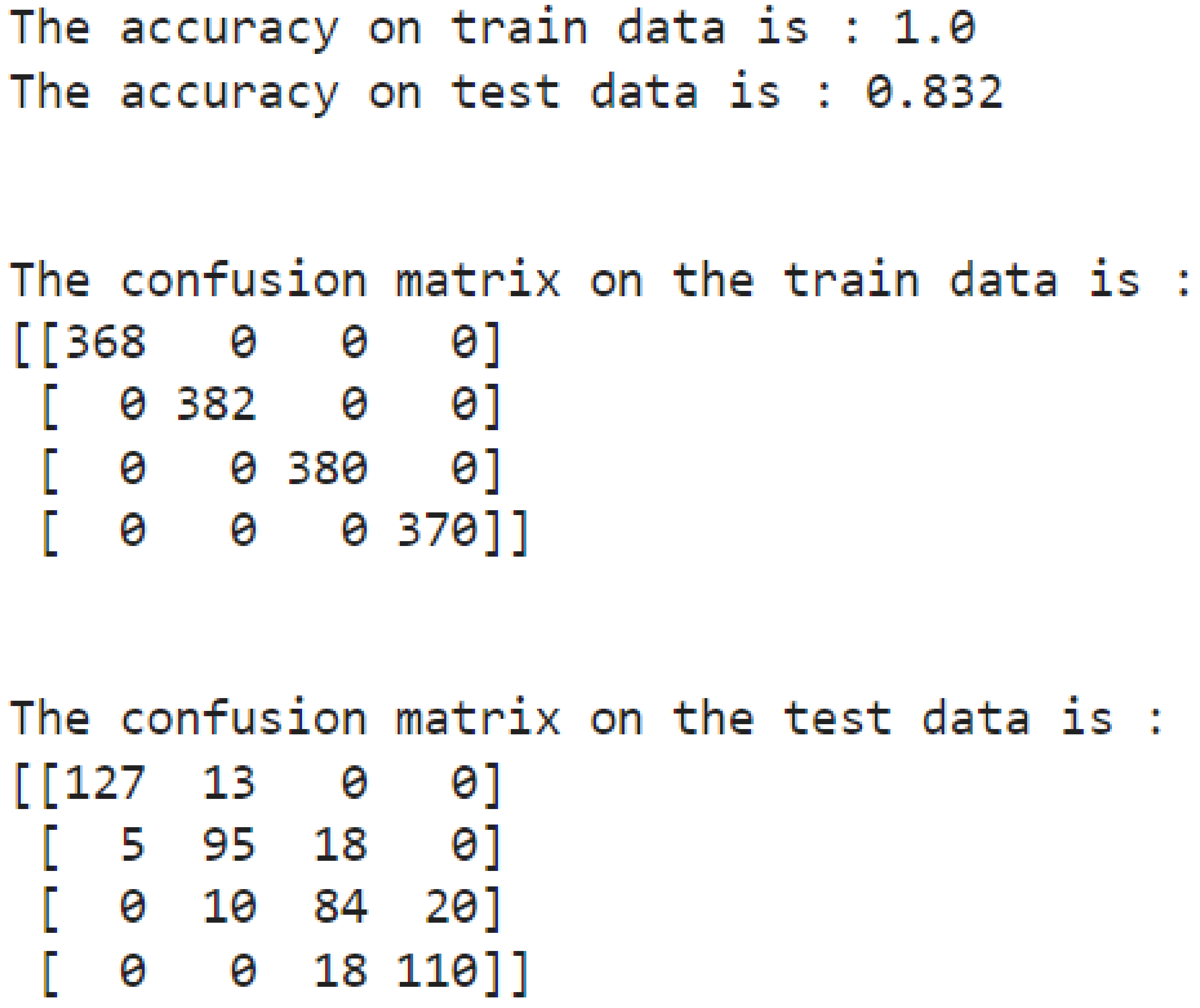
Observations

- Overall accuracy has improved to excellent levels
- Prediction accuracy for price_range 1 & 2 has also increased to excellent levels
- Accuracy can be further improved if max_iterations are increased, but we have capped it to 100 to keep run time under control for GOOGLE COLAB(Free Version)



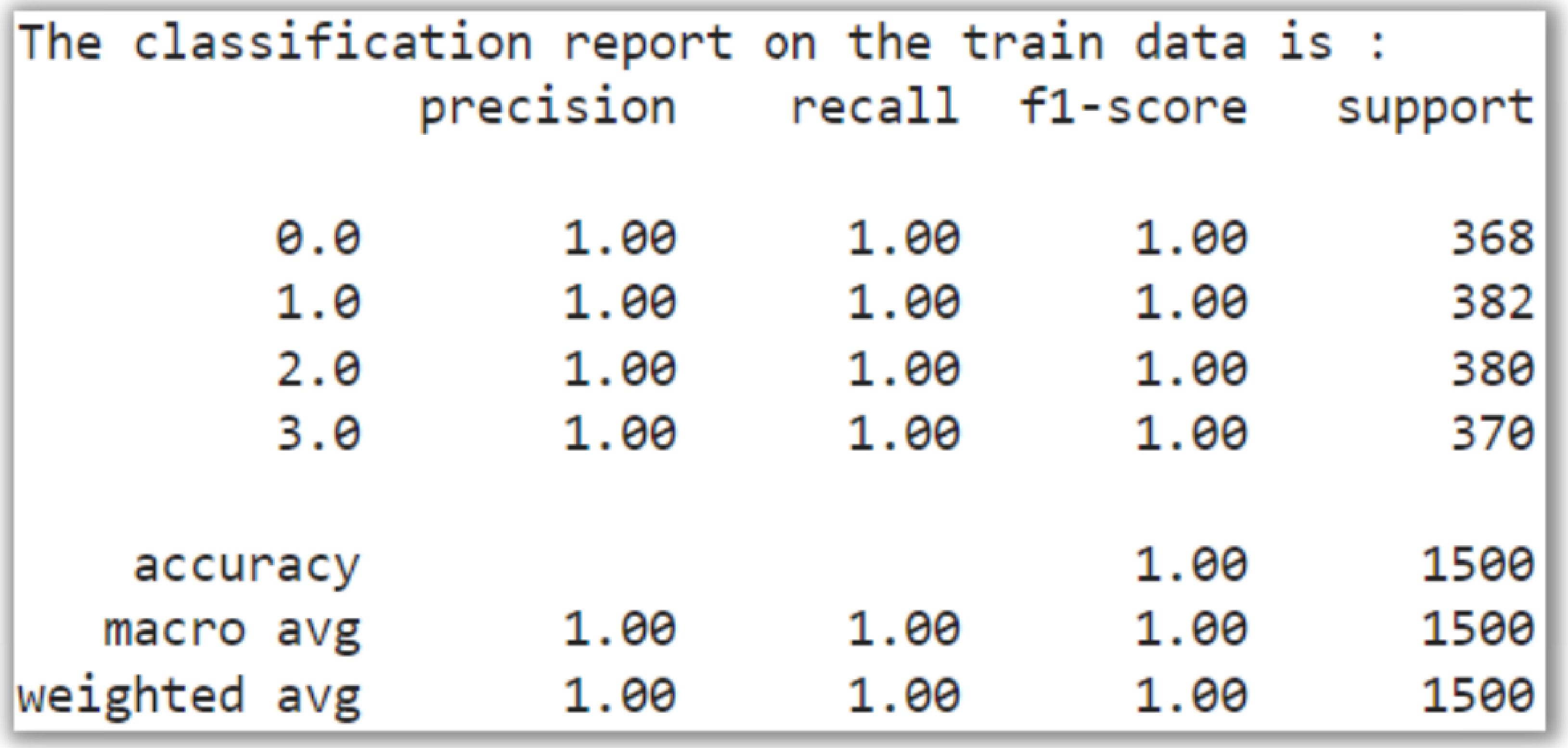
Random Forest Classification



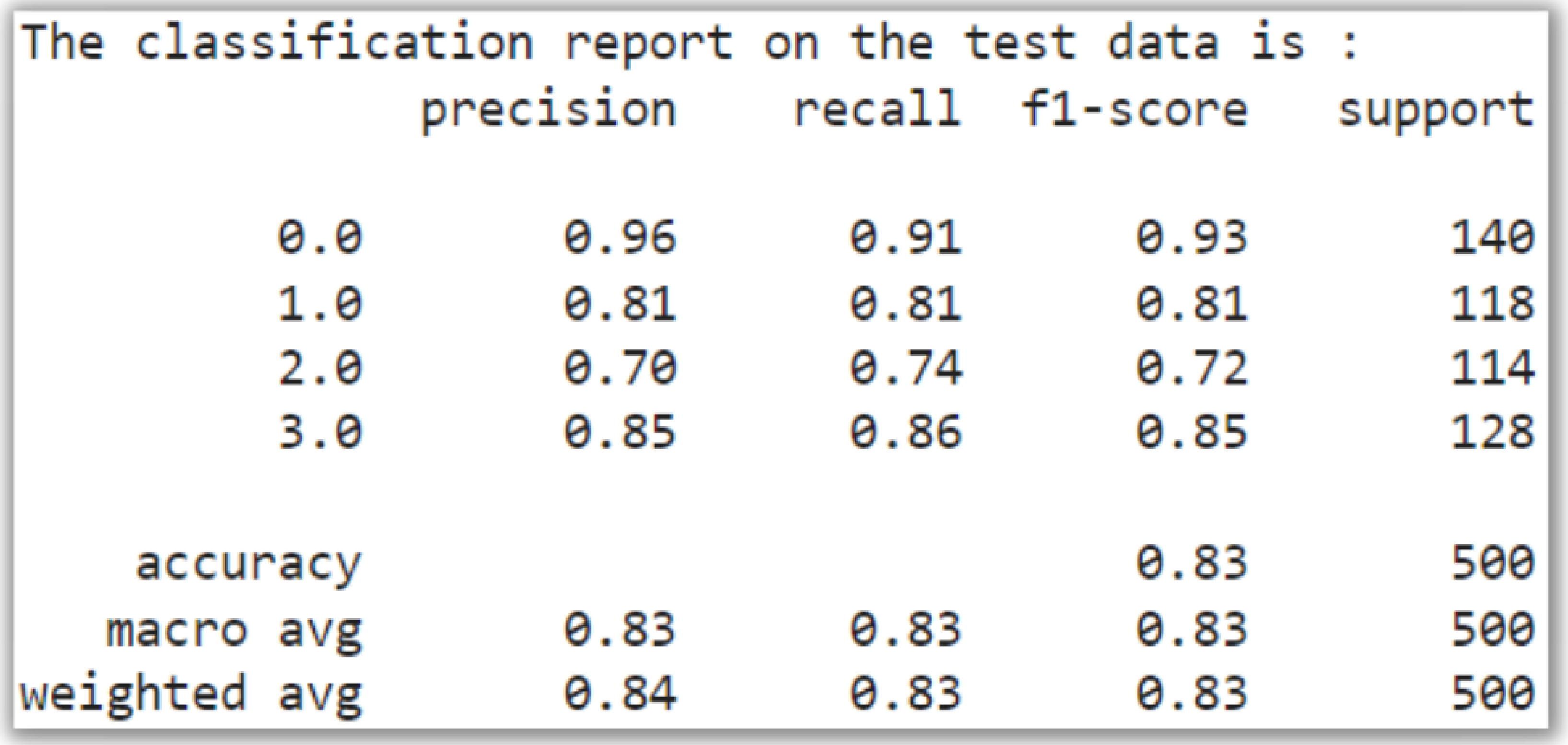


```
The accuracy on train data is : 1.0
The accuracy on test data is : 0.832


The confusion matrix on the train data is :
[[368   0   0   0]
 [  0 382   0   0]
 [  0   0 380   0]
 [  0   0   0 370]]


The confusion matrix on the test data is :
[[127  13   0   0]
 [  5  95  18   0]
 [  0  10  84  20]
 [  0   0  18 110]]
```

Random Forest Classification



```
The classification report on the train data is :
              precision    recall  f1-score   support

         0.0       1.00      1.00      1.00       368
         1.0       1.00      1.00      1.00       382
         2.0       1.00      1.00      1.00       380
         3.0       1.00      1.00      1.00       370

    accuracy                           1.00      1500
   macro avg       1.00      1.00      1.00      1500
weighted avg       1.00      1.00      1.00      1500
```



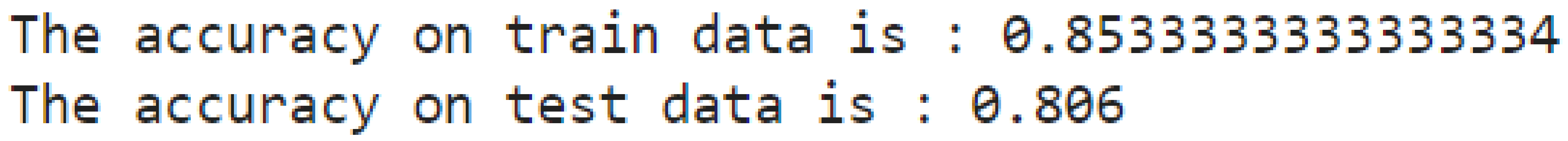
```
The classification report on the test data is :
              precision    recall  f1-score   support

         0.0       0.96      0.91      0.93       140
         1.0       0.81      0.81      0.81       118
         2.0       0.70      0.74      0.72       114
         3.0       0.85      0.86      0.85       128

    accuracy                           0.83       500
   macro avg       0.83      0.83      0.83       500
weighted avg       0.84      0.83      0.83       500
```

```
The ROC AUC score on the train data is:  1.0
The ROC AUC score on the test data is:  0.9658171969858055
```

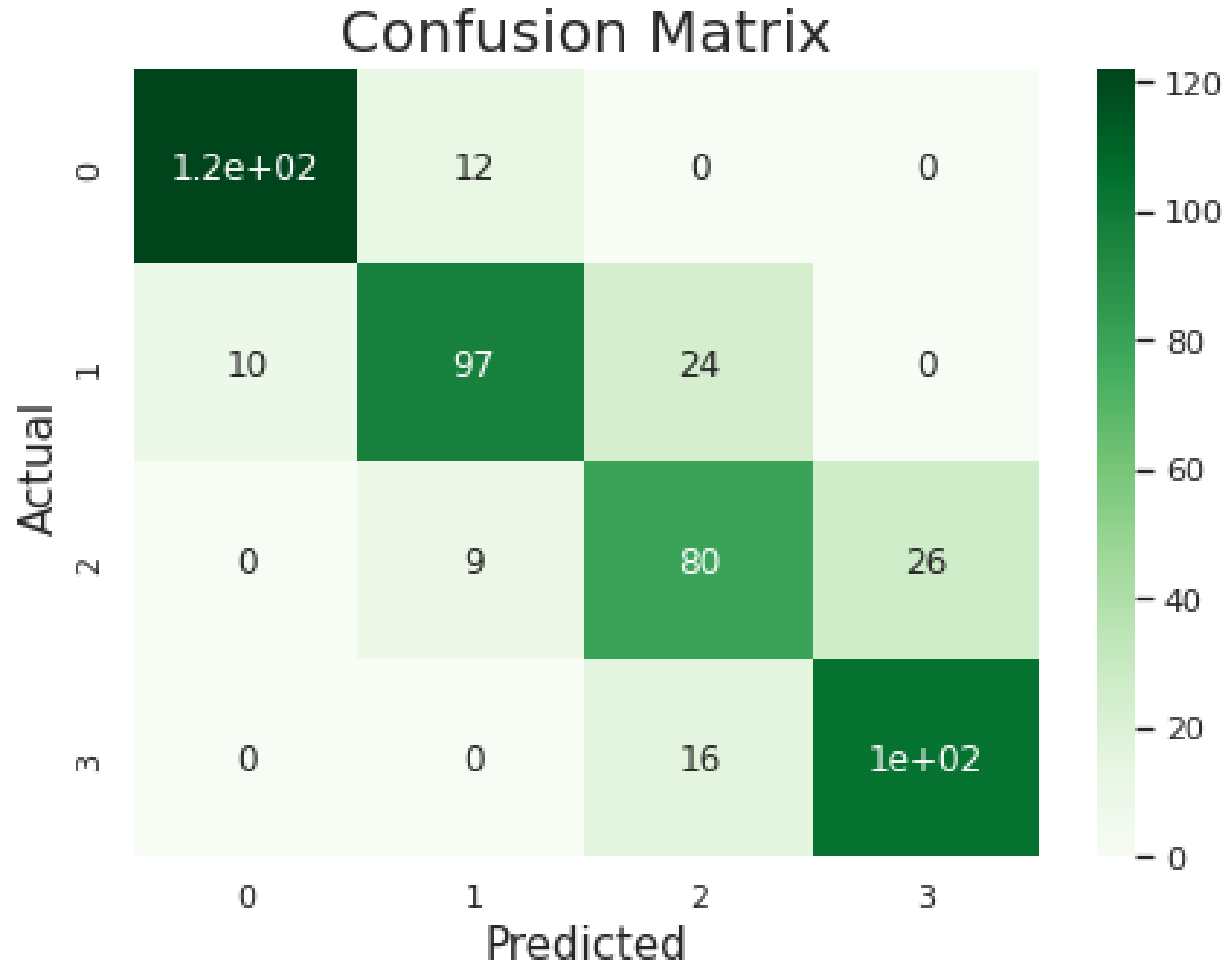
Observations

- Training accuracy is 1, Random Forest is overfitting
- Dropping random Forest and moving to next model

KNN Classification

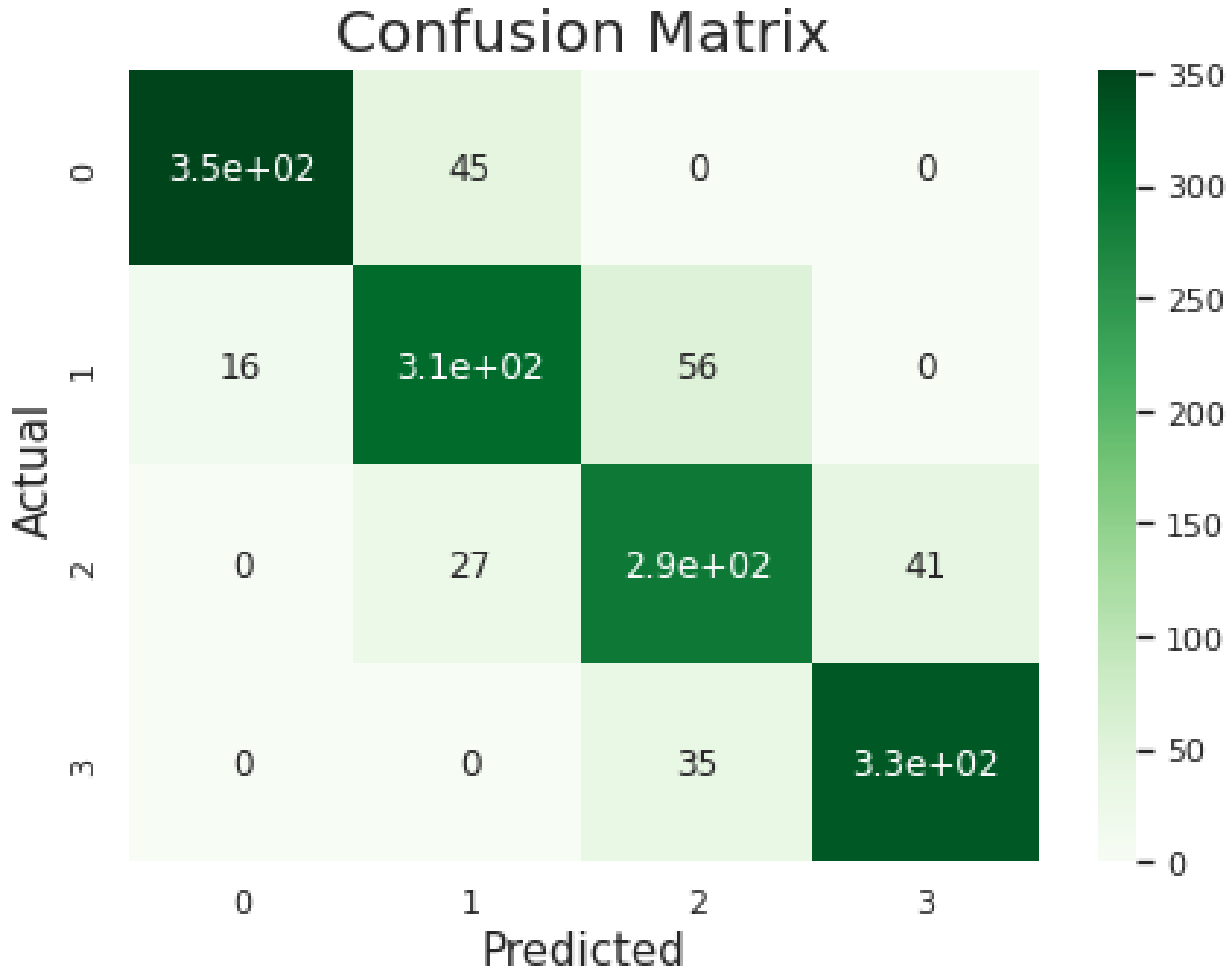


```
The accuracy on train data is : 0.8533333333333334
The accuracy on test data is : 0.806
```



Actual \ Predicted	0	1	2	3
0	1.2e+02	12	0	0
1	10	97	24	0
2	0	9	80	26
3	0	0	16	1e+02

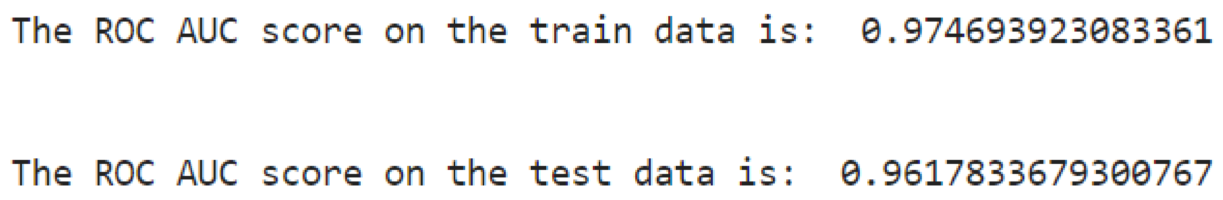
Confusion Matrix of Test Set



Actual \ Predicted	0	1	2	3
0	3.5e+02	45	0	0
1	16	3.1e+02	56	0
2	0	27	2.9e+02	41
3	0	0	35	3.3e+02

Confusion Matrix of Train Set

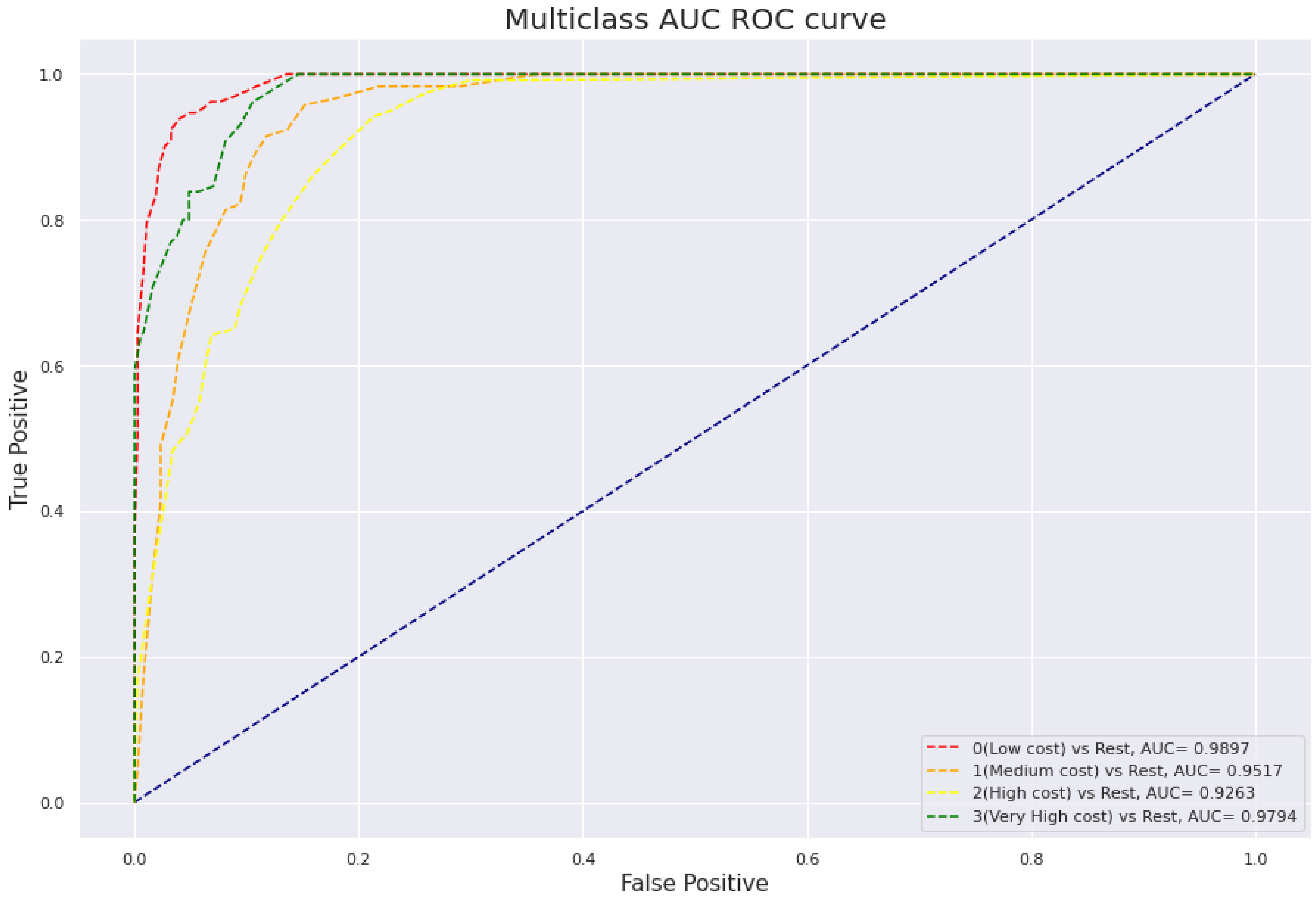
KNN Classification



```
The ROC AUC score on the train data is:  0.974693923083361

The ROC AUC score on the test data is:  0.9617833679300767
```

Observations

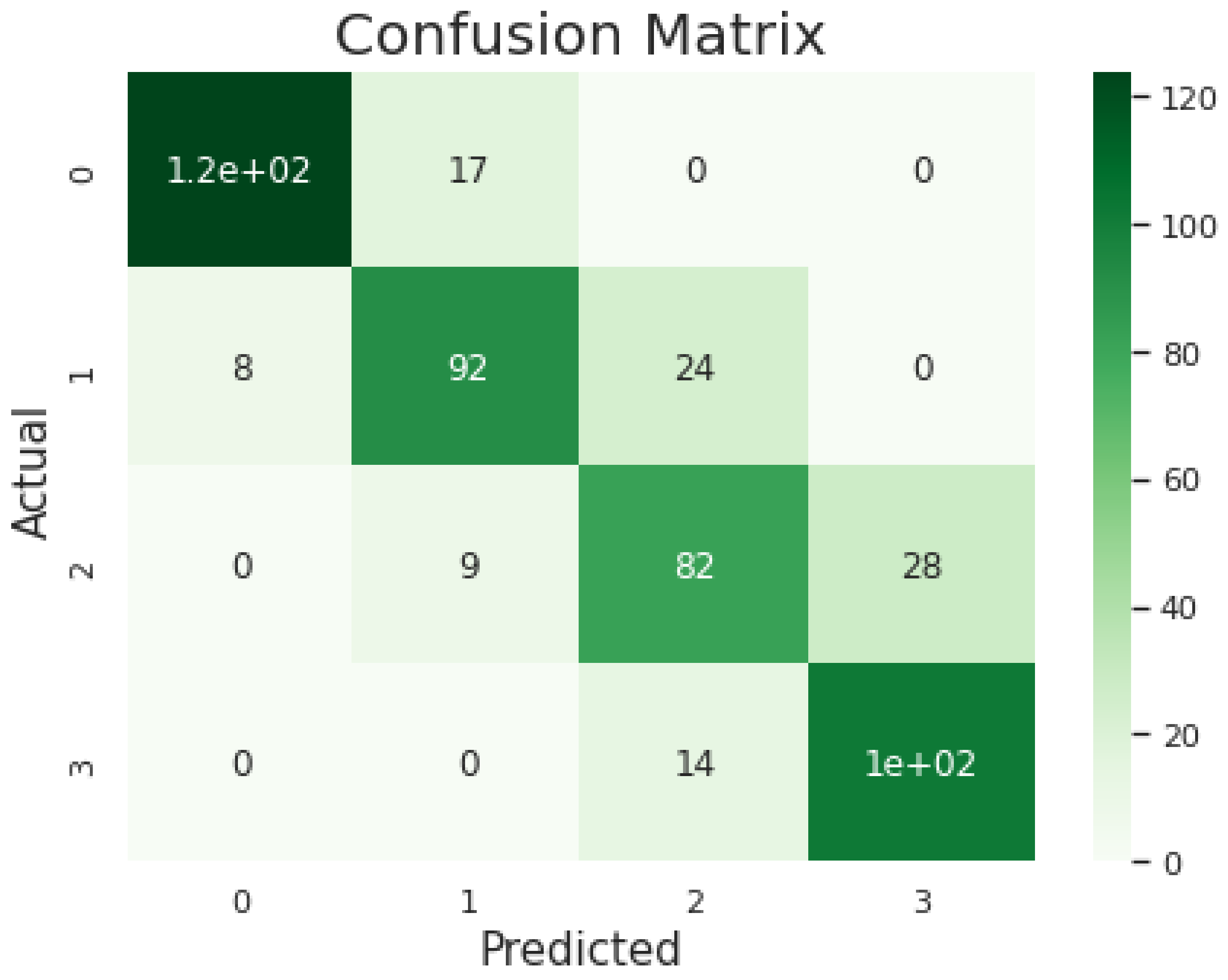
- Prediction accuracy is less than optimised Logistic classification
- AUC-ROC score is better than logistc classification on average for all 4 price_range
- Need to evaluate further with parameter tuning



KNN Classification

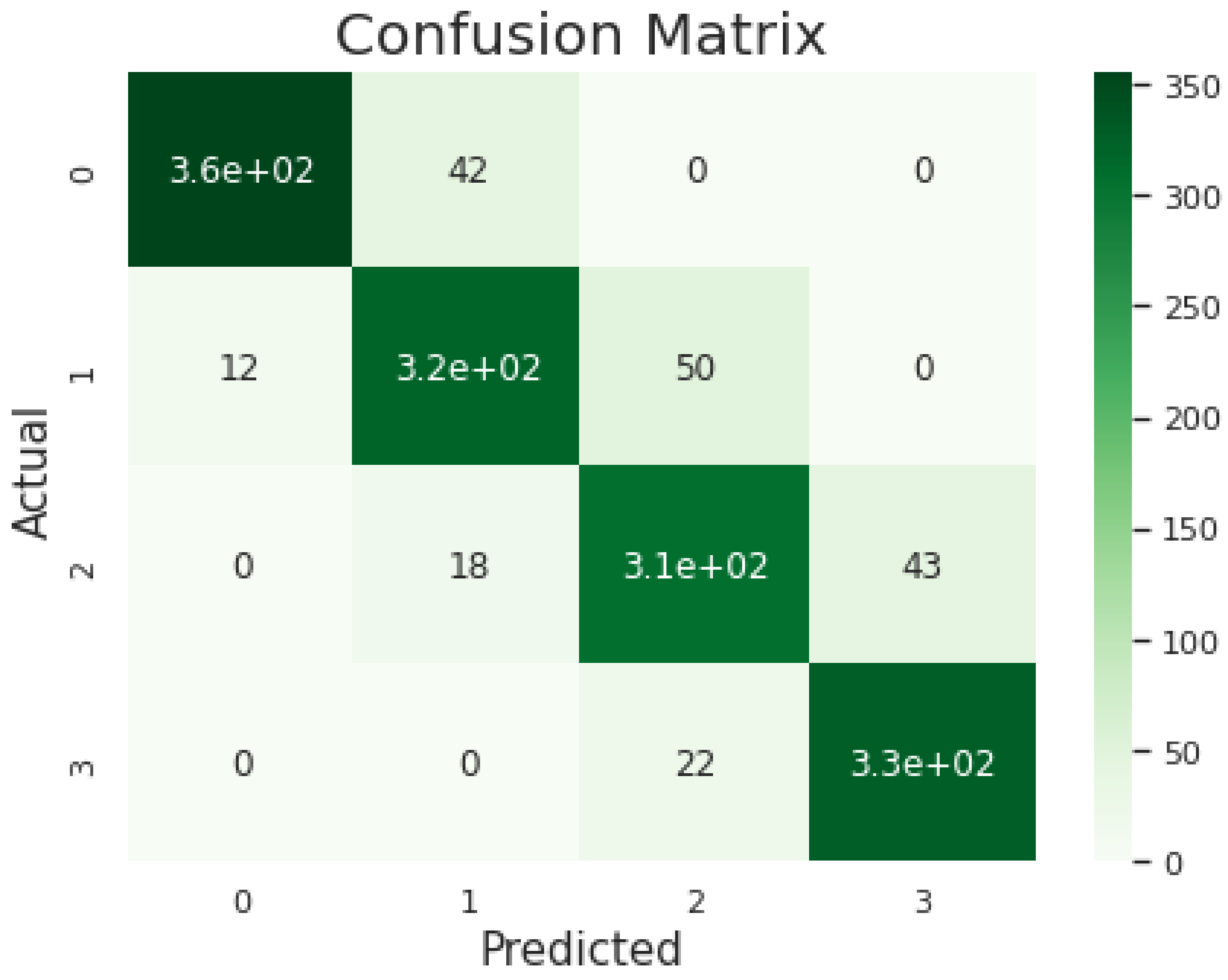
Hyper Parameter Tuning and Cross Validation of KNN Classification

```
The accuracy on train data is : 0.8753333333333333
The accuracy on test data is : 0.8
```



Actual \ Predicted	0	1	2	3
0	1.2e+02	17	0	0
1	8	92	24	0
2	0	9	82	28
3	0	0	14	1e+02

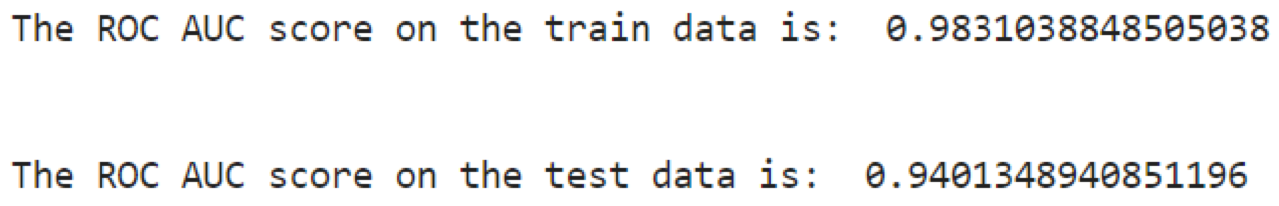
Confusion Matrix of Test Set



Actual \ Predicted	0	1	2	3
0	3.6e+02	42	0	0
1	12	3.2e+02	50	0
2	0	18	3.1e+02	43
3	0	0	22	3.3e+02

Confusion Matrix of Train Set

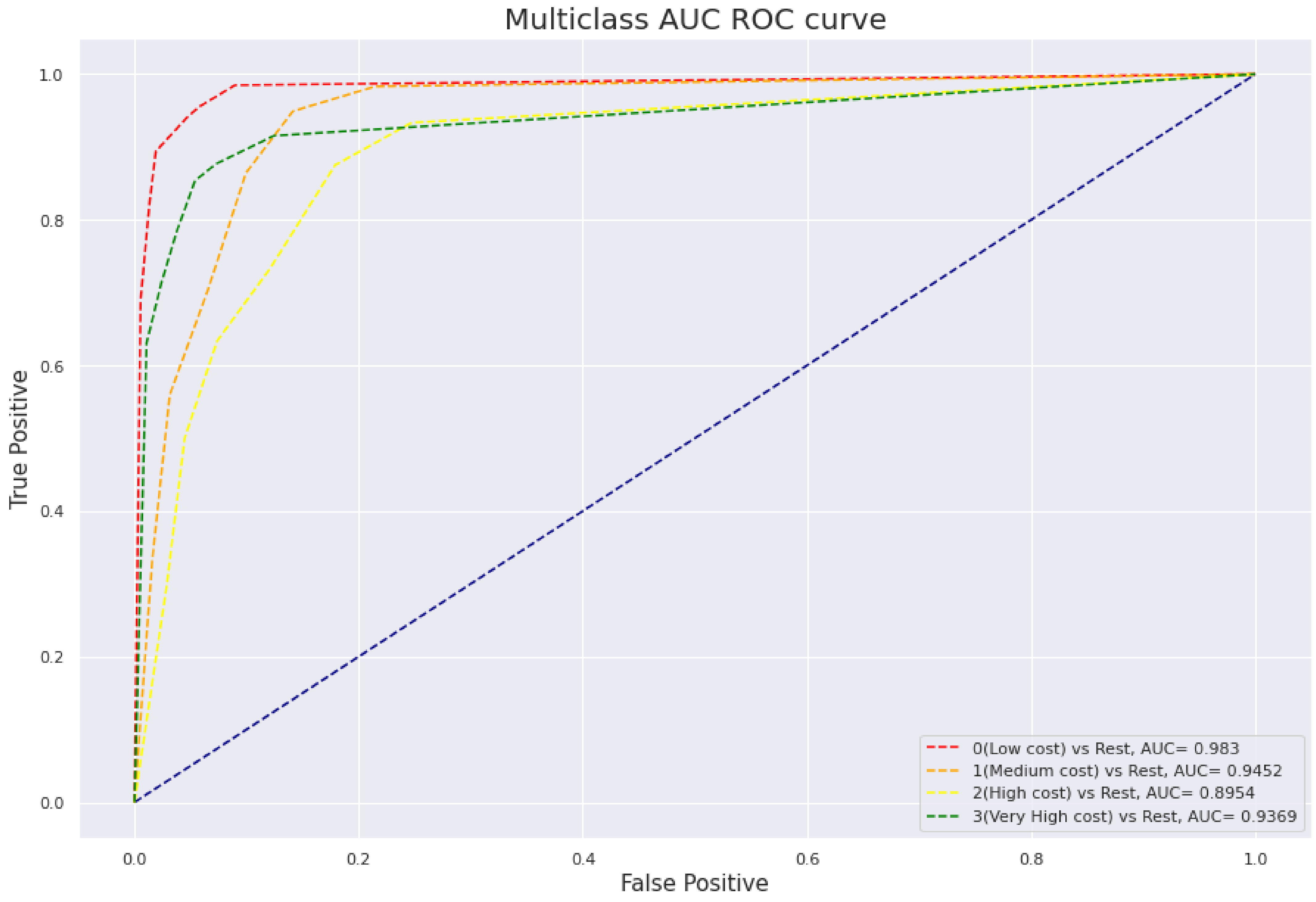
KNN Classification



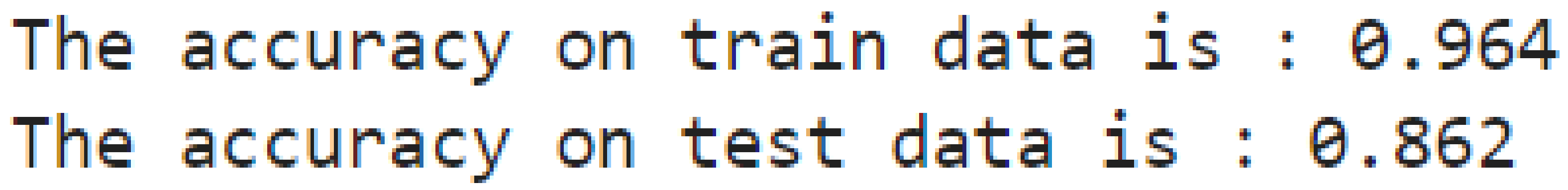
```
The ROC AUC score on the train data is:  0.9831038848505038

The ROC AUC score on the test data is:  0.9401348940851196
```

Observations

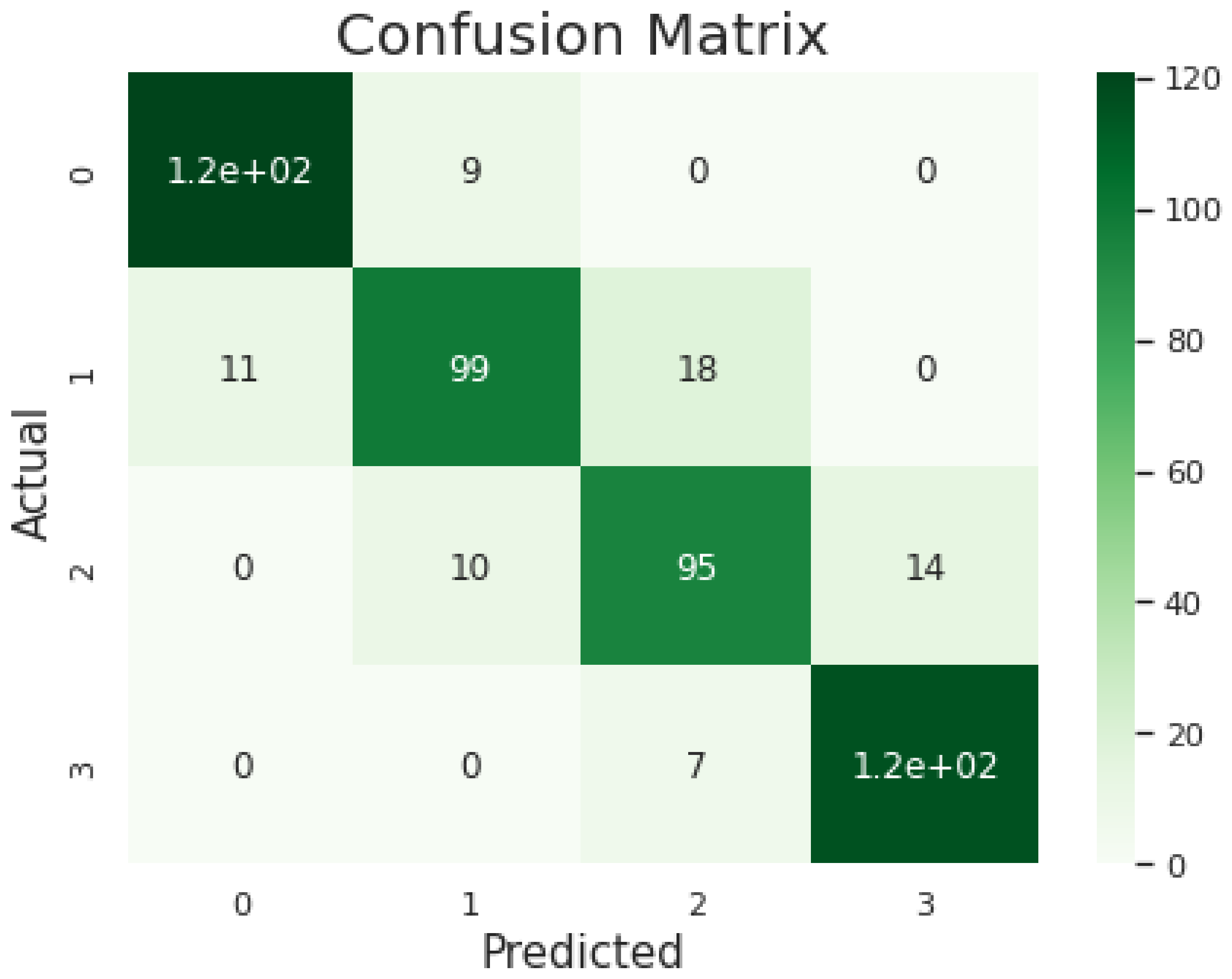
- After optimisation KNN improved very well.
- But for multiclass 'price_range' prediction of price_range = 2 is slightly less in comparisson with optimised logistic regression.



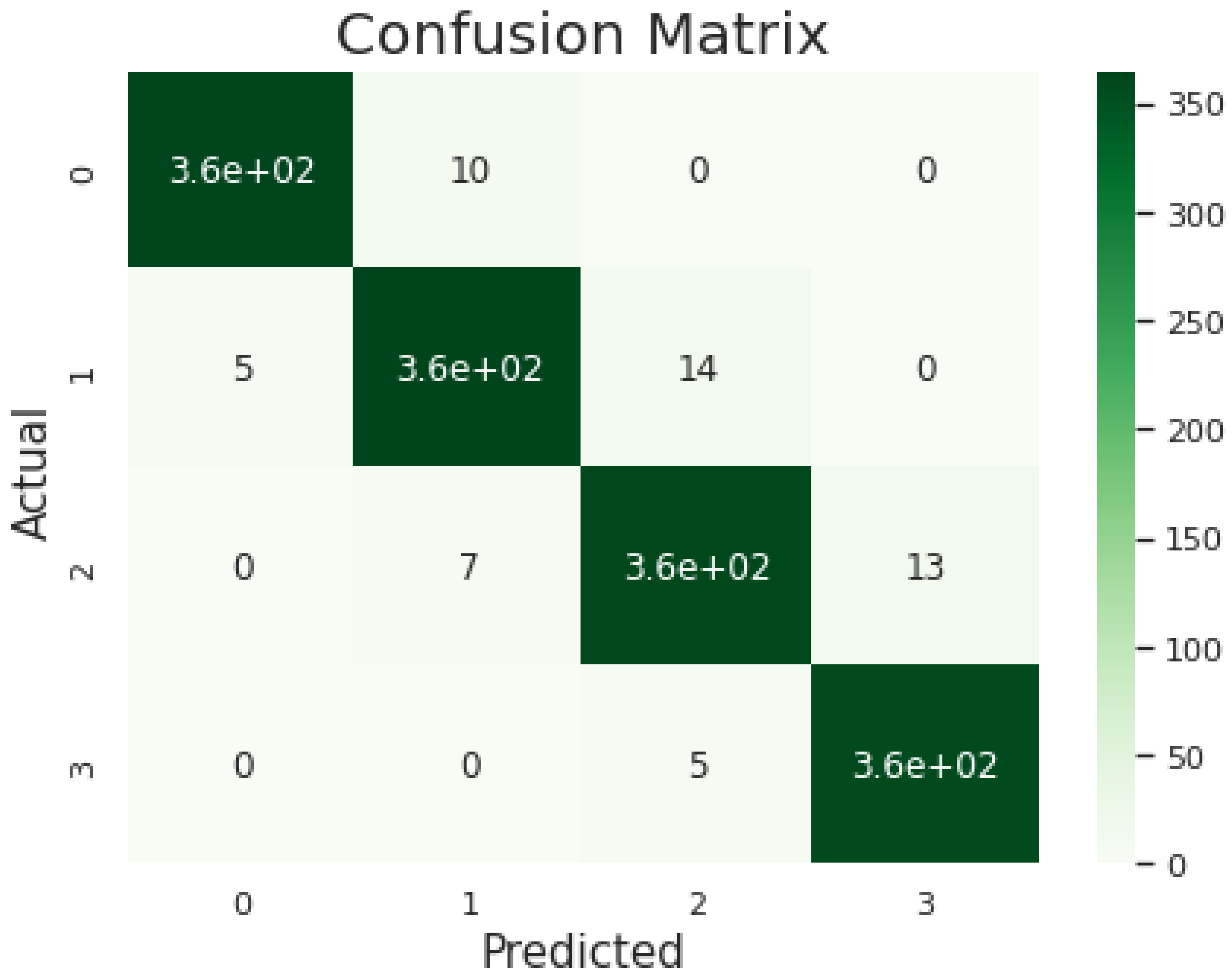
SVM Classification



```
The accuracy on train data is : 0.964
The accuracy on test data is : 0.862
```

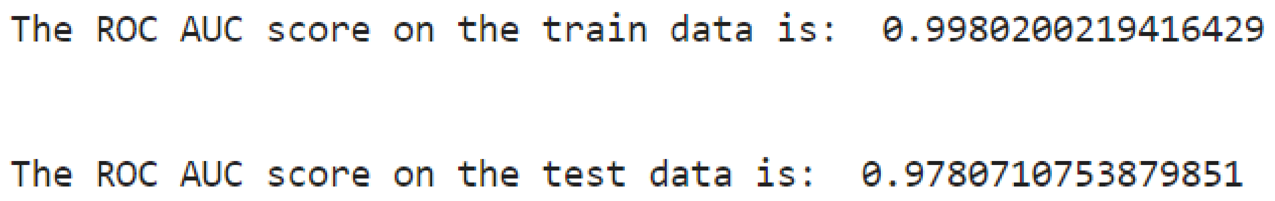


Confusion Matrix of Test Set

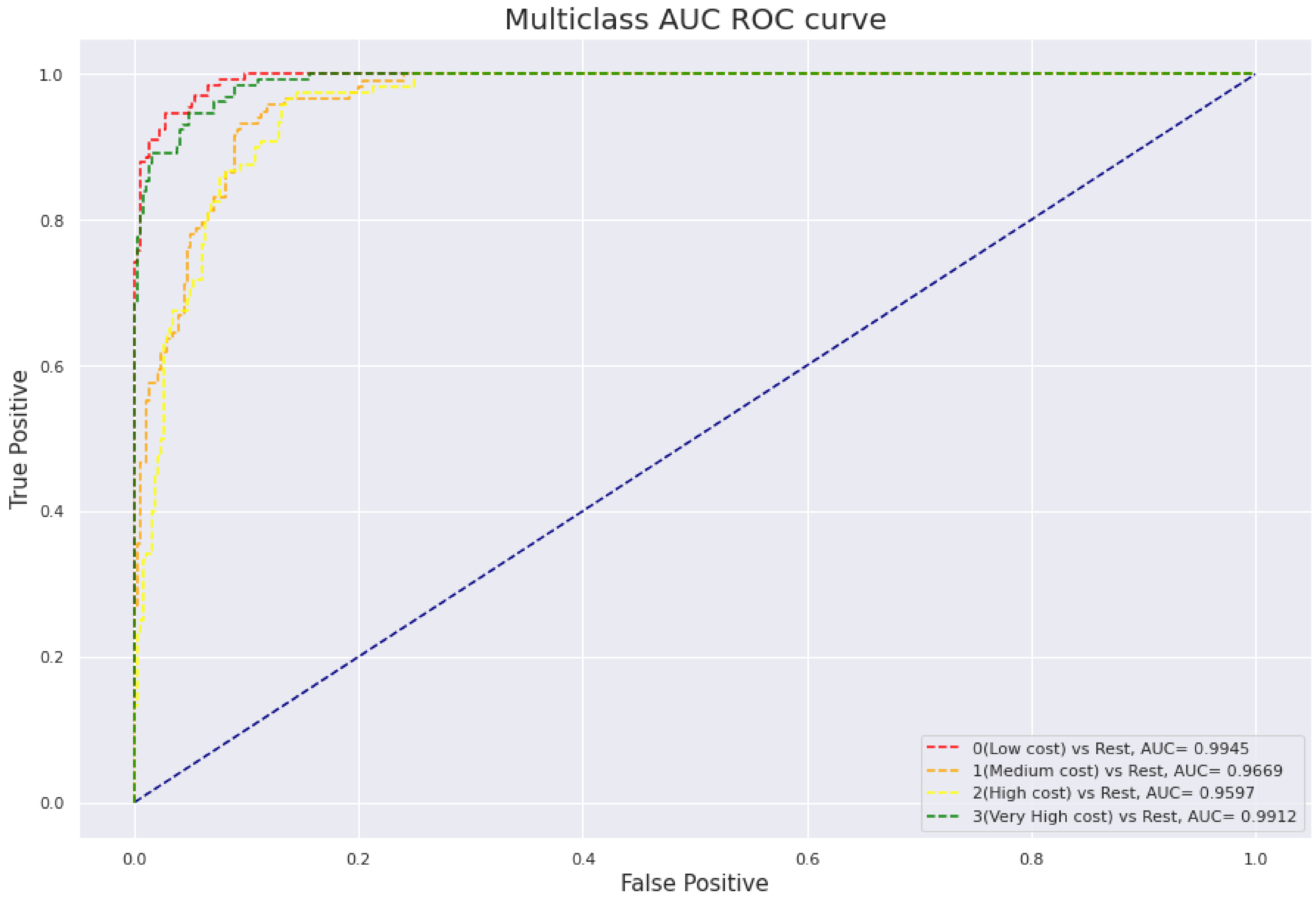


Confusion Matrix of Train Set

SVM Classification



```
The ROC AUC score on the train data is:  0.9980200219416429

The ROC AUC score on the test data is:  0.9780710753879851
```

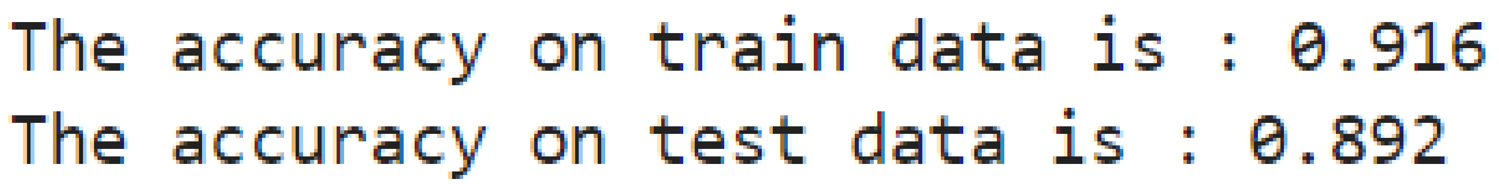


Observations

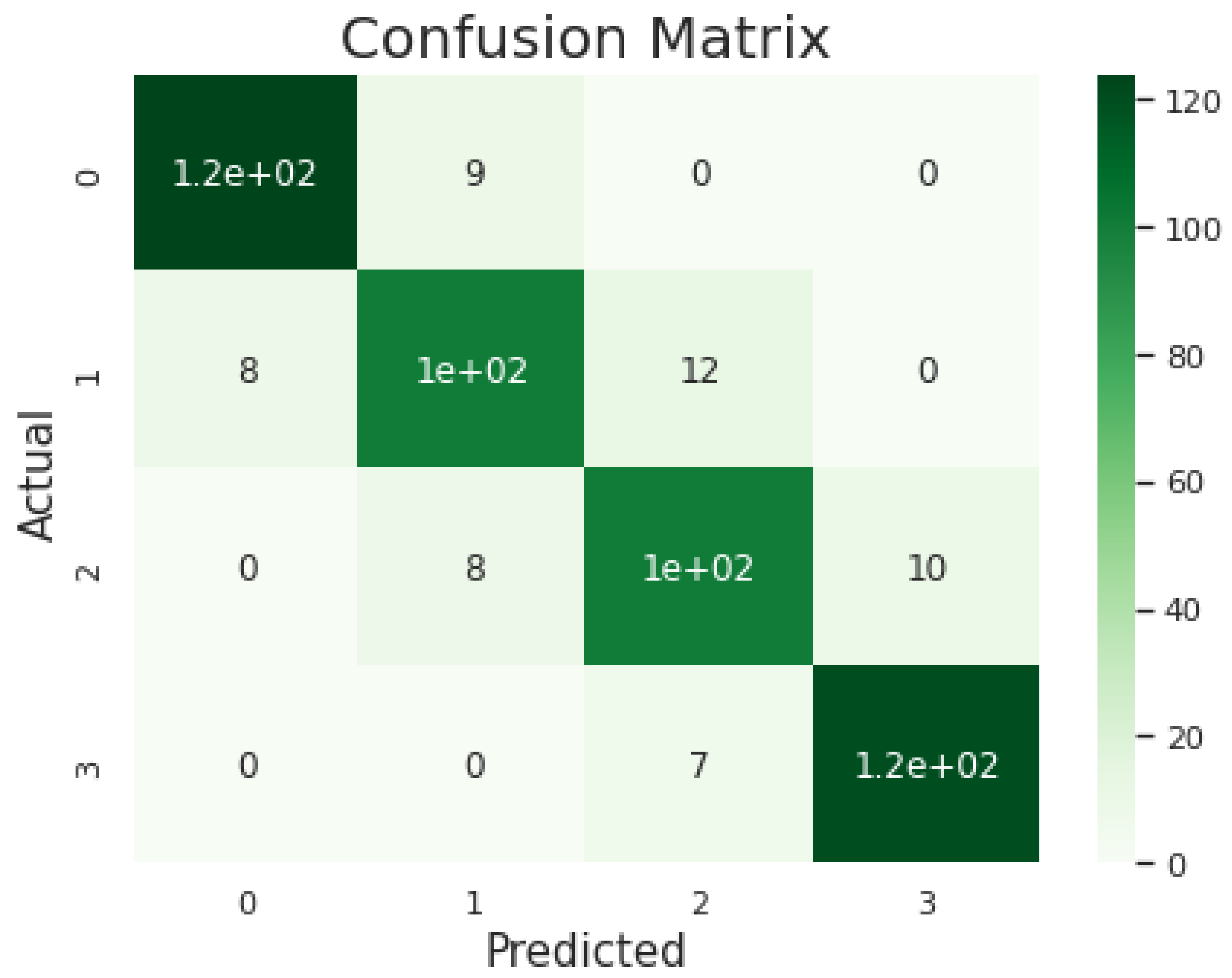
- Model is overfitting but can be fixed by optmisation.

SVM Classification

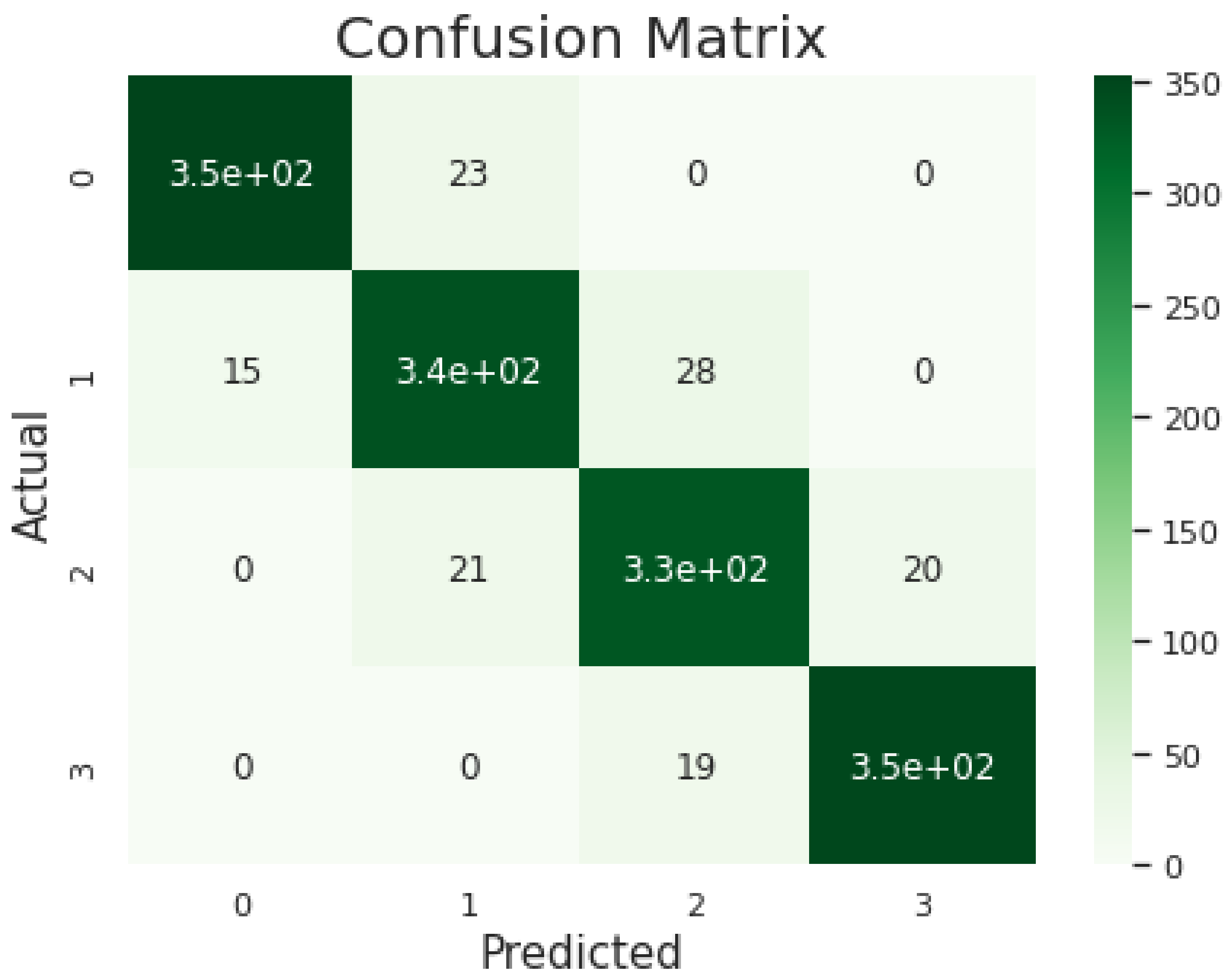
Hyper Parameter Tuning and Cross Validation of SVM Classification



```
The accuracy on train data is : 0.916
The accuracy on test data is : 0.892
```

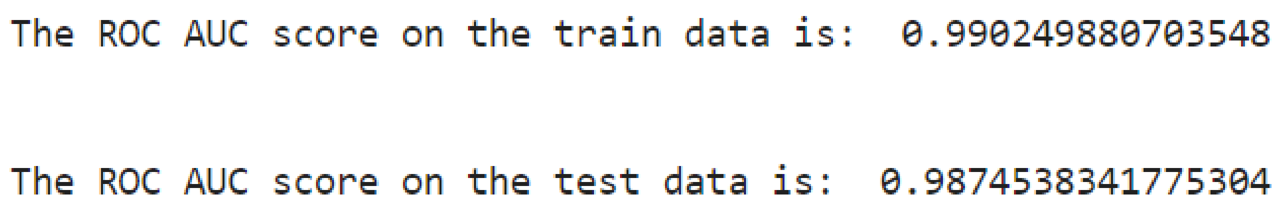


Confusion Matrix of Test Set

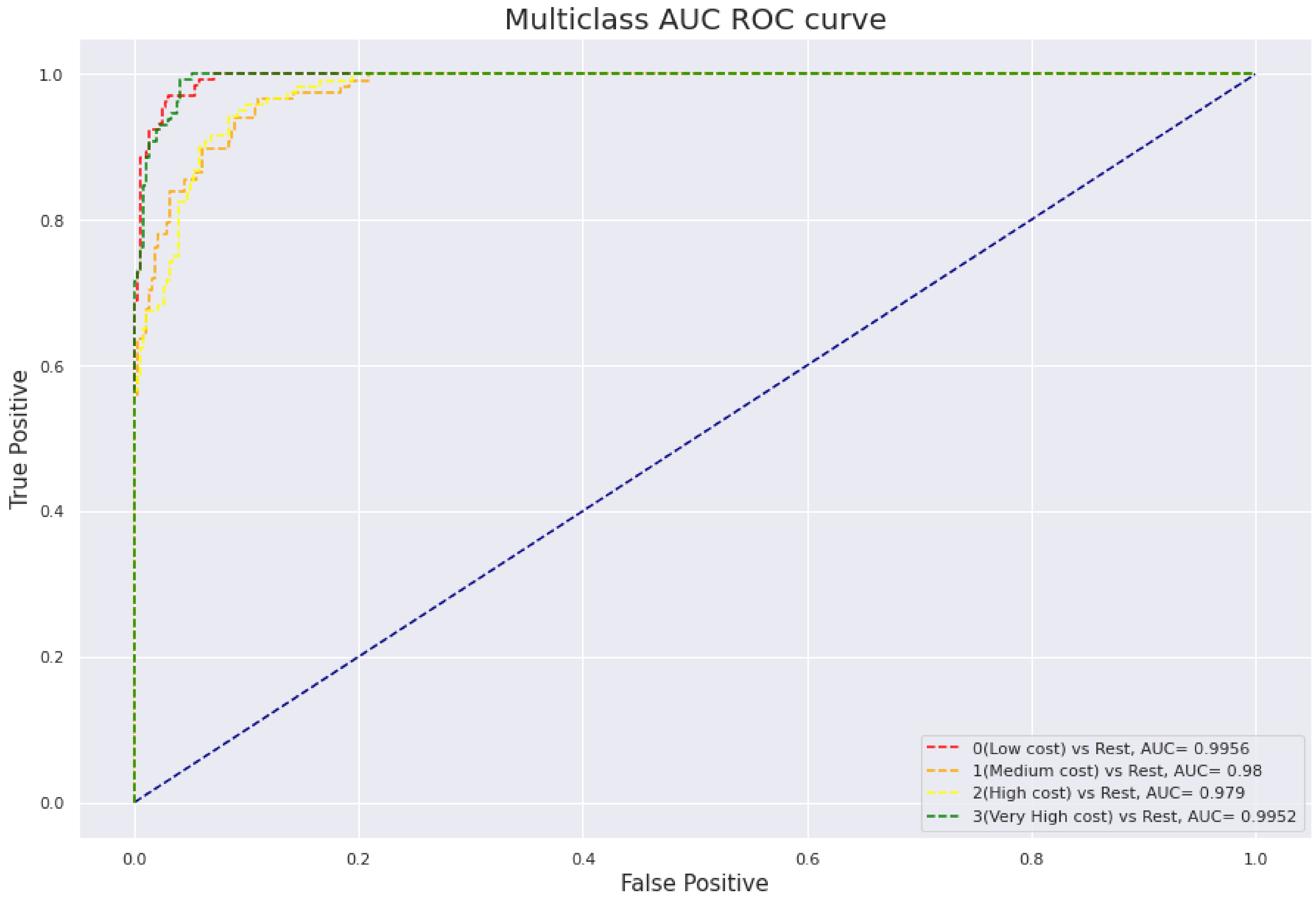


Confusion Matrix of Train Set

SVM Classification



```
The ROC AUC score on the train data is:  0.990249880703548


The ROC AUC score on the test data is:  0.9874538341775304
```



Observations

- Overfitting is reduced
- Prediction for all 4 price_range is good

Conclusion

- In EDA there were columns/features that were inter-related, we converted to new features using them.
- ram and batter_power has the highest impact on price_range.
- Using logistic regeression feature importance we observed that some of the columns were not relevant or had no impact negative/positive. Hence, they were dropped.
- Implemented various classification algorithms, Logistics and SVM accuracy was similar.
- Logistic regression classification model gave best results after hyper-parameter tuning with 91.5% train accuracy and 89.2% test accuracy score.
- SVM (Support vector machine) algorithm also gave equally best accuracy after hyper-parameter tuning with 91.6% train accuracy and 89.2 % test accuracy.
- Random Forest was Over-fitting
- KNN after optimization performed very well but for mutliclass price_range the prediction of price_range = 2 was lowered than Logistic and SVM in comparission.

We will go forward with Logistic regression classification model as using it increases the explainability of price_rage as per business requirement.

Thank You!