Capstone Project - 4

Supervised – ML – Mobile Price Range Classification

Sri harish A

Contents

Addressing the problem

The purpose of this project is to try a machine learning approach for **Mobile Price Range Classification** by given the mobile's processor, RAM, ROM and other physical details. This project contains: Exploratory data analysis, feature engineering, choosing appropriate features, cross algorithms, cross validation, tuning the algorithms, analysis of feature importance, analysis of model performance. Mobile phones come in all sorts of prices, features, specifications and all. Price estimation and prediction is an important part of consumer strategy. Deciding on the correct price of a product is very important for the market success of a product. A new product that has to be launched, must have the correct price so that consumers find it appropriate to buy the product.

Addressing the problem

How Mobile Price Range Classification helps?

In this Modern Era, Smartphones are an integral part of the lives of human beings. When a smartphone is purchased ,many factors like the Display, Processor, Memory, Camera, Thickness, Battery, Connectivity and others are taken into account . One factor that people do not consider is whether the product is worth the cost . As there are no resources to cross validate the price , people fail in taking the correct decision. This project looks to solve the problem by taking the historical data pertaining to the key features of smartphones along with its cost and develop a model that will predict the approximate price of the new smartphone with a reasonable accuracy.

Why Mobile Price Range Classification is important?

The price of a product is the most important attribute of marketing that product. One of those products where price matters a lot is a smartphone because it comes with a lot of features so that a company thinks a lot about how to price this mobile which can justify the features and also cover the marketing and manufacturing costs of the mobile.

Mobile phones are the best selling electronic devices as people keep updating their cell phones whenever they find new features in a new device. Thousands of mobiles are sold daily, in such a situation it is a very difficult task for someone who is planning to set up their own mobile phone business to decide what the price of the mobile should be.

since our task is to classify the price range of mobile phones and not to predict the actual prices

Features 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
- **Mobile_wt** - Weight of mobile phone
- **N_cores** - Number of cores of processor

Features Summary

Independent variables:

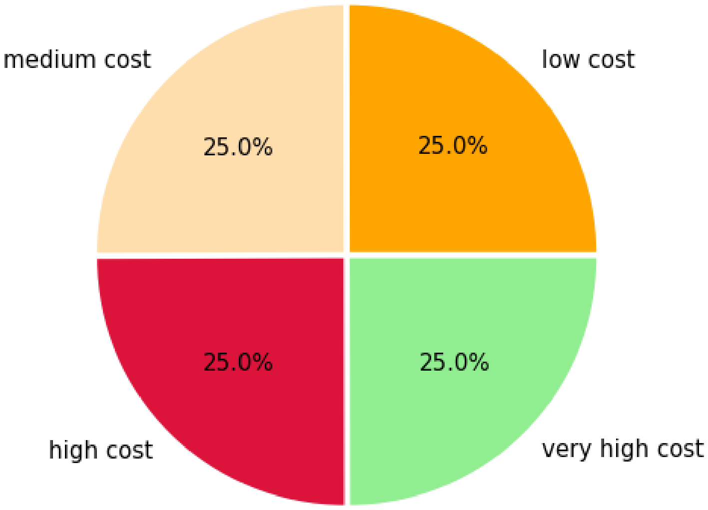
- **Pc** - Primary Camera mega pixels
- **Px_height** - Pixel Resolution Height
- **Px_width** - Pixel Resolution Width
- **Ram** - Random Access Memory in Mega Bytes
- **Sc_h** - Screen Height of mobile in cm
- **Sc_w** - Screen Width of mobile in cm
- **Talk_time** - longest time that a single battery charge will last when you are
- **Three_g** - Has 3G or not
- **Touch_screen** - Has touch screen or not
- **Wifi** - Has wifi or not

Features Summary (continued)

Dependent variable:

- **Price_range** - This is the target variable with value of
- 0(low cost),
- 1(medium cost),
- 2(high cost) and
- 3(very high cost).

Class distribution



Observations :

- we have almost equal number of observations for each category.
- Thus we don't have imbalanced target variable.
- Thus we don't have to worry about data imbalance and there is no need of oversampling or under sampling.
- Accuracy score will be the best evaluation metric for us to select the model.

Outliers

- An outlier is an **extremely high or extremely low data point** relative to the nearest data point and the rest of the neighboring co-existing values in a data graph or dataset you're working with.
- **Ways to detect outliers:**
- Interquartile range
- Box plot
- Scatter plot
- Z – score
- In the given dataset we have used Box plot to detect outliers.

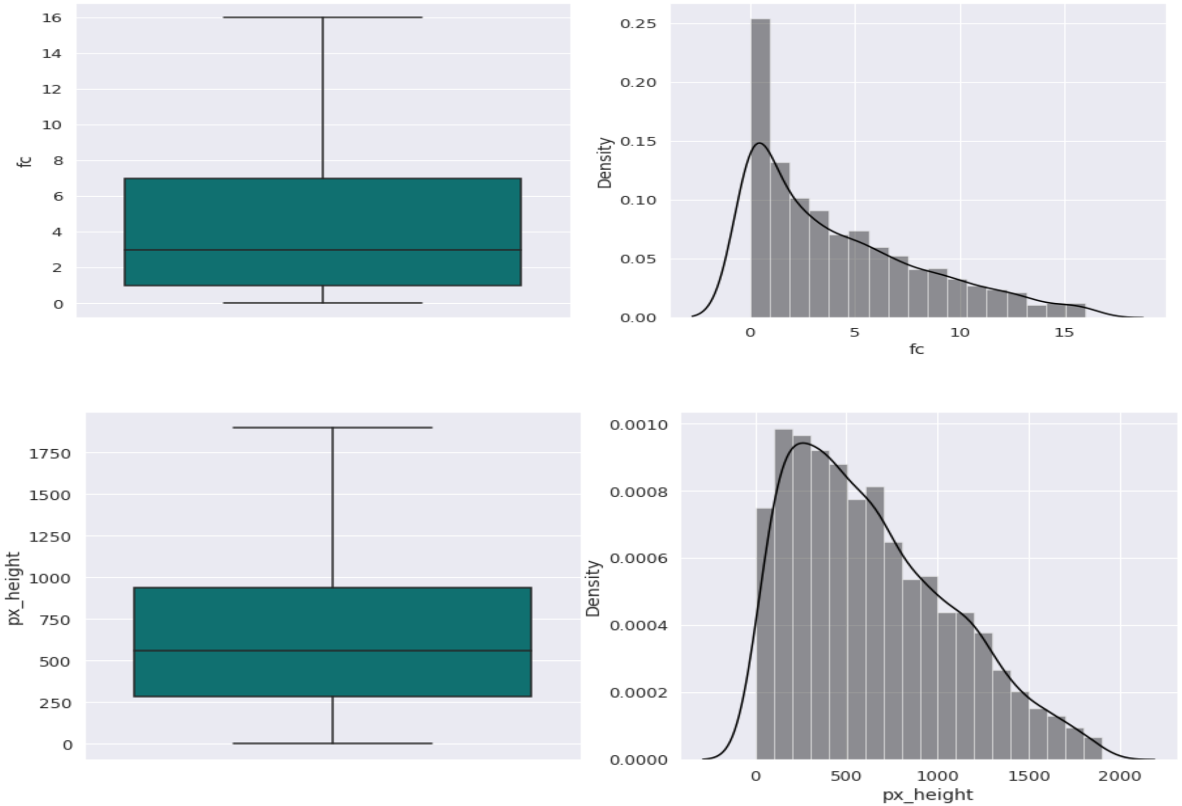
Outliers (continued)

IQR – To handle outliers.

```
# Outlier detection of fc column
Q1 = df["fc"].quantile(0)
Q3 = df['fc'].quantile(0.992)
IQR = Q3-Q1
# Outliers are present after Quartile 3. so we will take datapoints before Q3.
df = df[(df['fc'] <= Q3)]
```

```
#Outlier detection of px_height column
Q1 = df["px_height"].quantile(0)
Q3 = df['px_height'].quantile(0.999)
IQR = Q3-Q1
# Outliers are present after Quartile 3. so we will take datapoints before Q3.
df = df[(df['px_height'] <= Q3)]
```

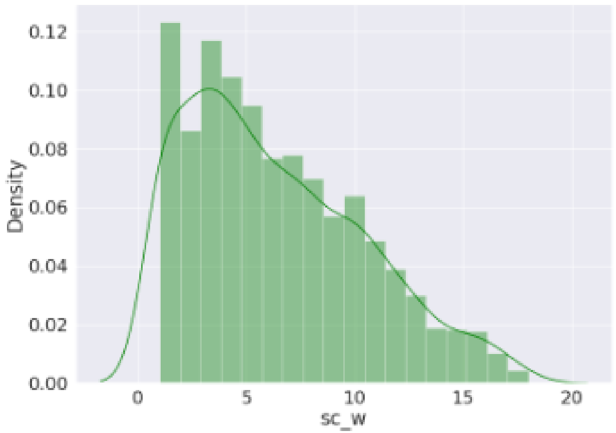
After outlier treatment



KNN-IMPUTATION

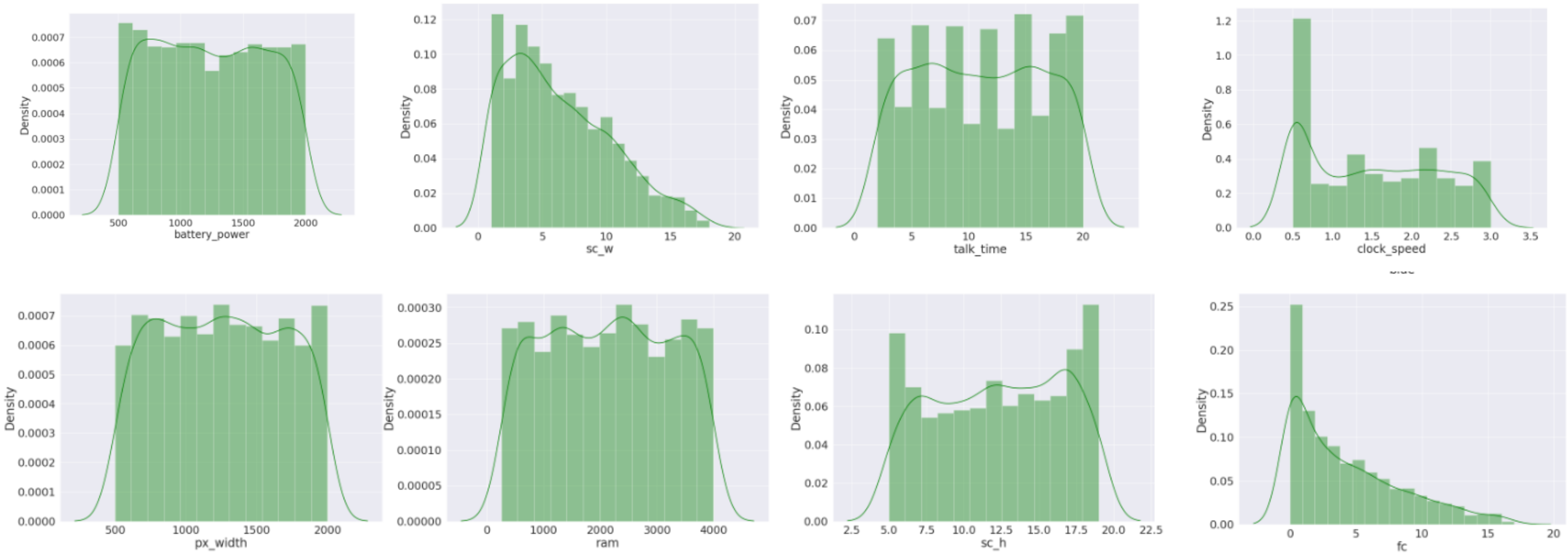
```
# Replacing 0 with NAN so that we can implement KNN Imputer.
df['sc_w']=df['sc_w'].replace(0,np.nan)
```

```
# import KNN imputer frio sklearn
# Missing values are imputed using the k-Nearest Neighbors approach where a Euclidean distance is used to find the nearest neighbors
from sklearn.impute import KNNImputer
impute_knn = KNNImputer(n_neighbors=1)
df=pd.DataFrame(impute_knn.fit_transform(df),columns=df.columns)
# The mismatched values has been imputed
```



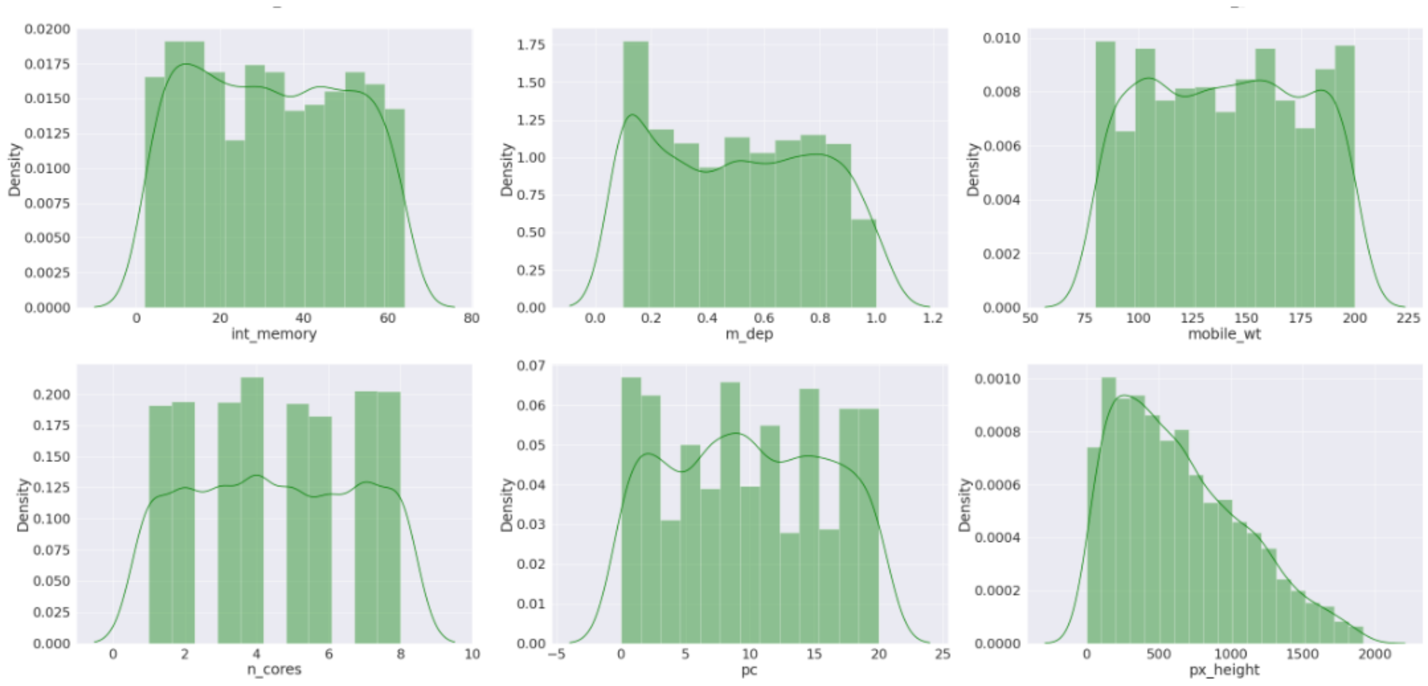
Exploratory Data Analysis

Distribution plot for independent variables.



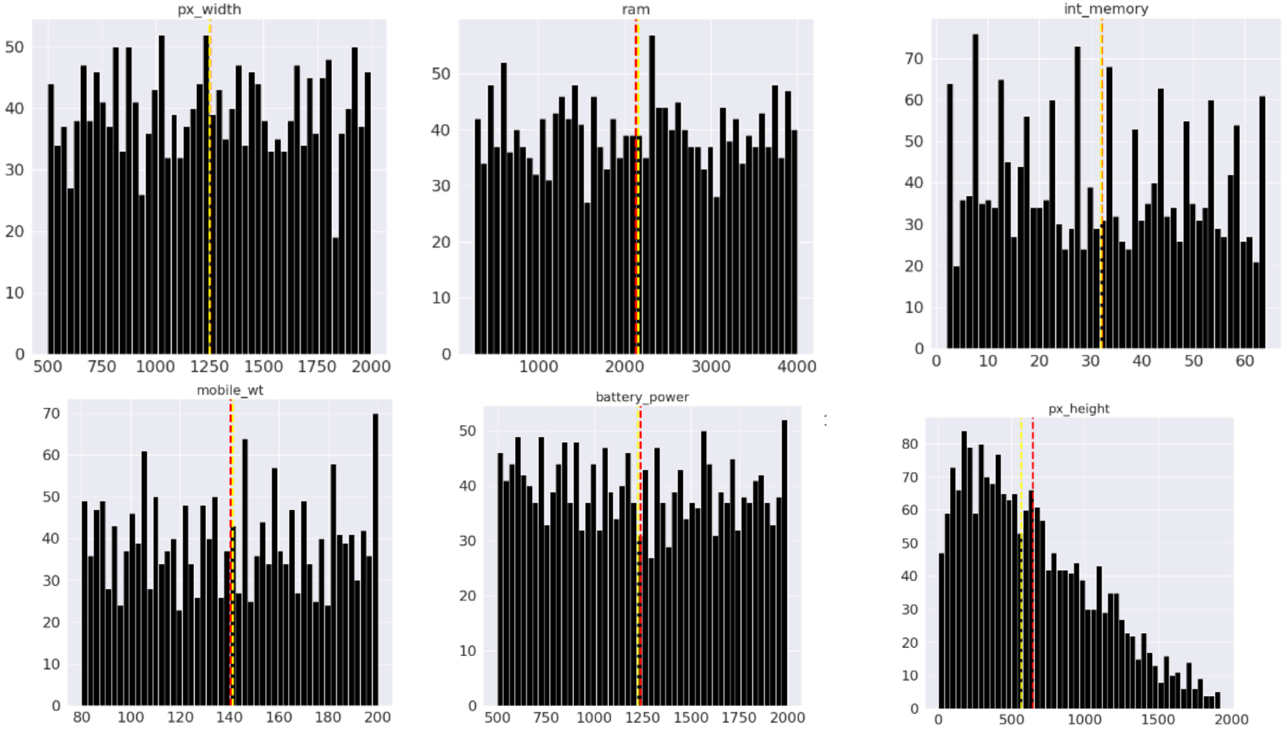
Exploratory Data Analysis

Distribution plot for independent variables.



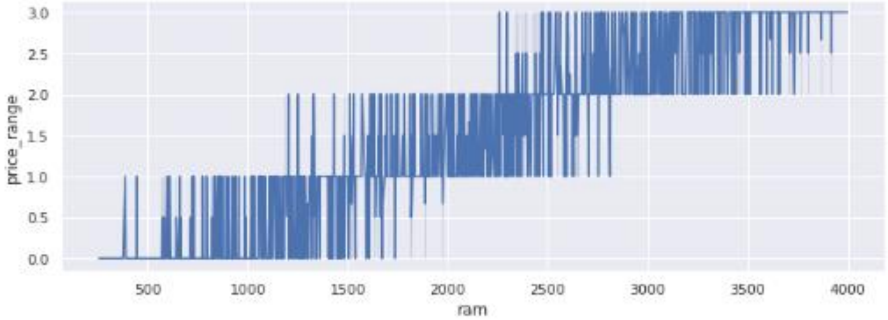
Exploratory Data Analysis

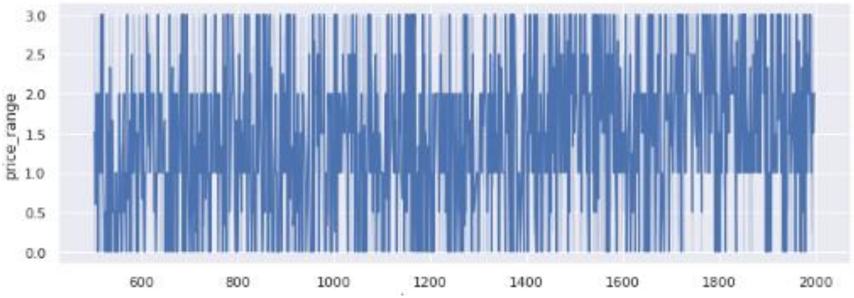
Mean, Median plot for independent variables.

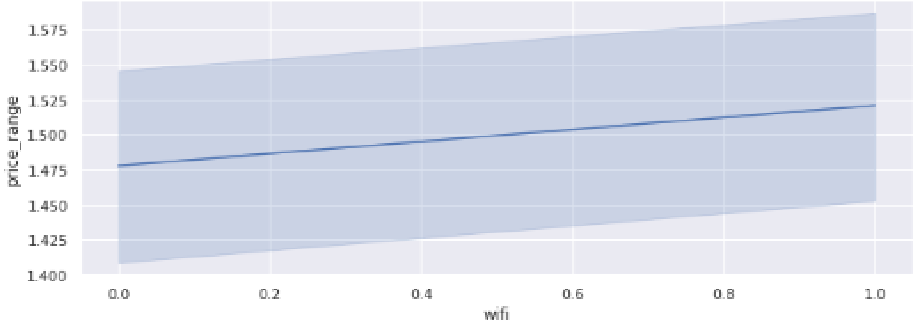


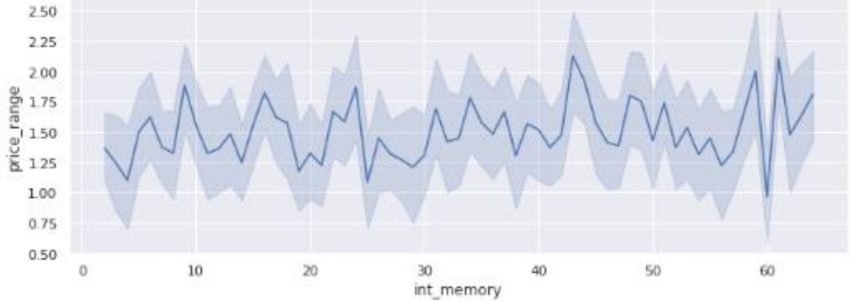
Exploratory Data Analysis

Line plot for independent numeric variables.



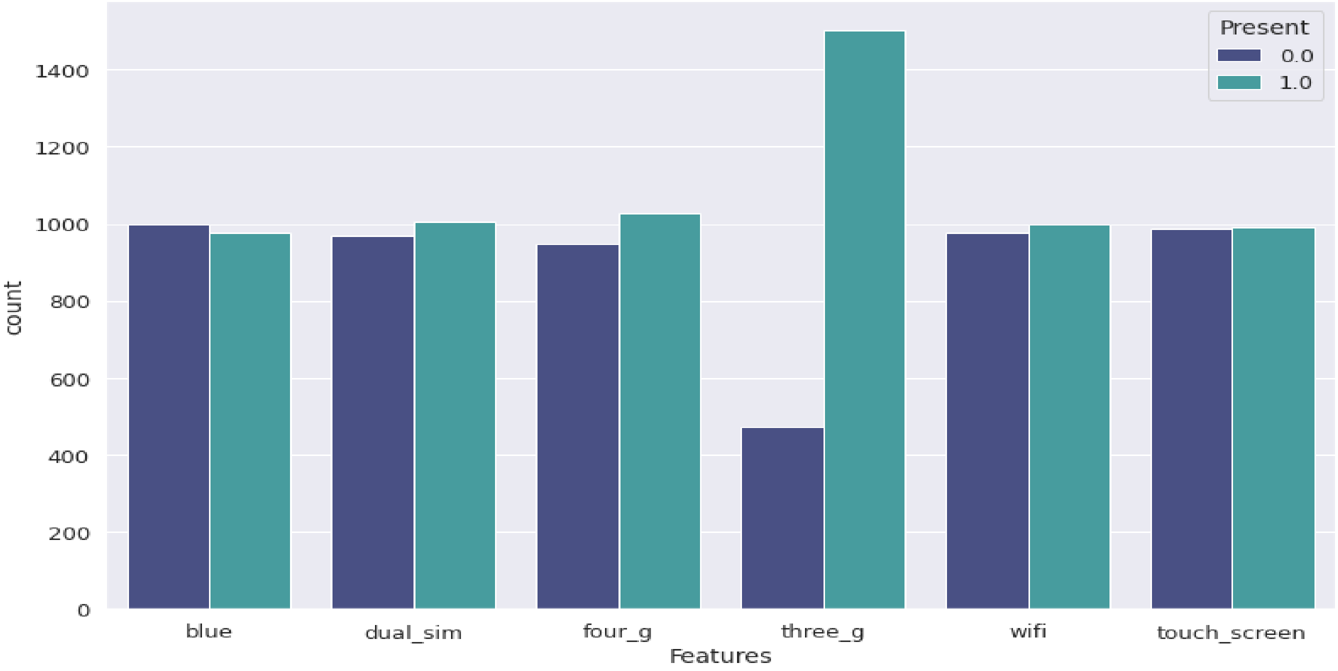






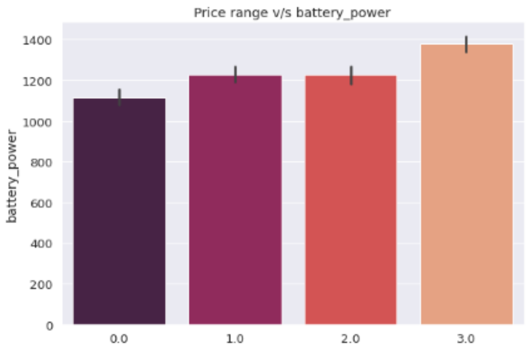
Exploratory Data Analysis (continued)

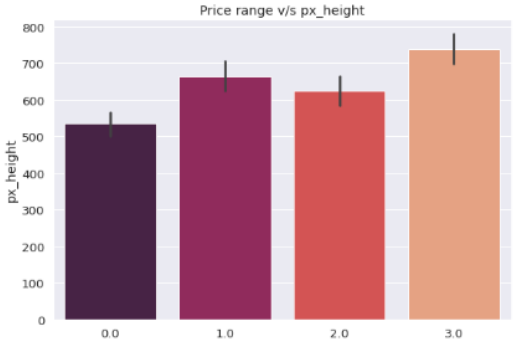
Count plot for all the binary categorical variables.

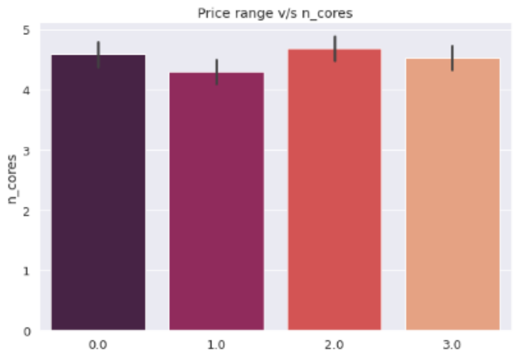


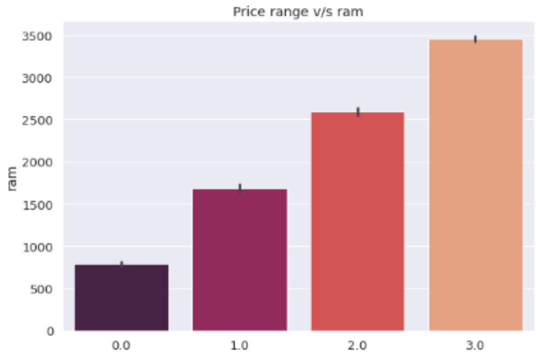
Exploratory Data Analysis (continued)

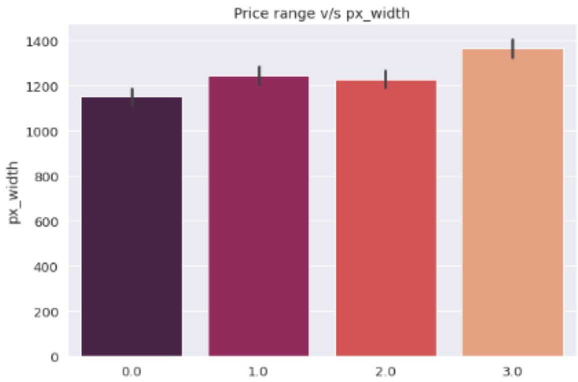
Price range vs Numerical features

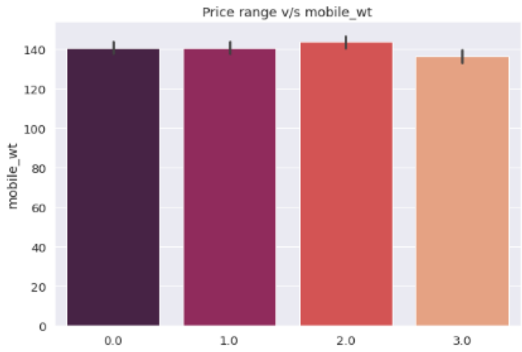






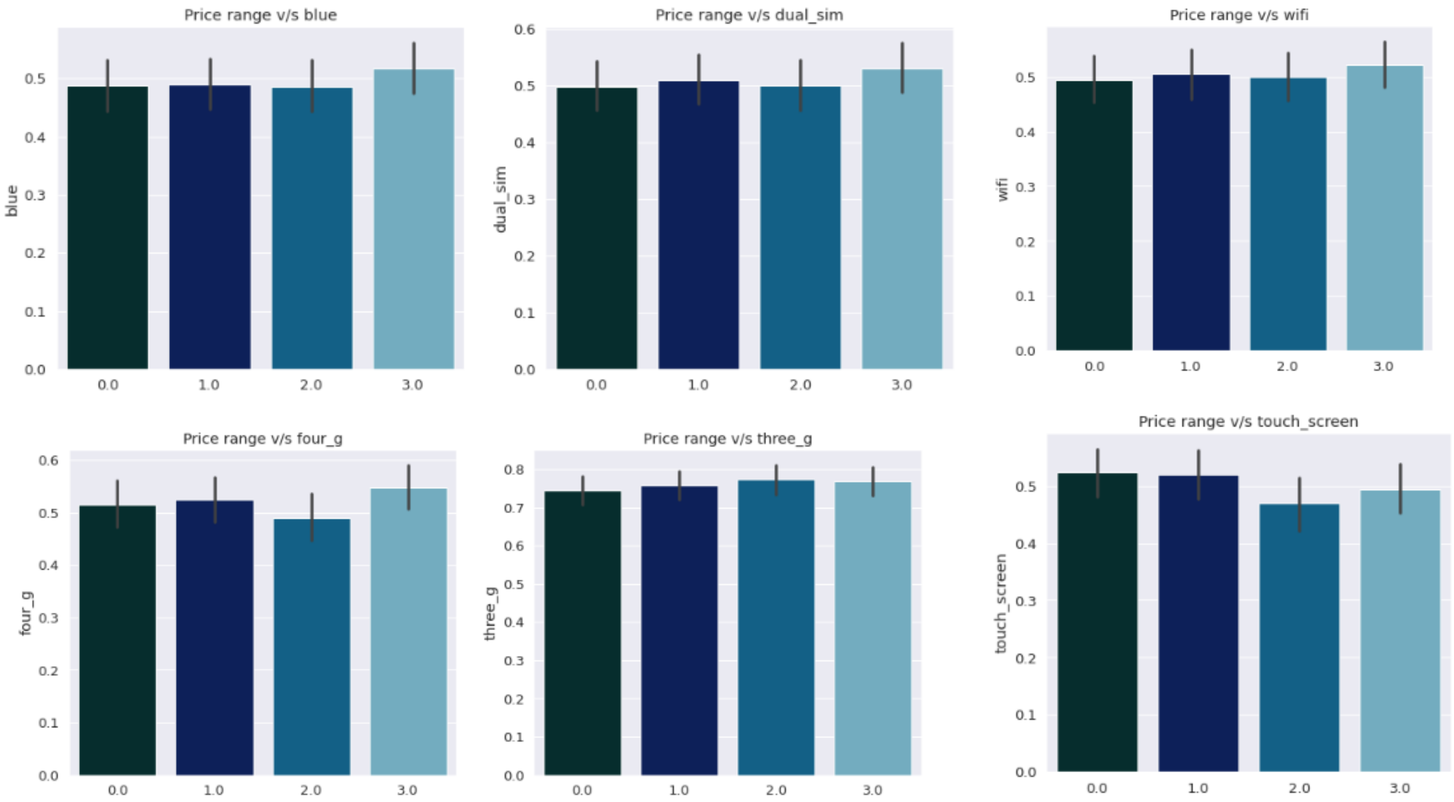




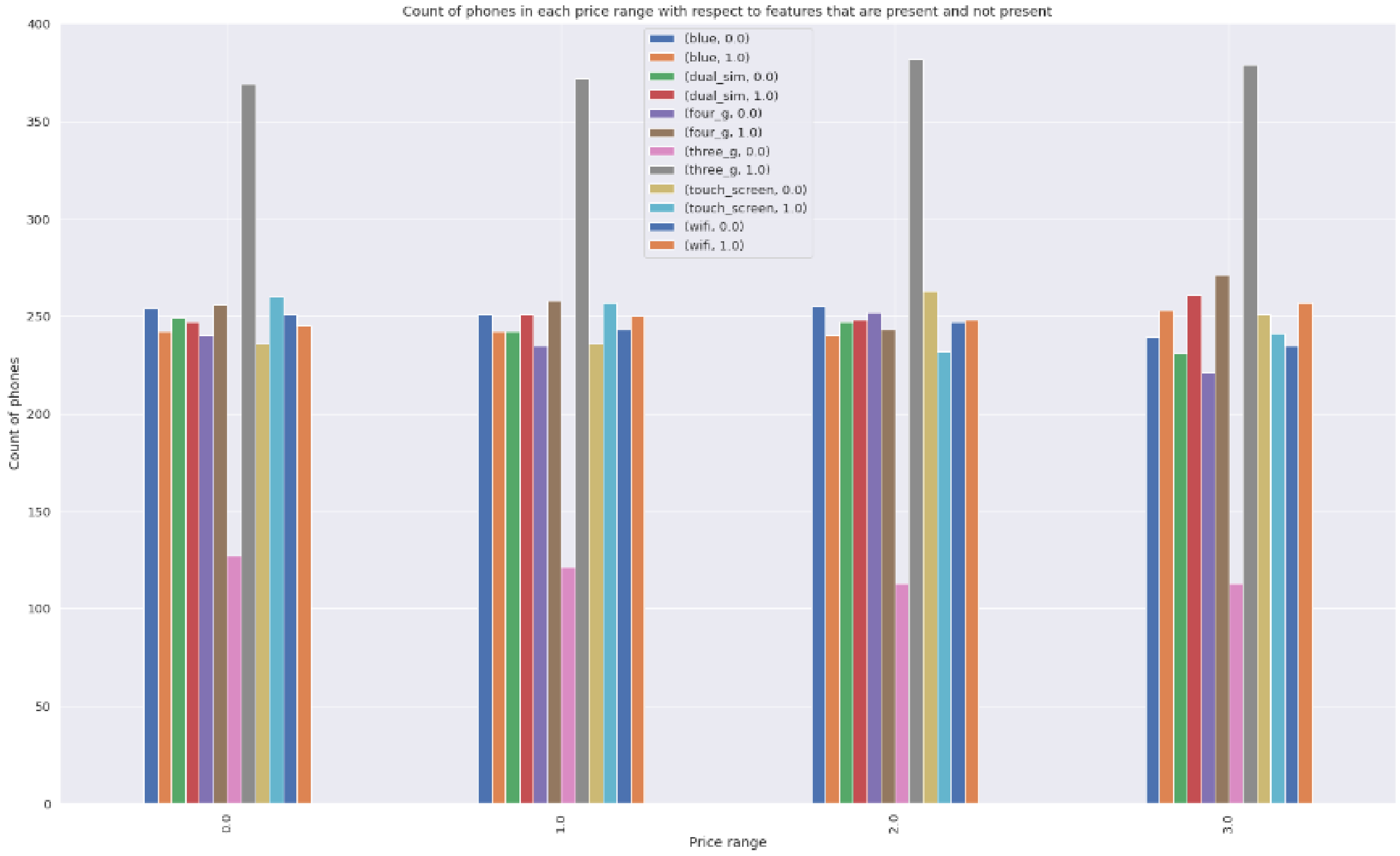


Exploratory Data Analysis (continued)

Price range vs Categorical feature

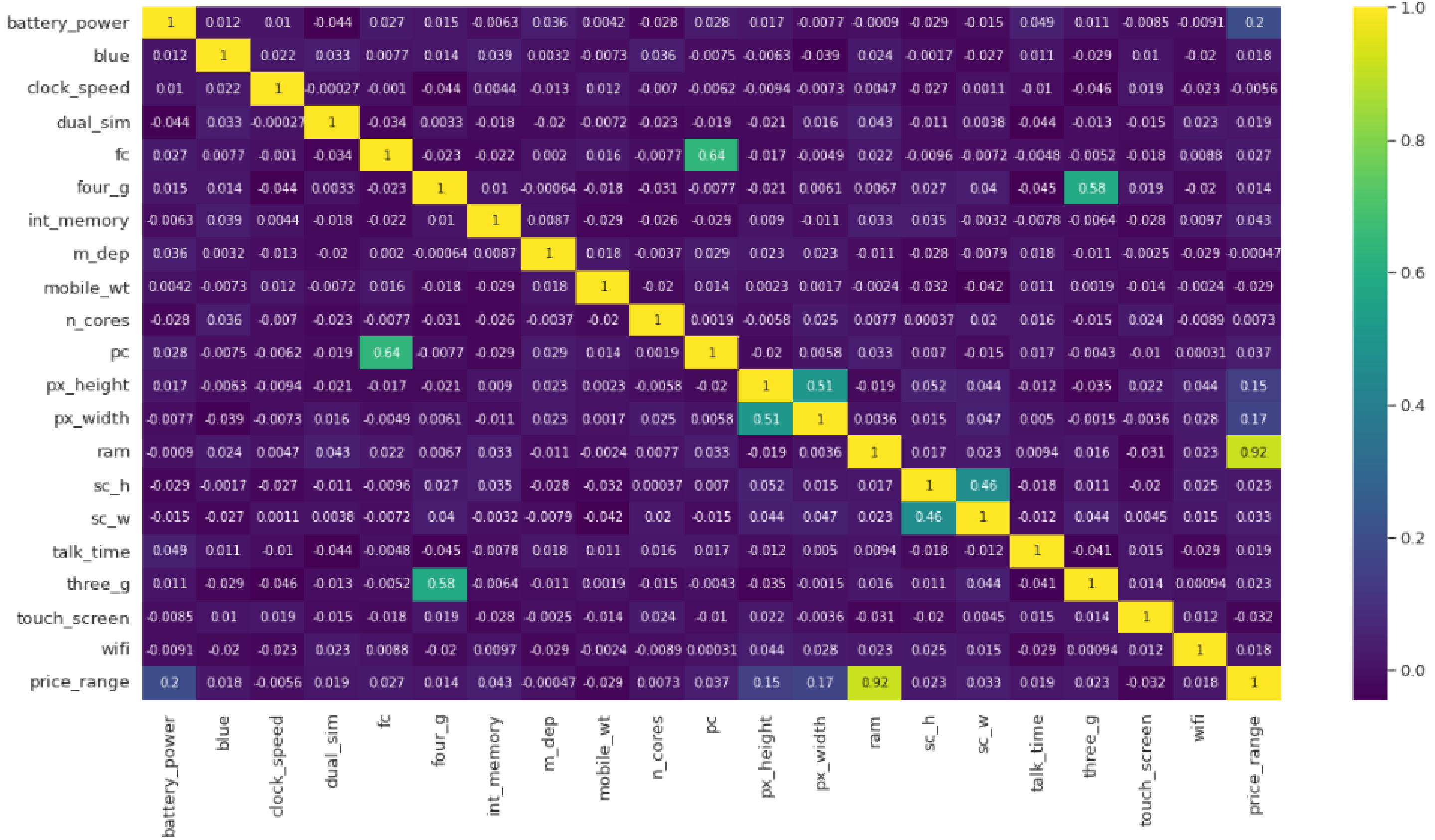


Count plot for feature present in each price range



Exploratory Data Analysis (continued)

Multicollinearity



Feature Engineering

Feature Selection

Selecting top most important feature

```
#Importing SelectKBest and chi2 for feature selection
from sklearn.feature_selection import SelectKBest
from sklearn.feature_selection import chi2
```

```
#Now we will select top 10 important features
bestfeatures = SelectKBest(score_func=chi2, k=10)
fit = bestfeatures.fit(X,y)
```

```
# creating dataframe for storing scores and column names
dfscores = pd.DataFrame(fit.scores_)
dfcolumns = pd.DataFrame(X.columns)
```

```
# concatenating the above two dataframes
featureScores = pd.concat([dfcolumns,dfscores],axis=1)
featureScores.columns = ['Specs','Score']
```

```
# Chi2 score of all the features in the datset
featureScores
```

	Specs	Score
0	battery_power	13782.770600
1	blue	0.537283
2	clock_speed	0.759809
3	dual_sim	0.624434
4	fc	12.413900
5	four_g	1.727665
6	int_memory	81.307799
7	m_dep	0.728500
8	mobile_wt	95.177627
9	n_cores	8.944628
10	pc	9.249370
11	px_height	16615.320424
12	px_width	9517.591468
13	ram	920045.447456
14	sc_h	10.437586
15	sc_w	9.786638
16	talk_time	11.395628
17	three_g	0.345598
18	touch_screen	2.091957
19	wifi	0.427920

Preparing dataset for modelling

Train Test Split

Train test split is a model validation procedure that allows you to simulate how a model would perform on new/unseen data.

```
X = data[independent_variables].values
y = data[dependent_variable].values
```

```
# Splitting the dataset into the Training set and Test set
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.25, random_state = 0)
```

- X_train: (2523, 15)
- X_test : (842, 15)
- y_train : (2523, 1)
- y_test : (842, 1)

Preparing dataset for modelling

Standardization

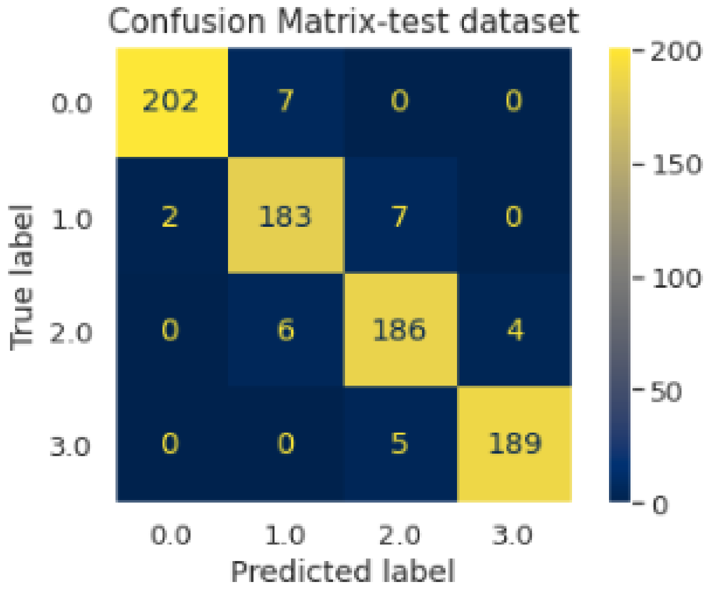
Standardizing a dataset involves rescaling the distribution of values so that the mean of observed values is 0 and the standard deviation is 1.

```
# Transforming data

# Standardization
scaler = MinMaxScaler()
X_train = scaler.fit_transform(X_train)
X_test= scaler.transform(X_test)
```

Applying Model

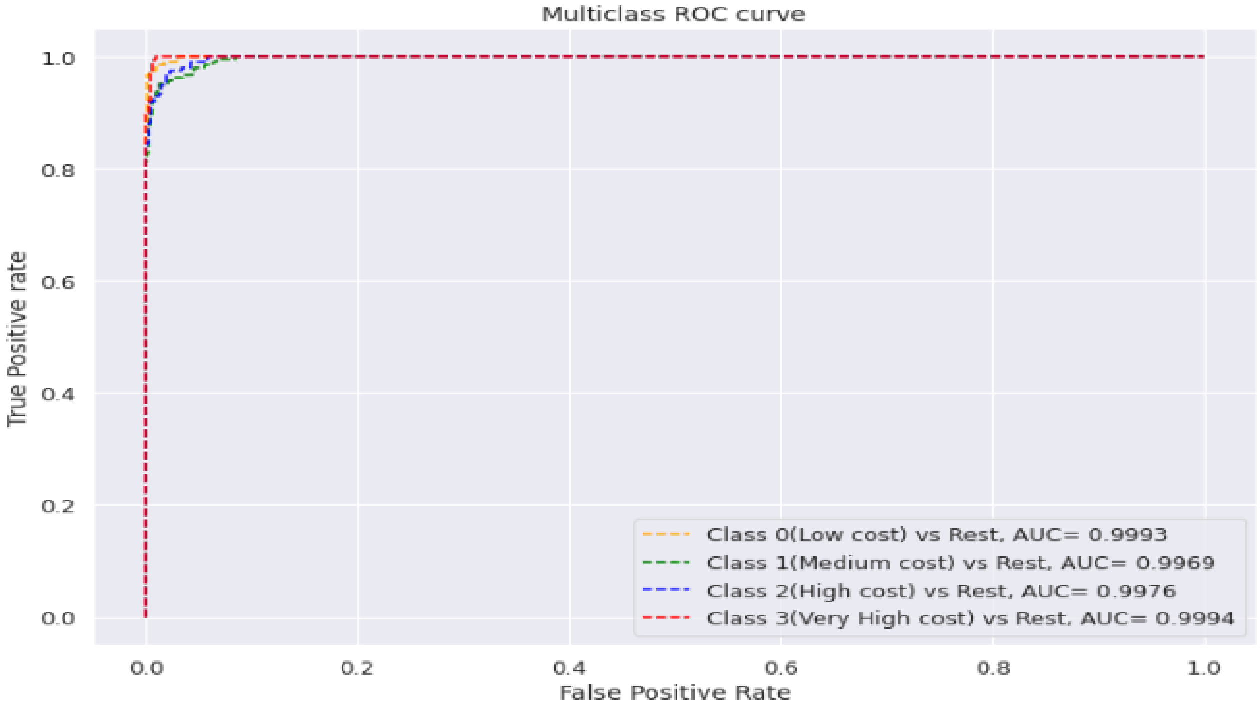
Logistic Regression

```
Train set accuracy score of is 0.9729957805907173
Test set accuracy score of  is 0.9608091024020228

Classification report

              precision    recall  f1-score   support

         0.0       0.98      0.99      0.98       287
         1.0       0.97      0.96      0.96       301
         2.0       0.96      0.96      0.96       299
         3.0       0.99      0.98      0.98       298

    accuracy                           0.97      1185
   macro avg       0.97      0.97      0.97      1185
weighted avg       0.97      0.97      0.97      1185

              precision    recall  f1-score   support

         0.0       0.99      0.97      0.98       209
         1.0       0.93      0.95      0.94       192
         2.0       0.94      0.95      0.94       196
         3.0       0.98      0.97      0.98       194

    accuracy                           0.96       791
   macro avg       0.96      0.96      0.96       791
weighted avg       0.96      0.96      0.96       791
```



Applying Model

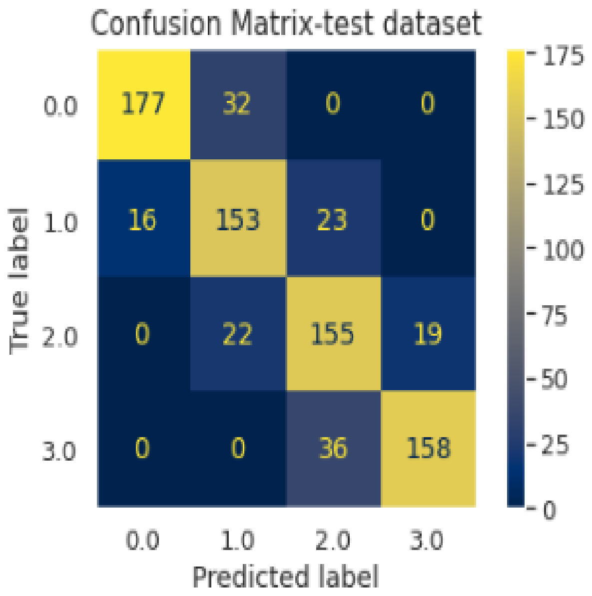
Model Validation(Logistic regression)



Applying Model

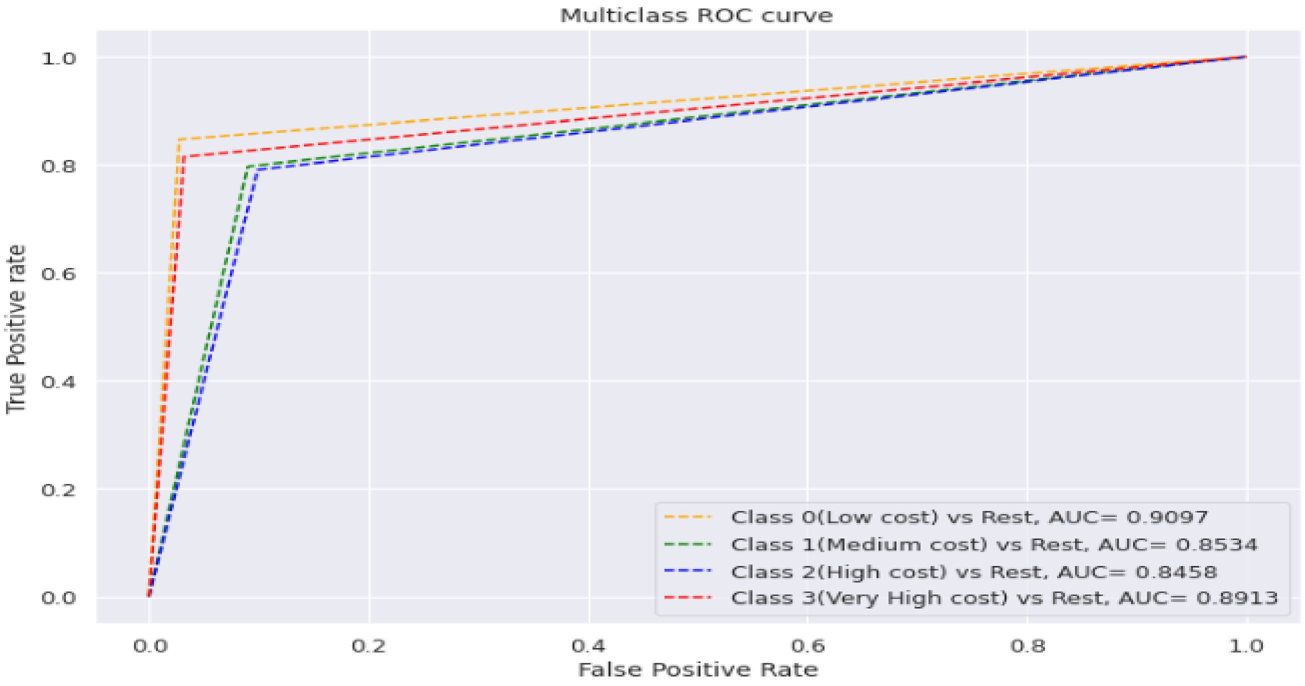
Decision Tree Classifier

```
Train set accuracy score of is 1.0
Test set accuracy score of  is 0.8128950695322377

Classification report

              precision    recall  f1-score   support

         0.0       1.00      1.00      1.00       287
         1.0       1.00      1.00      1.00       301
         2.0       1.00      1.00      1.00       299
         3.0       1.00      1.00      1.00       298

    accuracy                           1.00      1185
   macro avg       1.00      1.00      1.00      1185
weighted avg       1.00      1.00      1.00      1185
              precision    recall  f1-score   support

         0.0       0.92      0.85      0.88       209
         1.0       0.74      0.80      0.77       192
         2.0       0.72      0.79      0.76       196
         3.0       0.89      0.81      0.85       194

    accuracy                           0.81       791
   macro avg       0.82      0.81      0.81       791
weighted avg       0.82      0.81      0.82       791
```



Applying Model

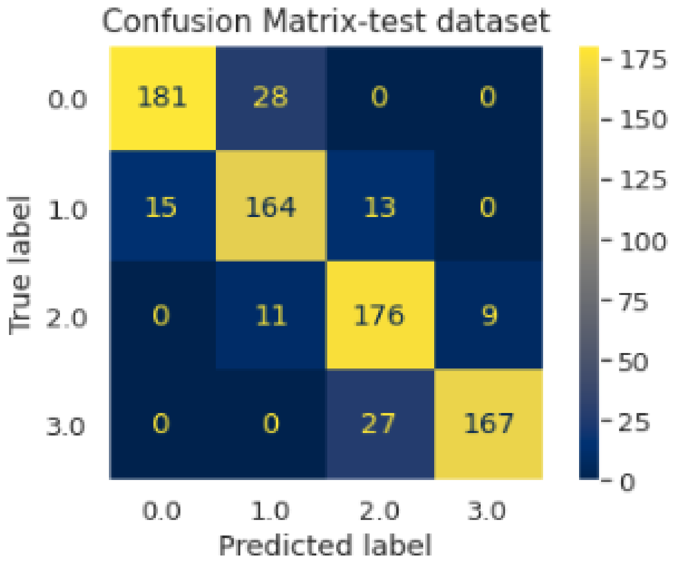
Decision Tree Classifier



Applying Model

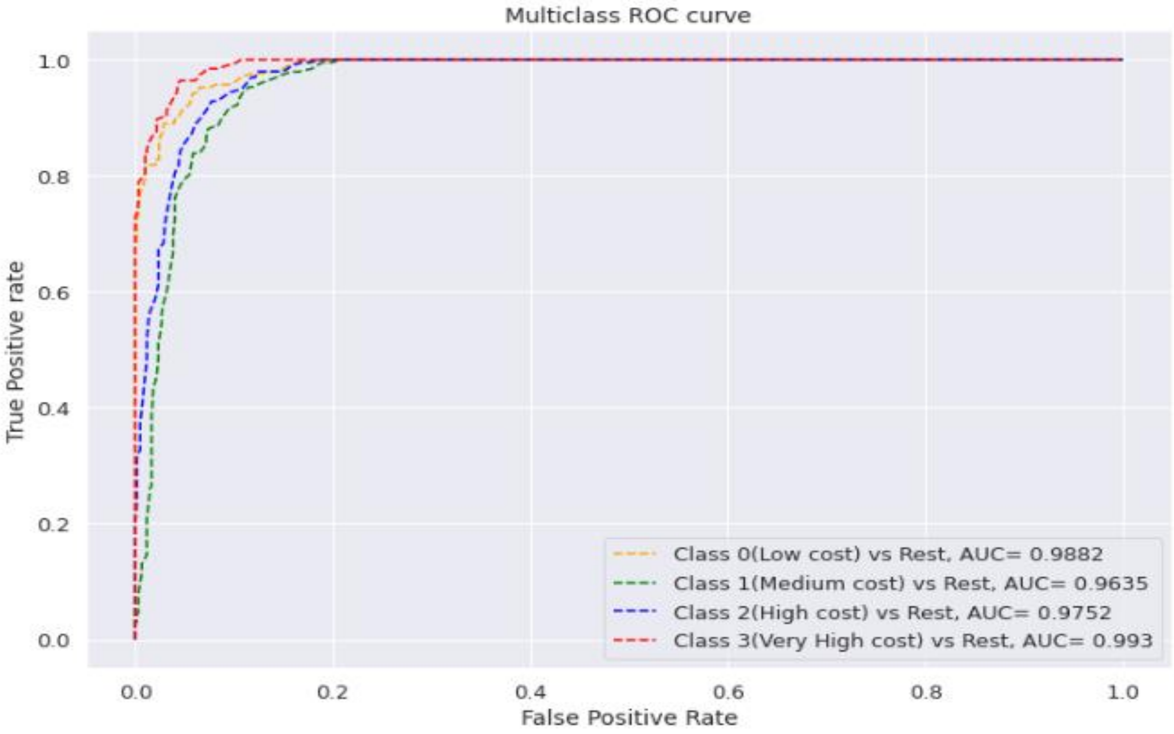
Random Forest Classifier

```
Train set accuracy score of is 1.0
Test set accuracy score of  is 0.8697850821744627

Classification report

              precision    recall  f1-score   support

         0.0       1.00      1.00      1.00       287
         1.0       1.00      1.00      1.00       301
         2.0       1.00      1.00      1.00       299
         3.0       1.00      1.00      1.00       298

    accuracy                           1.00      1185
   macro avg       1.00      1.00      1.00      1185
weighted avg       1.00      1.00      1.00      1185

              precision    recall  f1-score   support

         0.0       0.92      0.87      0.89       209
         1.0       0.81      0.85      0.83       192
         2.0       0.81      0.90      0.85       196
         3.0       0.95      0.86      0.90       194

    accuracy                           0.87       791
   macro avg       0.87      0.87      0.87       791
weighted avg       0.87      0.87      0.87       791
```



Applying Model

Random Forest Classifier



Applying Model

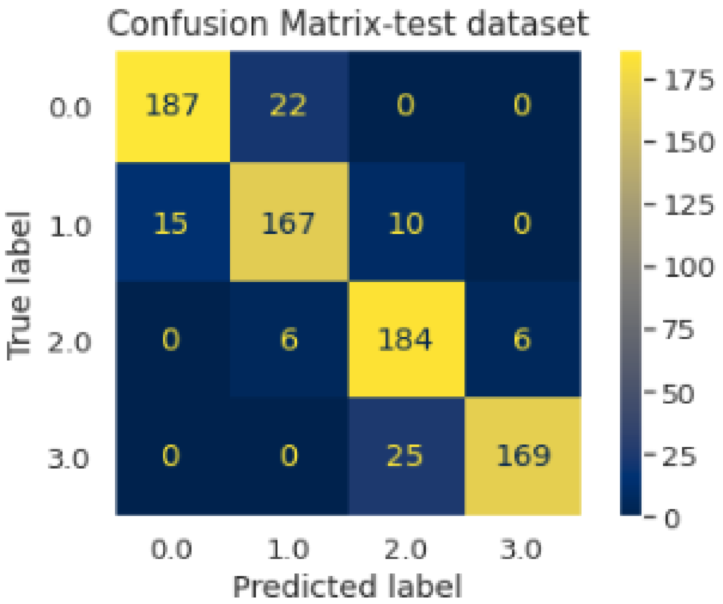
Gradient Boost Classifier

Train set accuracy score of is 1.0
Test set accuracy score of is 0.8938053097345132

Classification report

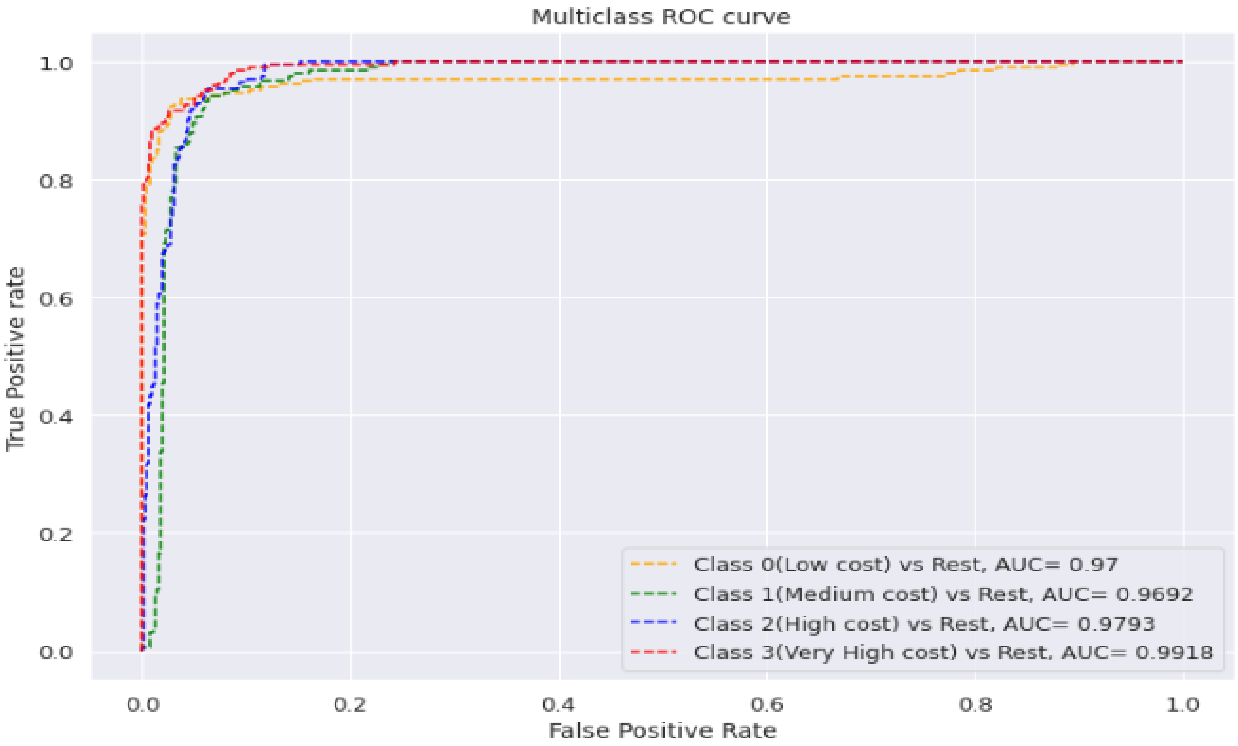
	precision	recall	f1-score	support
0.0	1.00	1.00	1.00	287
1.0	1.00	1.00	1.00	301
2.0	1.00	1.00	1.00	299
3.0	1.00	1.00	1.00	298
accuracy			1.00	1185
macro avg	1.00	1.00	1.00	1185
weighted avg	1.00	1.00	1.00	1185

	precision	recall	f1-score	support
0.0	0.93	0.89	0.91	209
1.0	0.86	0.87	0.86	192
2.0	0.84	0.94	0.89	196
3.0	0.97	0.87	0.92	194
accuracy			0.89	791
macro avg	0.90	0.89	0.89	791
weighted avg	0.90	0.89	0.89	791



Applying Model

Gradient Boost Classifier

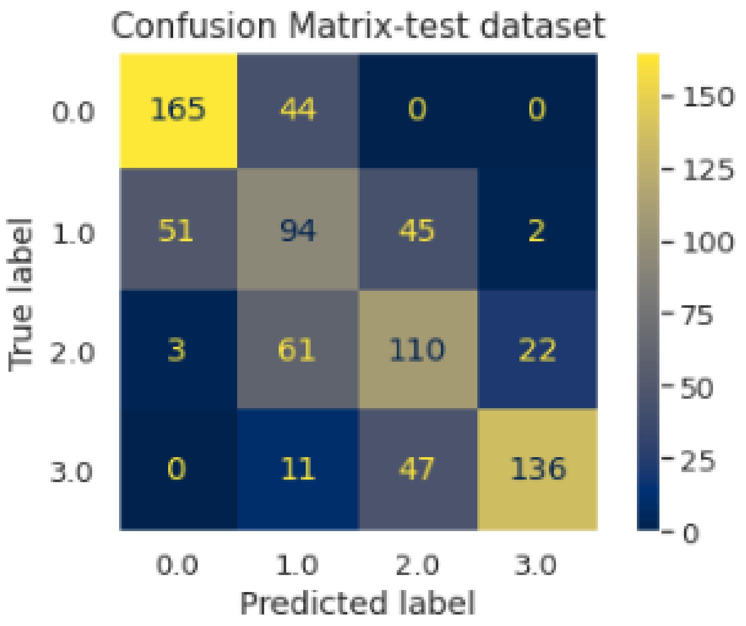


AI

Applying Model

KNNeighbors Classifier

```
Train set accuracy score of is 0.8185654008438819
Test set accuracy score of  is 0.638432364096081

Classification report

              precision    recall  f1-score   support

         0.0       0.86      0.91      0.88       287
         1.0       0.71      0.78      0.74       301
         2.0       0.78      0.72      0.75       299
         3.0       0.94      0.87      0.90       298

    accuracy                           0.82      1185
   macro avg       0.82      0.82      0.82      1185
weighted avg       0.82      0.82      0.82      1185

              precision    recall  f1-score   support

         0.0       0.75      0.79      0.77       209
         1.0       0.45      0.49      0.47       192
         2.0       0.54      0.56      0.55       196
         3.0       0.85      0.70      0.77       194

    accuracy                           0.64       791
   macro avg       0.65      0.64      0.64       791
weighted avg       0.65      0.64      0.64       791
```



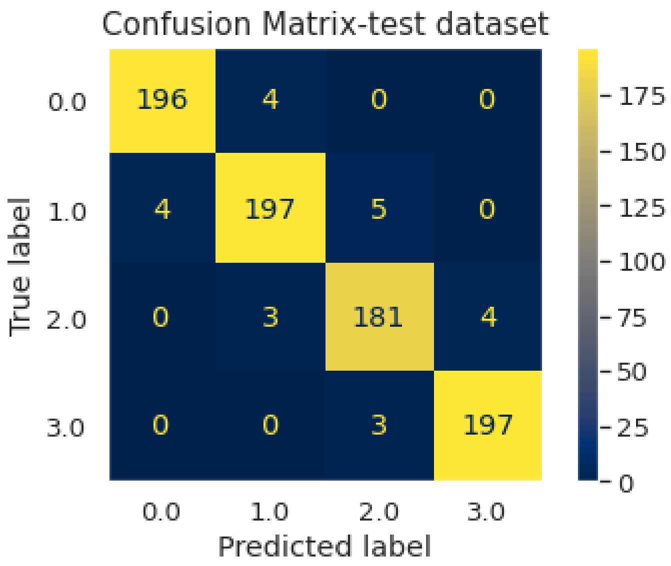
Applying Model

KNNeighbors Classifier

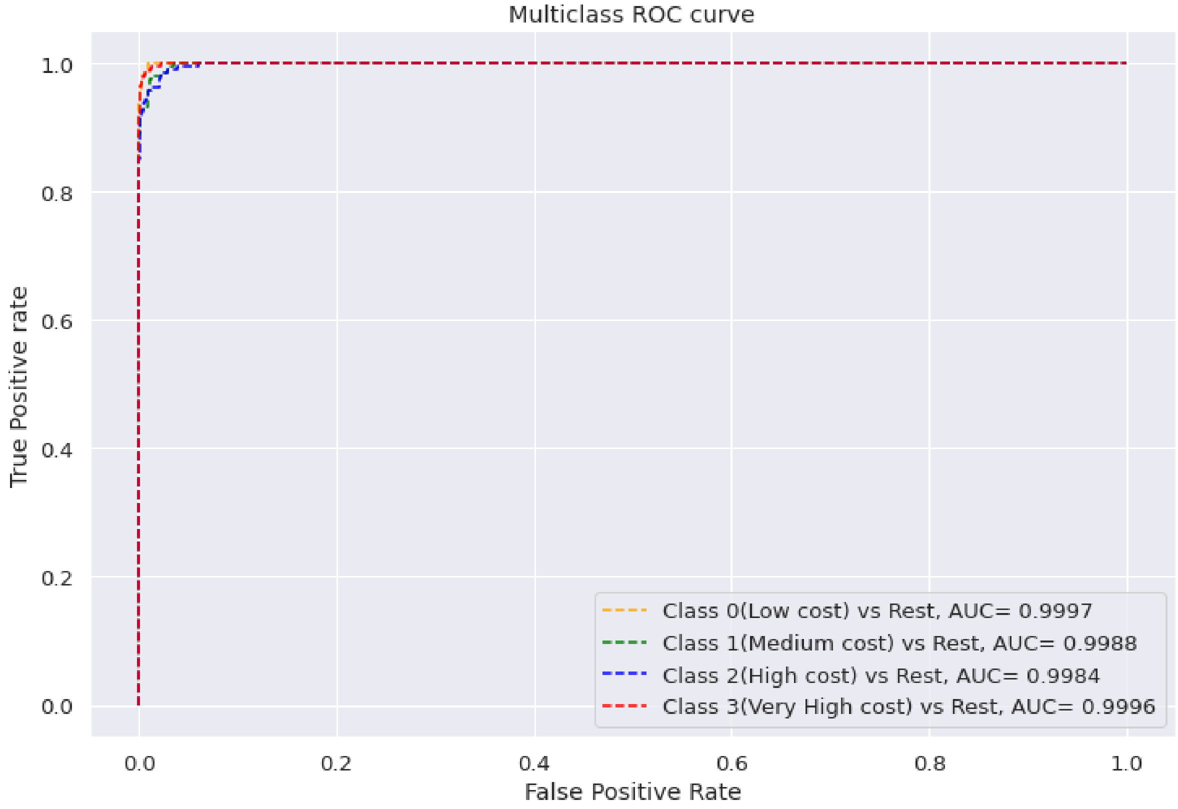
Hyper parameter tuning

Logistic Regression – optimal values

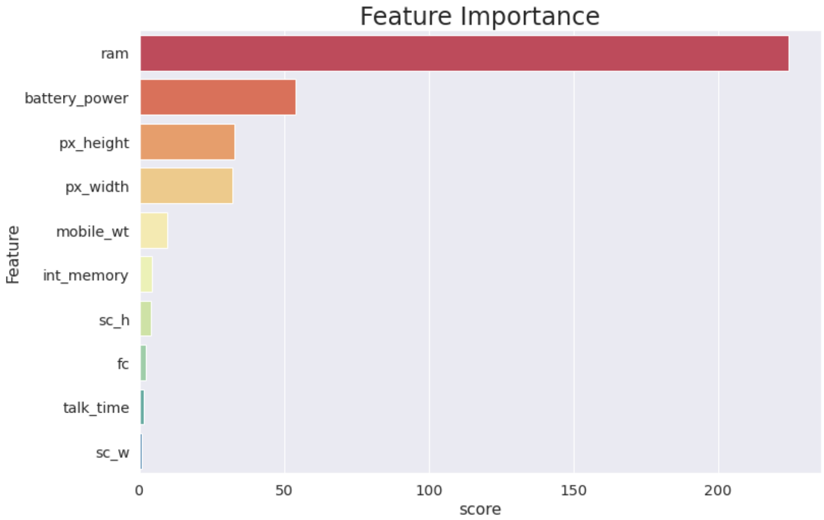
```
Train set accuracy score of is 0.9823529411764705
Test set accuracy score of  is 0.9710327455919395

Classification report

              precision    recall  f1-score   support

         0.0       0.99      0.99      0.99       296
         1.0       0.98      0.97      0.97       290
         2.0       0.97      0.98      0.97       308
         3.0       0.99      0.99      0.99       296

    accuracy                           0.98      1190
   macro avg       0.98      0.98      0.98      1190
weighted avg       0.98      0.98      0.98      1190

              precision    recall  f1-score   support

         0.0       0.98      0.98      0.98       200
         1.0       0.97      0.96      0.96       206
         2.0       0.96      0.96      0.96       188
         3.0       0.98      0.98      0.98       200

    accuracy                           0.97       794
   macro avg       0.97      0.97      0.97       794
weighted avg       0.97      0.97      0.97       794
```



ROC curve Logistic regression



Feature Importance- Logistic Regression



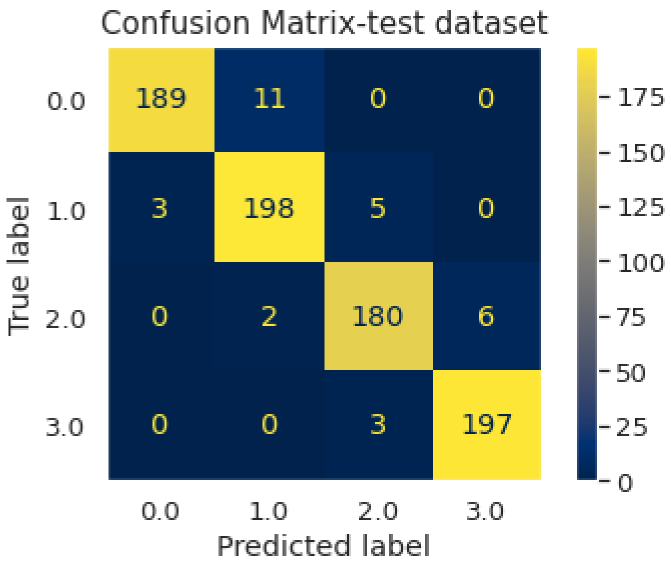
Observations - Logistic regression

- *Accuracy score on train set is 97% and Test score is 95%.* *Model is neither overfitted nor underfitting as the differance between train and test accuracy score is just 2% .
- *After Hyperparameter tuning train accuracy increased to 98.6 % and test accuracy score increased to 97%.*
- *Logistic regression performed the best when compared to other models taht were experimented.*
- *In terms of feature importance RAM,Battery power,px_height and px_weight are the imporatant features.*
- *f1 score for individual classes is also very good. Area under curve for each class prediction is also almost 1.*

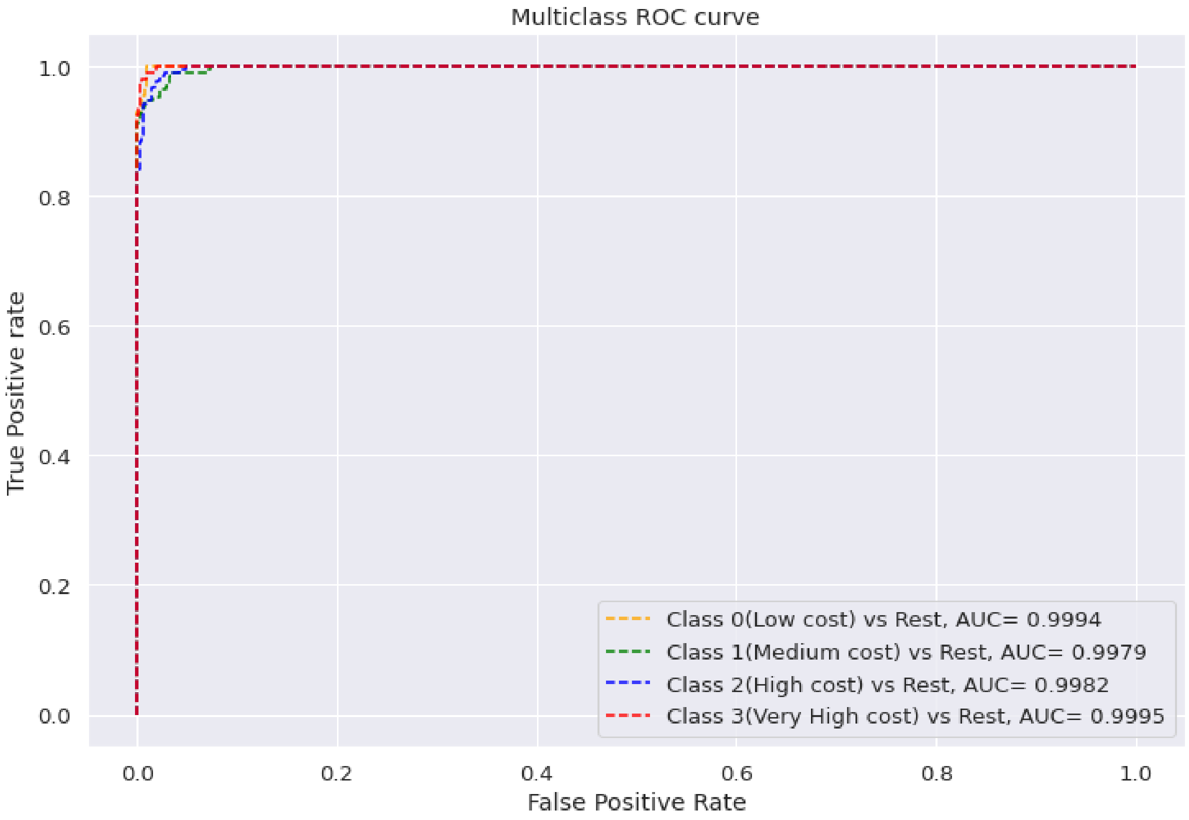
Hyper parameter tuning

SVC – optimal values

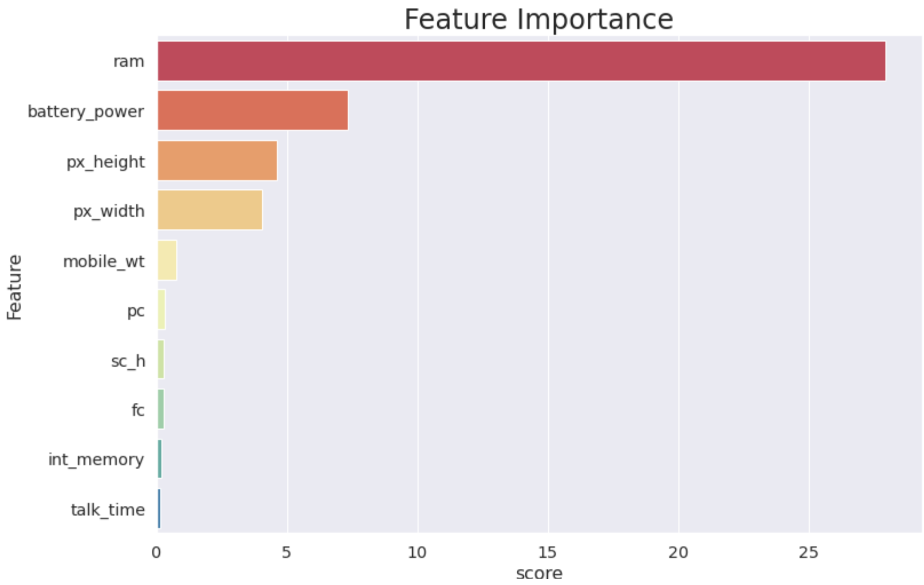
```
Train set accuracy score of is 0.9815126050420168
Test set accuracy score of  is 0.9622166246851386

Classification report

              precision    recall  f1-score   support

         0.0       0.99      0.99      0.99       296
         1.0       0.97      0.98      0.97       290
         2.0       0.97      0.97      0.97       308
         3.0       0.99      0.99      0.99       296

    accuracy                           0.98      1190
   macro avg       0.98      0.98      0.98      1190
weighted avg       0.98      0.98      0.98      1190
              precision    recall  f1-score   support

         0.0       0.98      0.94      0.96       200
         1.0       0.94      0.96      0.95       206
         2.0       0.96      0.96      0.96       188
         3.0       0.97      0.98      0.98       200

    accuracy                           0.96       794
   macro avg       0.96      0.96      0.96       794
weighted avg       0.96      0.96      0.96       794
```



ROC curve - SVC



Feature Importance- SVC

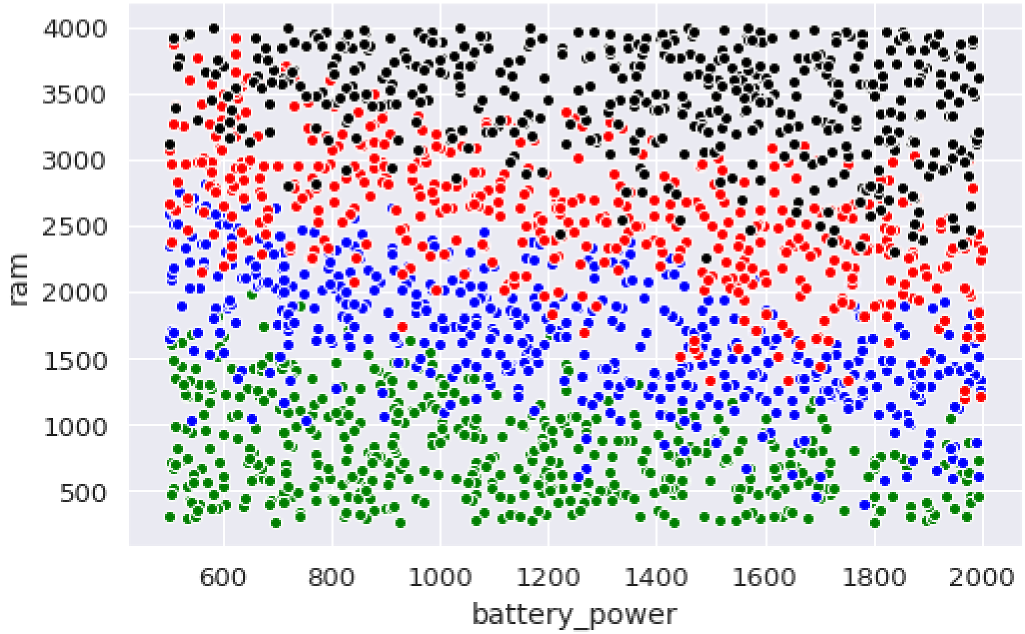


Observations - SVC

- ***Accuracy score on train set is 98% and Test score is 89%. Model seems to be overfitted as the differance between train and test accuracy score is almot 10%.***
- ***After Hyperparameter tuning train accuracy remained almost same 98% and test accuracy score increased to 97%.***
- ***SVM performed very well as compared to other alogorithms.***
- ***In terms of feature importance RAM,Battery power,px_height and px_weight are the imporatant features.***
- ***f1 score for individual classes is also very good. Area under curve for each class prediction is also almost 1.***

RAM and battery are the two most important features for both SVC and linear regression models

Visualizing class with respect to top two important features



Black- very high cost

Red-High cost

Blue-Medium cost

Green-low cost

Conclusion

This Project deals with the predication of the price range and the features of the mobile .It uses Feature selection to give precise features to be selected and get maximum accuracy results.

To find the optimal model, the accuracy score was chosen as the best statistic. The test accuracy for the tree based classification models hovered around 89%.

For this dataset, KNN produced the lowest accuracy score.

Linear classification models like SVM and Logistic regression gave high accuracy scores.

After performing hyper parameter tuning

SVM

Train accuracy: 98%

Test accuracy: 97%

Logistic Regression

Train accuracy: 98%

Test accuracy: 97%

Feature importance showed that **Ram** and **battery power** were the top two most significant features.

AUC scores for all four classes were almost close to 1 for both SVM and Logistic regression.

Thank you