AI

Supervised ML Classification Capstone Project

Mobile Price Range Prediction

Suraj Kad
Suraj.kad.90@gmail.com

- **classification in supervised learning**

The Classification algorithm is a Supervised Learning technique that is used to identify the category of new observations on the basis of training data. In Classification, a program learns from the given dataset or observations and then classifies new observation into a number of classes or groups.

- **difference of classification and regression**

The most significant difference between regression vs classification is that while regression helps predict a continuous quantity, classification predicts discrete class labels.

Problem Statement

- In the competitive mobile phone market companies want 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(e.g.:- RAM, Internal Memory, etc.) and its selling price. In this problem, we do not have to predict the actual price but a price range indicating how high the price is.

Points to discuss:

- Data description and summary
- Data Preprocessing

1. Getting the dataset
2. Importing libraries
3. Importing datasets
4. Finding Missing Data
5. Encoding Categorical Data

- Exploratory data analysis
- Heat map
- Machine learning algorithms

1. Logistic regression
2. Decision tree
3. Random forest classifier
4. SVM

- Conclusion

Data description

The data contains information regarding mobile phone features, specifications etc and their price range. The various features and information can be used to predict the price range of a mobile phone.

- 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 megapixels
- Four_g - Has 4G or not
- Int_memory - Internal Memory in Gigabytes
- M_dep - Mobile Depth in cm
- Mobile_wt - Weight of mobile phone

Data description

- N_cores - Number of cores of processor
- Pc - Primary Camera megapixels
- Px_height - Pixel Resolution Height
- Px_width - Pixel Resolution Width
- Ram - Random Access Memory in Megabytes
- 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
- Price_range - This is the target variable with value of 0(low cost), 1(medium cost),
- 2(high cost) and 3(very high cost).

Data Preprocessing

```
[2] MPR_Dataset=pd.read_csv("/content/drive/MyDrive/Mobile Price Range Prediction/data_mobile_price_range.csv")
```

```
MPR_Dataset.head()
```

	battery_power	blue	clock_speed	dual_sim	fc	four_g	int_memory	m_dep	mobile_wt	n_cores	...	px_height	px_width	ram
0	842	0	2.2	0	1	0	7	0.6	188	2	...	20	756	2549
1	1021	1	0.5	1	0	1	53	0.7	136	3	...	905	1988	2631
2	563	1	0.5	1	2	1	41	0.9	145	5	...	1263	1716	2603
3	615	1	2.5	0	0	0	10	0.8	131	6	...	1216	1786	2769
4	1821	1	1.2	0	13	1	44	0.6	141	2	...	1208	1212	1411

5 rows × 21 columns

```
MPR_Dataset.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 2000 entries, 0 to 1999
Data columns (total 21 columns):
 #   Column         Non-Null Count  Dtype
---  ------         --------------  -----
 0   battery_power  2000 non-null   int64
 1   blue           2000 non-null   int64
 2   clock_speed    2000 non-null   float64
 3   dual_sim       2000 non-null   int64
 4   fc             2000 non-null   int64
 5   four_g         2000 non-null   int64
 6   int_memory     2000 non-null   int64
 7   m_dep          2000 non-null   float64
 8   mobile_wt      2000 non-null   int64
 9   n_cores        2000 non-null   int64
 10  pc             2000 non-null   int64
 11  px_height      2000 non-null   int64
 12  px_width       2000 non-null   int64
 13  ram            2000 non-null   int64
 14  sc_h           2000 non-null   int64
 15  sc_w           2000 non-null   int64
 16  talk_time      2000 non-null   int64
 17  three_g        2000 non-null   int64
 18  touch_screen   2000 non-null   int64
 19  wifi           2000 non-null   int64
 20  price_range    2000 non-null   int64
dtypes: float64(2), int64(19)
memory usage: 328.2 KB
```

- **Read and write Mobile Price Range (tabular) data using pandas functions**

The info() method prints information about the Mobile Price Range Data Frame. The information contains the number of columns, column labels, column data types, memory usage, range index, and the number of cells in each column (non-null values).

```python
print(len(MPR_Dataset[MPR_Dataset.px_height==0]))
print(len(MPR_Dataset[MPR_Dataset.px_width==0]))
print(len(MPR_Dataset[MPR_Dataset.sc_h==0]))
print(len(MPR_Dataset[MPR_Dataset.sc_w==0]))
```

```
2
0
0
180
```

```python
[12] MPR_Dataset['sc_w'][MPR_Dataset[MPR_Dataset.sc_w == 0].index] = MPR_Dataset.sc_w.mean()
     MPR_Dataset['px_height'][MPR_Dataset[MPR_Dataset.px_height == 0].index] = MPR_Dataset.px_height.mean()
```

```python
[13] #Sucsessfully handle this values.
     print(len(MPR_Dataset[MPR_Dataset.sc_w==0]))
     print(len(MPR_Dataset[MPR_Dataset.px_height==0]))
```

```
0
0
```

Firstly check the minimum value of pixel width, pixel Height and Screen ,Width Screen Height is cannot be Zero.

I can found the zero value in pixel Height and screen width columns. So handle this value assigning mean .

```
MPR_Dataset.nunique()
```

```
battery_power     1094
blue                 2
clock_speed         26
dual_sim             2
fc                  20
four_g               2
int_memory          63
m_dep               10
mobile_wt          121
n_cores              8
pc                  21
px_height         1137
px_width          1109
ram               1562
sc_h                15
sc_w                19
talk_time           19
three_g              2
touch_screen         2
wifi                 2
price_range          4
dtype: int64
```

The pandas.unique() function returns the unique values present in a dataset.

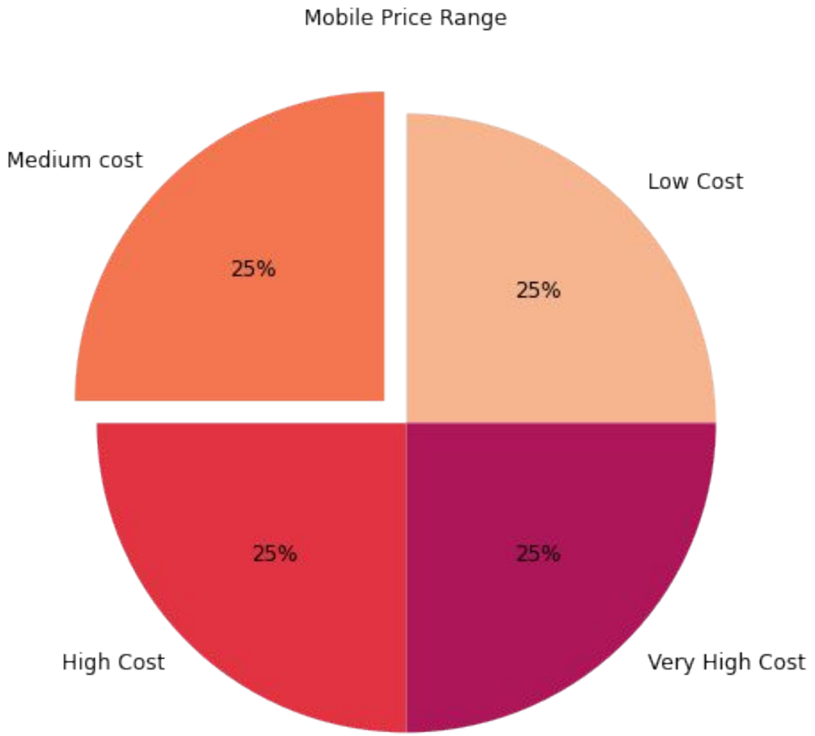
```
#Cheacking null/missing values Present in Dataset Or Not???
MPR_Dataset.isnull().sum()
#There are no missing values in the datasets.
```

```
battery_power    0
blue             0
clock_speed      0
dual_sim         0
fc               0
four_g           0
int_memory       0
m_dep            0
mobile_wt        0
n_cores          0
pc               0
px_height        0
px_width         0
ram              0
sc_h             0
sc_w             0
talk_time        0
three_g          0
touch_screen     0
wifi             0
price_range      0
dtype: int64
```

- We will count total number of NaN data present in Mobile Price Range dataset
and find out the number of NaN or missing values in each columns.

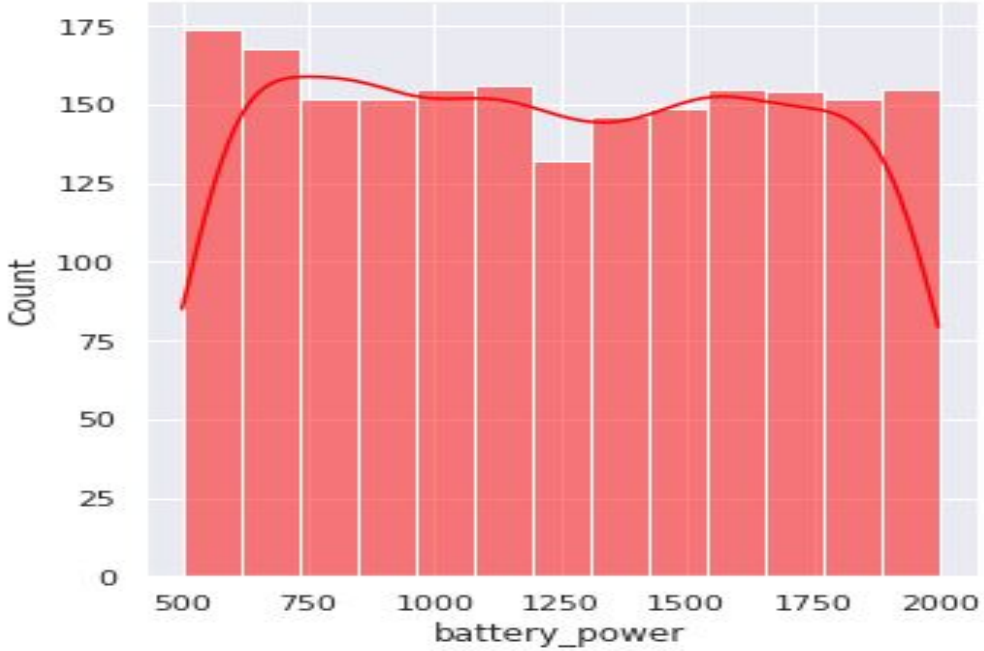
Exploratory data analysis

PRICE



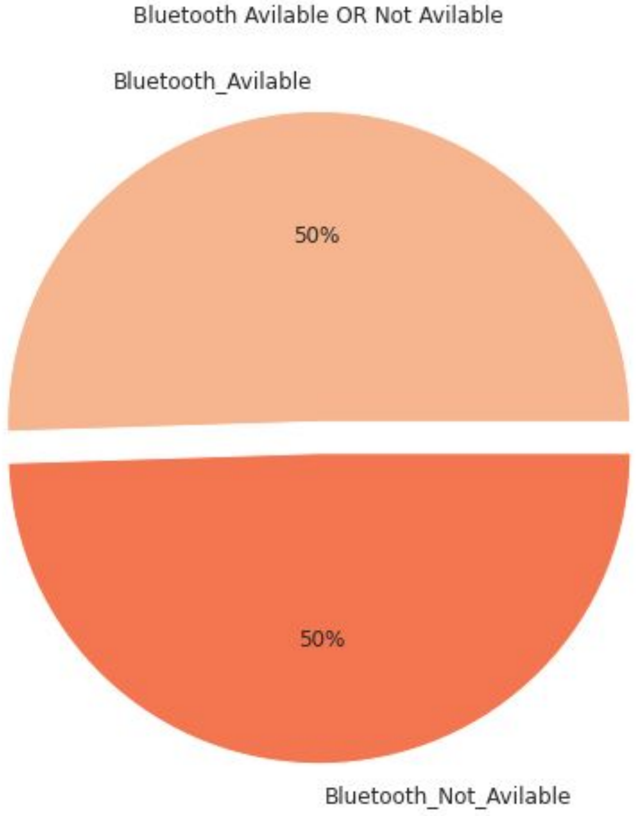
● we can see that ,this pie chart there are mobile phones in 4 price ranges. the number of elements is almost similar

BATTERY

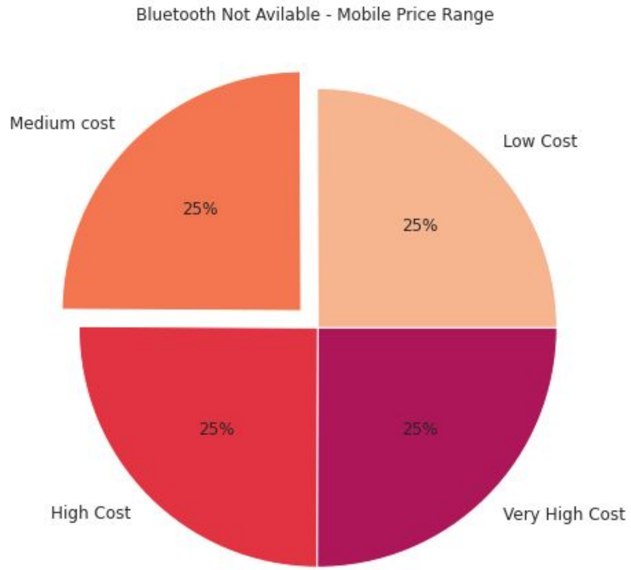


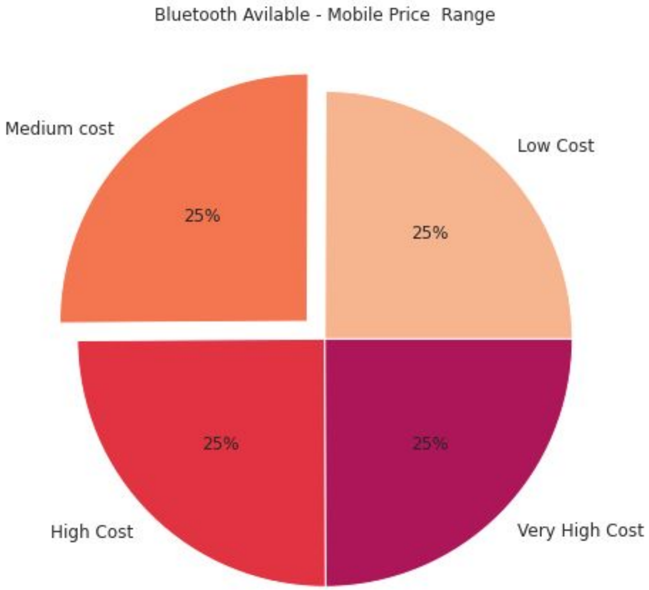
● This plot shows how the battery mAh is spread. there is a gradual increase as the price range increases

BLUETOOTH



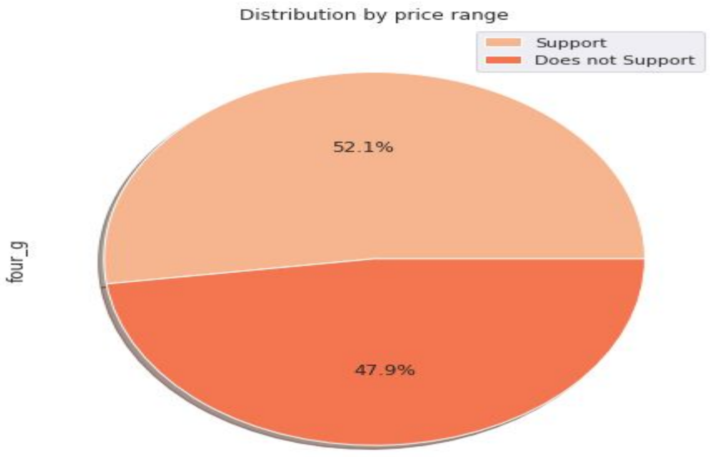
half the devices have Bluetooth, and half don’t

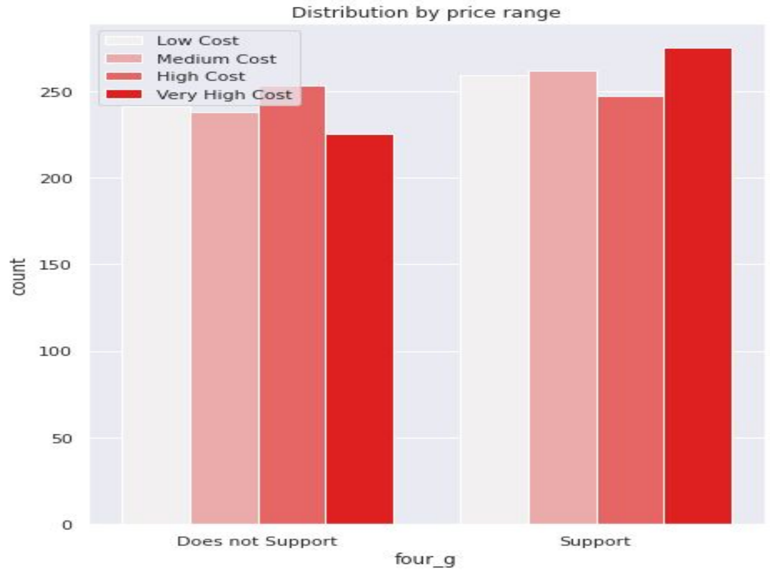


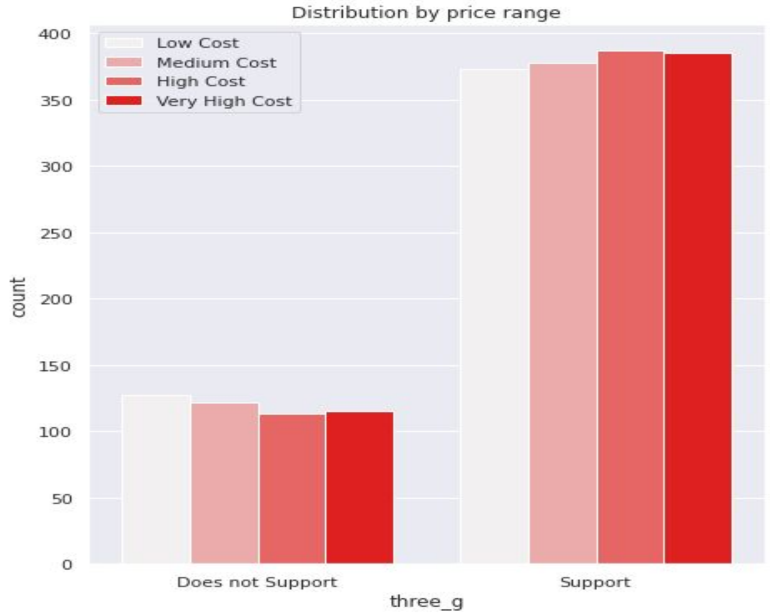


This Bluetooth features distribution is almost similar along all the price ranges variable, it may not be helpful in making predictions

3G AND 4G





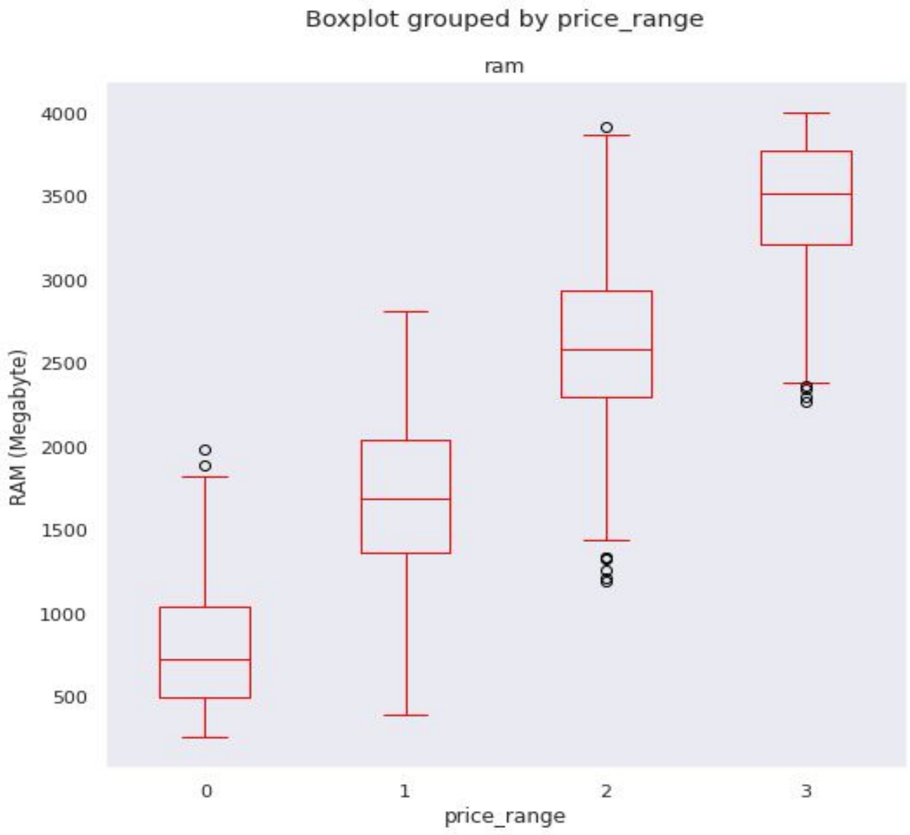


50% of the phones support 4_g and 76% of phones support 3_g

Distribution of price range almost similar of supported and unsupported feature in 4G . So that is not used full of prediction.

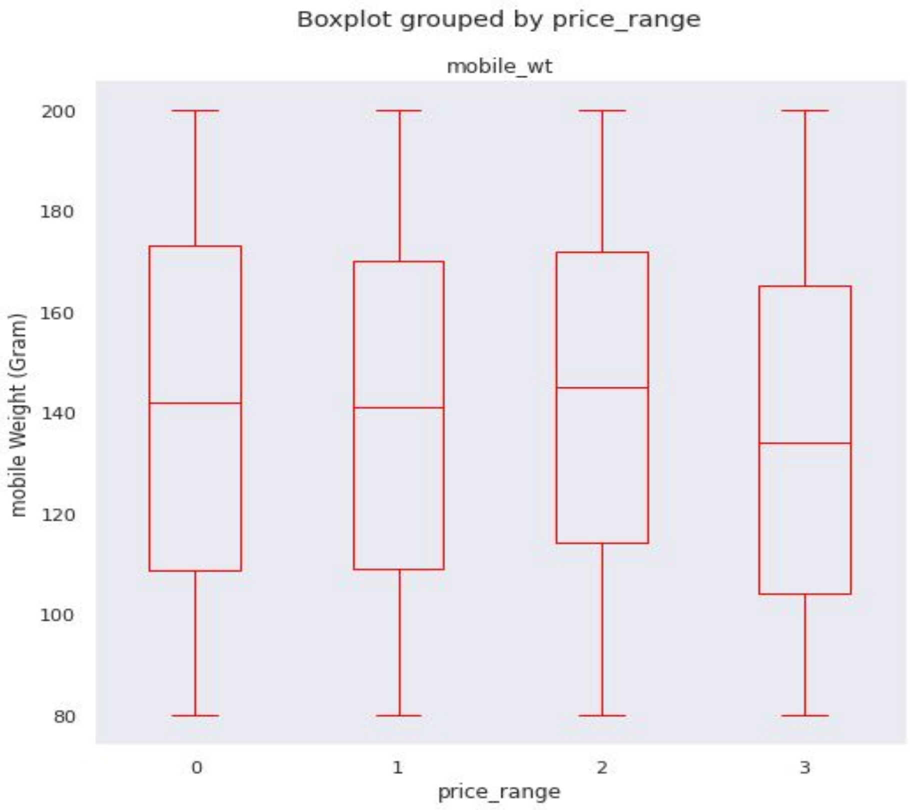
feature ‘ThreeG' play an important feature in prediction

RAM



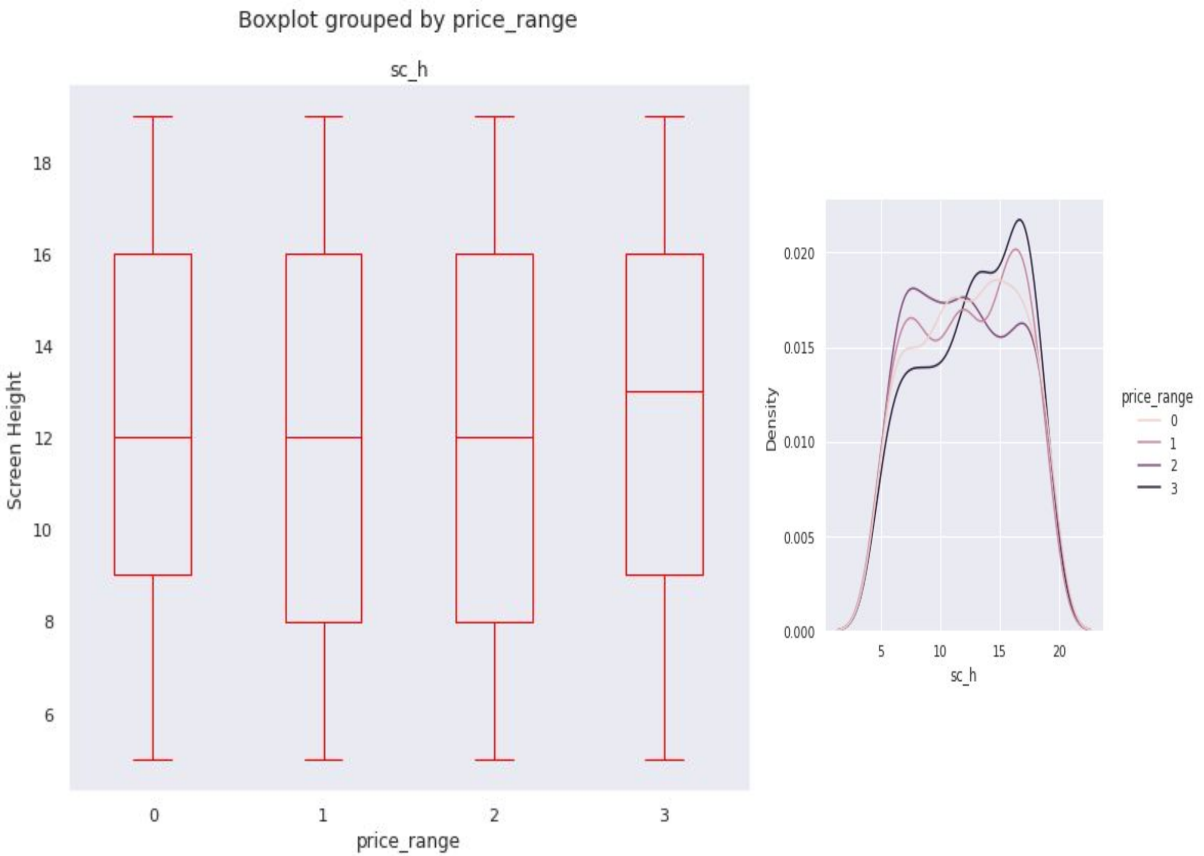
Ram has continuous increase with price range while moving from Low cost to Very high cost.

MOBILE WEIGHT

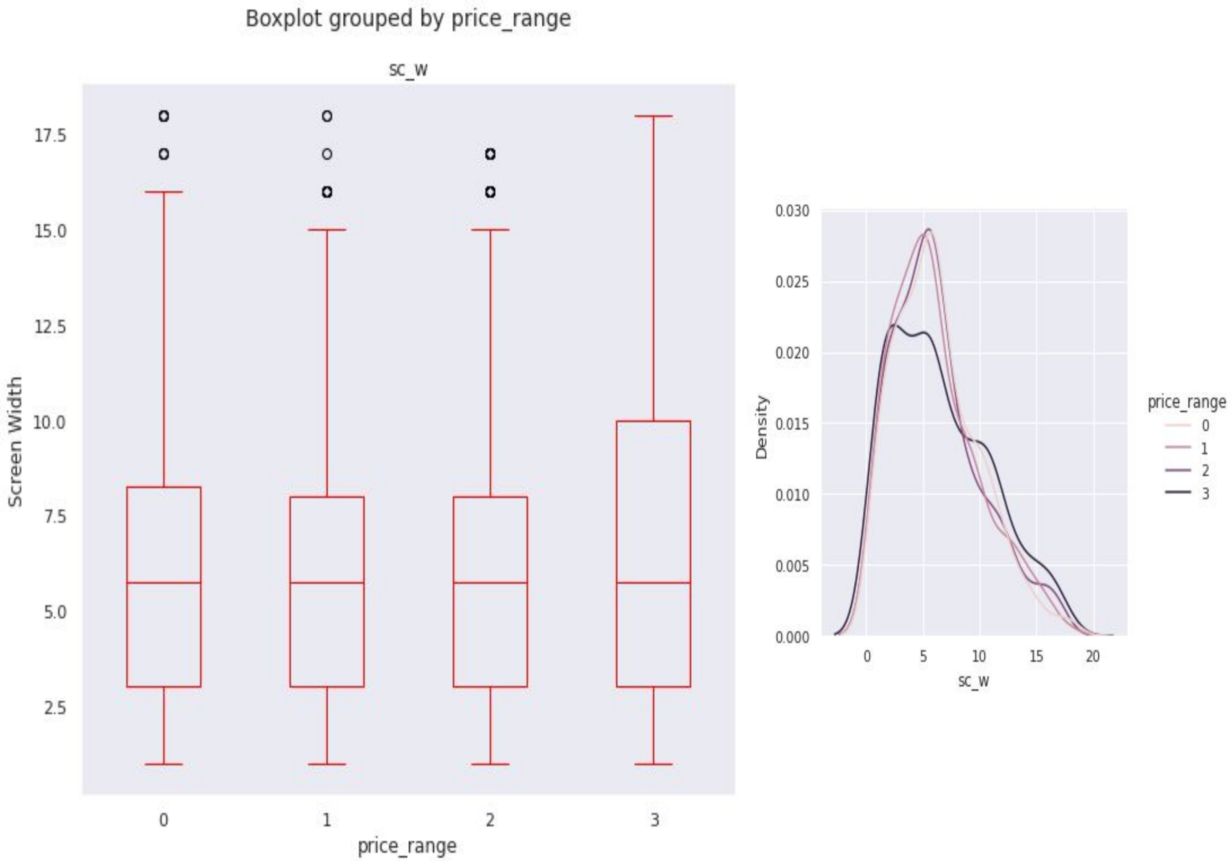


we can see that ,this boxplot costly phones are lighter weight

SCREEN HEIGHT



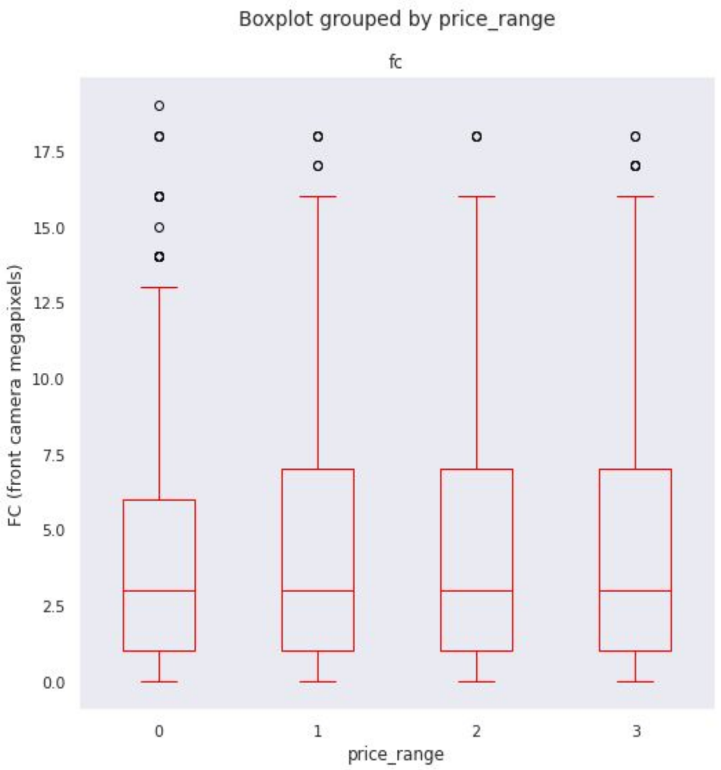
SCREEN WIDTH

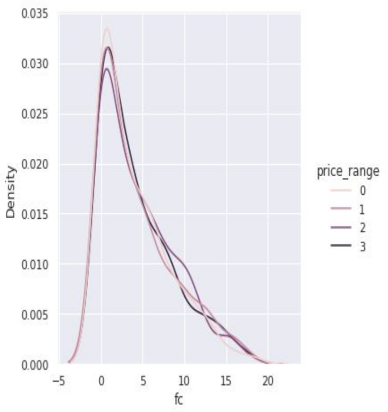


There is not a continuous increase in pixel width as we move from Low cost to Very high cost. Mobiles with 'Medium cost' and 'High cost' has almost equal pixel width. so we can say that it would be a driving factor in deciding price range.

Pixel height is almost similar as we move from Low cost to Very high cost. Little variation in pixel height

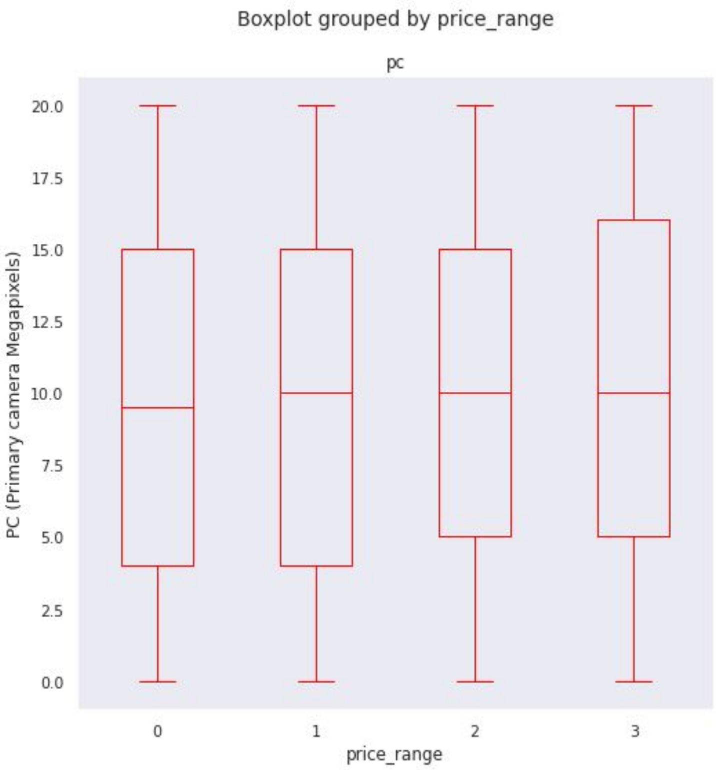
FC (front camera megapixels)





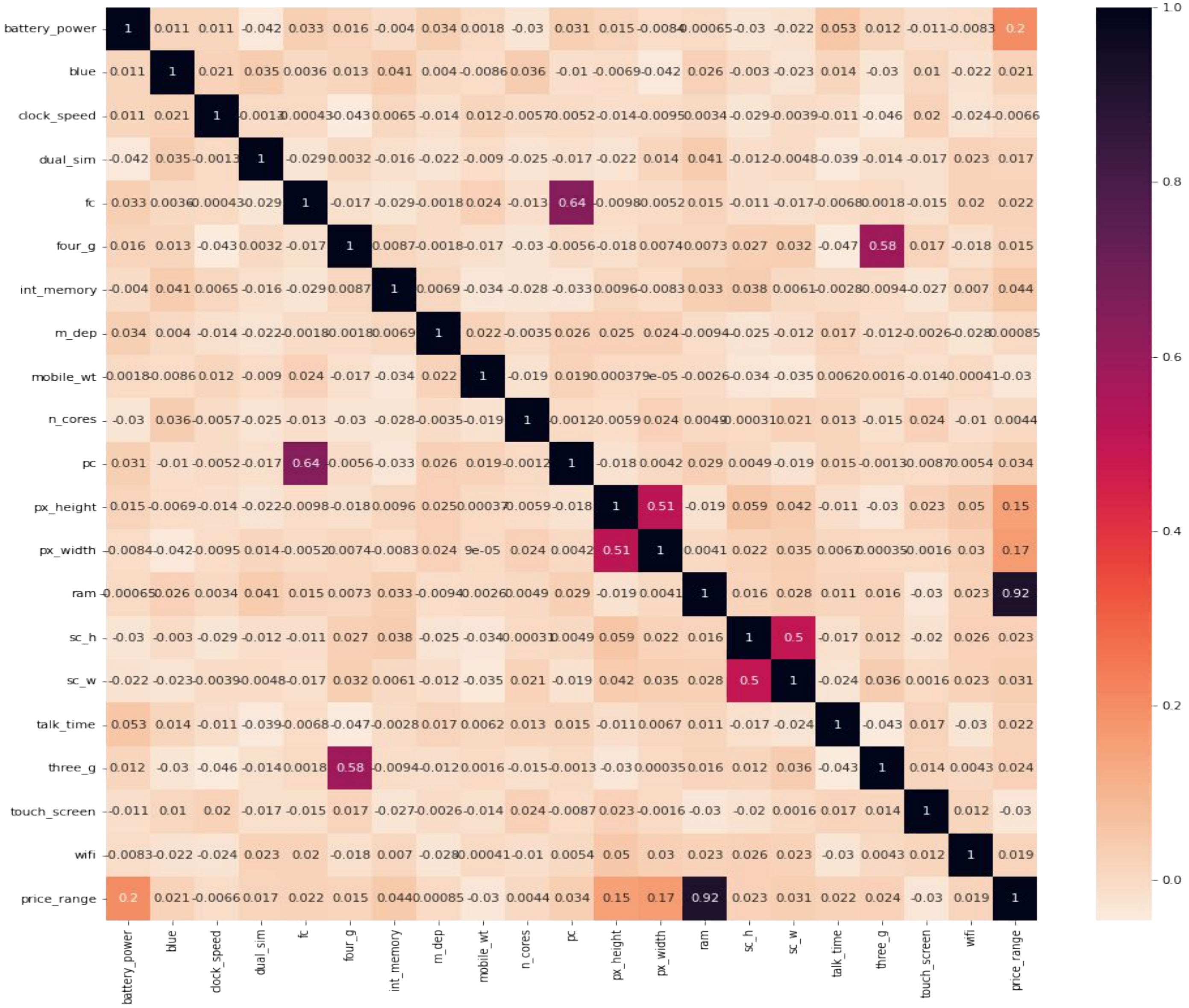
- This features distribution is almost similar along all the price ranges variable, it may not be helpful in making predictions

PC (Primary camera Megapixels)



- Primary camera megapixels are showing a little variation along the target categories, which is a good sign for prediction.

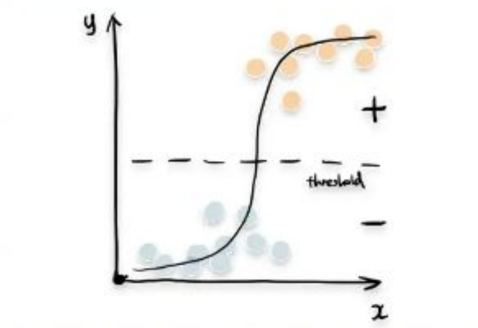
HEAT MAP



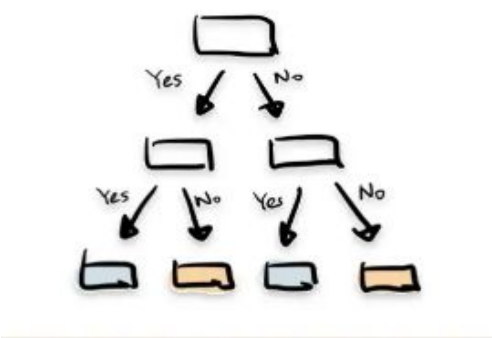
- **RAM and price range shows high correlation which is a good sign, it signifies that RAM will play major deciding factor in estimating the price range.**
- **There is some collinearity in feature pairs ('pc', 'fc') and ('px_width', 'px_height'). Both correlations are justified since there are good chances that if front camera of a phone is good, the back camera would also be good.**
- **Also, if px_height increases, pixel width also increases, that means the overall pixels in the screen. We can replace these two features with one feature. Front Camera megapixels and Primary camera megapixels are different entities despite of showing collinearity. So we'll be keeping them as they are.**

Supervised ML Classification Machine Learning algorithms

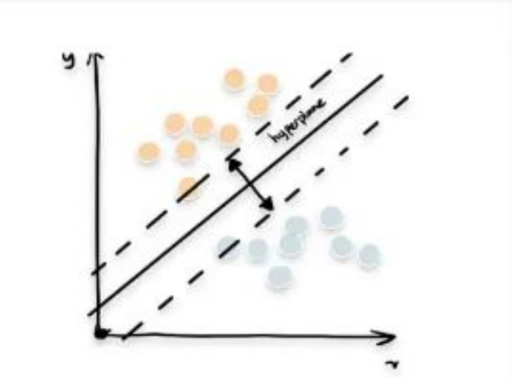
1. Logistic Regression



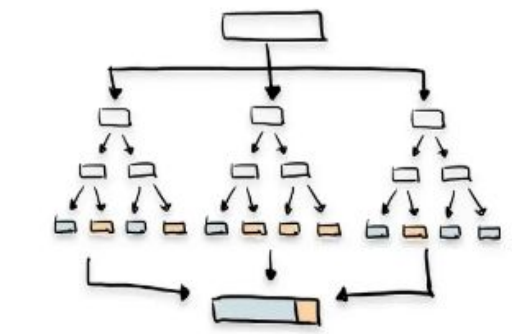
2. Decision Tree

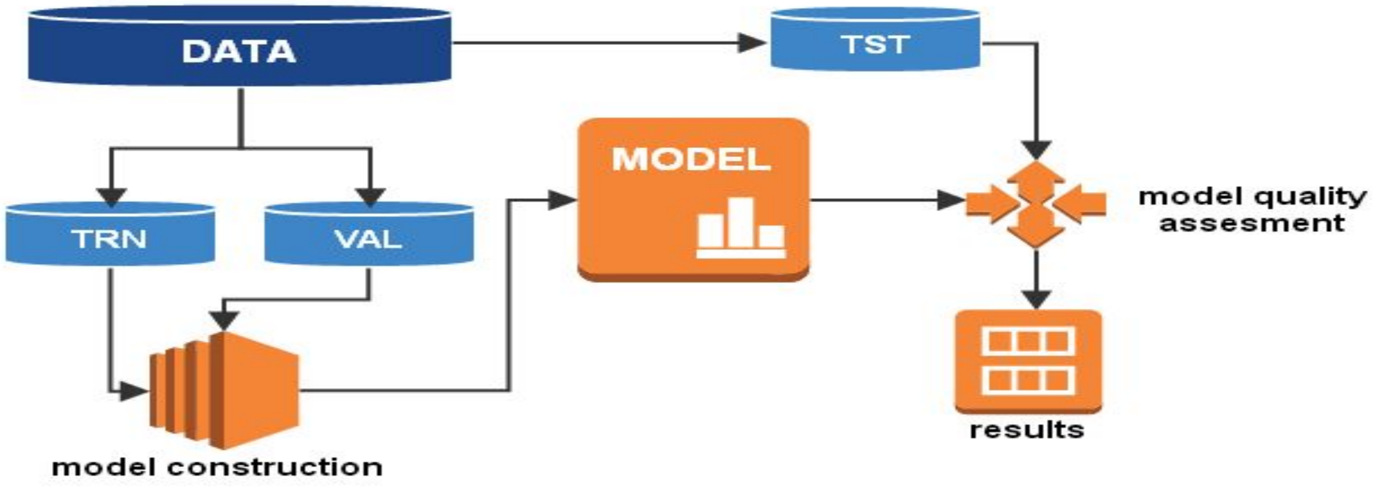


3. Support Vector Machine

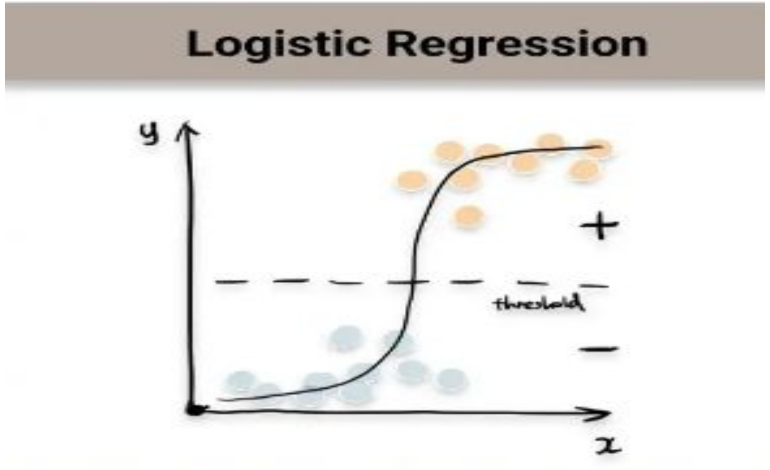


4. Random Forest





1. Logistic Regression



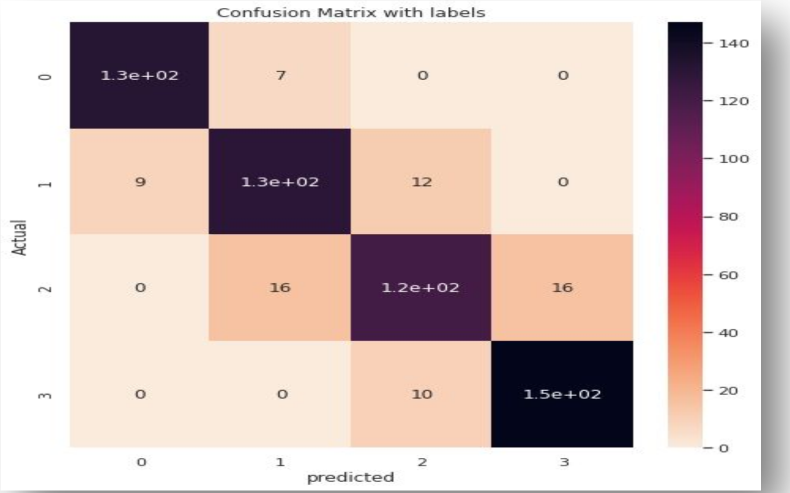
Logistic regression is a Machine Learning classification algorithm that is used to predict the probability of certain classes based on some dependent variables. In short, the logistic regression model computes a sum of the input features (in most cases, there is a bias term), and calculates the logistic of the result.

Classification report for Logistic Regression (Train set)=

	precision	recall	f1-score	support
0	0.97	0.94	0.95	375
1	0.86	0.89	0.88	338
2	0.83	0.89	0.86	326
3	0.96	0.91	0.94	361
accuracy			0.91	1400
macro avg	0.91	0.91	0.91	1400
weighted avg	0.91	0.91	0.91	1400

Classification report for Logistic Regression (Test set)=

	precision	recall	f1-score	support
0	0.95	0.94	0.94	141
1	0.86	0.85	0.86	153
2	0.79	0.85	0.82	143
3	0.94	0.90	0.92	163
accuracy			0.88	600
macro avg	0.88	0.88	0.88	600
weighted avg	0.89	0.88	0.88	600

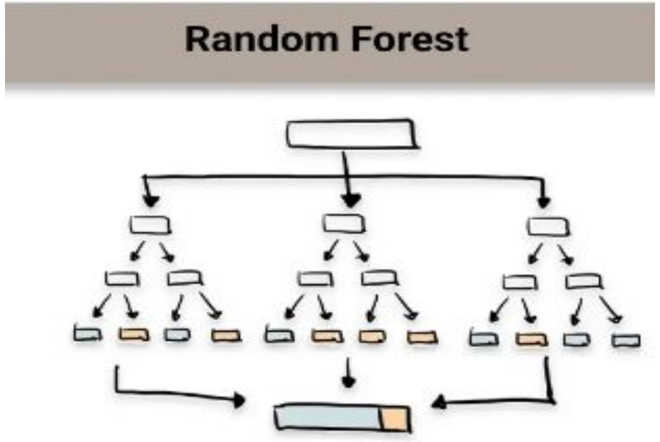


TRAIN ACCURACY : 91%

TEST ACCURACY : 88%

2. Random forest

A Random Forest Algorithm is a supervised machine learning algorithm which is extremely popular and is used for Classification and Regression problems in Machine Learning. We know that a forest comprises numerous trees, and the more trees more it will be robust.

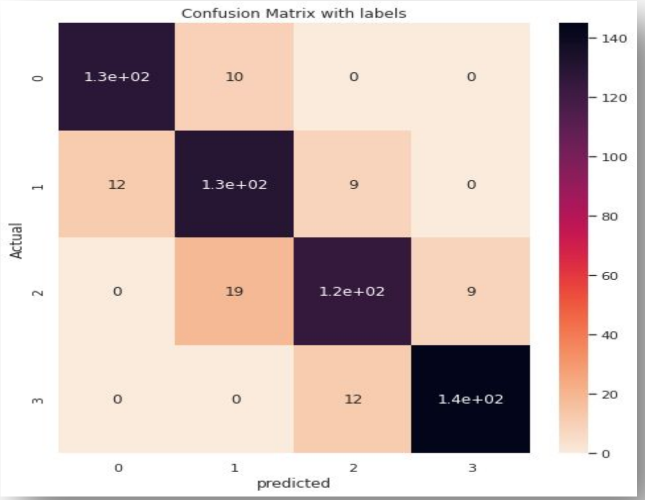


Classification report for Random Forest (Train set)=

	precision	recall	f1-score	support
0	1.00	1.00	1.00	361
1	1.00	1.00	1.00	349
2	1.00	1.00	1.00	347
3	1.00	1.00	1.00	343
accuracy			1.00	1400
macro avg	1.00	1.00	1.00	1400
weighted avg	1.00	1.00	1.00	1400

Classification report for Random Forest (Test set)=

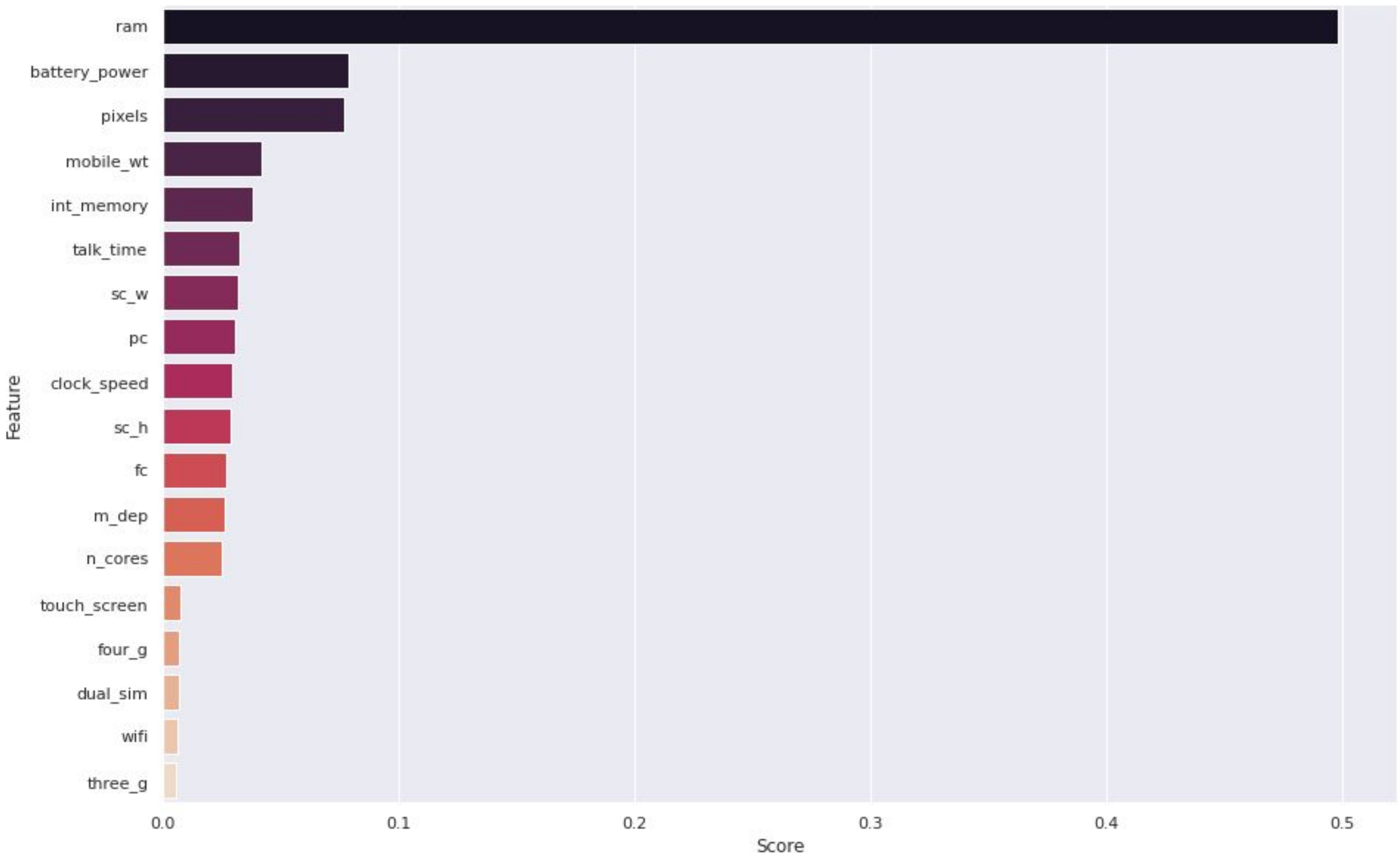
	precision	recall	f1-score	support
0	0.91	0.93	0.92	139
1	0.82	0.86	0.84	151
2	0.86	0.82	0.84	153
3	0.94	0.92	0.93	157
accuracy			0.88	600
macro avg	0.88	0.88	0.88	600
weighted avg	0.88	0.88	0.88	600



TRAIN ACCURACY : 100%

TEST ACCURACY : 88%

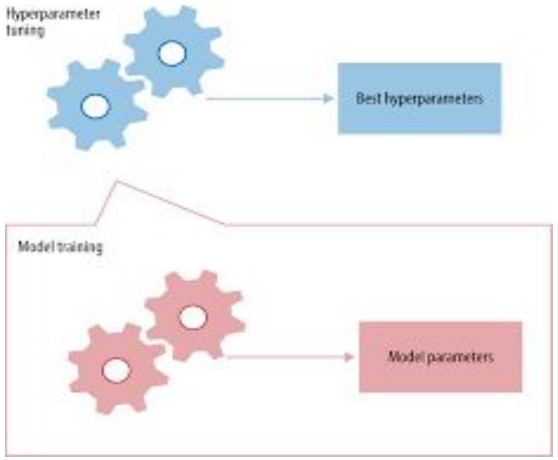
Feature importance Decision tree



Feature importances are provided by the fitted attribute feature_importances_ and they are computed as the mean and standard deviation of accumulation of the impurity decrease within each tree.

Hyperparameter tuning for Random Forest

In the case of a random forest, hyperparameters include the number of decision trees in the forest and the number of features considered by each tree when splitting a node. (The parameters of a random forest are the variables and thresholds used to split each node learned during training)



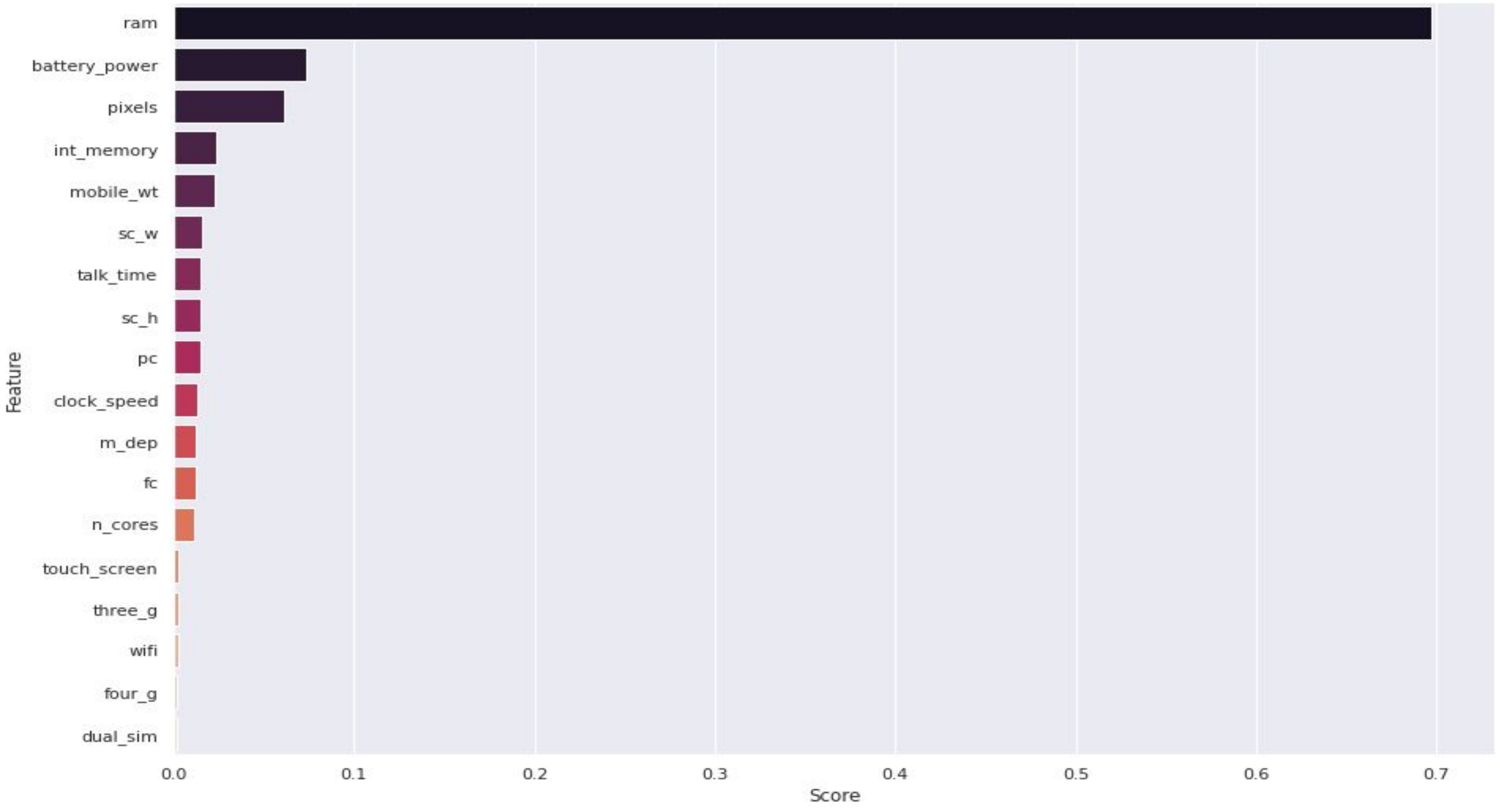

```
print('Classification report for Hyperparameter tuning for Random Forest (Train set)= ')
print(classification_report(y_train, y_pred))

Classification report for Hyperparameter tuning for Random Forest (Train set)=
              precision    recall  f1-score   support

           0       0.97      0.98      0.97       395
           1       0.92      0.93      0.93       409
           2       0.93      0.92      0.92       408
           3       0.97      0.97      0.97       388

    accuracy                           0.95      1600
   macro avg       0.95      0.95      0.95      1600
weighted avg       0.95      0.95      0.95      1600
```

```
print('Classification report for Hyperparameter tuning for Random Forest (Test set)= ')
print(classification_report(y_test, y_pred2))

Classification report for Hyperparameter tuning for Random Forest (Test set)=
              precision    recall  f1-score   support

           0       0.94      0.96      0.95       105
           1       0.84      0.80      0.82        91
           2       0.76      0.79      0.78        92
           3       0.92      0.89      0.90       112

    accuracy                           0.87       400
   macro avg       0.86      0.86      0.86       400
weighted avg       0.87      0.87      0.87       400
```

TRAIN ACCURACY : 95%

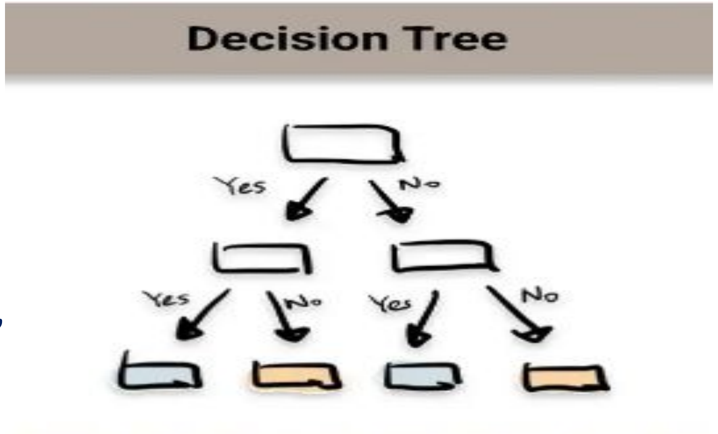
TEST ACCURACY : 87%

Feature importance for Hyperparameter tuning for Random Forest



3.Decision tree

A decision tree is a non-parametric supervised learning algorithm, which is utilized for both classification and regression tasks. It has a hierarchical, tree structure, which consists of a root node, branches, internal nodes and leaf nodes.

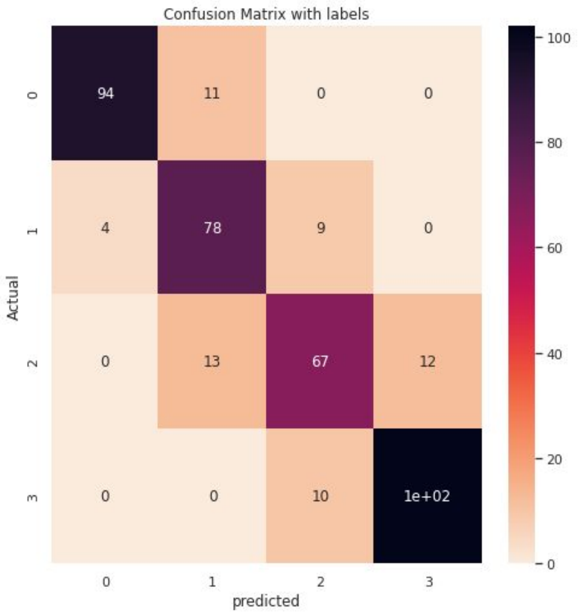



```
print('Classification Report for Decision Tree (train set)= ')
print(classification_report(y_test, y_pred_test))
```

```
Classification Report for Decision Tree (train set)=
              precision    recall  f1-score   support

           0       0.96      0.90      0.93       105
           1       0.76      0.86      0.81        91
           2       0.78      0.73      0.75        92
           3       0.89      0.91      0.90       112

    accuracy                           0.85       400
   macro avg       0.85      0.85      0.85       400
weighted avg       0.86      0.85      0.85       400
```

```
print('Classification report for Decision Tree (Test set)= ')
print(classification_report(y_pred_test, y_test))
```

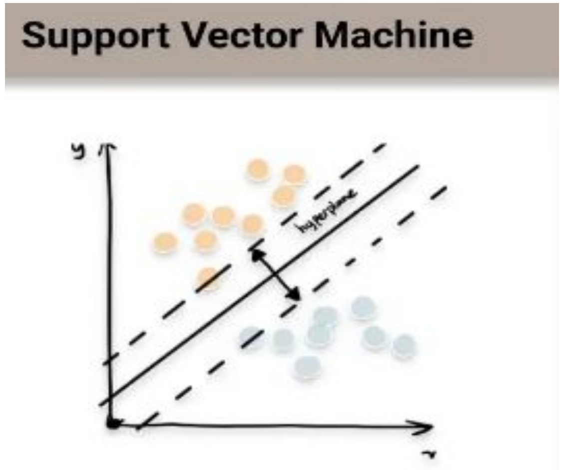
```
Classification report for Decision Tree (Test set)=
              precision    recall  f1-score   support

           0       0.87      0.98      0.92        93
           1       0.81      0.73      0.77       101
           2       0.78      0.67      0.72       108
           3       0.81      0.93      0.87        98

    accuracy                           0.82       400
   macro avg       0.82      0.83      0.82       400
weighted avg       0.82      0.82      0.82       400
```



TRAIN ACCURACY : 85%

TEST ACCURACY : 82%

4. SUPPORT VECTOR MACHINE



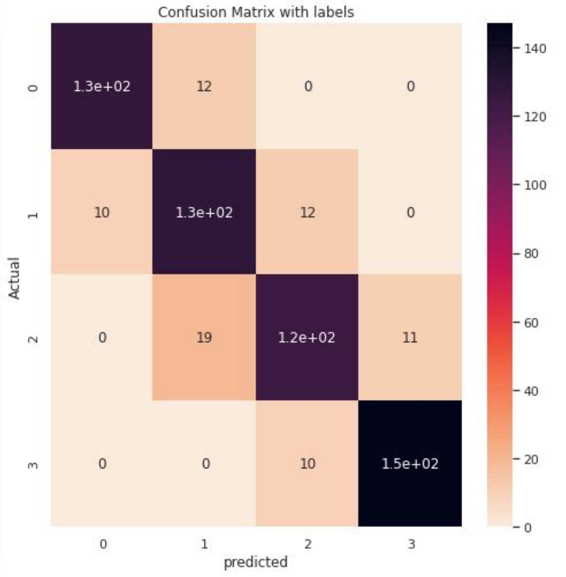
Support Vector Machine(SVM) is a supervised machine learning algorithm used for both classification and regression. Though we say regression problems as well its best suited for classification. The objective of SVM algorithm is to find a hyperplane in an N-dimensional space that distinctly classifies the data points.

```
[87] print('Classification Report for Decision Tree (Train set)= ')
     print(classification_report(y_train, y_pred_train))

     Classification Report for Decision Tree (Train set)=
                   precision    recall  f1-score   support

                0       0.99      0.99      0.99       395
                1       0.97      0.97      0.97       409
                2       0.97      0.97      0.97       408
                3       0.99      0.98      0.99       388

         accuracy                           0.98      1600
        macro avg       0.98      0.98      0.98      1600
     weighted avg       0.98      0.98      0.98      1600
```

TRAIN ACCURACY : 98%

```
print('Classification report for Support Vector Machine (Test set)= ')
print(classification_report(y_pred_test, y_test))

Classification report for Support Vector Machine (Test set)=
              precision    recall  f1-score   support

           0       0.93      0.97      0.95       101
           1       0.87      0.81      0.84        98
           2       0.82      0.77      0.79        97
           3       0.88      0.95      0.92       104

    accuracy                           0.88       400
   macro avg       0.88      0.88      0.87       400
weighted avg       0.88      0.88      0.88       400
```

TEST ACCURACY : 88%



Conclusion

1. From EDA we can see that here are mobile phones in 4 price ranges. The number of elements is almost similar.
2. half the devices have Bluetooth, and half don't
3. There is a gradual increase in battery as the price range increases Ram has continuous increase with price range while moving from Low cost to Very high cost
4. costly phones are lighter
5. RAM, battery power, pixels played more significant role in deciding the price range of mobile phone.
6. form all the above experiments we can conclude that logistic regression , SVM and Hyperparameter tuning for Random Forest we got the best results
7. This project model could be improved by developing software that could predict by selecting features so that it could be used while launching the new product.

Thank You