Mobile Price Range Prediction

Capstone Project

Submitted By
Poovarasan
shipfriend0368@gmail.com

TABLE OF CONTENTS

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 (eg:- 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.

Data description:

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

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).

Independent variables

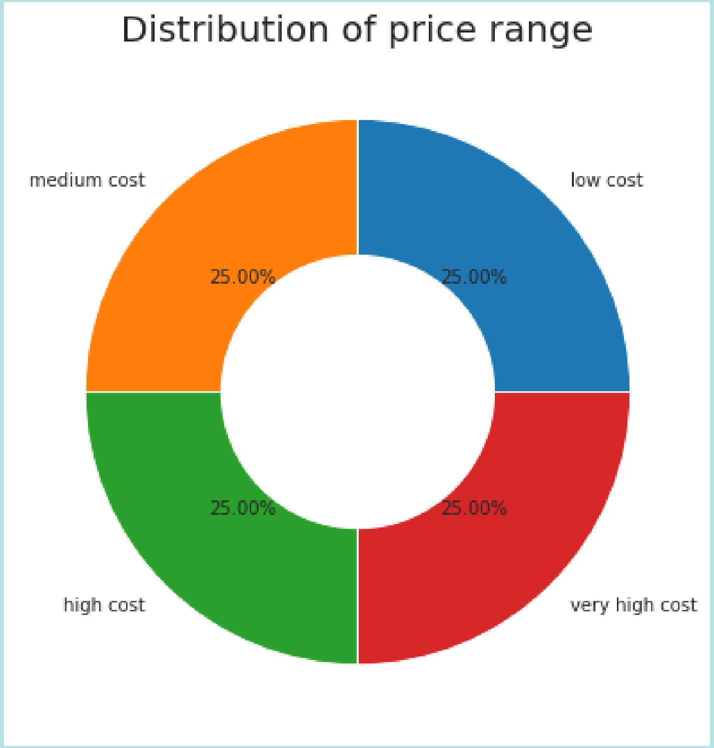
- Battery power - Total energy a battery can store in one time measured in mAh
- Blue - Has Bluetooth or not
- Clock_speed - speed at which microprocessor executes instructions
- Dual_sim - Has dual sim support or not
- Fc - Front Camera mega pixels

- 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
- 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

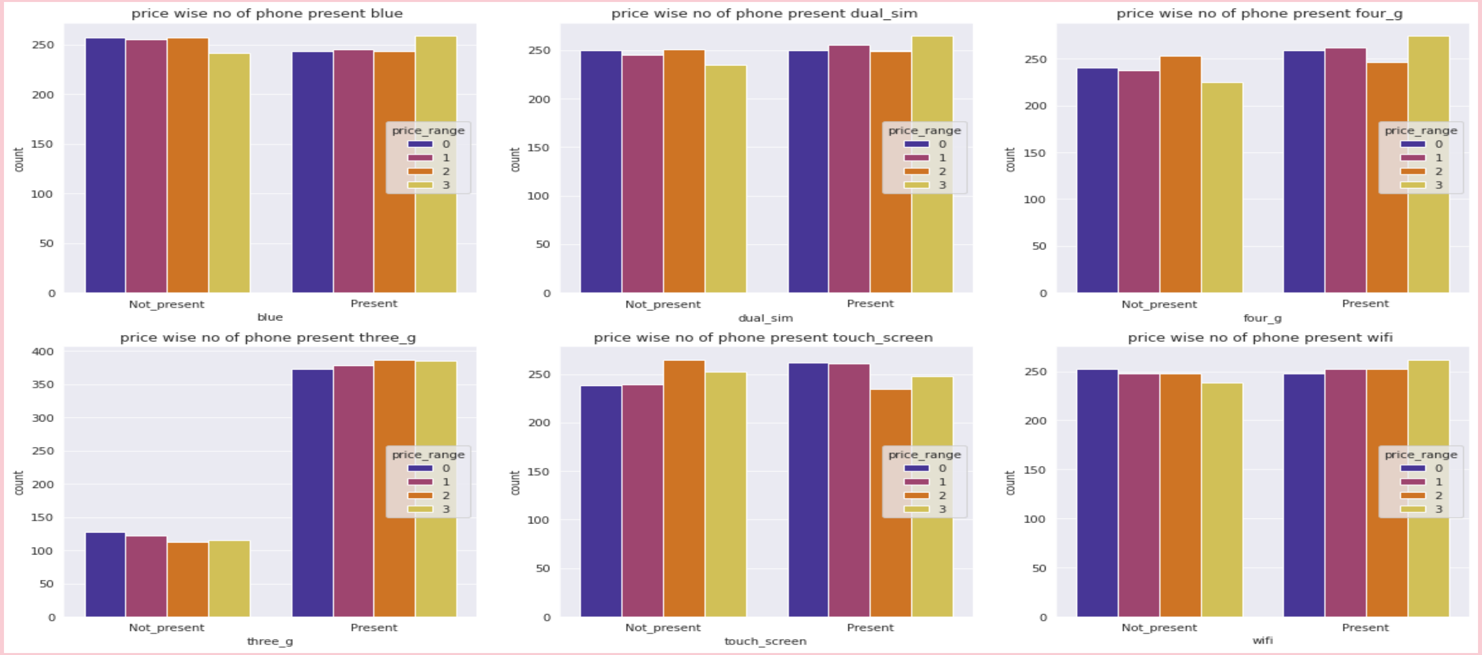
Exploratory Data Analysis:

Price range

there are mobile phones in 4 price ranges. the number of elements are very similar to each others.

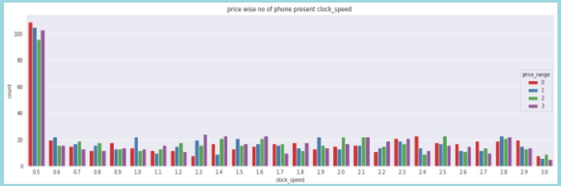


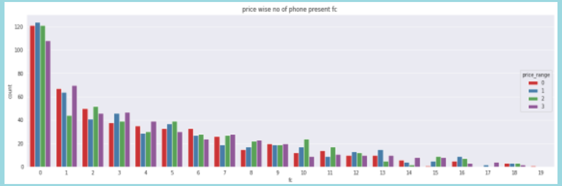
Categorical variables

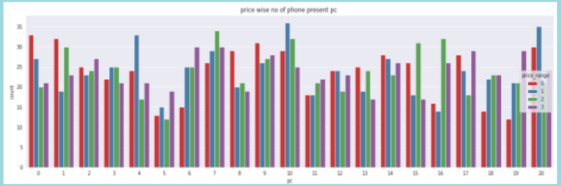


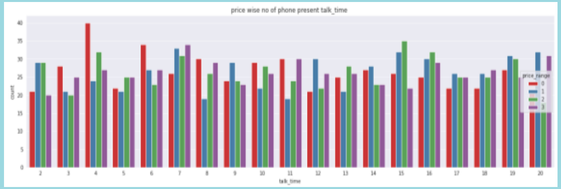
With the exception of 3G, all of the categories (Bluetooth, Dual-SIM, 4G, touch screen, and Wifi) are distributed equally based on their existence rather than absence, in 3G category, 27% of mobile phones do not have 3G, but 63% of phones do have the 3G option.

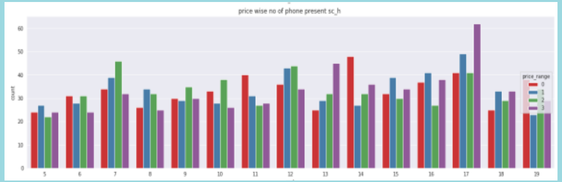
clock_speed, fc, pc, talk_time, sc_h, sc_w, m_dep, int_memory by price range wise

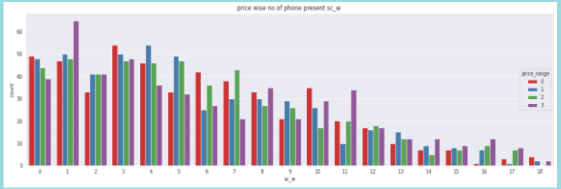


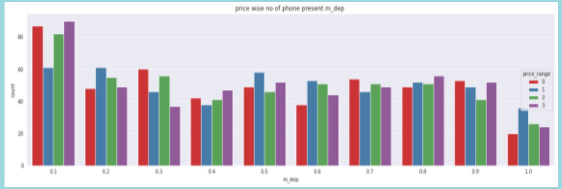


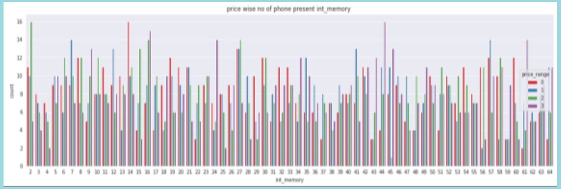






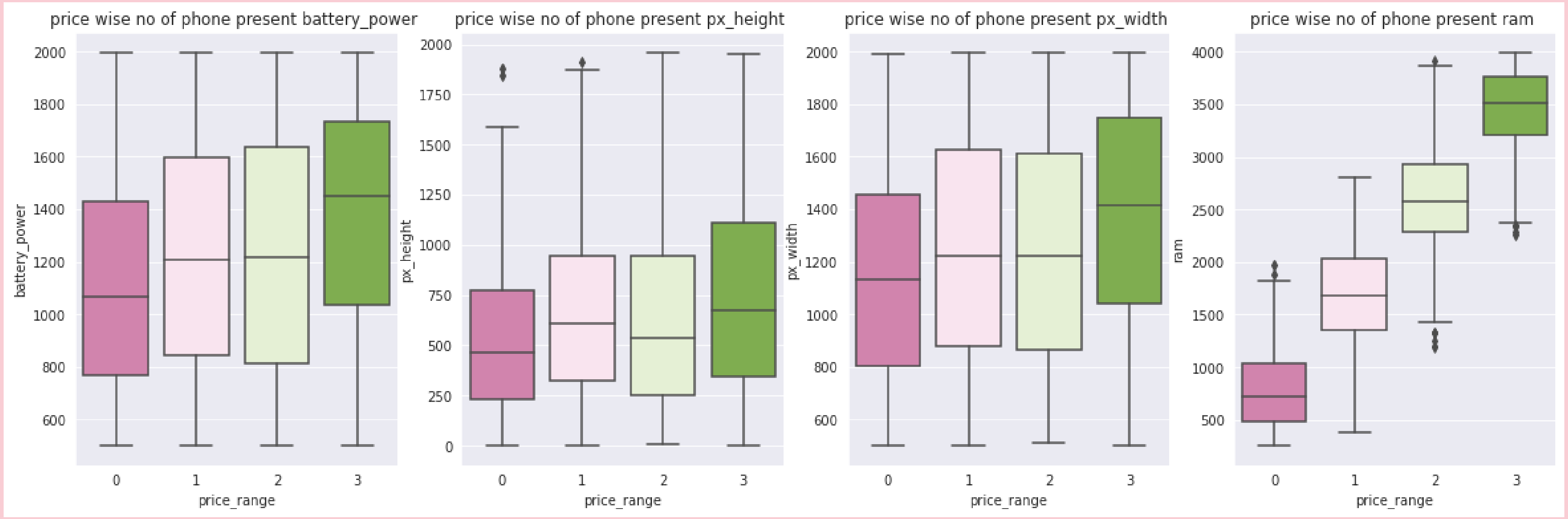






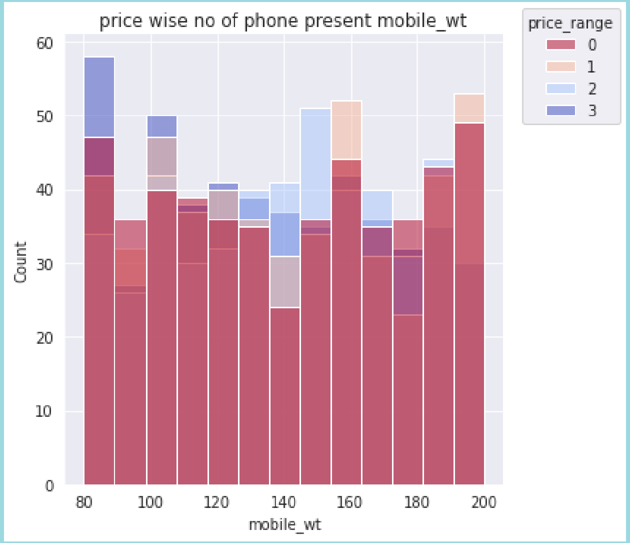
The columns for clock speed, front camera, screen width, and mobile depth are left skewed in the above chart. Other columns (Primary camera, Talk time, and internal memory) have zero-skew, and the screen height is slightly right-skewed.

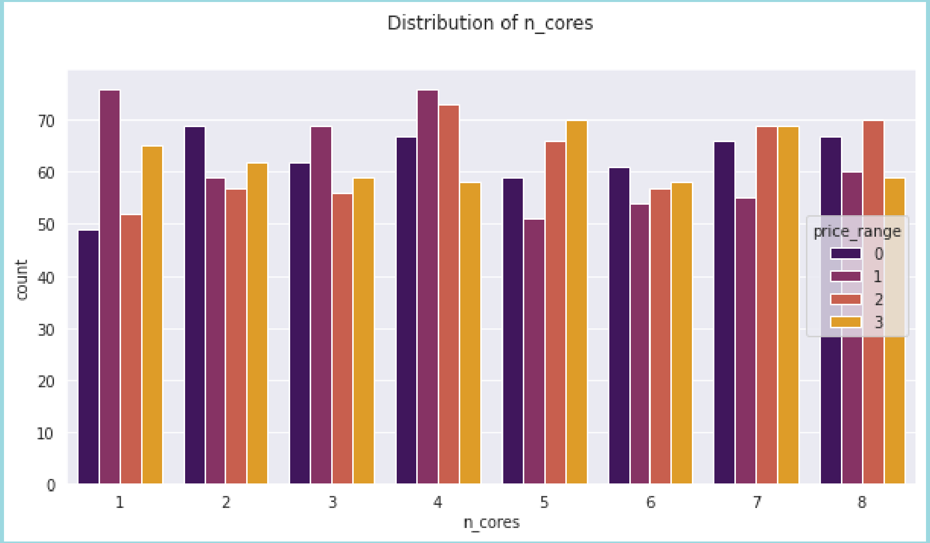
Battery, pixels and ram



According to the above chart, the expensive phone has a large battery life, high-performance RAM, and large pixels both in width and height. High- and medium-priced mobiles almost have the same levels of battery life, RAM, and pixels, although they perform less well than very expensive devices. Compared to the other three categories, low-cost mobile devices' performance is very poor.

Mobile Weight and N_core



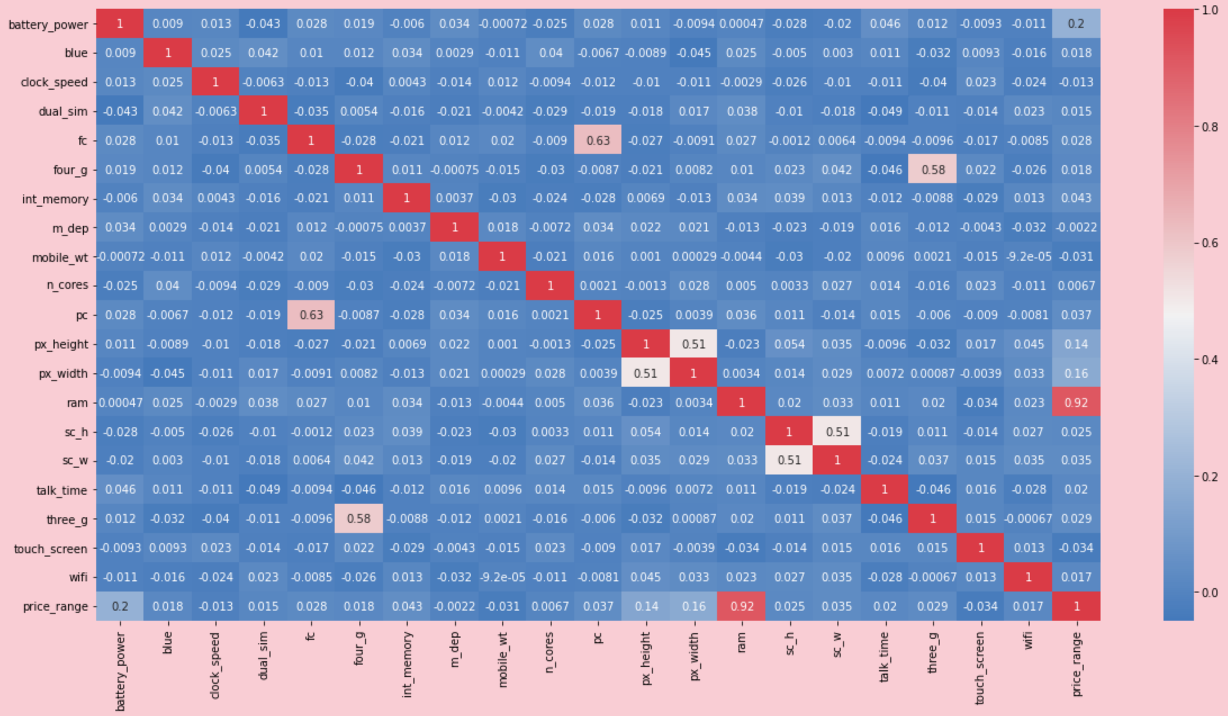


- Phones that weigh between 80 and 100g are very expensive. and the phone, which weights between 160 and 200g are expensive.
- Phone which have 4 N_Cores highly present in data sets but its has minimaly sold in very high cost, But the phone which has 5 N_Core has highly sold in Very high cost. and this N_core column almost Zero-Skewed.

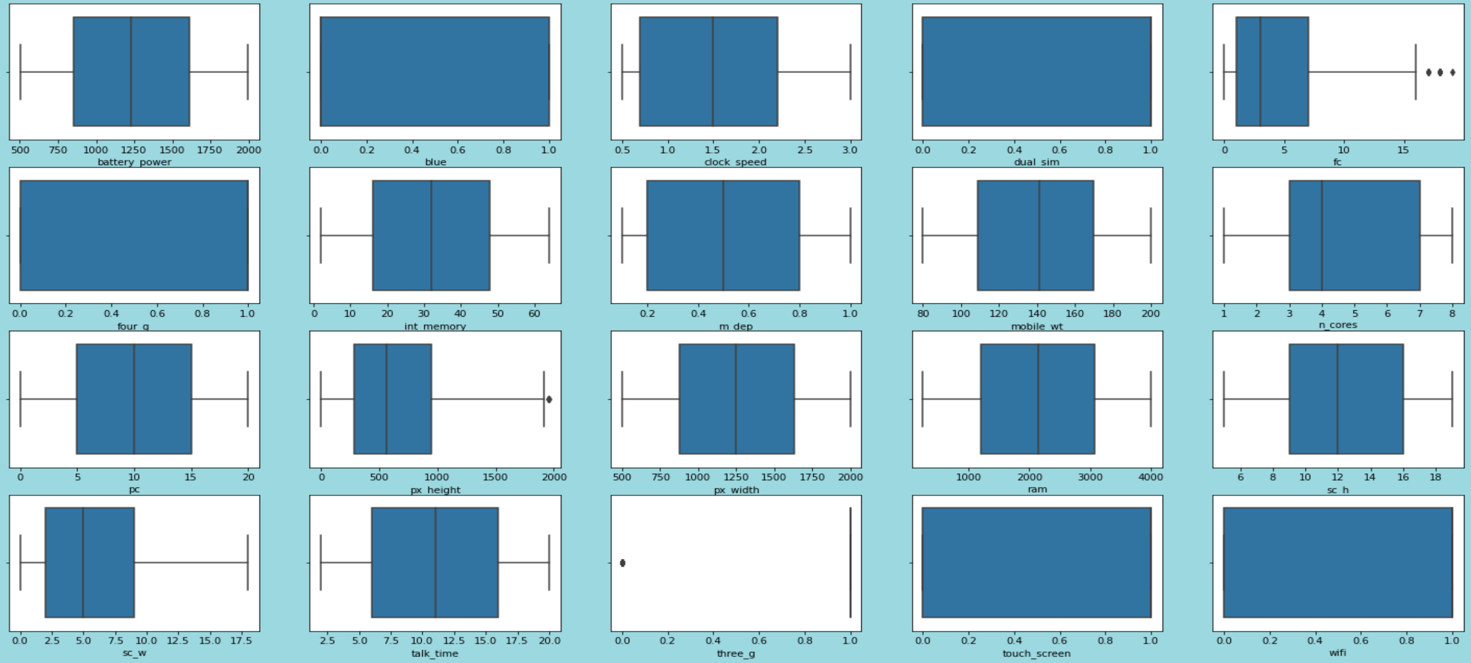
Data Preprocessing:

1. Feature selection

According to the heatmap above, the columns 3g, 4g, fc, and pc are slightly correlated, but ram and price range are highly correlated. However, since the goal of this project is to predict price range, the correlation between ram and price range has no affect on the model. so we leave this.



2. Out layer finding

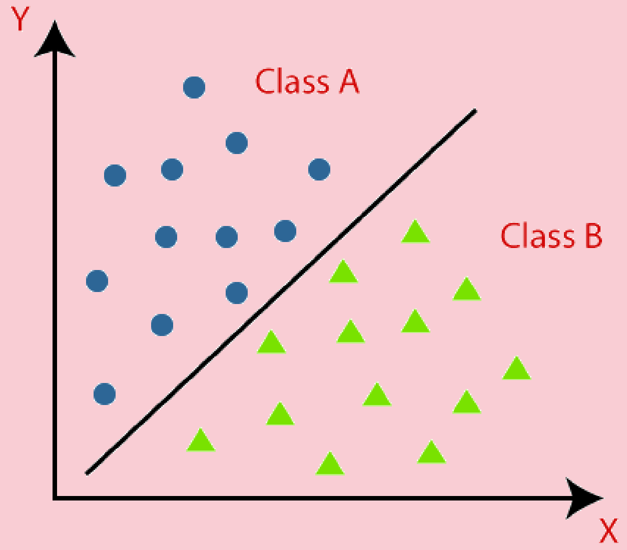


The above chart shows out layer present in each columns, the Front camera and Pixel Height columns has outliers we have the remove that.

Classification models:

What is Classification?

A classification algorithm is a supervised learning technique that uses data training to determine data into different classes. Classification predictive modeling is trained using data or observations, and new observations are categorized into classes or groups. Classification predictive modeling is the task of a mapping function (f) from input variables (x) to discrete output variables (y). In this approach, the algorithm generates a probability score and assigns this score to the input. For example, email service providers use classification to generate probability scores for email identification to determine if the email is in the spam class or not.



Decision Tree Classifier :

Decision Tree is a Supervised learning technique that can be used for both classification and Regression problems, but mostly it is preferred for solving Classification problems. It is a tree-structured classifier, where internal nodes represent the features of a dataset, branches represent the decision rules and each leaf node represents the outcome.

Random Forest Classifier :

Random Forest is a popular machine learning algorithm that belongs to the supervised learning technique. It can be used for both Classification and Regression problems in ML. It is based on the concept of ensemble learning, which is a process of combining multiple classifiers to solve a complex problem and to improve the performance of the model.

Gradient Boosting Classifier :

Gradient boosting algorithm is one of the most powerful algorithms in the field of machine learning. As we know that the errors in machine learning algorithms are broadly classified into two categories i.e. Bias Error and Variance Error. As gradient boosting is one of the boosting algorithms it is used to minimize bias error of the model.

K-Nearest Neighbours Classifier :

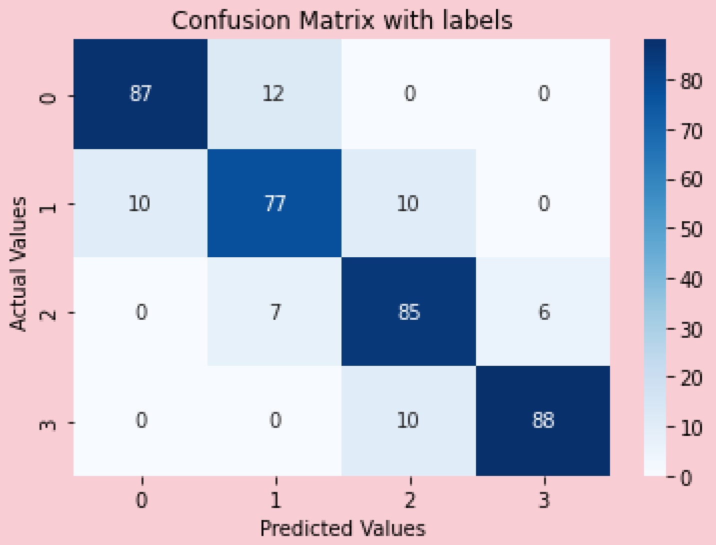
The k-nearest neighbors classifier (kNN) is a non-parametric supervised machine learning algorithm. It’s distance-based: it classifies objects based on their proximate neighbors’ classes. kNN is most often used for classification, but can be applied to regression problems as well

Decision Tree Classifier

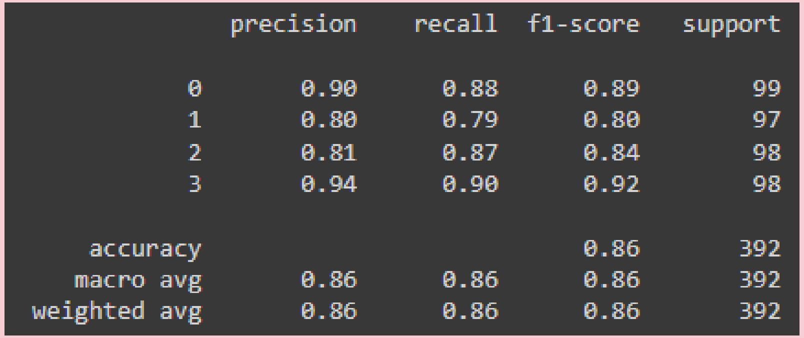
- The overall accuracy score on our test data is 85%.
- Prediction accuracy on Low Cost is 90%.
- Prediction accuracy on Medium Cost is 80%.
- Prediction accuracy on High Cost is 81%.
- Prediction accuracy on Very High Cost is 94%.

we can see that the Decision Tree Classifier has performed good overall, best are low cost and Very High Cost. but in confusion matrix its performance not good in medium cost. and this model trend to overfit.

Model_score : 1.000000
Test_score : 0.859693



$$\begin{array}{c|cccc} & 0 & 1 & 2 & 3 \\ \hline 0 & 87 & 12 & 0 & 0 \\ 1 & 10 & 77 & 10 & 0 \\ 2 & 0 & 7 & 85 & 6 \\ 3 & 0 & 0 & 10 & 88 \end{array}$$

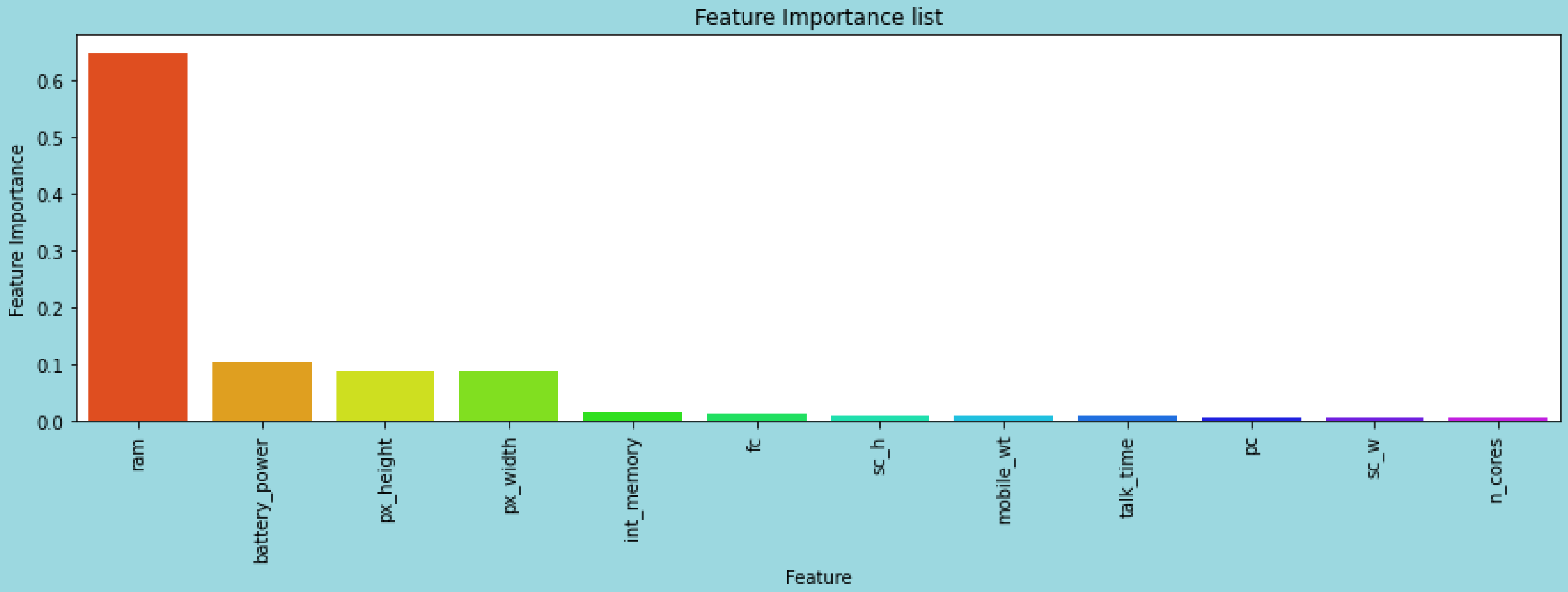


```
              precision    recall  f1-score   support

           0       0.90      0.88      0.89        99
           1       0.80      0.79      0.80        97
           2       0.81      0.87      0.84        98
           3       0.94      0.90      0.92        98

    accuracy                           0.86       392
   macro avg       0.86      0.86      0.86       392
weighted avg       0.86      0.86      0.86       392
```

Feature Importance



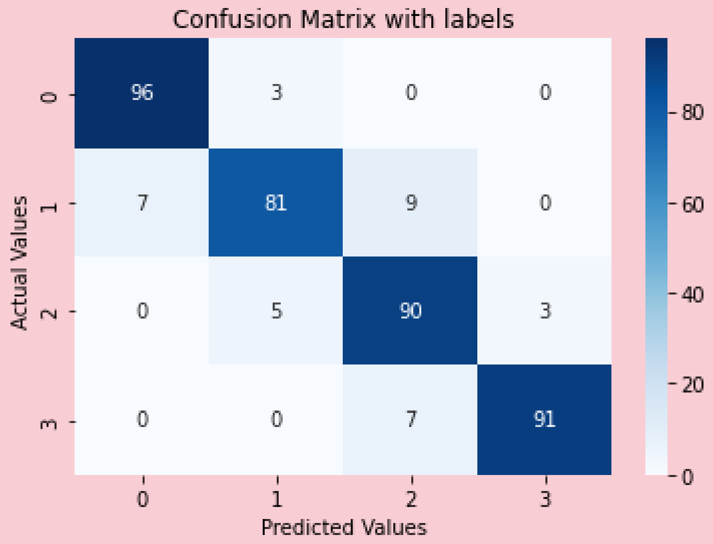
Decision Tree Classifier shows ram, battery power, px height and px width has more importance compare to all other features

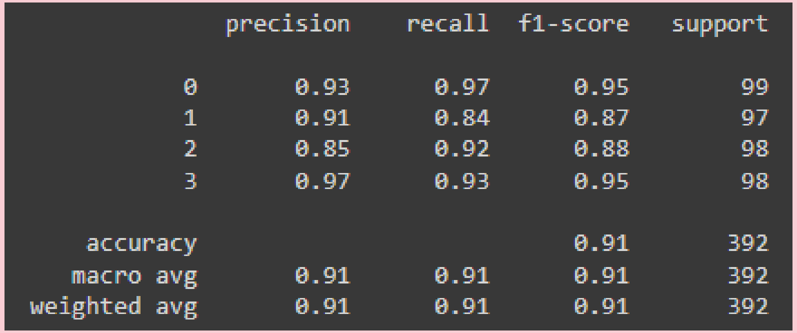
Random Forest Classifier

- The overall accuracy score on our test data is 91%.
- Prediction accuracy on Low Cost is 93%.
- Prediction accuracy on Medium Cost is 91%.
- Prediction accuracy on High Cost is 85%.
- Prediction accuracy on Very High Cost is 97%.

Random Forest Classifier has performed good overall, and best in Low Cost and Very High Cost. this model also trend to overfit.

Model_score : 1.000000
Test_score : 0.913265

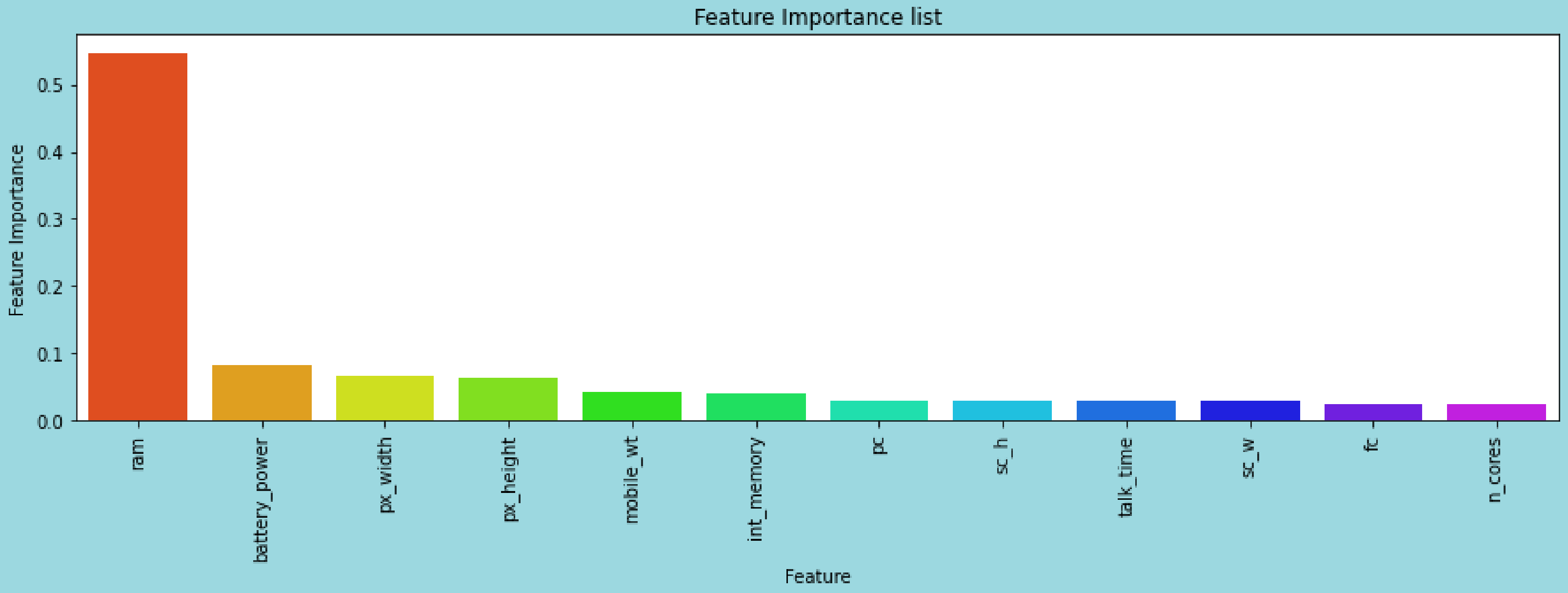



```
              precision    recall  f1-score   support

           0       0.93      0.97      0.95        99
           1       0.91      0.84      0.87        97
           2       0.85      0.92      0.88        98
           3       0.97      0.93      0.95        98

    accuracy                           0.91       392
   macro avg       0.91      0.91      0.91       392
weighted avg       0.91      0.91      0.91       392
```

Feature Importance



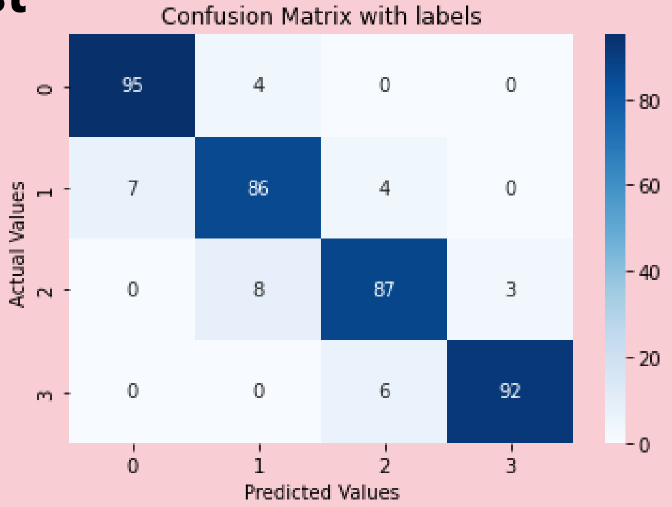
Random Forest Classifier shows ram, battery power, px height, px width, mobile weight and int memory has more importance compare to all other features

Hyperparameter tuning with Random forest

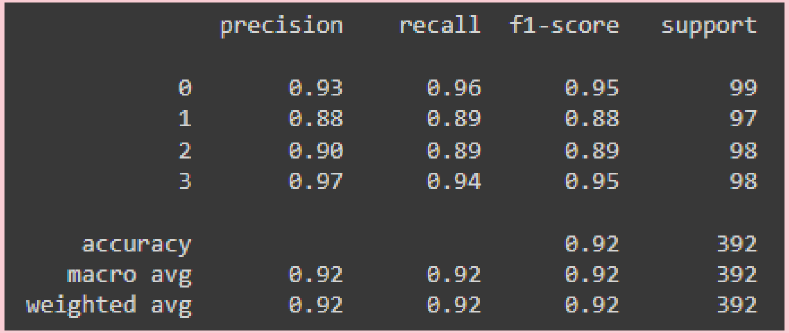
- The overall accuracy score on our test data is 91%.
- Prediction accuracy on Low Cost is 93%.
- Prediction accuracy on Medium Cost is 88%.
- Prediction accuracy on High Cost is 90%.
- Prediction accuracy on Very High Cost is 97%.

Hyperparameter tuning with Random forest has performed good overall, and best in Low Cost and Very High Cost.

Model_score : 1.000000
Test_score : 0.918367



	0	1	2	3
0	95	4	0	0
1	7	86	4	0
2	0	8	87	3
3	0	0	6	92

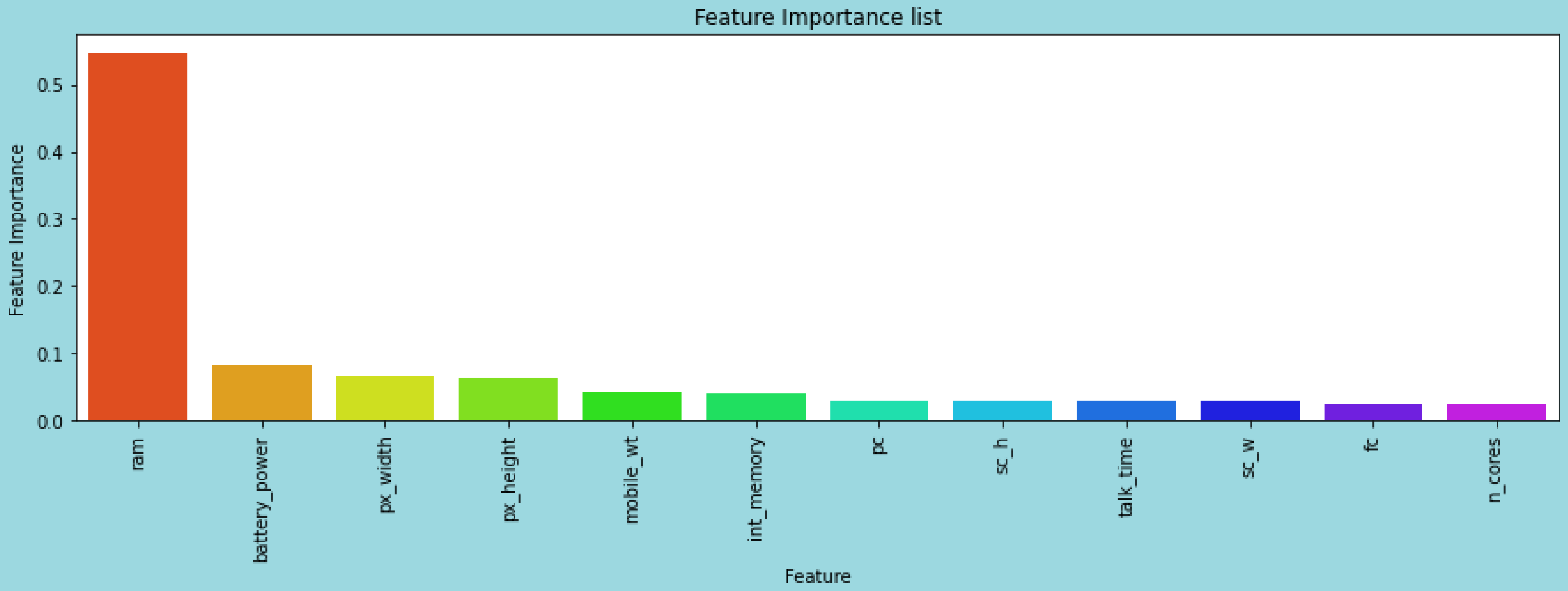


```
              precision    recall  f1-score   support

           0       0.93      0.96      0.95        99
           1       0.88      0.89      0.88        97
           2       0.90      0.89      0.89        98
           3       0.97      0.94      0.95        98

    accuracy                           0.92       392
   macro avg       0.92      0.92      0.92       392
weighted avg       0.92      0.92      0.92       392
```

Feature Importance



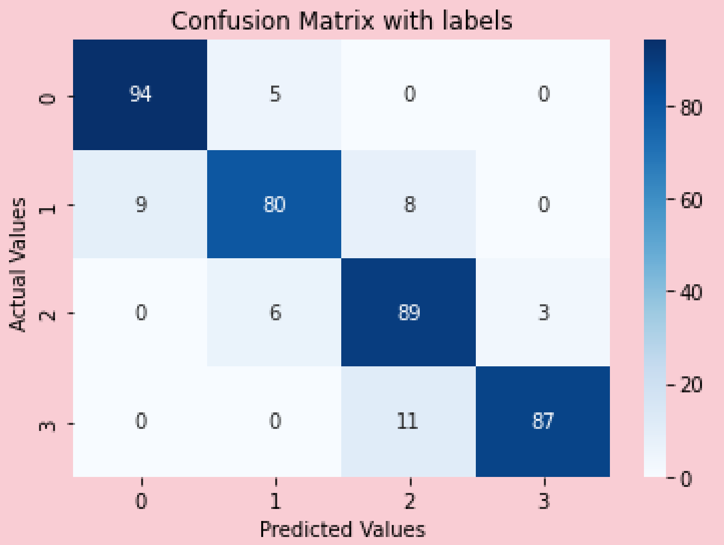
Hyperparameter tuning with Random forest Classifier shows ram, battery power, px height, px width, mobile weight and int memory has more importance compare to all other features same like random forest. Its looks like there nothing change after hyperparameter tuning.

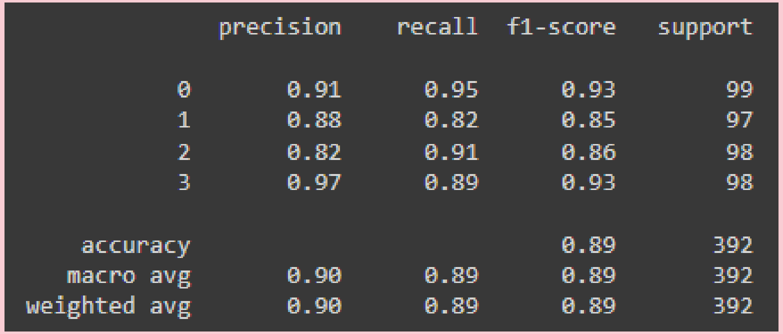
Gradient Boosting Classifier

- The overall accuracy score on our test data is 89%.
- Prediction accuracy on Low Cost is 91%.
- Prediction accuracy on Medium Cost is 88%.
- Prediction accuracy on High Cost is 82%.
- Prediction accuracy on Very High Cost is 97%.

Gradient Boosting Classifier has performed good overall, and best in Very High Cost. but in confusion matrix Medium Cost performance not good.

Model_score : 0.999360
Test_score : 0.892857



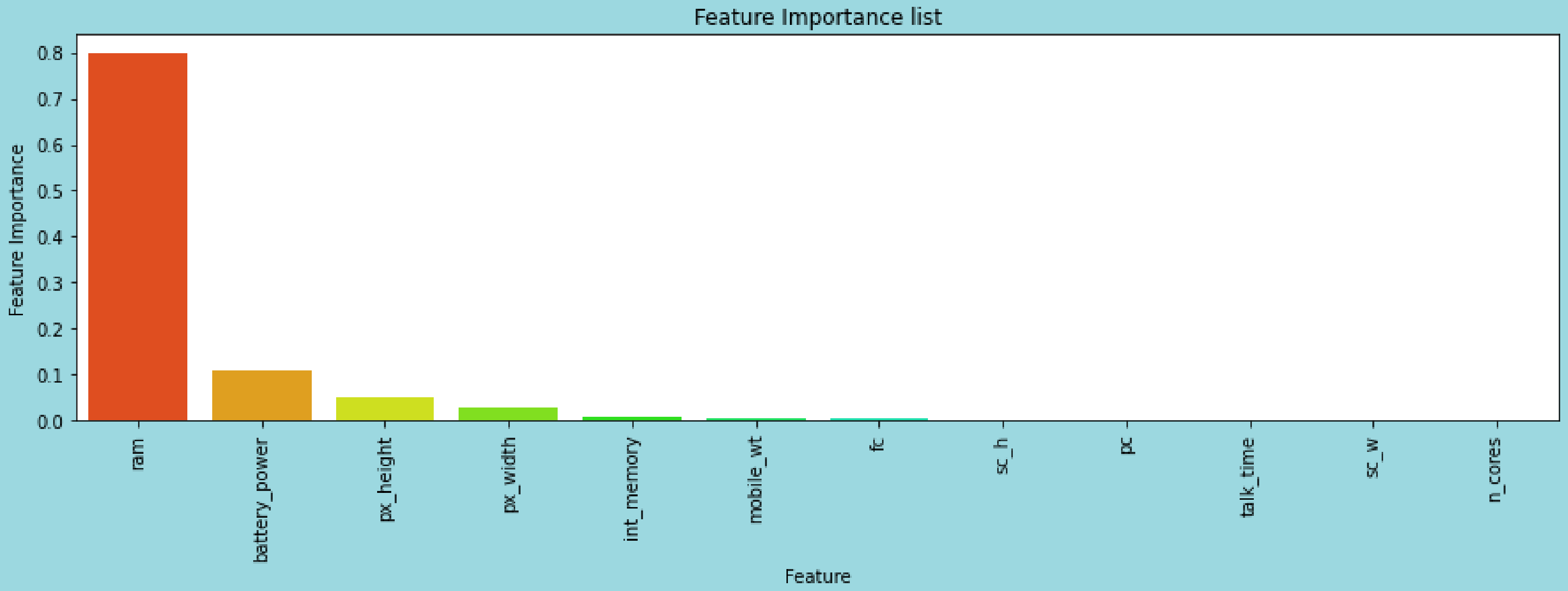


```
              precision    recall  f1-score   support

           0       0.91      0.95      0.93        99
           1       0.88      0.82      0.85        97
           2       0.82      0.91      0.86        98
           3       0.97      0.89      0.93        98

    accuracy                           0.89       392
   macro avg       0.90      0.89      0.89       392
weighted avg       0.90      0.89      0.89       392
```

Feature Importance



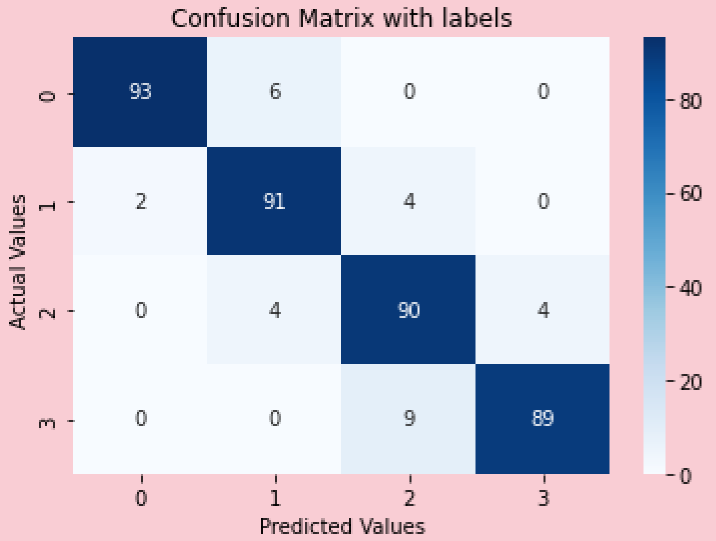
Gradient Boosting Classifier shows ram, battery power and px height has more importance compare to all other features.

K-Nearest Neighbors Classifier

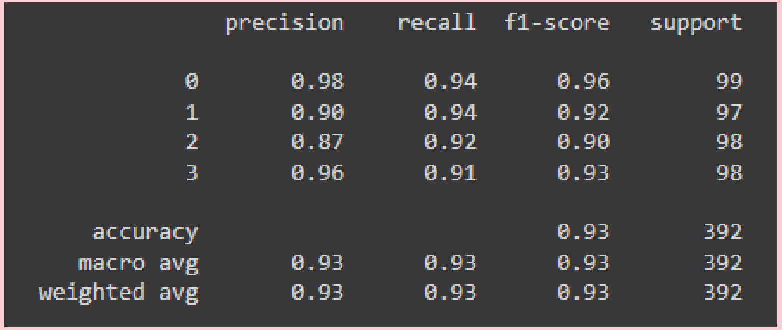
- The overall accuracy score on our test data is 92%.
- Prediction accuracy on Low Cost is 98%.
- Prediction accuracy on Medium Cost is 90%.
- Prediction accuracy on High Cost is 87%.
- Prediction accuracy on Very High Cost is 96%.

K-Nearest Neighbours has performed good overall, and its best are Low Cost and Medium Cost.

Model_score : 0.953964
Test_score : 0.926020



	0	1	2	3
0	93	6	0	0
1	2	91	4	0
2	0	4	90	4
3	0	0	9	89

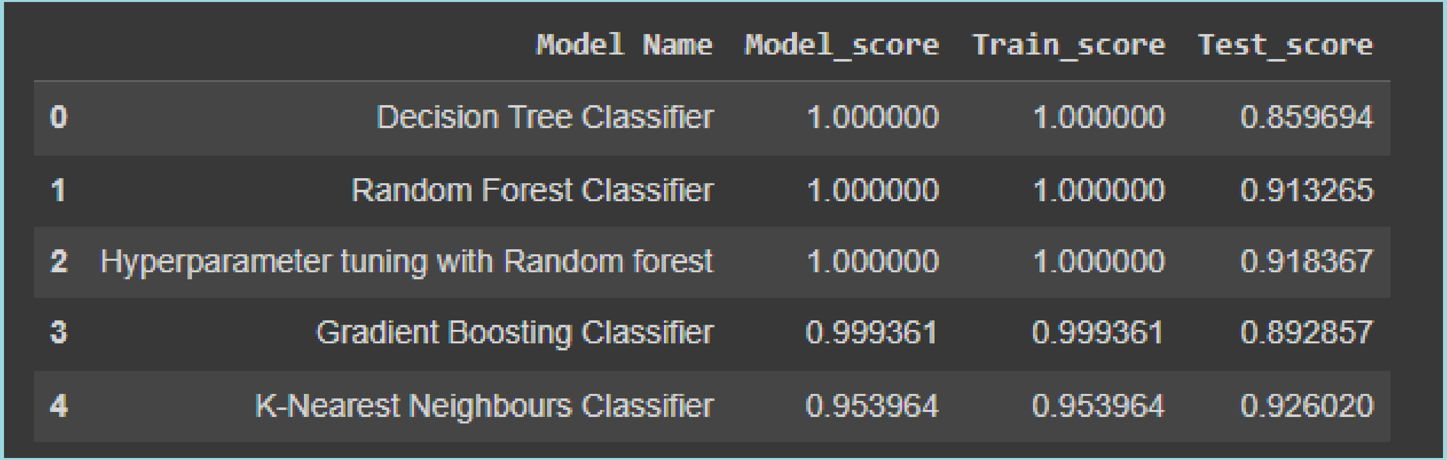


```
              precision    recall  f1-score   support

           0       0.98      0.94      0.96        99
           1       0.90      0.94      0.92        97
           2       0.87      0.92      0.90        98
           3       0.96      0.91      0.93        98

    accuracy                           0.93       392
   macro avg       0.93      0.93      0.93       392
weighted avg       0.93      0.93      0.93       392
```

Conclusions:



	Model Name	Model_score	Train_score	Test_score
0	Decision Tree Classifier	1.000000	1.000000	0.859694
1	Random Forest Classifier	1.000000	1.000000	0.913265
2	Hyperparameter tuning with Random forest	1.000000	1.000000	0.918367
3	Gradient Boosting Classifier	0.999361	0.999361	0.892857
4	K-Nearest Neighbours Classifier	0.953964	0.953964	0.926020

We tried a variety of models, and the table above summarizes the results of one set of models. K-Nearest Neighbours and Random Forest Classifier has the best overall accuracy of 92 and 91 percents, respectively. The optimal accuracy for Hyperparameter tunned Random forest, Decision Tree, Gradient Boosting was 91 percent, 80 percent, and 89 percent respectively. However, we'll make K-Nearest Neighbours as our best model because it provides good overall and individual class accuracy.

Thanks!