Capstone Project-3

Mobile Price Range Prediction

Navin Kodam

Content :

The Dilemma :

How price range prediction works :

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 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

Data Summary contd..

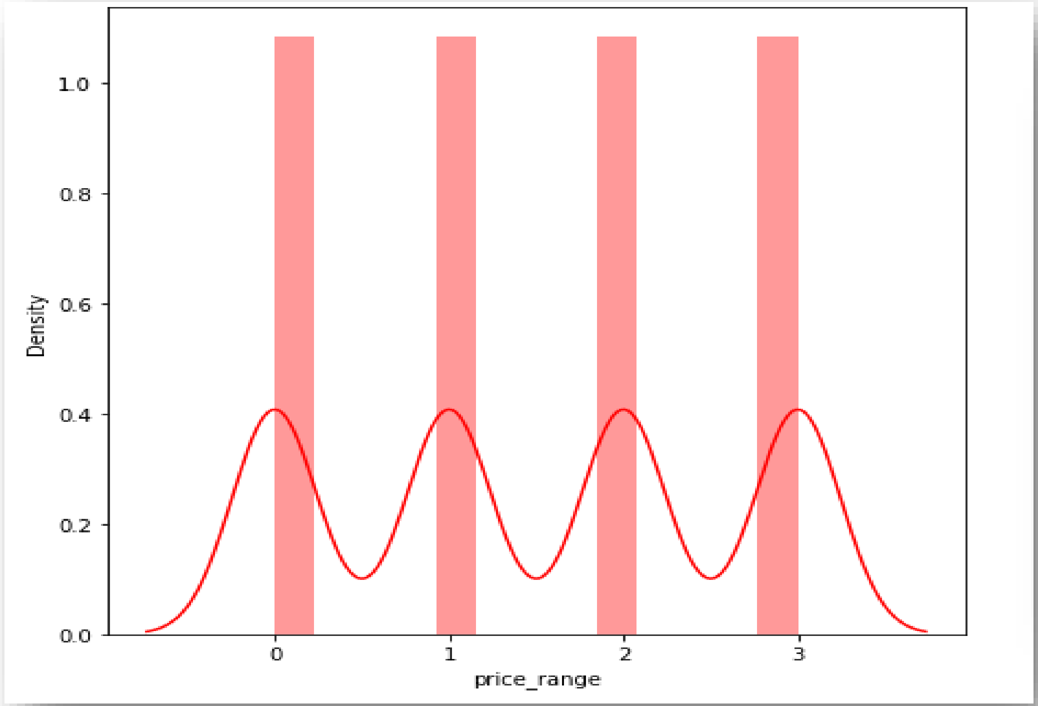
- **Mobile_wt** - Weight of mobile phone
- **N_cores** - Number of cores of processor
- **Pc** - Primary Camera mega pixels
- **Px_height** - Pixel Resolution Height
- **Px_width** - Pixel Resolution Width
- **Ram** - Random Access Memory in Mega Bytes
- **Sc_h** - Screen Height of mobile in cm
- **Sc_w** - Screen Width of mobile in cm
- **Talk_time** - longest time that a single battery charge will last when you are

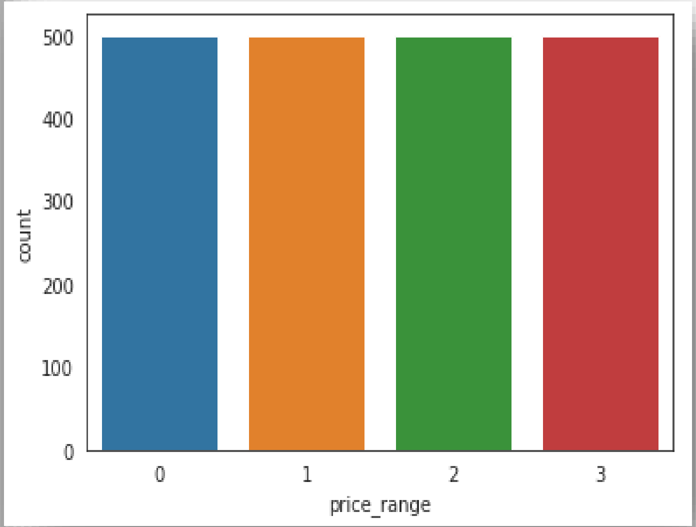
Data Summary contd..

Three_g - Has 3G or not
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).

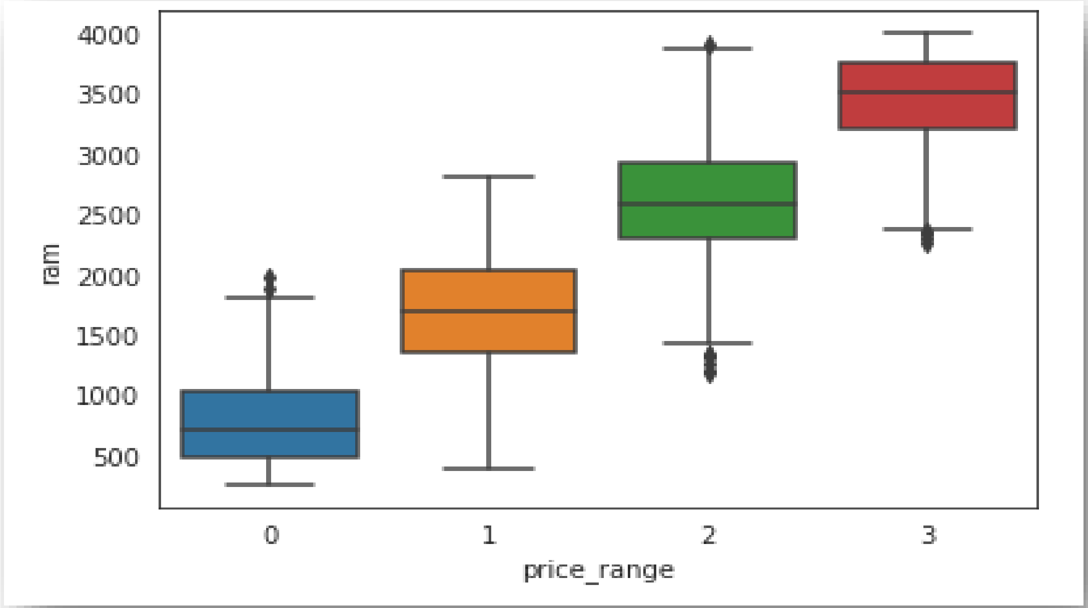
Define Dependent variable :





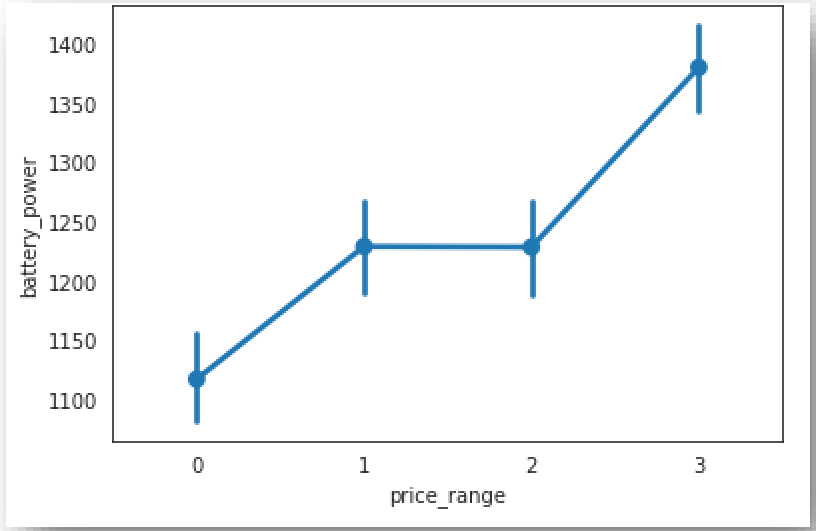
EDA

Ram vs price_range



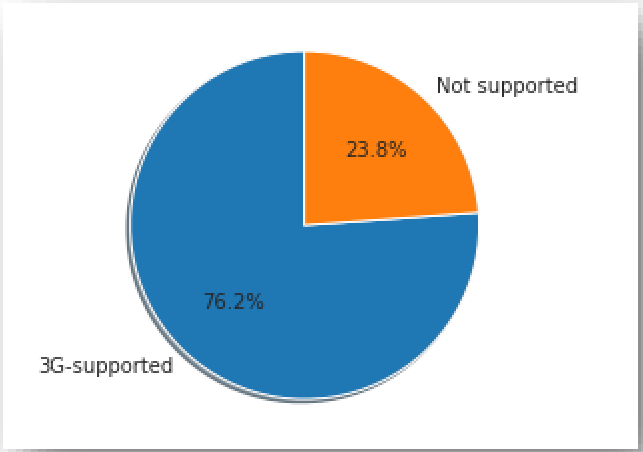
EDA contd...

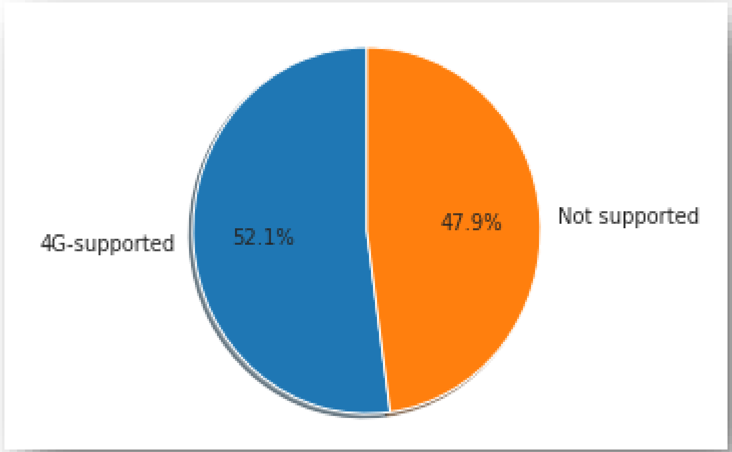
Battery_power vs price_range



EDA contd..

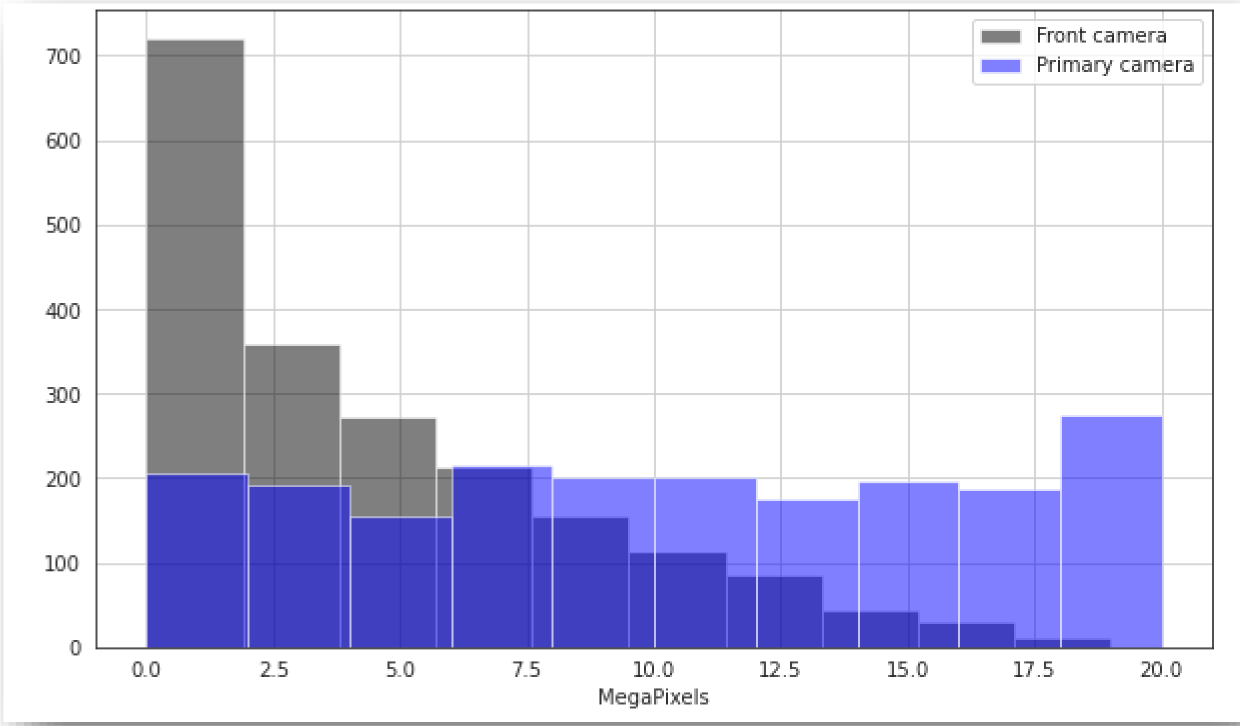
- **3G-4G supported and Non-supported**



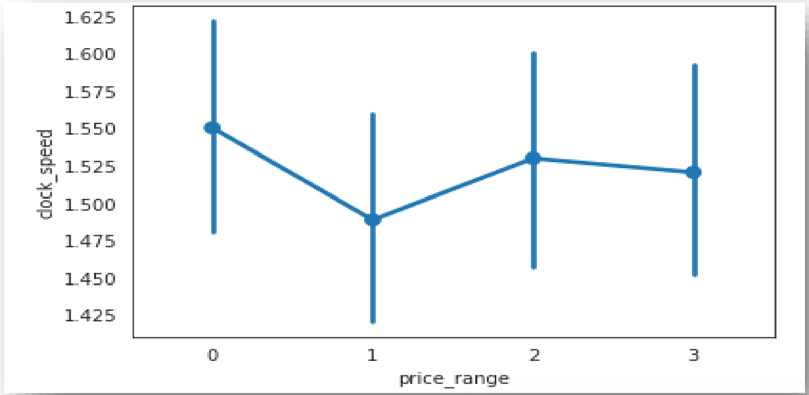


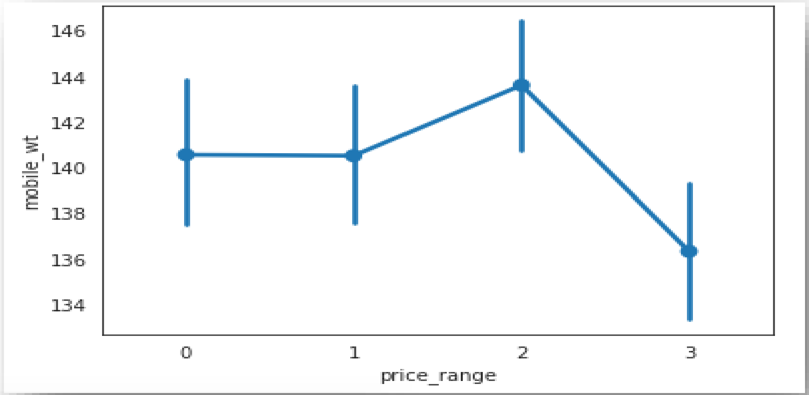
EDA contd...

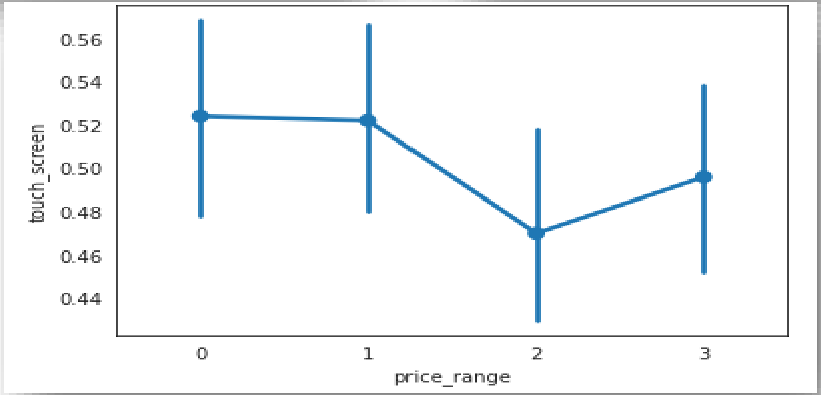
Histogram of Front and primary Camera in MegaPixels



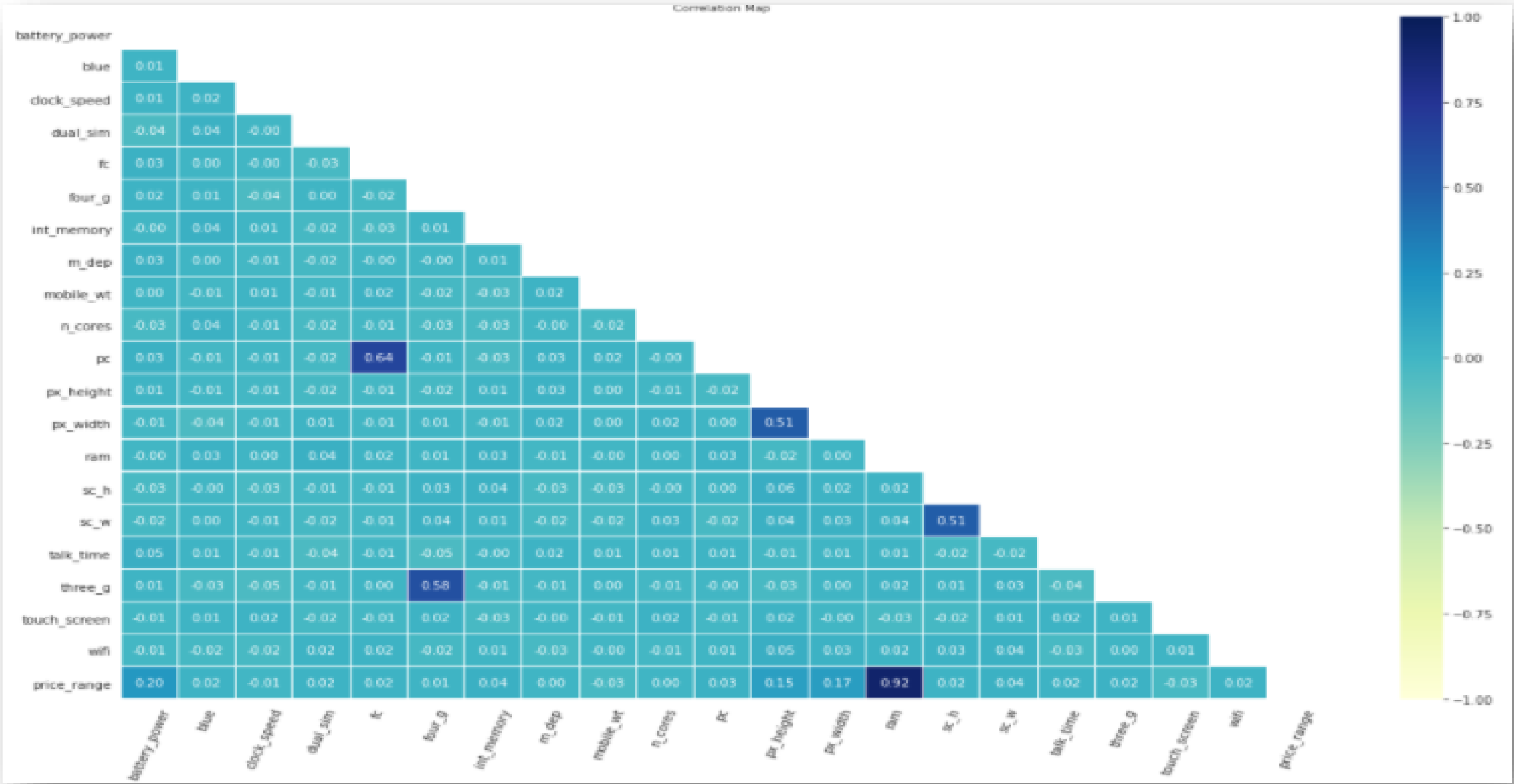
EDA contd...(Plotting negative corr features)







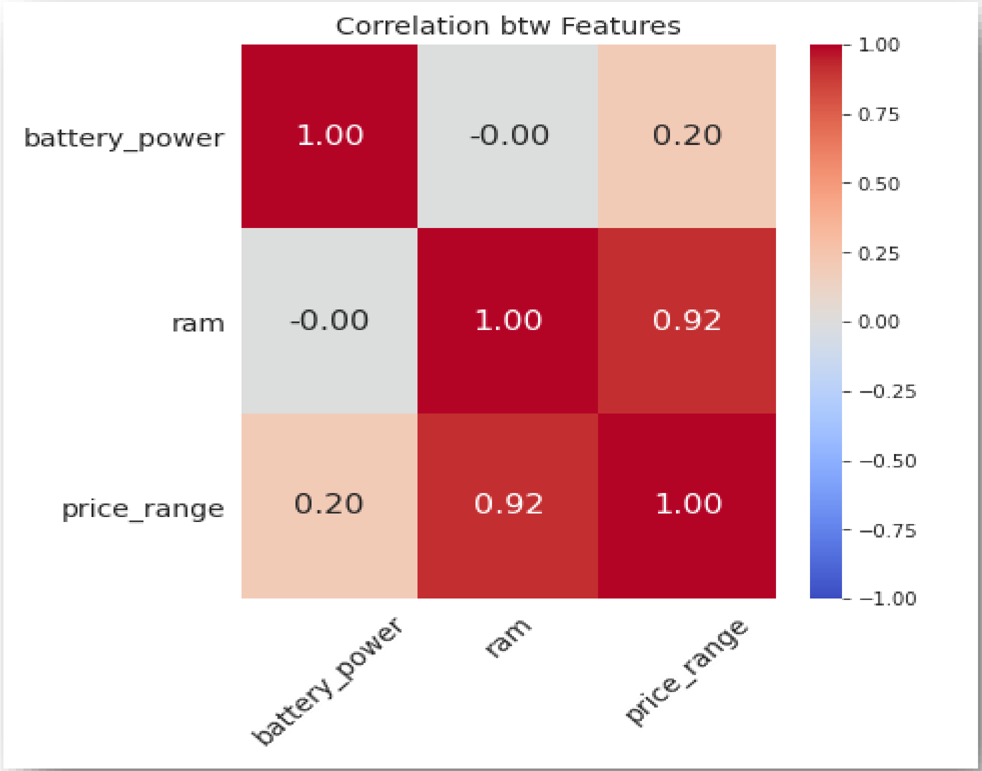
EDA contd...



	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																					
blue	0.01																				
clock_speed	0.01	0.02																			
dual_sim	-0.04	0.04	-0.00																		
fc	0.03	0.00	-0.00	-0.03																	
four_g	0.02	0.01	-0.04	0.00	-0.02																
int_memory	-0.00	0.04	0.01	-0.02	-0.03	0.01															
m_dep	0.03	0.00	-0.01	-0.02	-0.00	-0.00	0.01														
mobile_wt	0.00	-0.01	0.01	-0.01	0.02	-0.02	-0.03	0.02													
n_cores	-0.03	0.04	-0.01	-0.02	-0.01	-0.03	-0.03	-0.00	-0.02												
pc	0.03	-0.01	-0.01	-0.02	0.64	-0.01	-0.03	0.03	0.02	-0.00											
px_height	0.01	-0.01	-0.01	-0.02	-0.01	-0.02	0.01	0.03	0.00	-0.01	-0.02										
px_width	-0.01	-0.04	-0.01	0.01	-0.01	0.01	-0.01	0.02	0.00	0.02	0.00	0.51									
ram	-0.00	0.03	0.00	0.04	0.02	0.01	0.03	-0.01	-0.00	0.00	0.03	-0.02	0.00								
sc_h	-0.03	-0.00	-0.03	-0.01	-0.01	0.03	0.04	-0.03	-0.03	-0.00	0.00	0.06	0.02	0.02							
sc_w	-0.02	0.00	-0.01	-0.02	-0.01	0.04	0.01	-0.02	-0.02	0.03	-0.02	0.04	0.03	0.04	0.51						
talk_time	0.05	0.01	-0.01	-0.04	-0.01	-0.05	-0.00	0.02	0.01	0.01	0.01	-0.01	0.01	0.01	-0.02	-0.02					
three_g	0.01	-0.03	-0.05	-0.01	0.00	0.58	-0.01	-0.01	0.00	-0.01	-0.00	-0.03	0.00	0.02	0.01	0.03	-0.04				
touch_screen	-0.01	0.01	0.02	-0.02	-0.01	0.02	-0.03	-0.00	-0.01	0.02	-0.01	0.02	-0.00	-0.03	-0.02	0.01	0.02	0.01			
wifi	-0.01	-0.02	-0.02	0.02	0.02	-0.02	0.01	-0.03	-0.00	-0.01	0.01	0.05	0.03	0.02	0.03	0.04	-0.03	0.00	0.01		
price_range	0.20	0.02	-0.01	0.02	0.02	0.01	0.04	0.00	-0.03	0.00	0.03	0.15	0.17	0.92	0.02	0.04	0.02	0.02	-0.03	0.02	

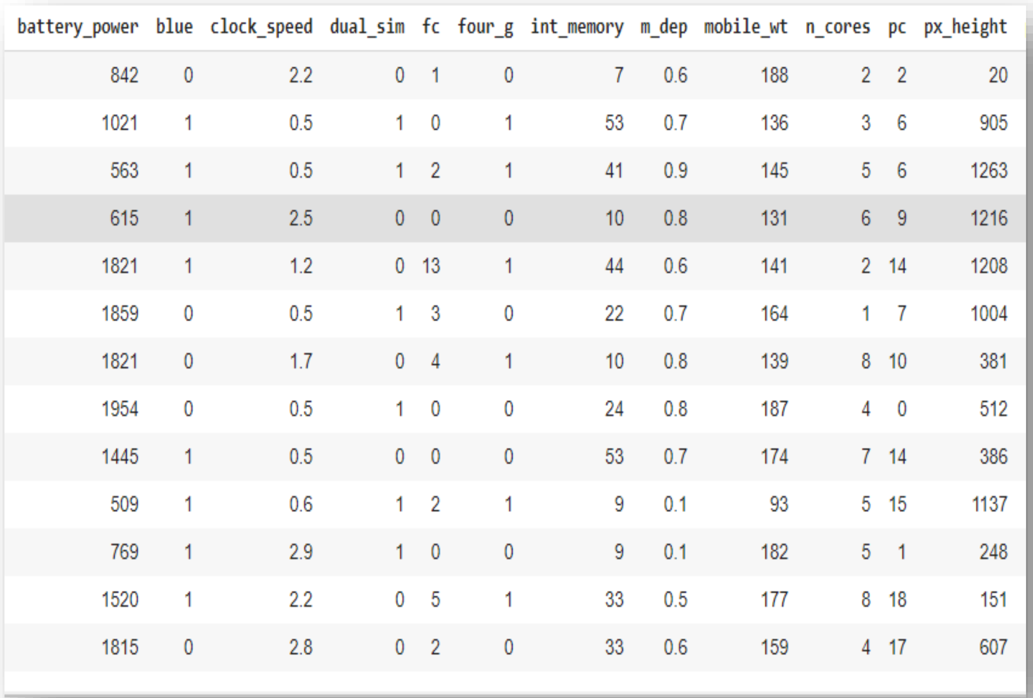
EDA contd...

Heatmap of maximum Correlation features



Preparing dataset for modeling

Task : multiclass classification
Train set : (1340 , 17)
Test set : (660 , 17)
Response : 0-1-2-3

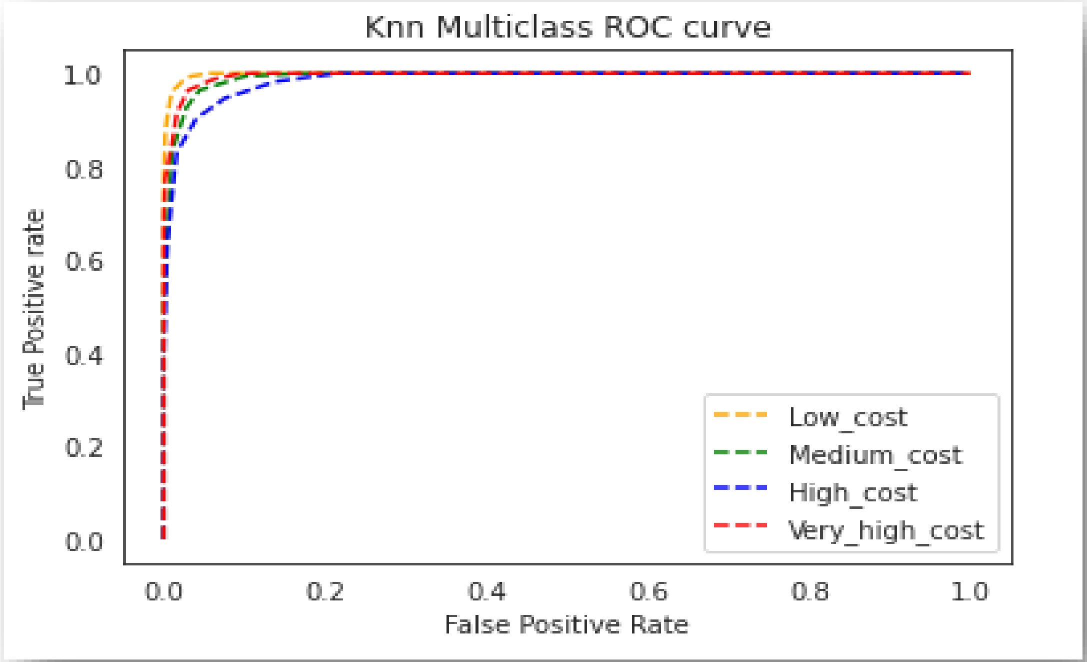


battery_power	blue	clock_speed	dual_sim	fc	four_g	int_memory	m_dep	mobile_wt	n_cores	pc	px_height
842	0	2.2	0	1	0	7	0.6	188	2	2	20
1021	1	0.5	1	0	1	53	0.7	136	3	6	905
563	1	0.5	1	2	1	41	0.9	145	5	6	1263
615	1	2.5	0	0	0	10	0.8	131	6	9	1216
1821	1	1.2	0	13	1	44	0.6	141	2	14	1208
1859	0	0.5	1	3	0	22	0.7	164	1	7	1004
1821	0	1.7	0	4	1	10	0.8	139	8	10	381
1954	0	0.5	1	0	0	24	0.8	187	4	0	512
1445	1	0.5	0	0	0	53	0.7	174	7	14	386
509	1	0.6	1	2	1	9	0.1	93	5	15	1137
769	1	2.9	1	0	0	9	0.1	182	5	1	248
1520	1	2.2	0	5	1	33	0.5	177	8	18	151
1815	0	2.8	0	2	0	33	0.6	159	4	17	607

Implementing KNeighbours Classifier

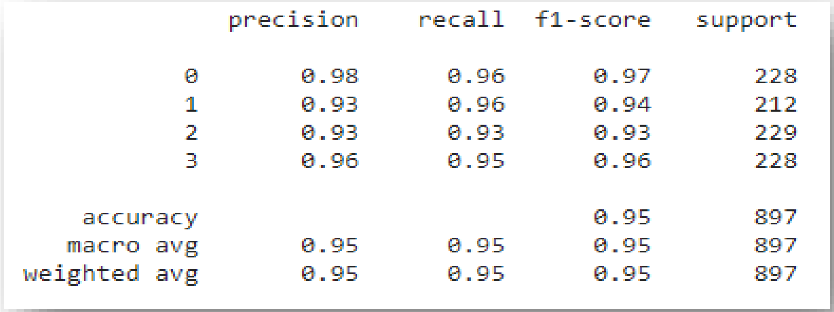
TPR = TP/(TP+FN)

FPR = FP/(FP+TN)



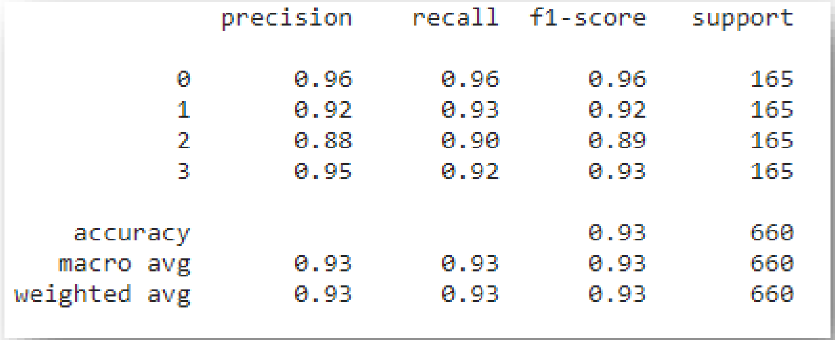
Implementing KNeighbours Classifier contd.

Train metrics

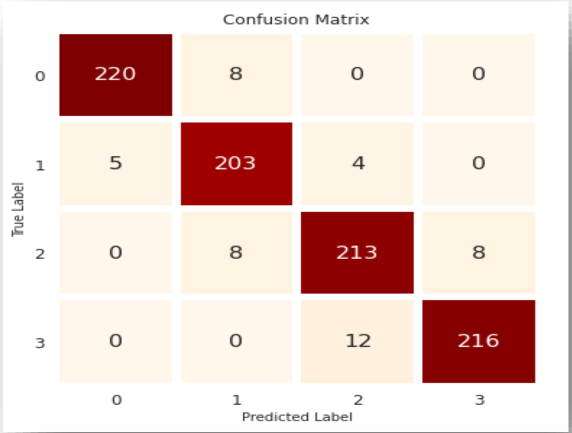


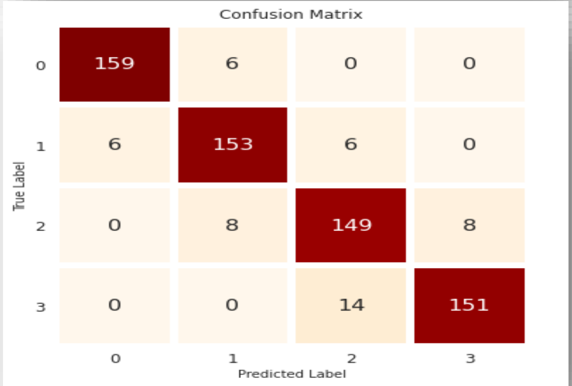
```
              precision    recall  f1-score   support

           0       0.98      0.96      0.97       228
           1       0.93      0.96      0.94       212
           2       0.93      0.93      0.93       229
           3       0.96      0.95      0.96       228

    accuracy                           0.95       897
   macro avg       0.95      0.95      0.95       897
weighted avg       0.95      0.95      0.95       897
```

Test metrics



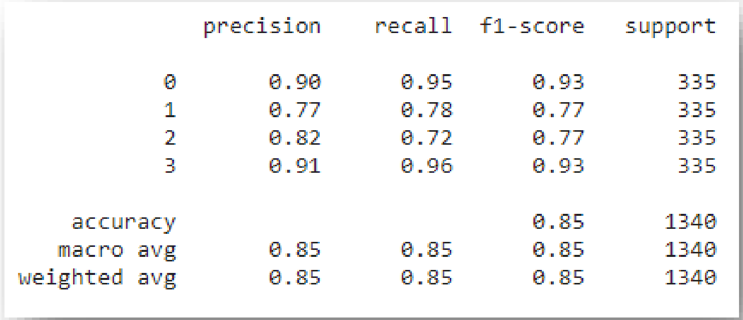
```
              precision    recall  f1-score   support

           0       0.96      0.96      0.96       165
           1       0.92      0.93      0.92       165
           2       0.88      0.90      0.89       165
           3       0.95      0.92      0.93       165

    accuracy                           0.93       660
   macro avg       0.93      0.93      0.93       660
weighted avg       0.93      0.93      0.93       660
```



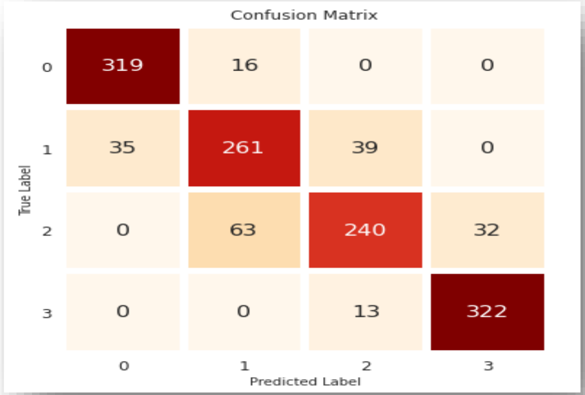


Implementing Random Forest Classifier

Train metrics

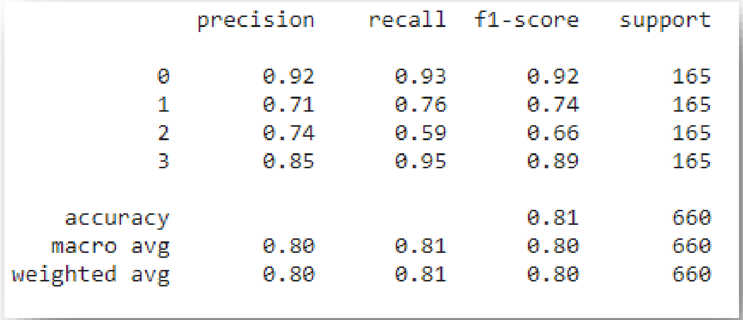


```
              precision    recall  f1-score   support

           0       0.90      0.95      0.93       335
           1       0.77      0.78      0.77       335
           2       0.82      0.72      0.77       335
           3       0.91      0.96      0.93       335

    accuracy                           0.85      1340
   macro avg       0.85      0.85      0.85      1340
weighted avg       0.85      0.85      0.85      1340
```

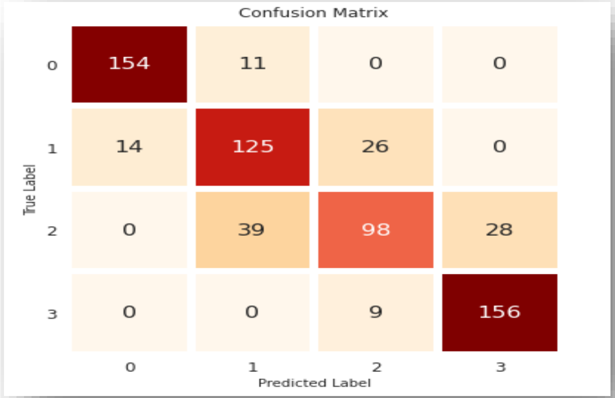


$$\begin{array}{c|cccc} & 0 & 1 & 2 & 3 \\ \hline 0 & 319 & 16 & 0 & 0 \\ 1 & 35 & 261 & 39 & 0 \\ 2 & 0 & 63 & 240 & 32 \\ 3 & 0 & 0 & 13 & 322 \end{array}$$

Test metrics



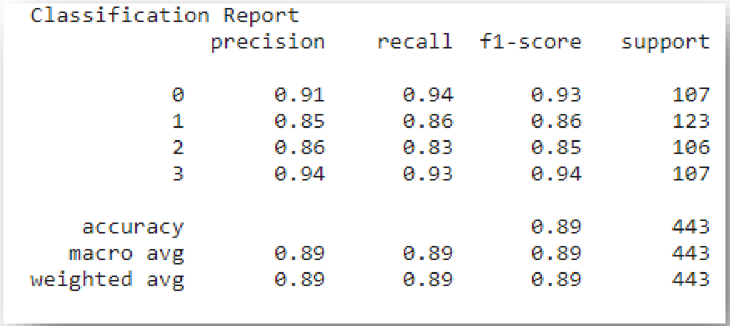
```
              precision    recall  f1-score   support

           0       0.92      0.93      0.92       165
           1       0.71      0.76      0.74       165
           2       0.74      0.59      0.66       165
           3       0.85      0.95      0.89       165

    accuracy                           0.81       660
   macro avg       0.80      0.81      0.80       660
weighted avg       0.80      0.81      0.80       660
```



$$\begin{array}{c|cccc} & 0 & 1 & 2 & 3 \\ \hline 0 & 154 & 11 & 0 & 0 \\ 1 & 14 & 125 & 26 & 0 \\ 2 & 0 & 39 & 98 & 28 \\ 3 & 0 & 0 & 9 & 156 \end{array}$$

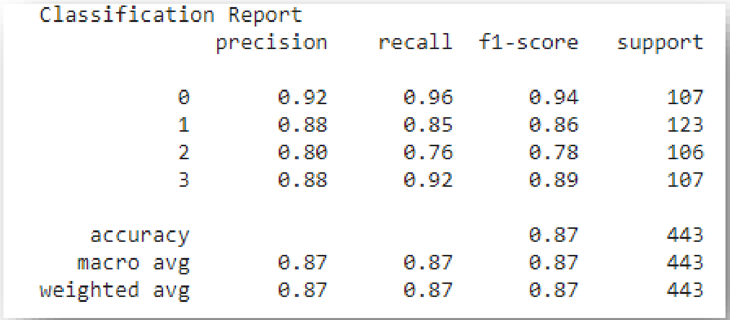
Implementing GradientBoostingClassifier

Train metrics

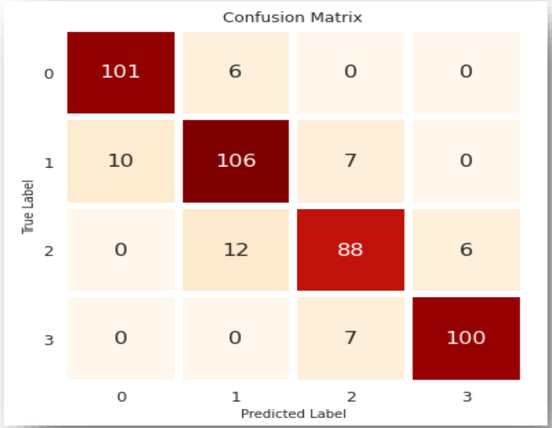


```
Classification Report
              precision    recall  f1-score   support

           0       0.91      0.94      0.93       107
           1       0.85      0.86      0.86       123
           2       0.86      0.83      0.85       106
           3       0.94      0.93      0.94       107

    accuracy                           0.89       443
   macro avg       0.89      0.89      0.89       443
weighted avg       0.89      0.89      0.89       443
```

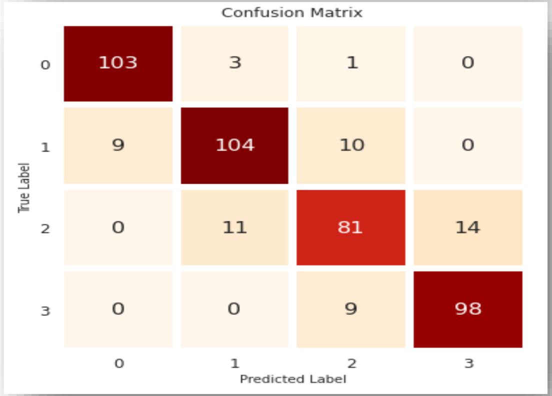
Test metrics



```
Classification Report
              precision    recall  f1-score   support

           0       0.92      0.96      0.94       107
           1       0.88      0.85      0.86       123
           2       0.80      0.76      0.78       106
           3       0.88      0.92      0.89       107

    accuracy                           0.87       443
   macro avg       0.87      0.87      0.87       443
weighted avg       0.87      0.87      0.87       443
```



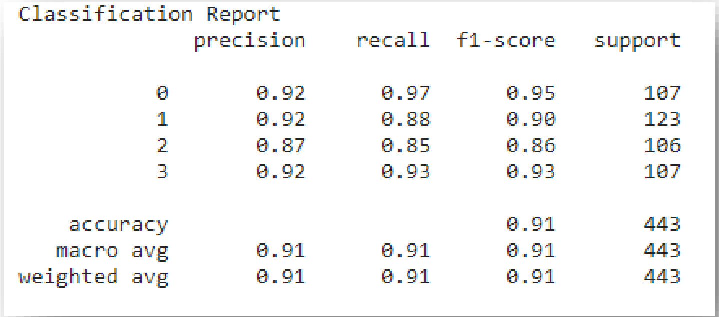
	0	1	2	3
0	101	6	0	0
1	10	106	7	0
2	0	12	88	6
3	0	0	7	100



	0	1	2	3
0	103	3	1	0
1	9	104	10	0
2	0	11	81	14
3	0	0	9	98

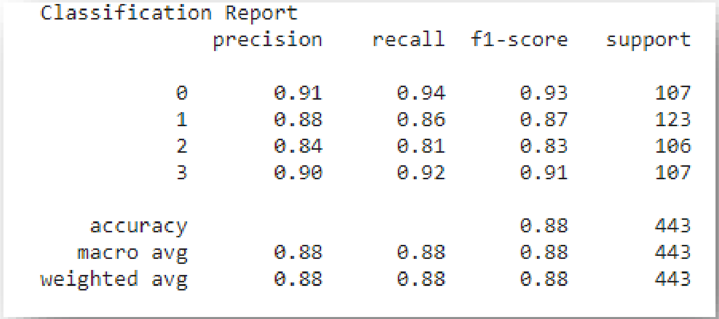
Implementing XGBClassifier

Train metrics

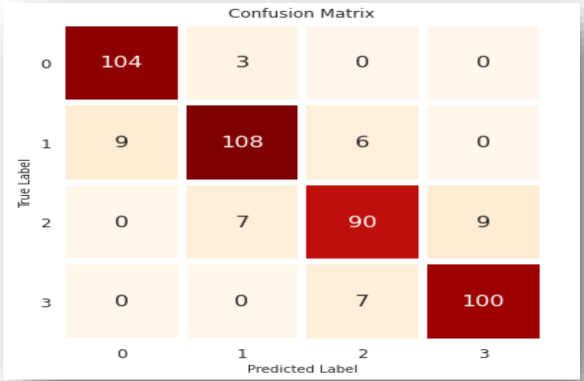


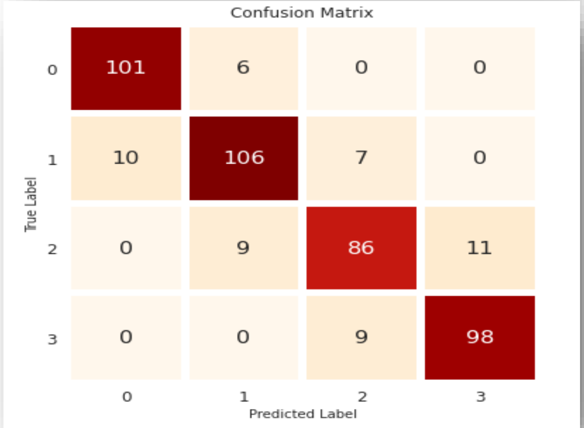
```
Classification Report
              precision    recall  f1-score   support

           0       0.92      0.97      0.95       107
           1       0.92      0.88      0.90       123
           2       0.87      0.85      0.86       106
           3       0.92      0.93      0.93       107

    accuracy                           0.91       443
   macro avg       0.91      0.91      0.91       443
weighted avg       0.91      0.91      0.91       443
```

Test metrics



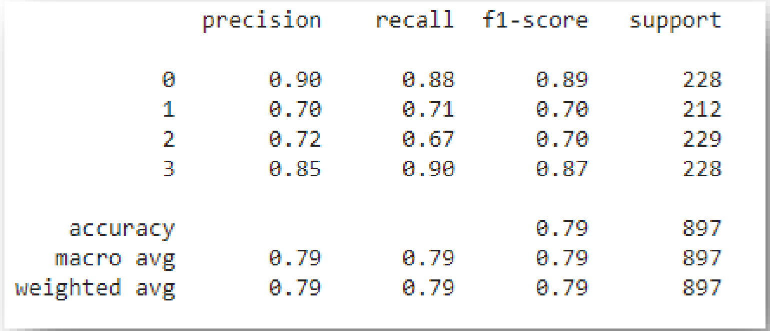
```
Classification Report
              precision    recall  f1-score   support

           0       0.91      0.94      0.93       107
           1       0.88      0.86      0.87       123
           2       0.84      0.81      0.83       106
           3       0.90      0.92      0.91       107

    accuracy                           0.88       443
   macro avg       0.88      0.88      0.88       443
weighted avg       0.88      0.88      0.88       443
```



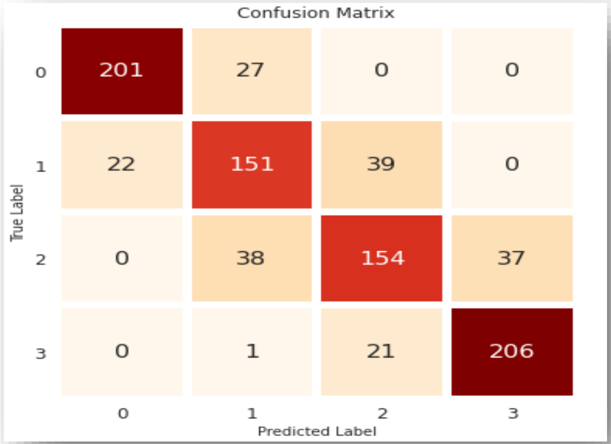


Implementing Logistic regression

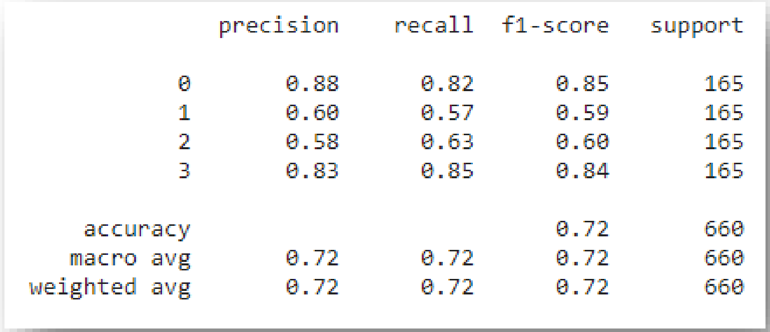
Train metrics



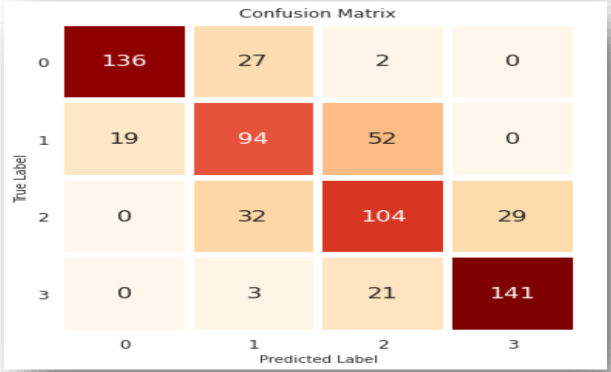
```
              precision    recall  f1-score   support

           0       0.90      0.88      0.89       228
           1       0.70      0.71      0.70       212
           2       0.72      0.67      0.70       229
           3       0.85      0.90      0.87       228

    accuracy                           0.79       897
   macro avg       0.79      0.79      0.79       897
weighted avg       0.79      0.79      0.79       897
```



Test metrics



```
              precision    recall  f1-score   support

           0       0.88      0.82      0.85       165
           1       0.60      0.57      0.59       165
           2       0.58      0.63      0.60       165
           3       0.83      0.85      0.84       165

    accuracy                           0.72       660
   macro avg       0.72      0.72      0.72       660
weighted avg       0.72      0.72      0.72       660
```



Model Validation & Selection contd…

Observations:

1. As seen in the above slides Random forest classifier is not giving great results , GradientBoostingClassifier is bit better than Random forest in recall and precision

2. XGboost classifier is giving the better results than GB but the recall of random forest classifier is somewhat similar

3. KNeighbors is giving the best results among all of the algorithms

4. Logistic regression is giving low results among all of them

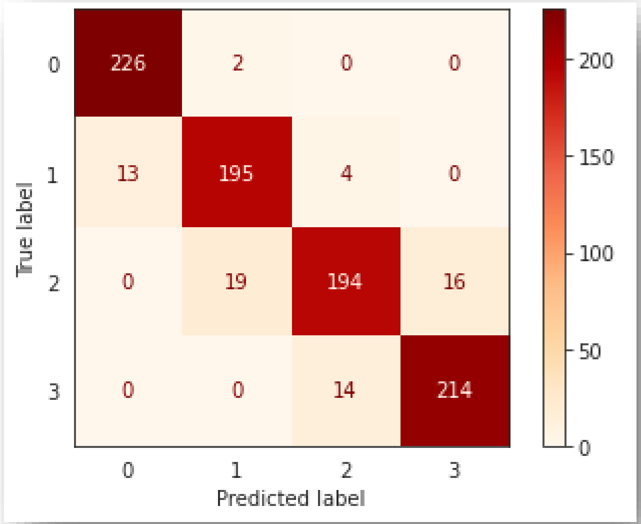
Model Validation & Selection contd...

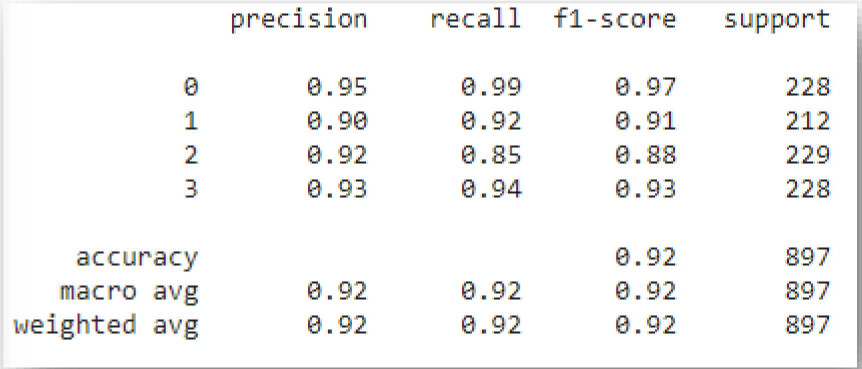
So we had chosen Kneighbors classifier for the prediction and the best hyperparameters obtained are as below

Best hyperparameters :
Train : (algorithm='auto', leaf_size=30, metric=‘Euclidean', metric_params=None, n_jobs=None, n_neighbors=11, p=2, weights='distance’)

Test : (algorithm='auto', leaf_size=30, metric='euclidean', metric_params=None, n_jobs=None, n_neighbors=17, p=2, weights='distance')

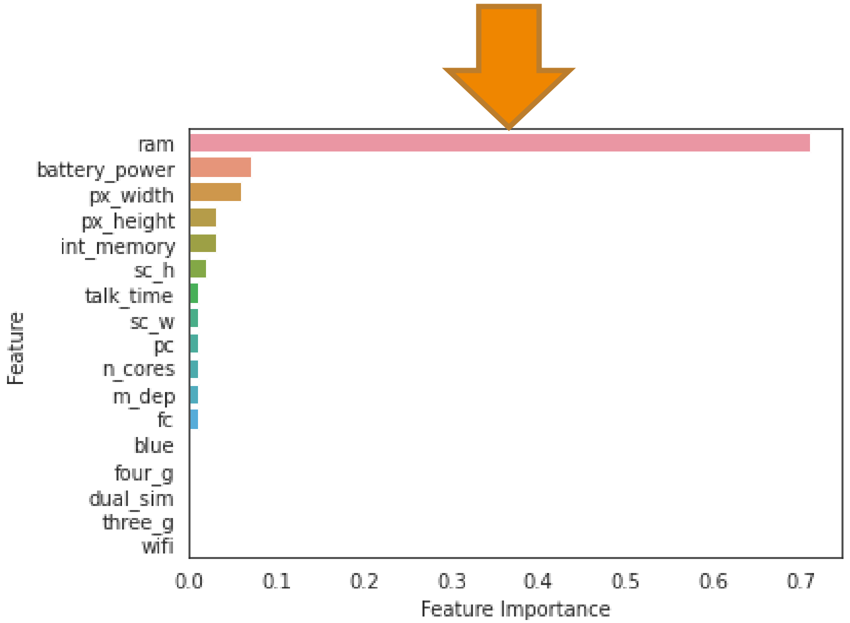
Model Validation & Selection(Hyperparamter tuned)



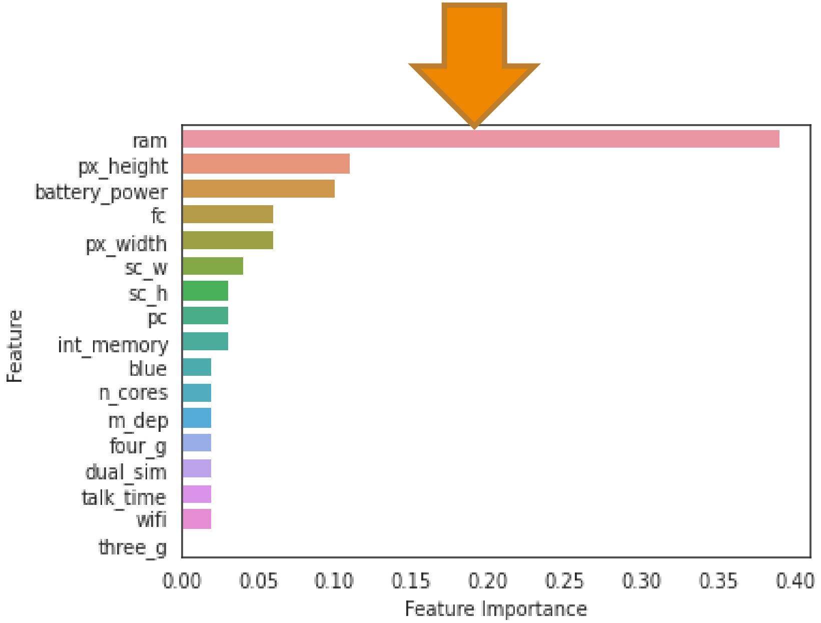

```
              precision    recall  f1-score   support

           0       0.95      0.99      0.97       228
           1       0.90      0.92      0.91       212
           2       0.92      0.85      0.88       229
           3       0.93      0.94      0.93       228

    accuracy                           0.92       897
   macro avg       0.92      0.92      0.92       897
weighted avg       0.92      0.92      0.92       897
```

Feature Importance

Random Forest Classifier



XGBoost Classifier



Conclusion

- Ram , Battery_power features were found to be the most relevant features for predicting price range of mobiles and dropping negative correlation features which are clock speed , mobile_wt , touch_screen
- Kneighbors and Xgboost are given best accuracy score 95% test ,93% train and 91% train , 88% test respectively and roc_auc score for kneighbors is 99%
- Tuning the hyperparameters by GridSearchCV on kneighbors but not getting much difference in results but the best parameters n_neighbors for train and test are 11 and 17
- So we conclude that kneighbors classifier is giving the best results for these dataset
- So we can say that in the price range prediction as the ram and battery_power increases the price range will increase for sure

Q & A