Capstone Project-3

Mobile Price Range Prediction

Classification of mobiles based on Price Range

Team Members

Palash Pathak

Yogesh Dubey

Overview of 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.
- We do not have to predict the actual price but a price range indicating how high the price is.
- Price_range - This is the target variable with value :
- 0(low cost), 1(medium cost), 2(high cost)
- 3(very high cost).

Data Summary

#	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

No of observations & no of features
(2000, 21)
Great!!! We don't have any null values.
Date type also looks correct.

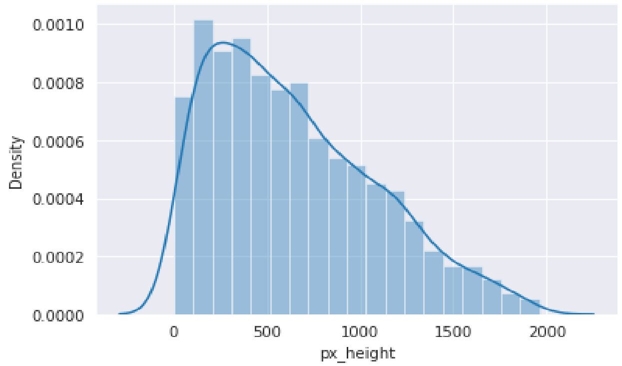
Lets do some sanity check..

Dealing With Zero values:

1. Pixel Height :

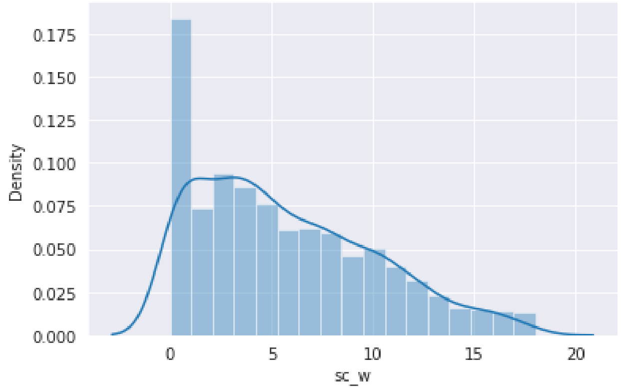
We have two - zero entries for pixel height

We will replace these zero values with median.



2. Screen Width :

We have 180 entries for zero screen width.
We will replace these zero values with mean.



Dealing With abnormal values:

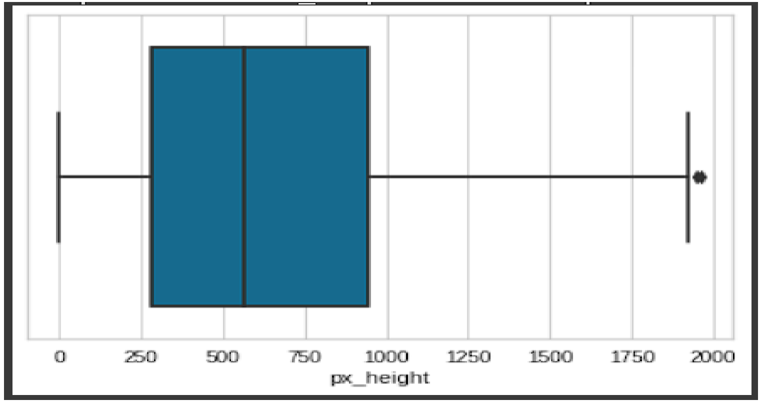
Looking at Pixel Height & Pixel width:

Pixel height has lots of abnormally small values.

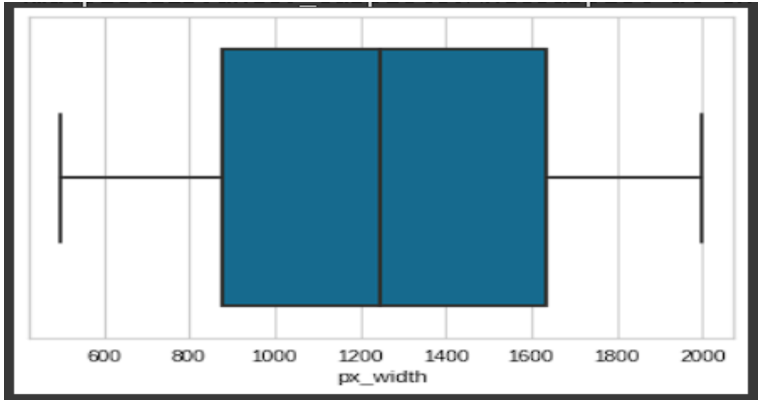
After doing some research on mobile resolution standards, we decided to clean all entries with less than 200 pixel height by replacing with average based on its price range.

```
mob_df['px_height'].fillna(mob_df.groupby('price_range')['px_height'].transform('mean')
```

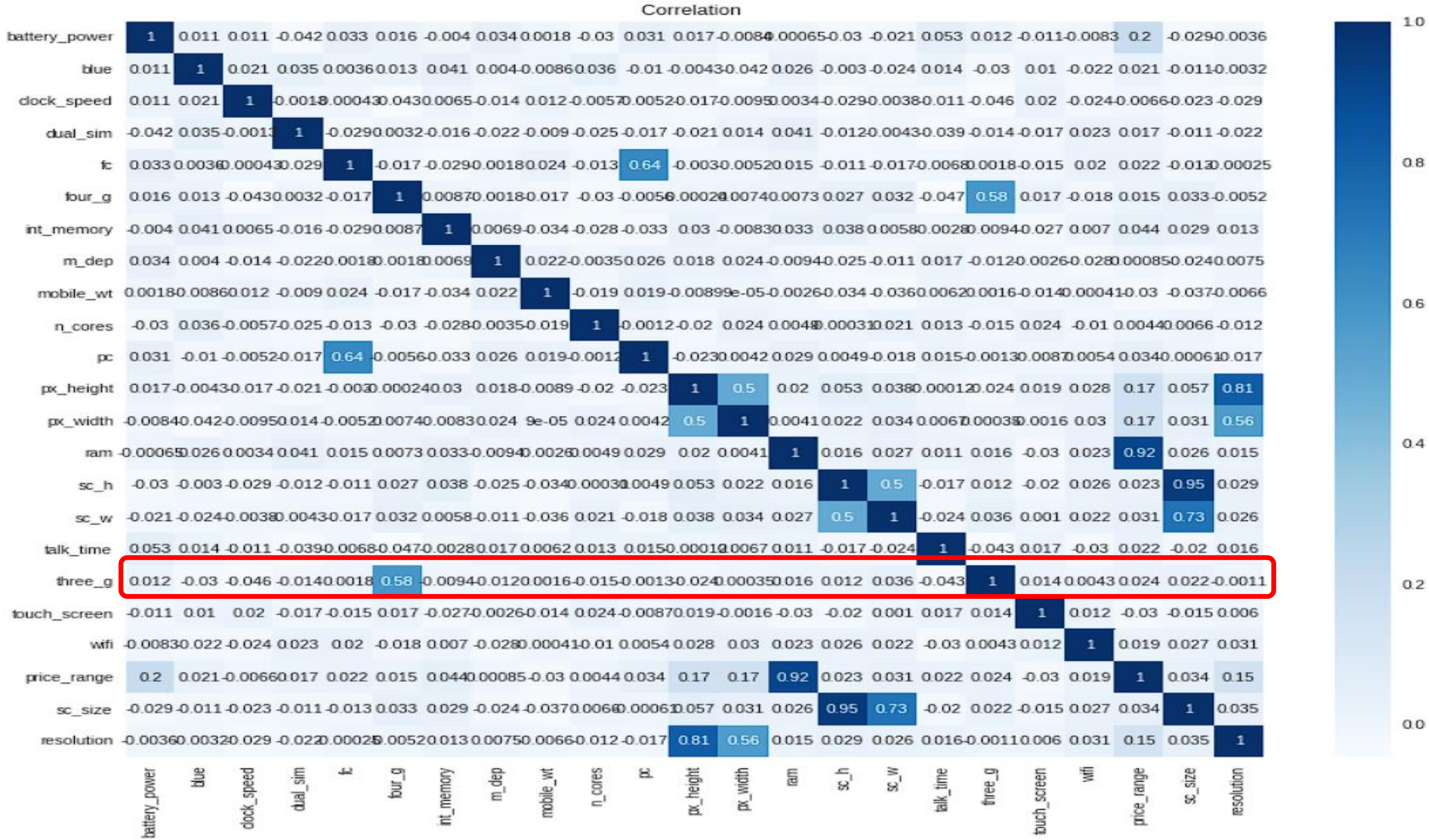
Pixel Height



Pixel Width



Correlation Heatmap



Correlation with Price Range :

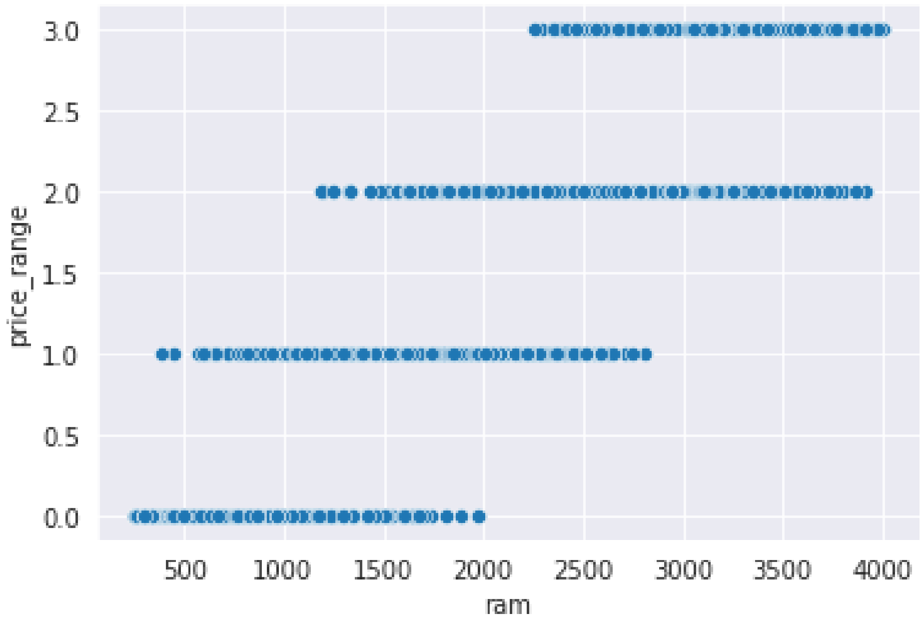
Our target variable Price Range has the highest correlation with RAM.

Apart from RAM, other imp factors are battery power and pixel resolution

Some features have high correlation namely:

1. Front camera and primary camera
2. Pixel height and width
3. Screen height and width
4. 3G and 4G

Price range vs RAM

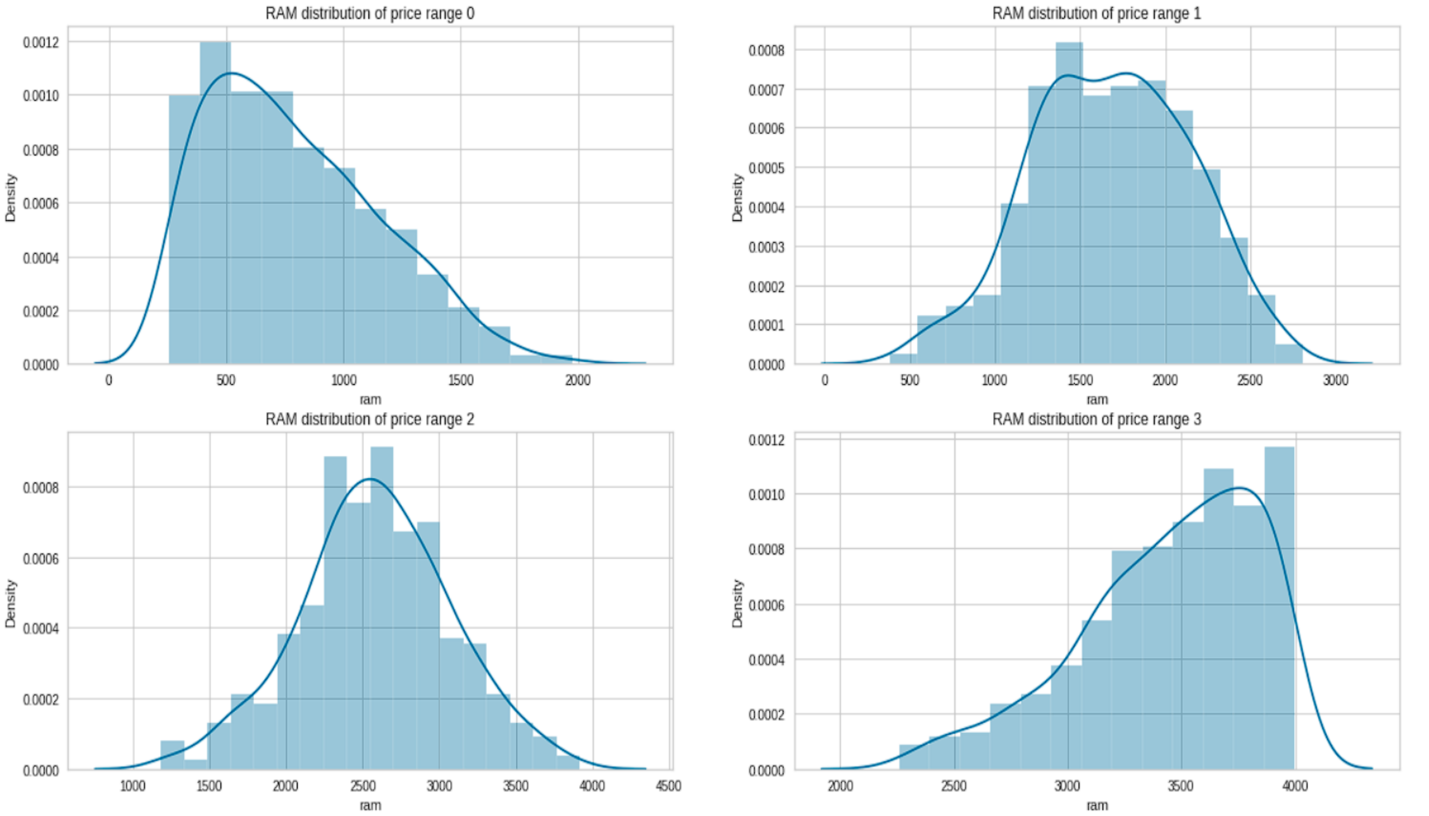


From plot we can say that :

- Mobiles with RAM upto 2GB falls into low price category
- Mobiles with RAM from 0.5GB to 3GB falls into medium price category
- Mobiles with RAM from 1GB to 4GB falls into high price category
- Mobiles with RAM above 2GB falls into very high price category

Also, there is one interesting observation that RAM of 2GB can be a border line to separate low and very high price tags directly irrespective of other mobile specifications

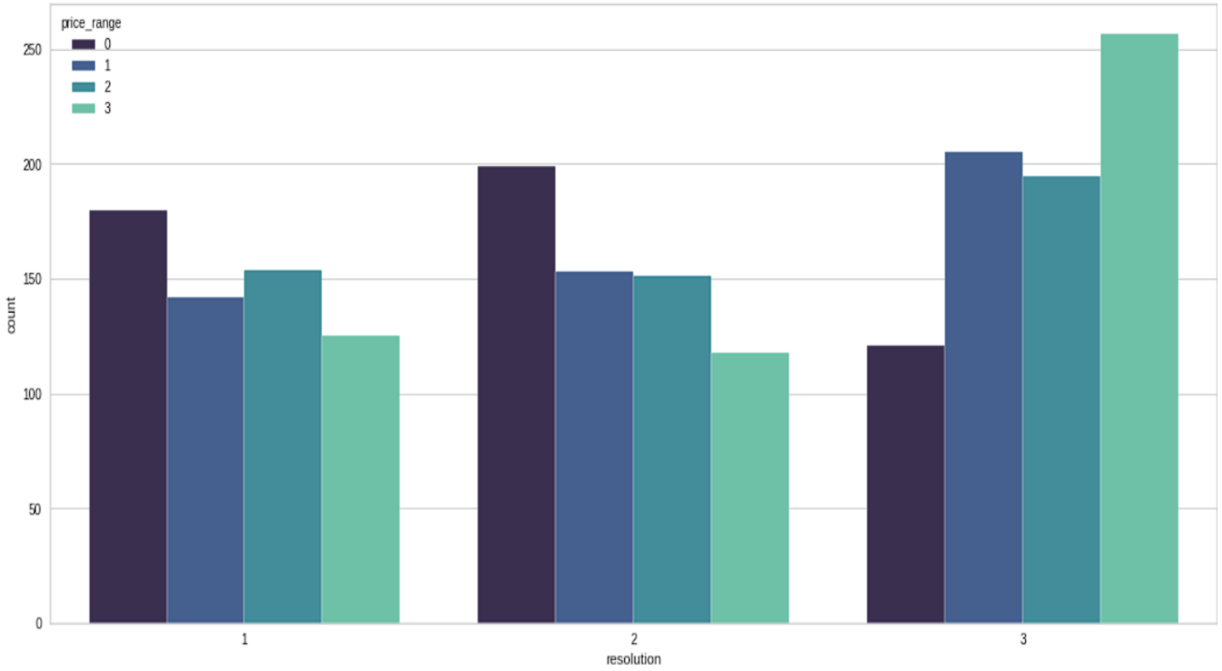
RAM for different price range:



Price range vs Resolution:

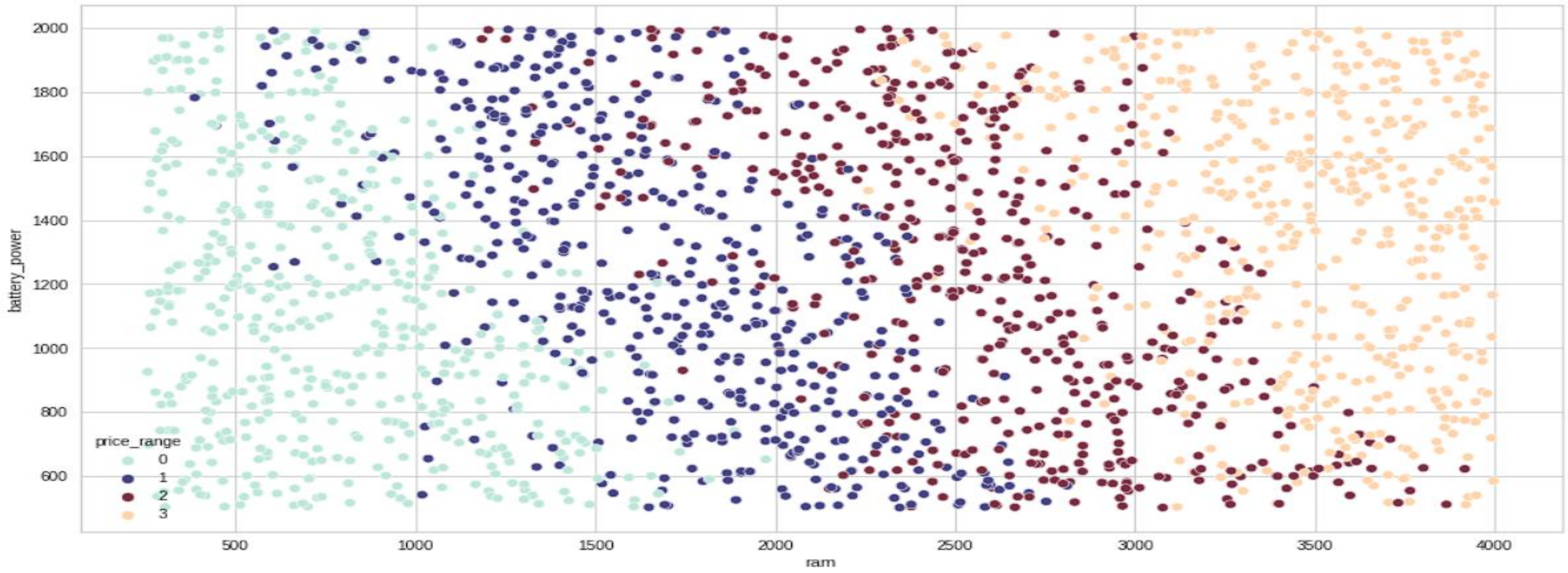
After doing some research for mobile resolution standards in market, we decided to group mobiles based on resolutions into 3 categories : CGA,VGA,HD

We will nominate
1: CGA (< 640*480p)
2: VGA
3: HD (>1080*720p)



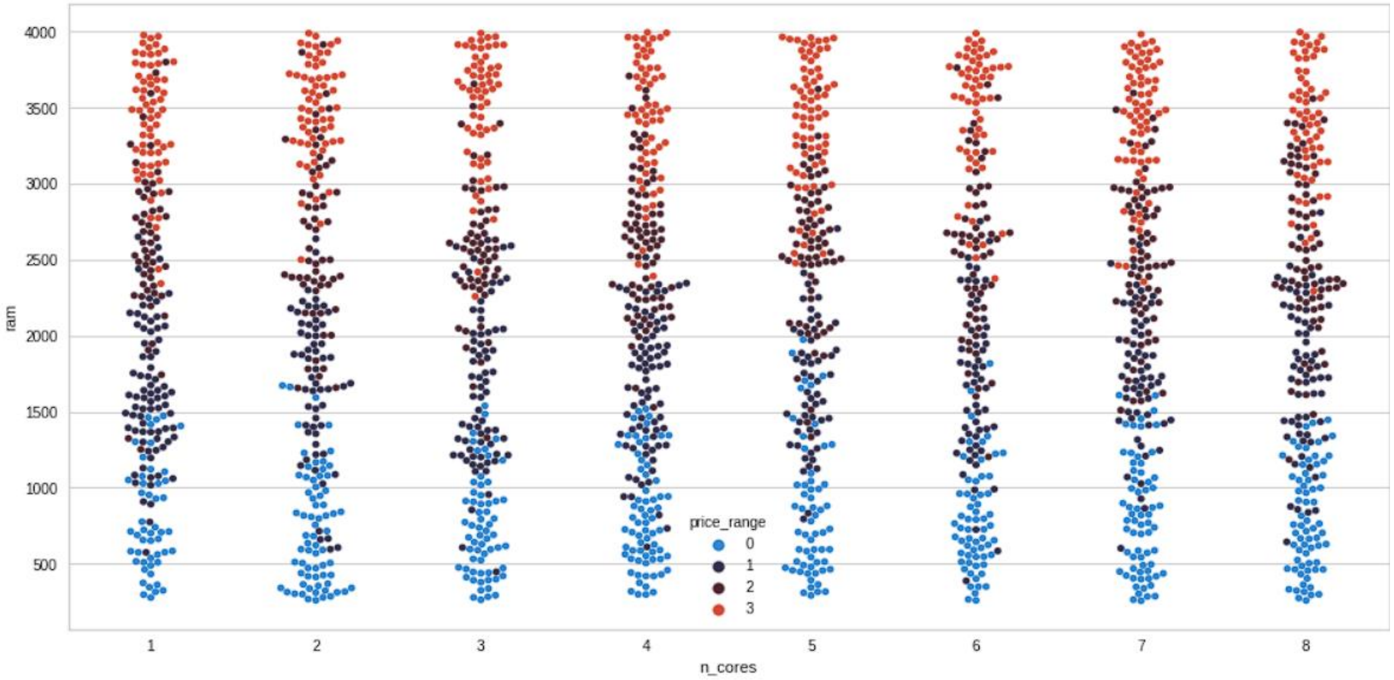
So for higher price ranges, there are more higher resolution mobiles.

Battery Power vs RAM:



As we already know that RAM is our most imp feature, we compared other features with Ram. Above plot shows how battery power varies with Ram for all price ranges

CPU cores vs RAM:



Above plot that we have any number of CPU cores ranging from 1 to 8 for all values of RAM

Class Balance

Splitting the data and checking class balance:
Train df : (1600, 20)
Test df : (400, 20)

Class doesn't seem too unbalanced, so no need to forcibly do balanced split, we will go forward with random split.

Training Model

1.Naive Bayes Classifier(base model):

First 10 actual classes : [0 2 1 2 3 1 3 3 2 1]
First 10 predicted classes : [0 3 0 2 3 2 3 3 1 2]

As we can see in the report we have accuracy of 0.52 & 0.54 on train & test set respectively.

Also, as we have multi-class target variable, we will check the weighted f1_score which is found to be 0.51 and 0.52 for train & test set resp.

What is Hamming Loss ?
Hamming loss is the fraction of targets that are Miss-classified.

Training hamming loss : 0.482
Testing hamming loss : 0.465

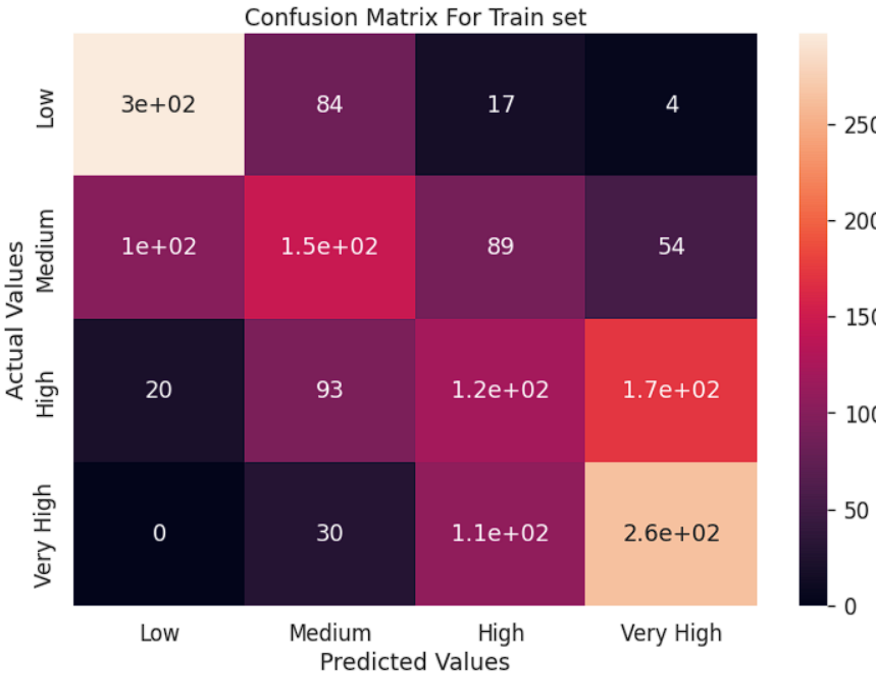
Report on Train Set

	precision	recall	f1-score	support
0	0.71	0.74	0.72	402
1	0.42	0.38	0.39	391
2	0.36	0.30	0.33	406
3	0.53	0.66	0.59	401
accuracy			0.52	1600
macro avg	0.51	0.52	0.51	1600
weighted avg	0.51	0.52	0.51	1600

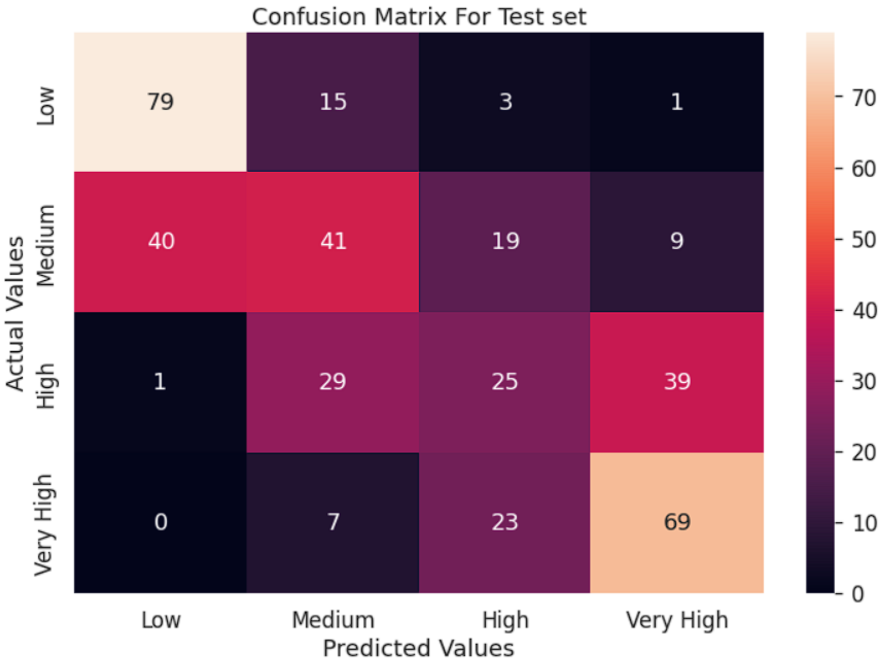
Report on Test Set

	precision	recall	f1-score	support
0	0.66	0.81	0.72	98
1	0.45	0.38	0.41	109
2	0.36	0.27	0.30	94
3	0.58	0.70	0.64	99
accuracy			0.54	400
macro avg	0.51	0.54	0.52	400
weighted avg	0.51	0.54	0.52	400

Naive Bayes Classifier - CM



Confusion Matrix for train set



Confusion Matrix for test set

We can say that our model is performing more poorly on medium & high classes.

2.Decision Tree:

After using cross-validation, following hyper-parameters were tuned to get the best-fit:

Max depth: 8
Max leaf nodes : 50
Min samples leaf : 5
Criterion : entropy

Results of best-fit Decision Tree:

First 10 actual classes : [0 2 1 2 3 1 3 3 2 1]
First 10 predicted classes : [0 2 1 2 3 1 3 3 2 2]

Training accuracy : 0.93
Testing accuracy : 0.89
Training weighted f1 score : 0.93
Testing weighted f1 score : 0.89
Training hamming loss : 0.069
Testing hamming loss : 0.112

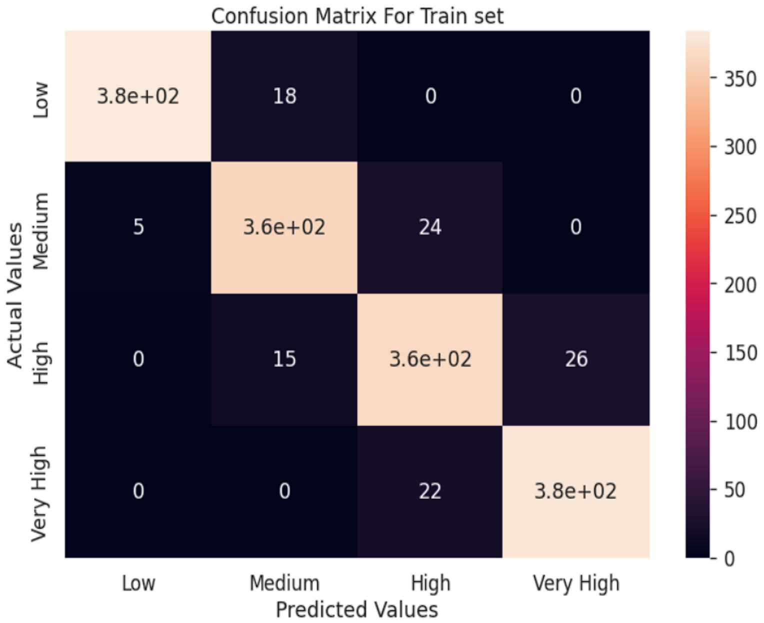
Report on Train Set

	precision	recall	f1-score	support
0	0.99	0.96	0.97	402
1	0.92	0.93	0.92	391
2	0.89	0.90	0.89	406
3	0.94	0.95	0.94	401
accuracy			0.93	1600
macro avg	0.93	0.93	0.93	1600
weighted avg	0.93	0.93	0.93	1600

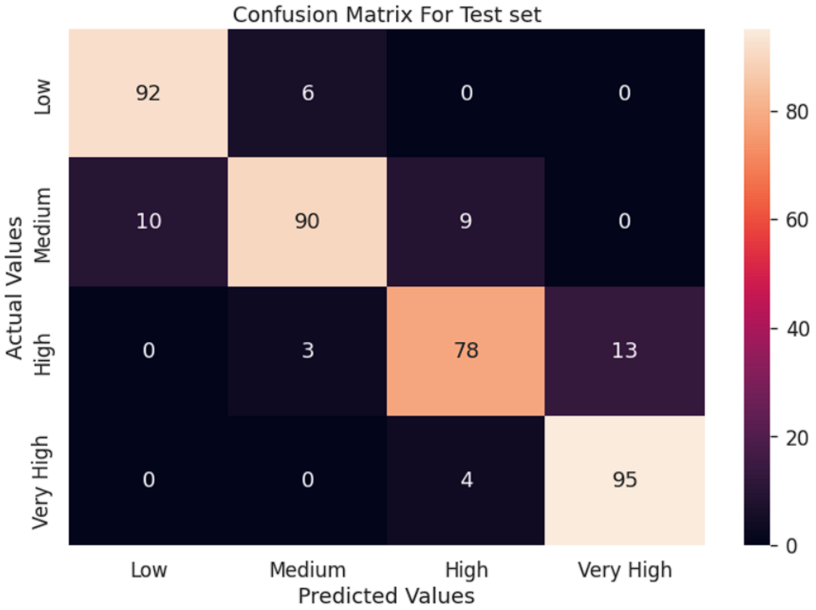
Report on Test Set

	precision	recall	f1-score	support
0	0.90	0.94	0.92	98
1	0.91	0.83	0.87	109
2	0.86	0.83	0.84	94
3	0.88	0.96	0.92	99
accuracy			0.89	400
macro avg	0.89	0.89	0.89	400
weighted avg	0.89	0.89	0.89	400

Decision Trees Classifier - CM



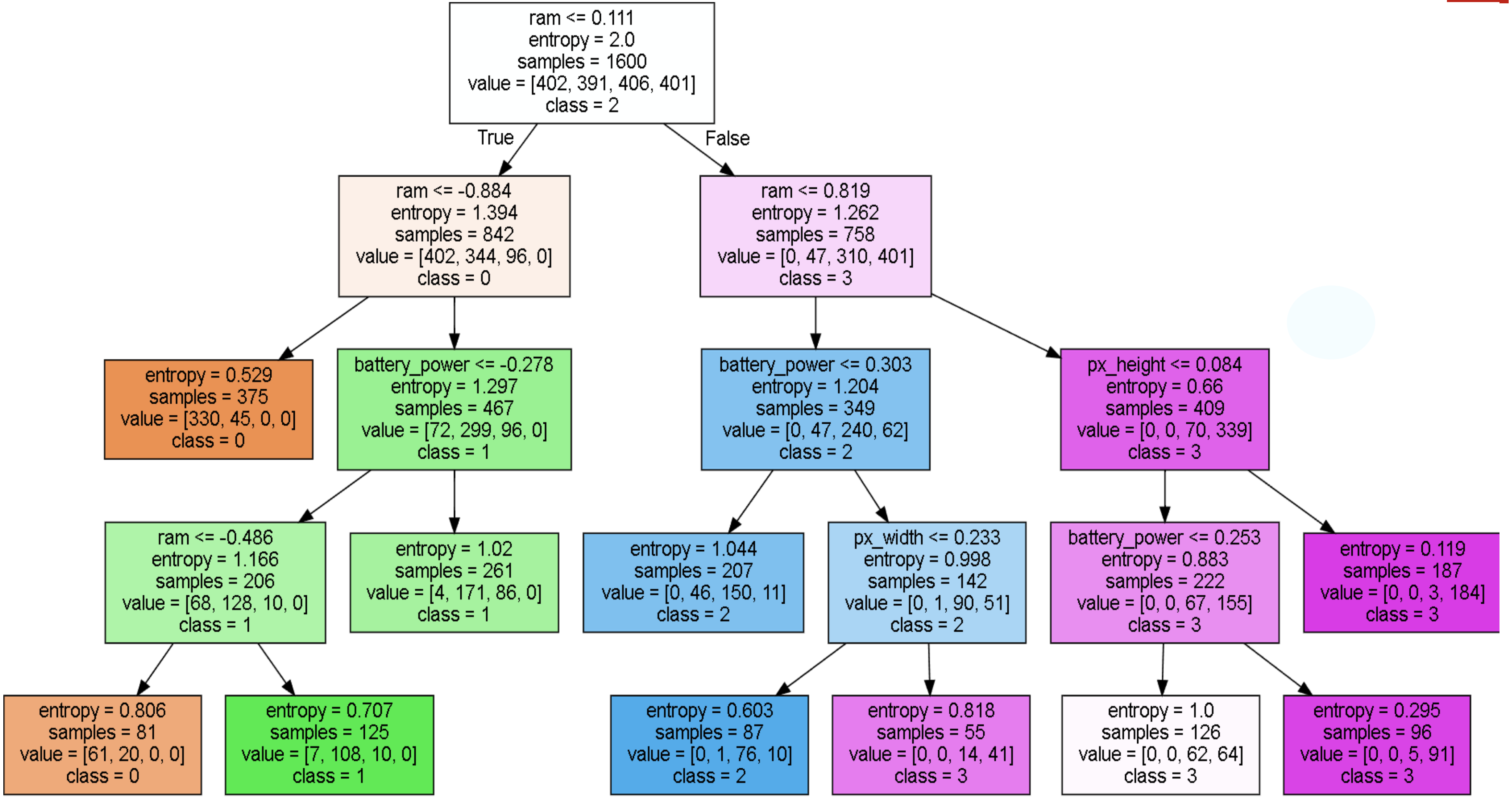
Confusion Matrix for train set



Confusion Matrix for test set

We can say that the classes low and very high are predicted more correctly.

Visualizing the Decision Trees



3.Random Forest :

After using cross-validation, following hyper-parameters were tuned to get the best-fit:

Max depth: 10
N estimators: 110
Min samples leaf : 4
Criterion : gini

Results of best-fit Random Forest:

First 10 actual classes : [0 2 1 2 3 1 3 3 2 1]
First 10 predicted classes : [0 2 0 2 3 1 3 3 1 1]

Training accuracy : 0.99
Testing accuracy : 0.90
Training weighted f1 score : 0.99
Testing weighted f1 score : 0.90
Training hamming loss : 0.011
Testing hamming loss : 0.098

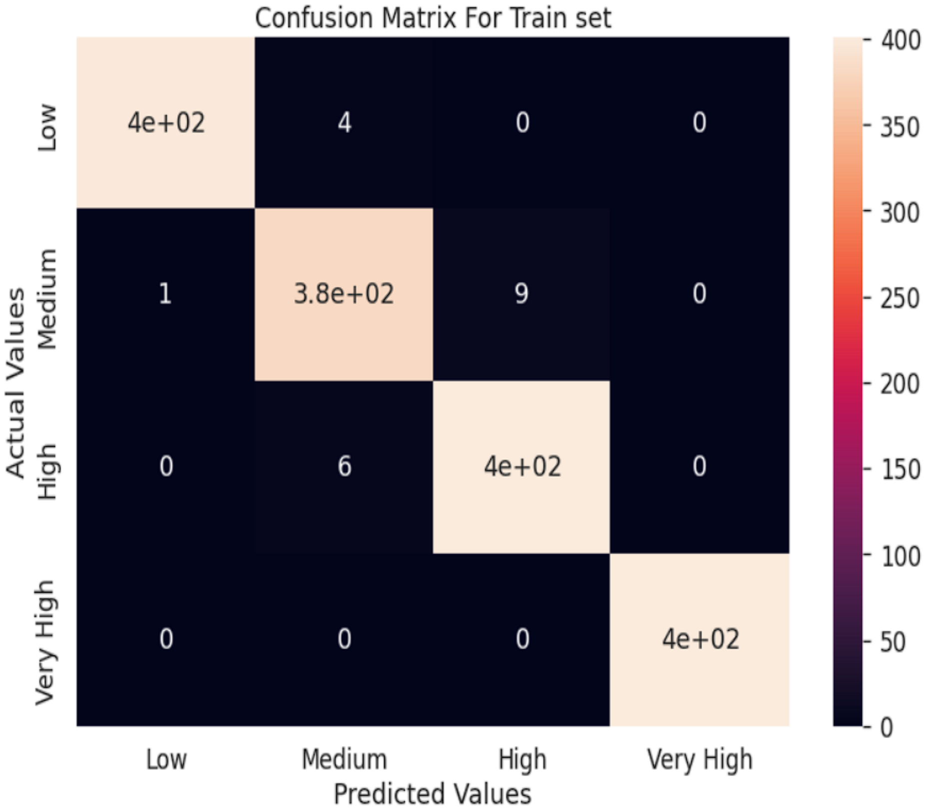
Report on Train Set

	precision	recall	f1-score	support
0	1.00	0.99	0.99	402
1	0.97	0.98	0.98	391
2	0.99	0.99	0.99	406
3	1.00	1.00	1.00	401
accuracy			0.99	1600
macro avg	0.99	0.99	0.99	1600
weighted avg	0.99	0.99	0.99	1600

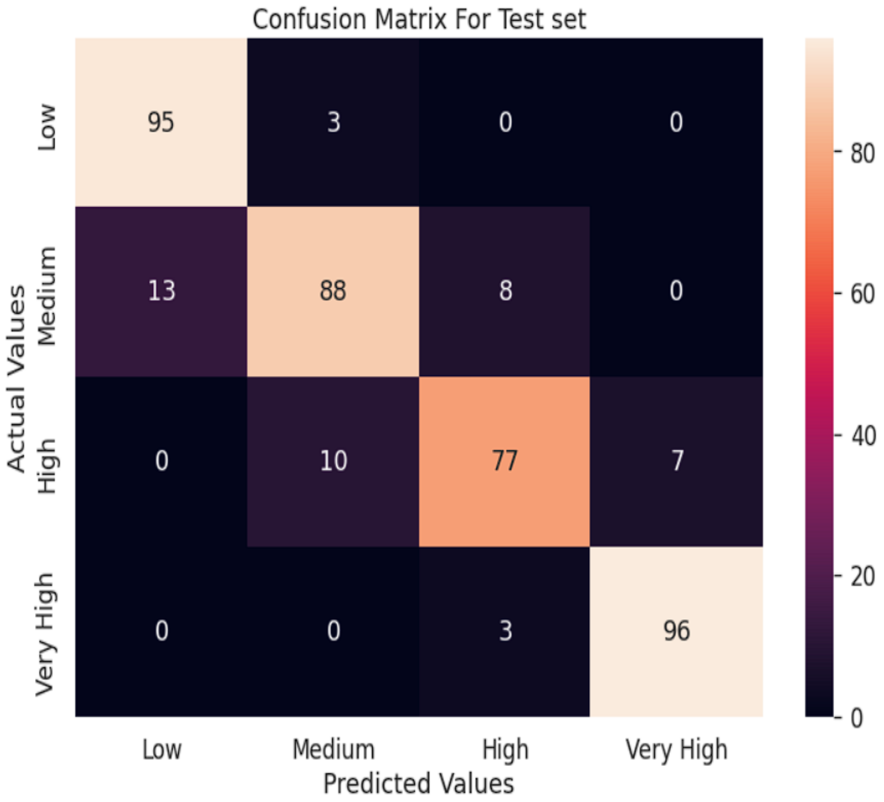
Report on Test Set

	precision	recall	f1-score	support
0	0.88	0.97	0.92	98
1	0.88	0.83	0.85	109
2	0.91	0.84	0.87	94
3	0.94	0.98	0.96	99
accuracy			0.90	400
macro avg	0.90	0.90	0.90	400
weighted avg	0.90	0.90	0.90	400

Random Forest Classifier - CM



train set



test set

4.Logistic regression:

After using cross-validation, following hyper-parameters were tuned to get the best-fit:

C : 0.7
Max iteration : 100
Multi-class : auto
Penalty : L1
Solver : saga

Results of best-fit Logistic regression:

First 10 actual classes : [0 2 1 2 3 1 3 3 2 1]
First 10 predicted classes : [0 2 1 2 3 1 3 3 2 2]

Training accuracy : 0.96
Testing accuracy : 0.96
Training weighted f1 score : 0.96
Testing weighted f1 score : 0.96
Training hamming loss : 0.041
Testing hamming loss : 0.042

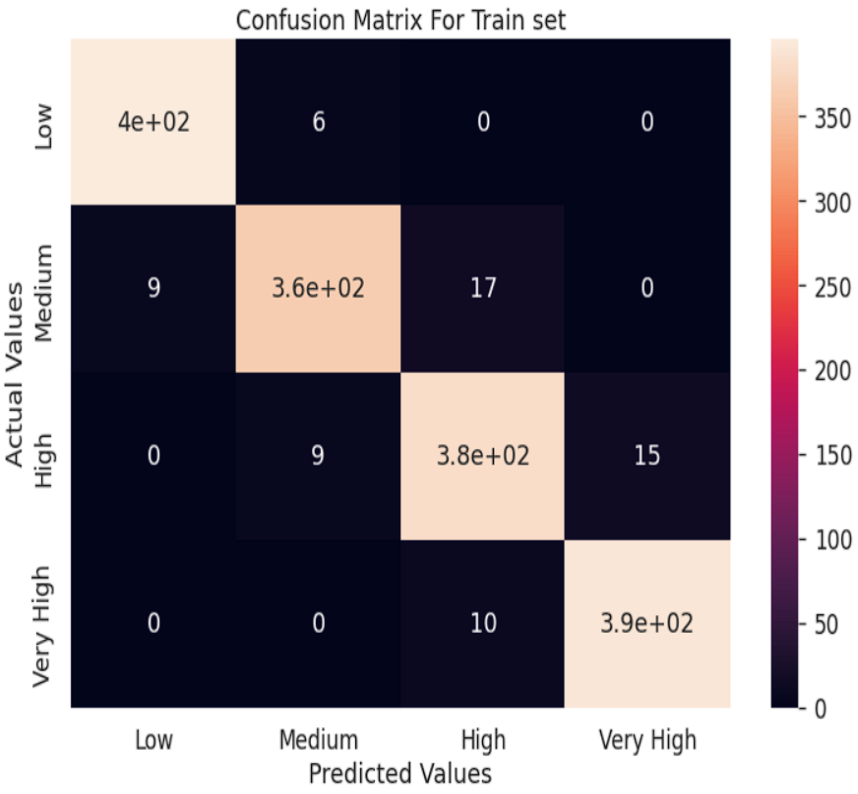
Report on Train Set

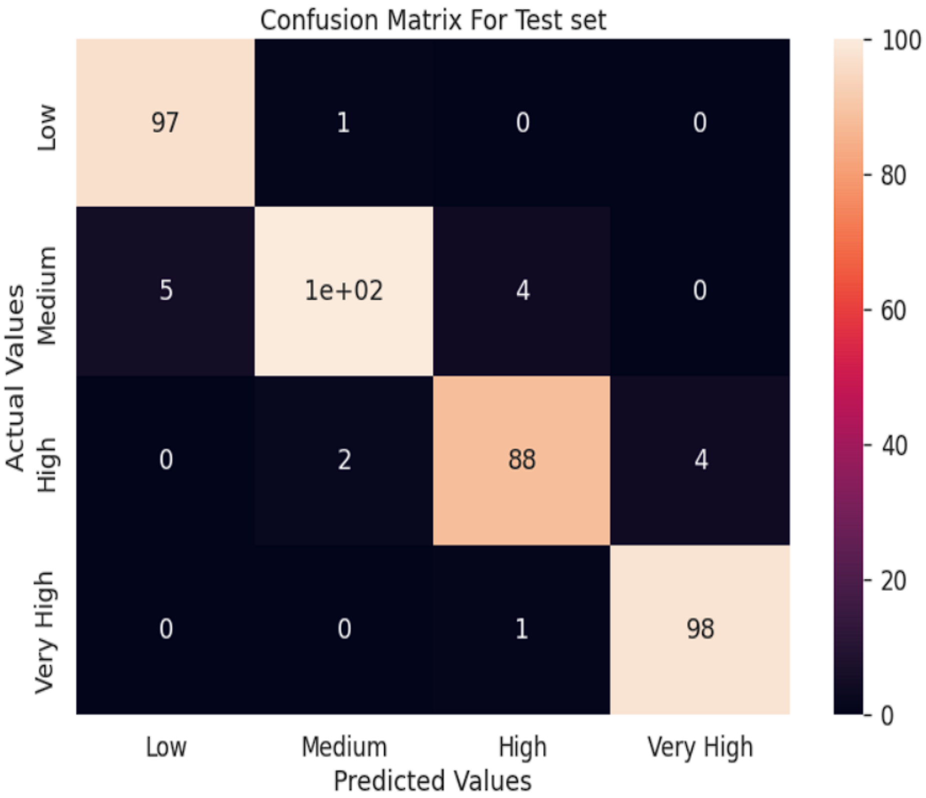
	precision	recall	f1-score	support
0	0.98	0.99	0.98	402
1	0.96	0.93	0.95	391
2	0.93	0.94	0.94	406
3	0.96	0.98	0.97	401
accuracy			0.96	1600
macro avg	0.96	0.96	0.96	1600
weighted avg	0.96	0.96	0.96	1600

Report on Test Set

	precision	recall	f1-score	support
0	0.95	0.99	0.97	98
1	0.97	0.92	0.94	109
2	0.95	0.94	0.94	94
3	0.96	0.99	0.98	99
accuracy			0.96	400
macro avg	0.96	0.96	0.96	400
weighted avg	0.96	0.96	0.96	400

Logistic Regression- CM





train set

test set

4.Support Vector Machines:

After using cross-validation, following hyper-parameters were tuned to get the best-fit:

C : 14
Kernel : linear
Gamma : auto
Decision function shape : ovo

Results of best-fit SVM:

First 10 actual classes : [0 2 1 2 3 1 3 3 2 1]
First 10 predicted classes : [0 2 1 2 3 1 3 3 2 2]

Training accuracy : 0.96
Testing accuracy : 0.96
Training weighted f1 score : 0.96
Testing weighted f1 score : 0.96
Training hamming loss : 0.039
Testing hamming loss : 0.04

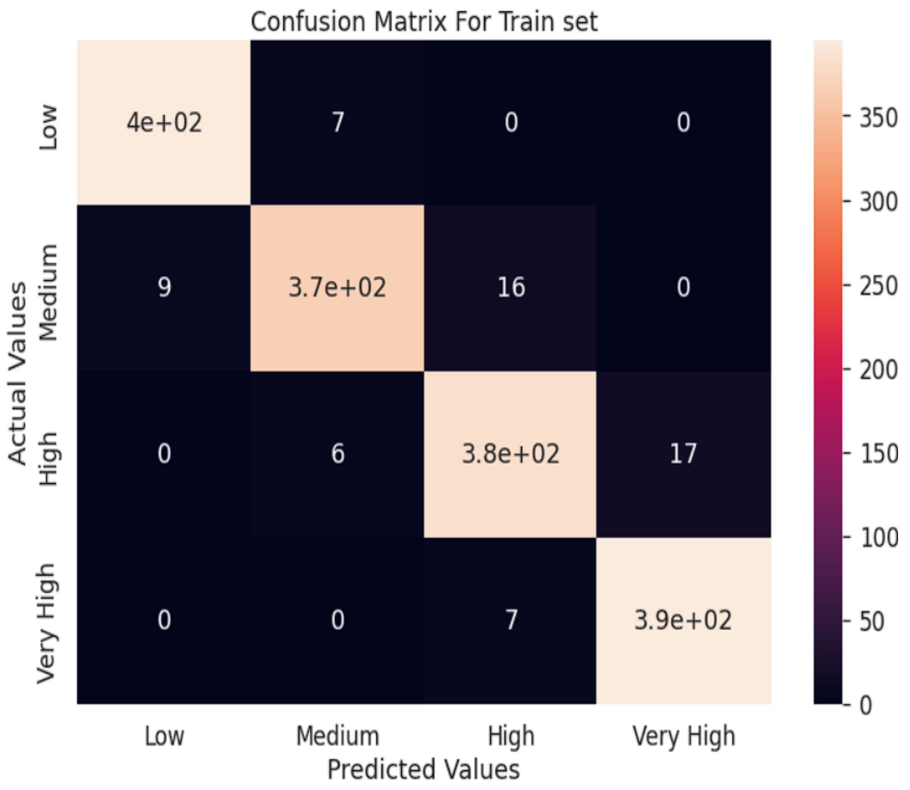
Report on Train Set

	precision	recall	f1-score	support
0	0.98	0.98	0.98	402
1	0.97	0.94	0.95	391
2	0.94	0.94	0.94	406
3	0.96	0.98	0.97	401
accuracy			0.96	1600
macro avg	0.96	0.96	0.96	1600
weighted avg	0.96	0.96	0.96	1600

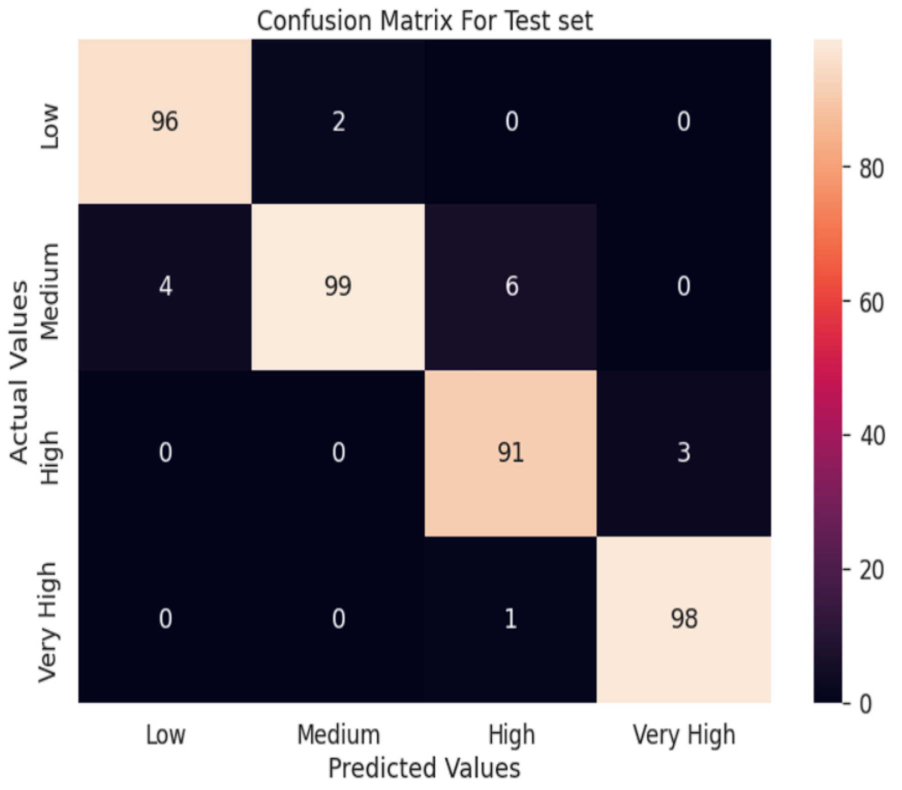
Report on Test Set

	precision	recall	f1-score	support
0	0.96	0.98	0.97	98
1	0.98	0.91	0.94	109
2	0.93	0.97	0.95	94
3	0.97	0.99	0.98	99
accuracy			0.96	400
macro avg	0.96	0.96	0.96	400
weighted avg	0.96	0.96	0.96	400

SVM - CM



train set



test set

Model Selection :

Lets compare the performance:

As we can see that both SVM and LR are giving similar performance.
Both are having a score of around 96% on both train/test set .

	Weighted_f1_score-Train	hamming_loss-Train	Weighted_f1_score-Test	hamming_loss-Test
Naive_Bayes	0.509279	0.481875	0.517781	0.4650
Decision_Tree	0.931471	0.068750	0.886553	0.1125
Random_Forest	0.989395	0.010625	0.901271	0.0975
Logistic_Reg	0.958692	0.041250	0.957245	0.0425
SVM	0.961170	0.038750	0.959815	0.0400

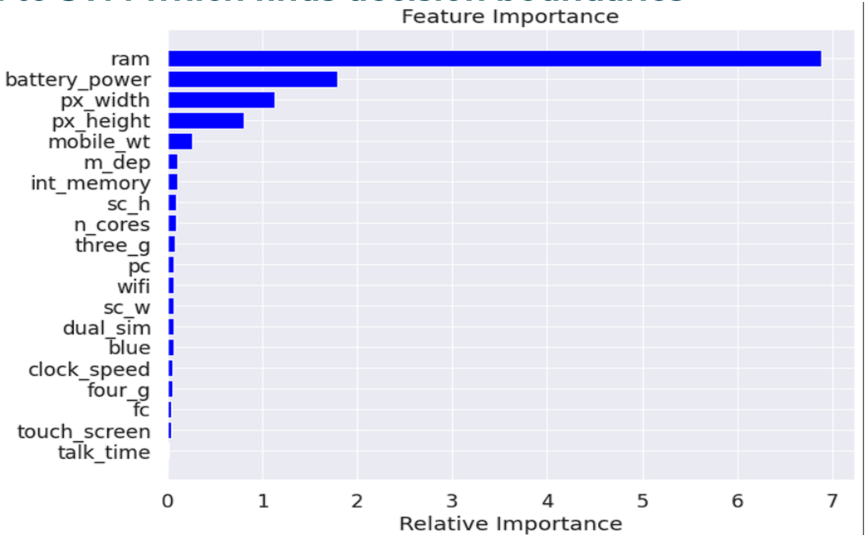
LR vs SVM:

We decided to choose LR over SVM based on following points:

- As we saw while training SVM, linear kernel fitted our data the best, so we don't need non-linear kernel for our data.
- In case of linear kernel SVM, it performs similar to LR, but for non-linear relation SVM beats Logistic Regression.
- Here, LR is preferred as it can output probabilities instead of hard labels and if needed we can fine tune our performance by plotting the ROC curve and figuring out the right threshold.
- Logistic Regression is like linear regression with a activation function added. This makes it more easy to interpret as compared to SVM which finds decision boundaries to separate the classes.

Figure here displays the important features extracted from Logistic Regression .

We can comment that the most important features are RAM, battery power, pixel resolution.



Summary:

We had data for 2000 mobile phones grouped into 4 categories based on Price range.

- We first cleaned data by removing zeroes values and abnormally small values(Pixel Height).
- EDA summarized to following takeaway points:
  * This dataset seems older as the specifications of mobiles are not from recent technologies like max 2000mah battery, 4gb ram, etc.
  * Almost half of the mobiles in our dataset have Bluetooth, dual sim, 4G, touch screen, Wi-Fi.
  * Majority of mobiles(~75%) have 3G connectivity.
  * Mobile weight is up to 200g.
  * RAM of 2GB can be a border line to separate low and very high price tags directly irrespective of other mobile specifications.
  * Mobiles in higher price range have high pixel resolution.
  * Even with low ram , it's possible to have up to 3 GHz clock speed.
  * We can have any number of cores from 1 to 8 for any range of RAM.
- Our base model , Naïve Bayes Classifier gave around 50% accuracy.
- Decision Trees and Random forest over fitted our data and we are not able to reach good accuracy levels for Test sets.
- Decision Trees and Random forest worked quite well on Class0 & 3 (low & very high price)
- Logistic Regression and SVM(with linear kernel) worked best on our classification task and delivered around 96% accuracy on both Train/Test sets.
- We ended up selecting Logistic Regression over SVM as our best model.
- Most important features affecting mobile price are found to be RAM, battery power, pixel resolution (height & width) and mobile weight.

THANK YOU