Capstone Project-3

Mobile Price Range Prediction

Meet Delvadiya

CONTENT :

1. 1. Defining Problem Statement
2. 2. EDA and Feature Engineering
3. 3. Feature Selection
4. 4. Preparing Dataset for Modeling
5. 5 . Apply Model
6. 6. Model Validation and Selection
7. 7. Conclusion

THE DILEMMA:

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.

DATA PIPELINE :

Data processing : In this part, we manually go through each features and encoded with numerical features. After that we have removed unnecessary features.

EDA : In this part, we do some exploratory data analysis (EDA) on the selected features to see the trend.

Create a model : Finally, in this part, we will create models. Creating a model is also not an easy task. It is an iterative process. We show how to start a simple model, and slowly add complexity for better performance.

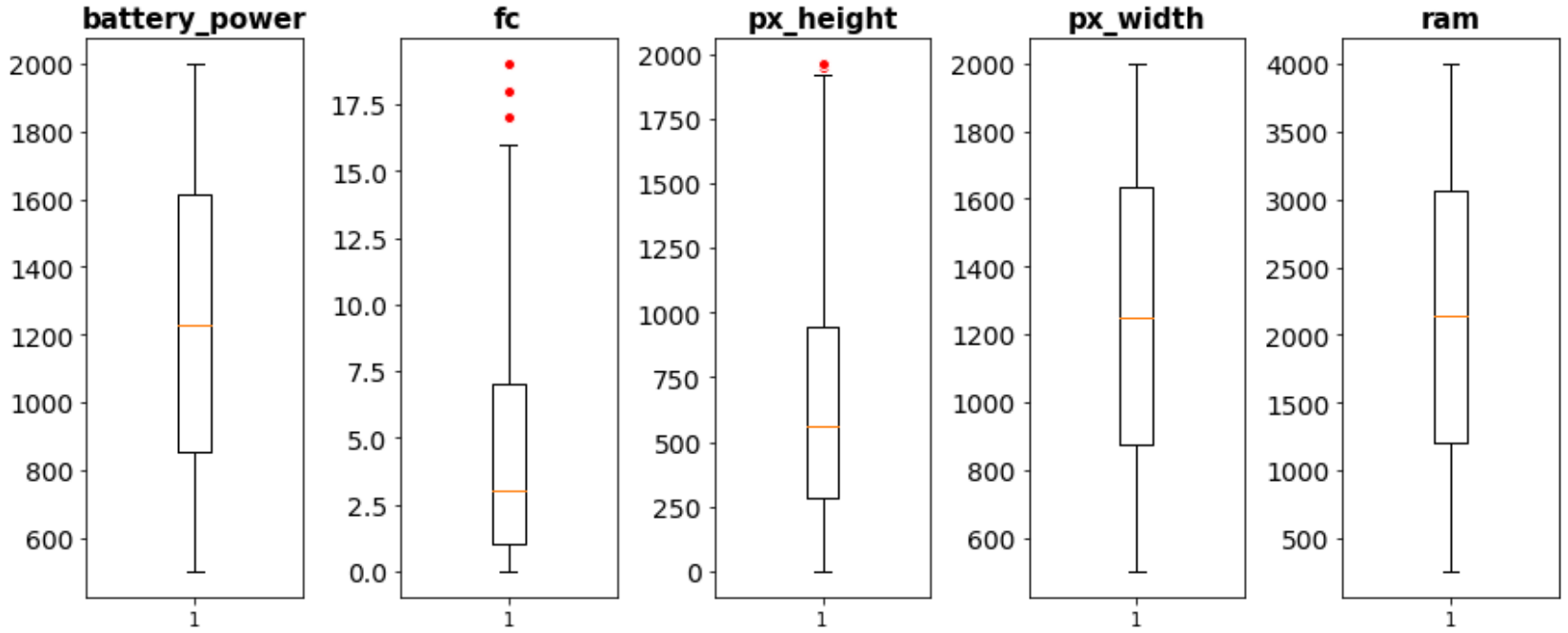
DATA SUMMARY :

- **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

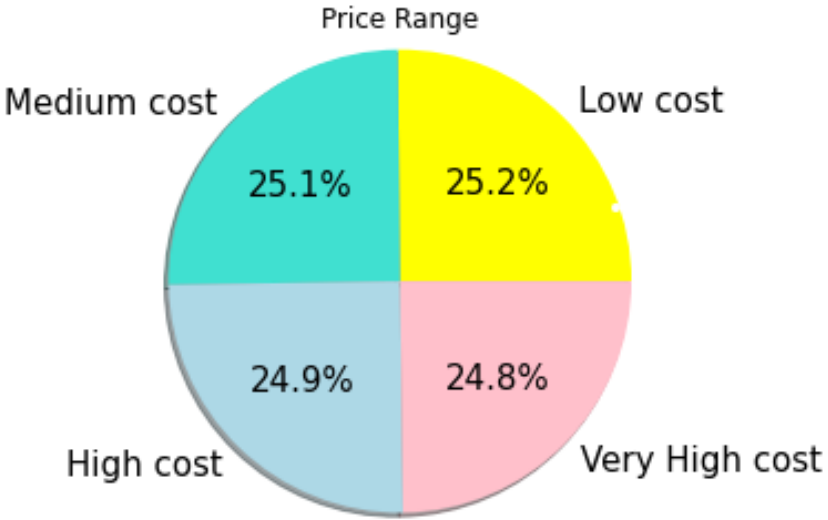
DATA SUMMARY(Continue) :

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

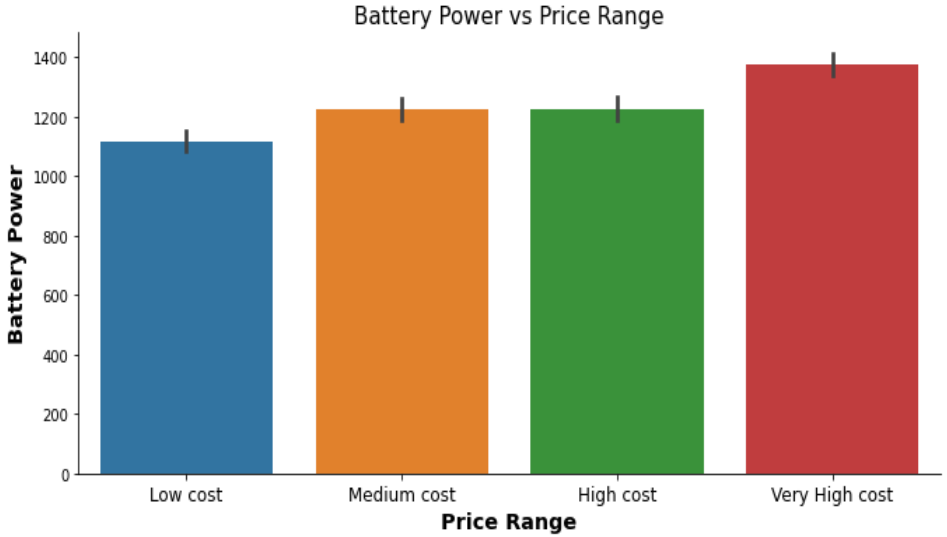
Box plot:

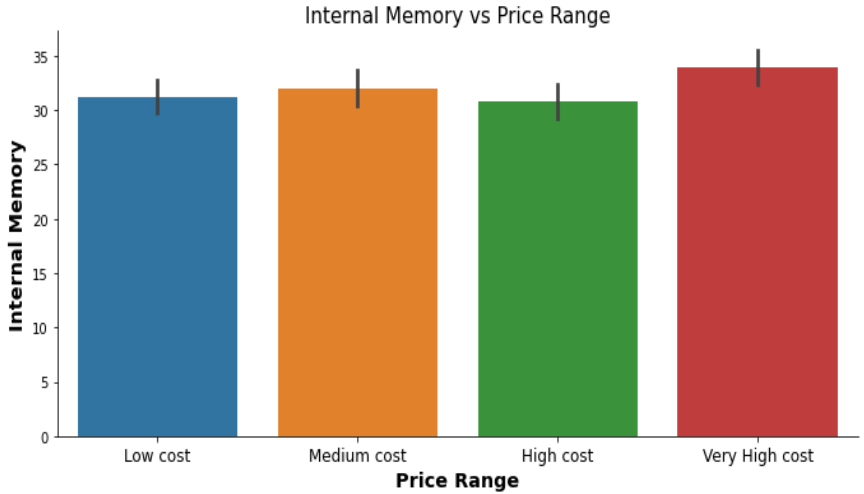

battery_power
2000
1800
1600
1400
1200
1000
800
600
1
fc
17.5
15.0
12.5
10.0
7.5
5.0
2.5
0.0
1
px_height
2000
1750
1500
1250
1000
750
500
250
0
1
px_width
2000
1800
1600
1400
1200
1000
800
600
1
ram
4000
3500
3000
2500
2000
1500
1000
500
1

Dependent variable Pie Chart:

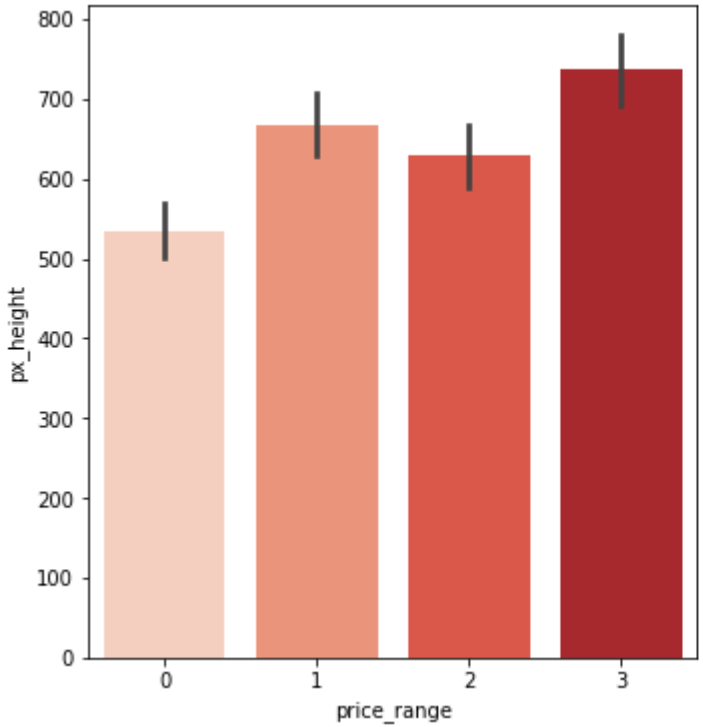


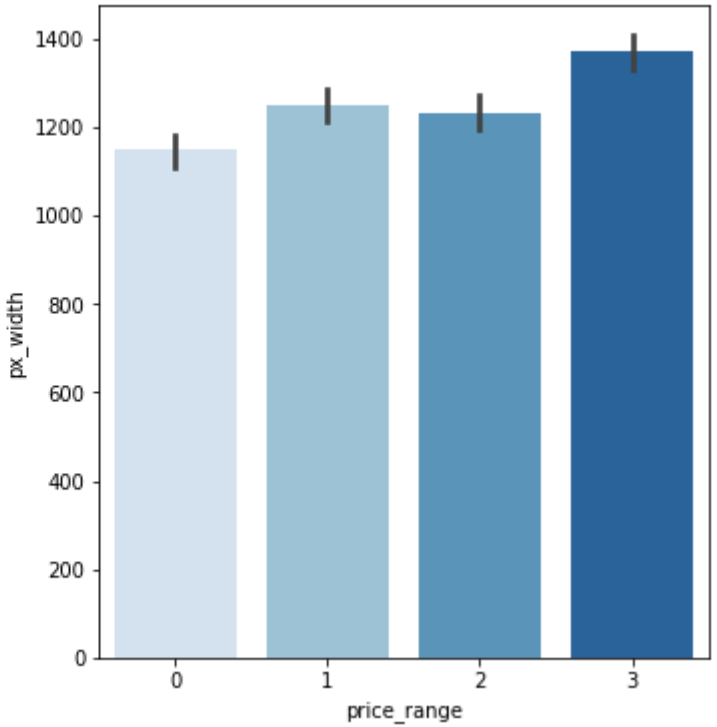
Price Range w.r.t Battery and Storage :



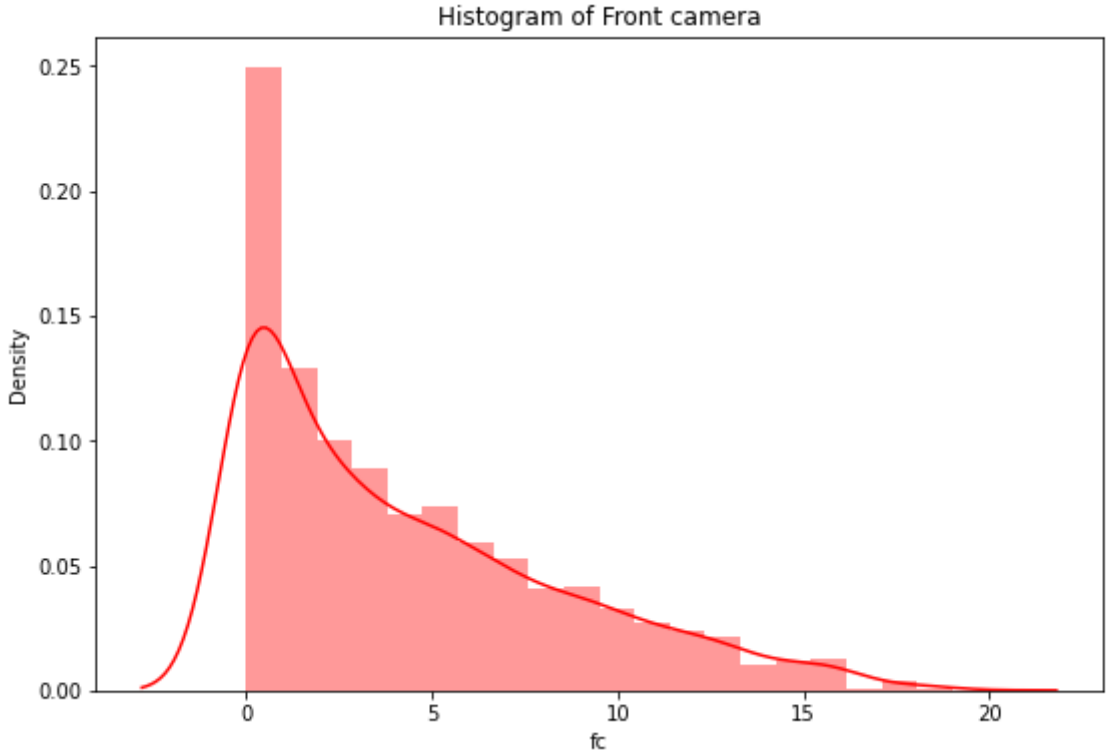


Price Range w.r.t Pixel Height and Width:

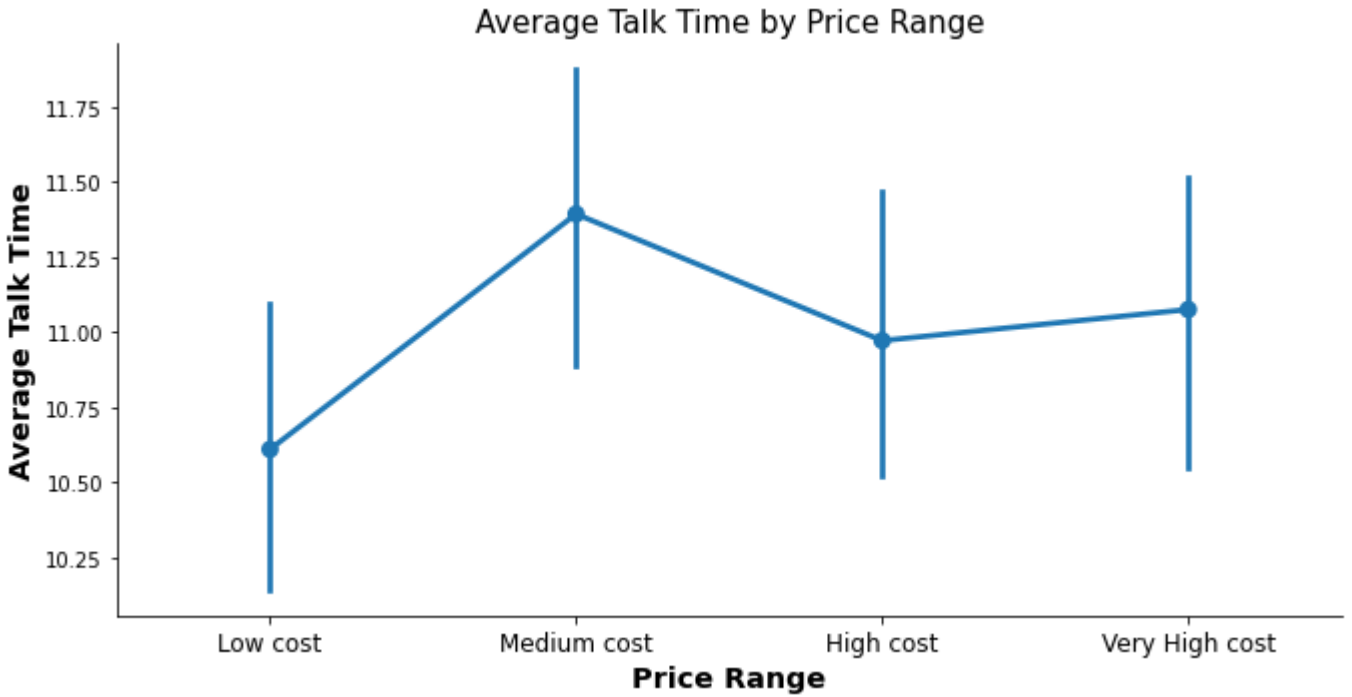




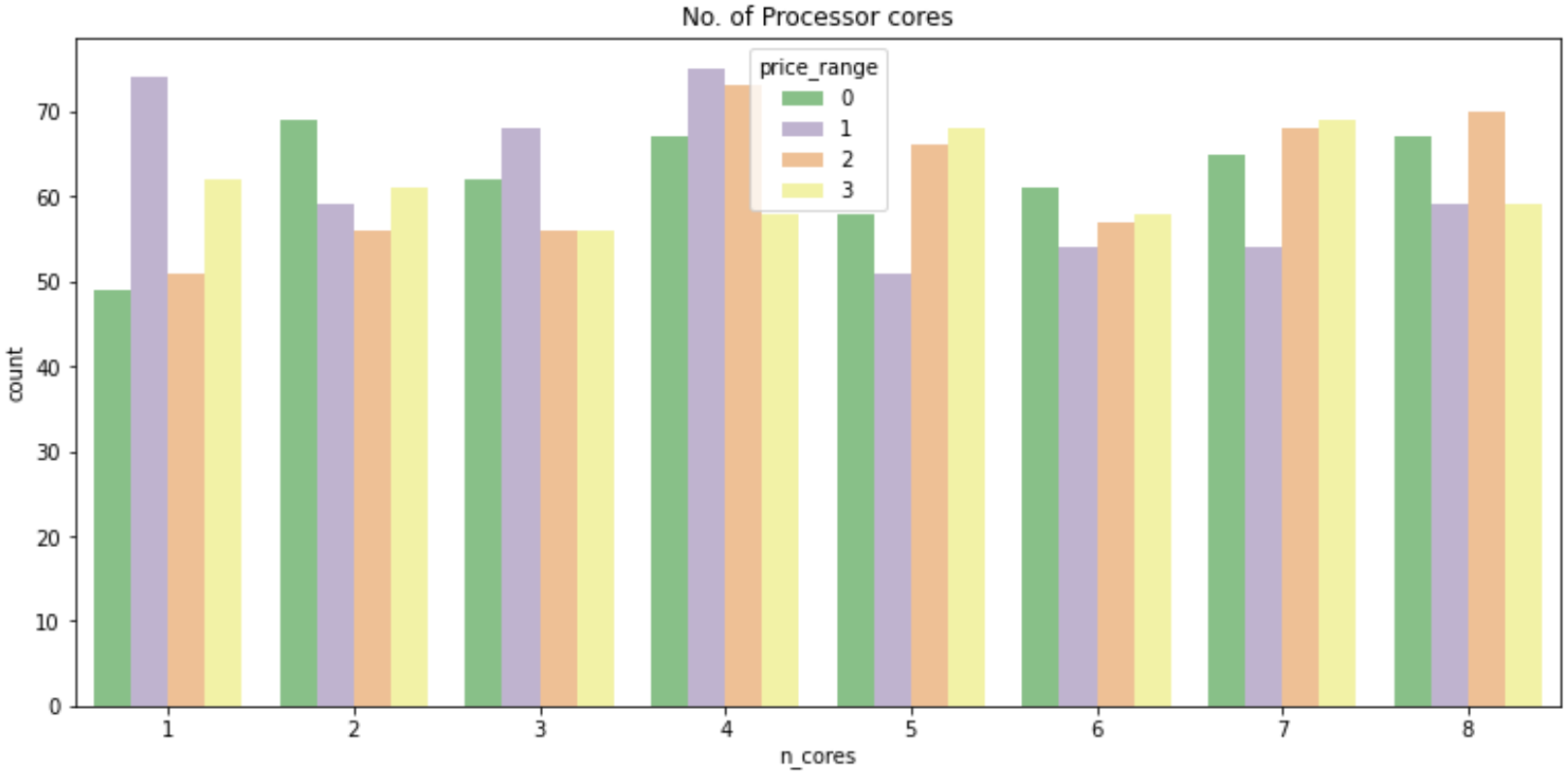
Front Camera Histogram:



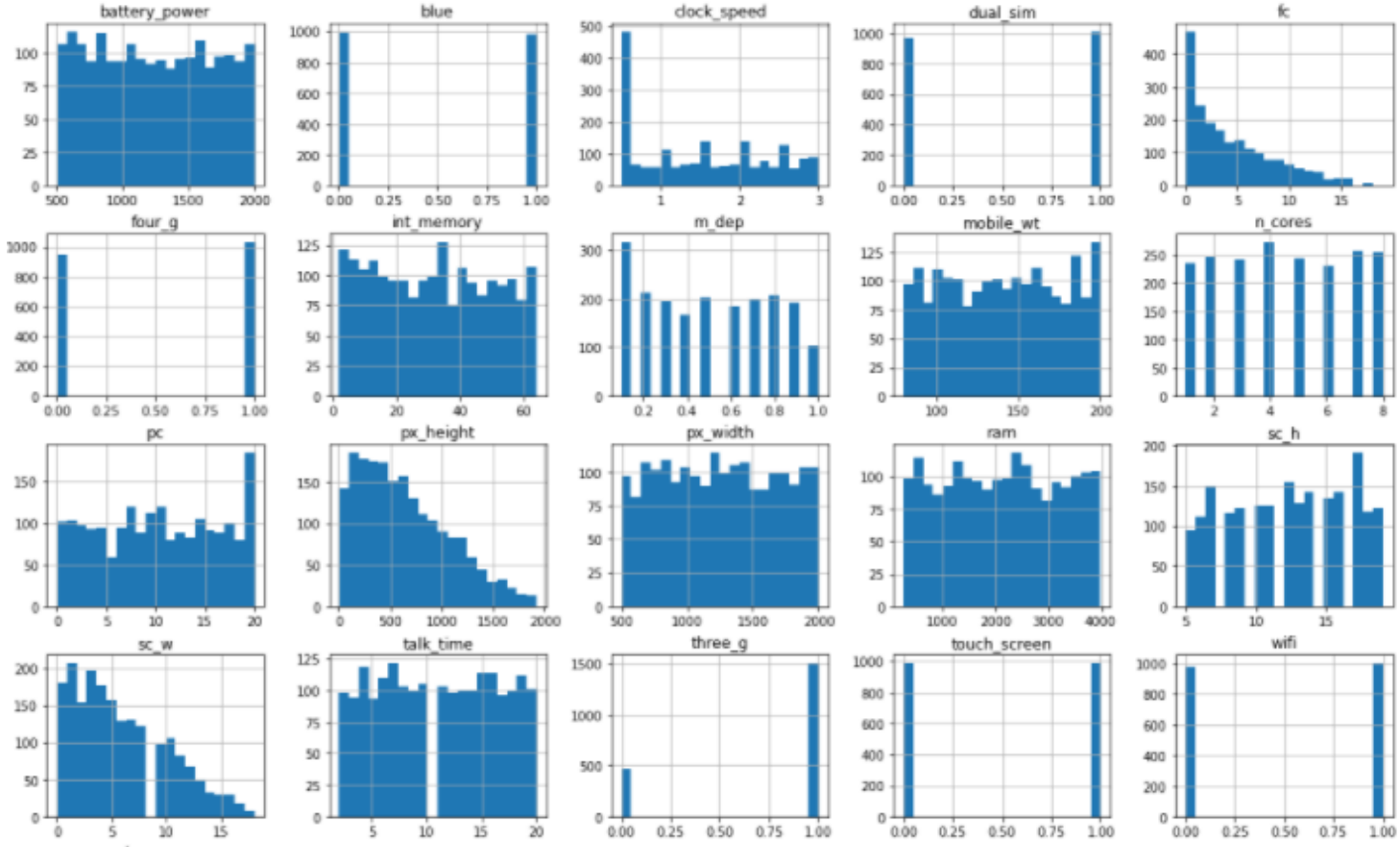
Price Range vs Average Talk Time:



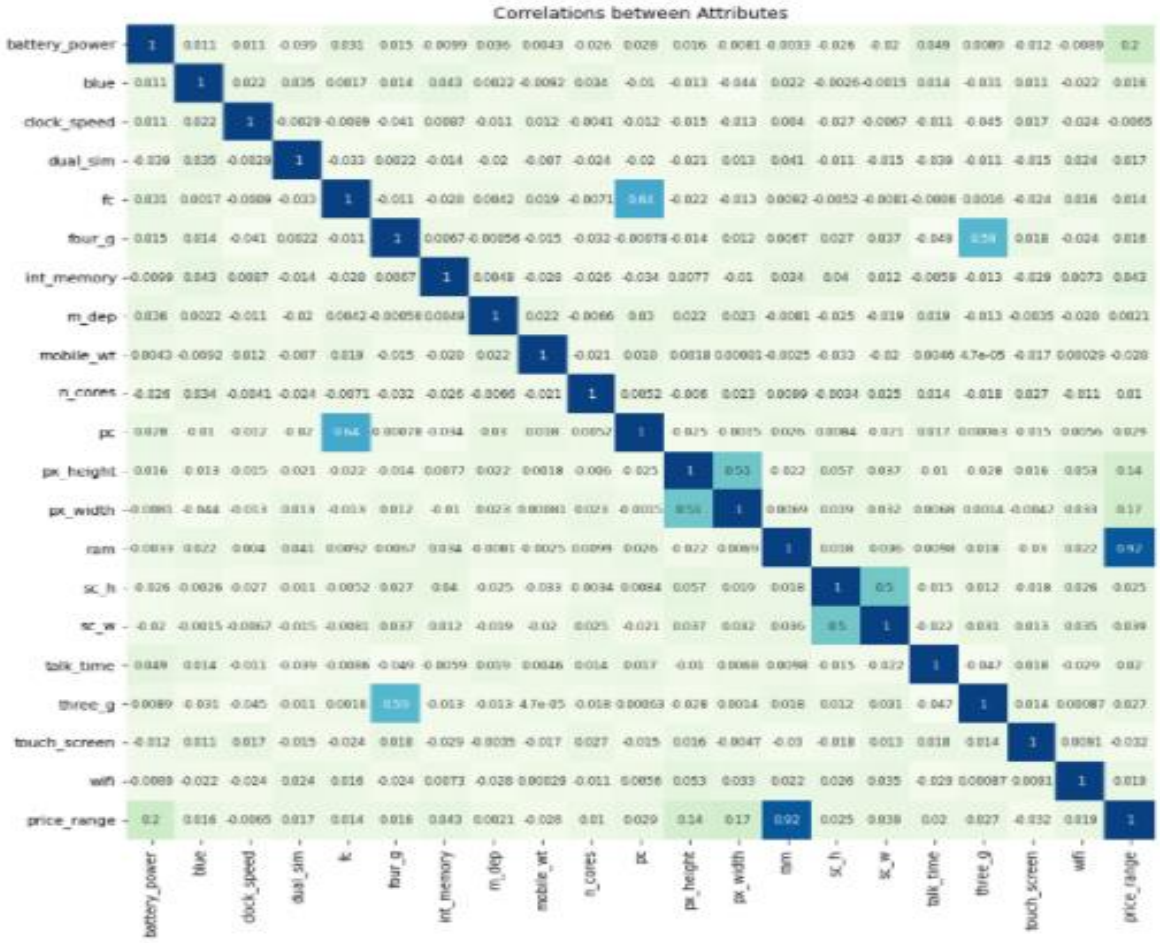
Price Range vs No. of Processor Cores:



Histogram of All Features:



Correlation:



Preparing dataset for modeling :

Task :

- Logistic Regression
- K Nearest neighbors
- Support vector machine
- Decision Tree

Train test split (70%-30%)

Train Set : (1386, 18)

Test Set : (594, 18)

Dependent Variable :

Price Range

```
[ ] X = dataset.drop("price_range" , axis = 1)
    y = dataset["price_range"]
```

```
[ ] X.shape
```

```
(1980, 18)
```

```
[ ] y.shape
```

```
(1980,)
```

```
[ ] #Train-test split

    X_train, X_test, y_train , y_test = train_test_split(X , y , test_size = 0.3 , random_state = 42)
```

```
[ ] X_train.shape
```

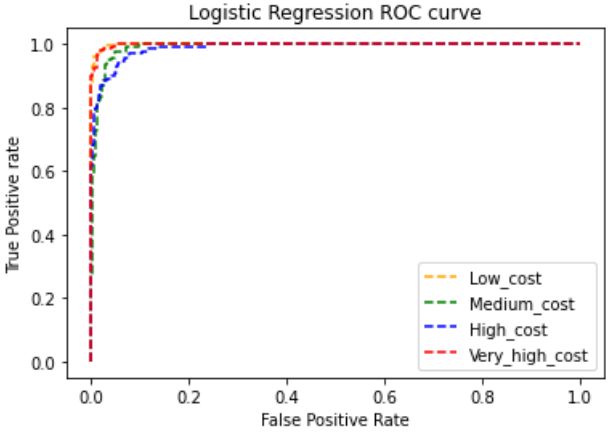
```
(1386, 18)
```

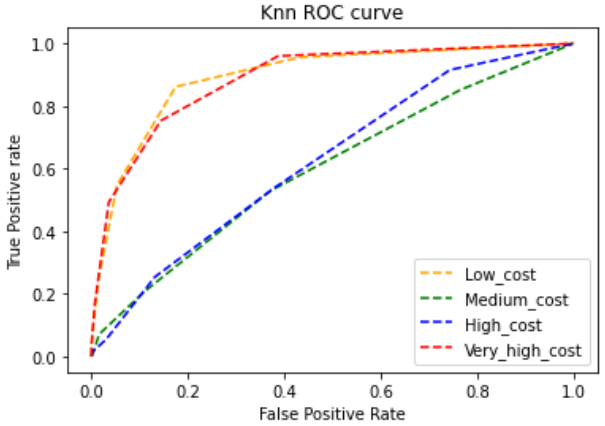
```
[ ] X_test.shape
```

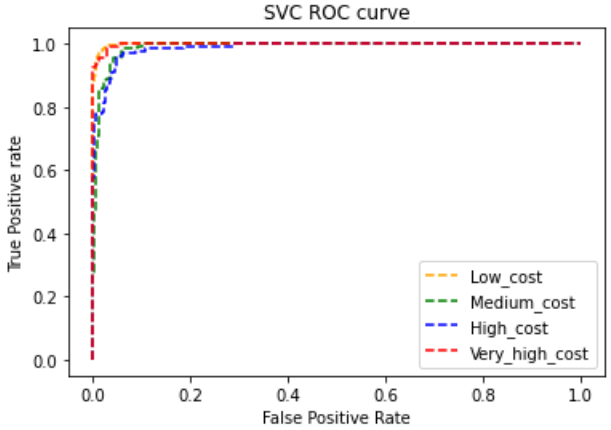
```
(594, 18)
```

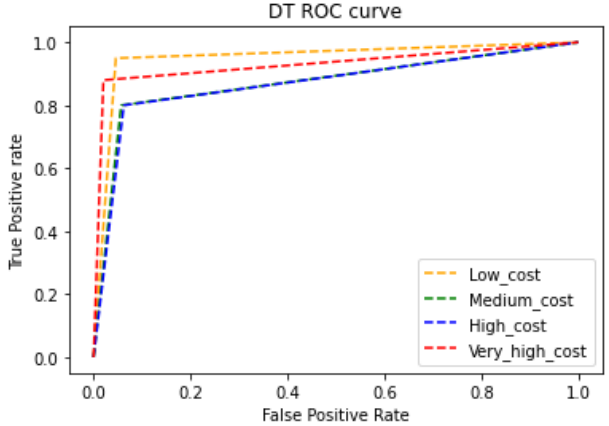
```
[ ] # Standardization using Standard scaler

    from sklearn.preprocessing import StandardScaler
    scaler = StandardScaler()

    X_train = scaler.fit_transform(X_train)
    X_test = scaler.transform(X_test)
```

ROC AUC Curve:

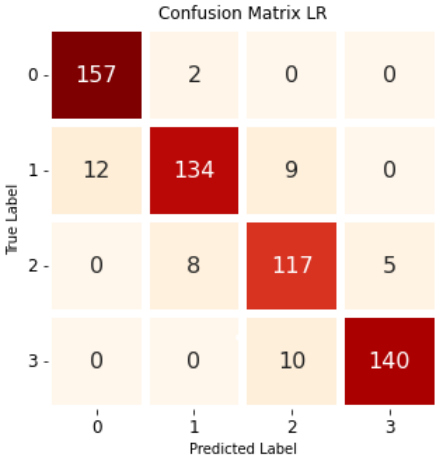


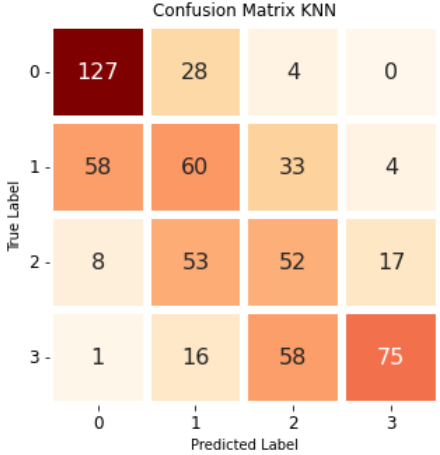


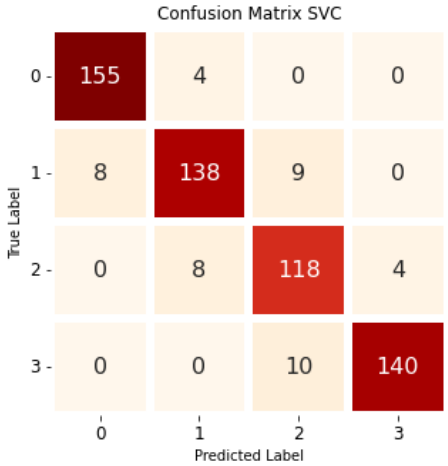


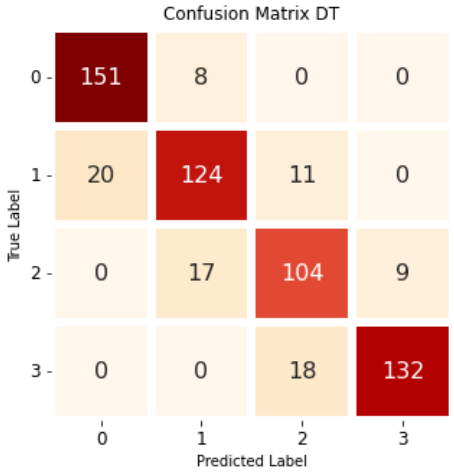


Confusion Matrix

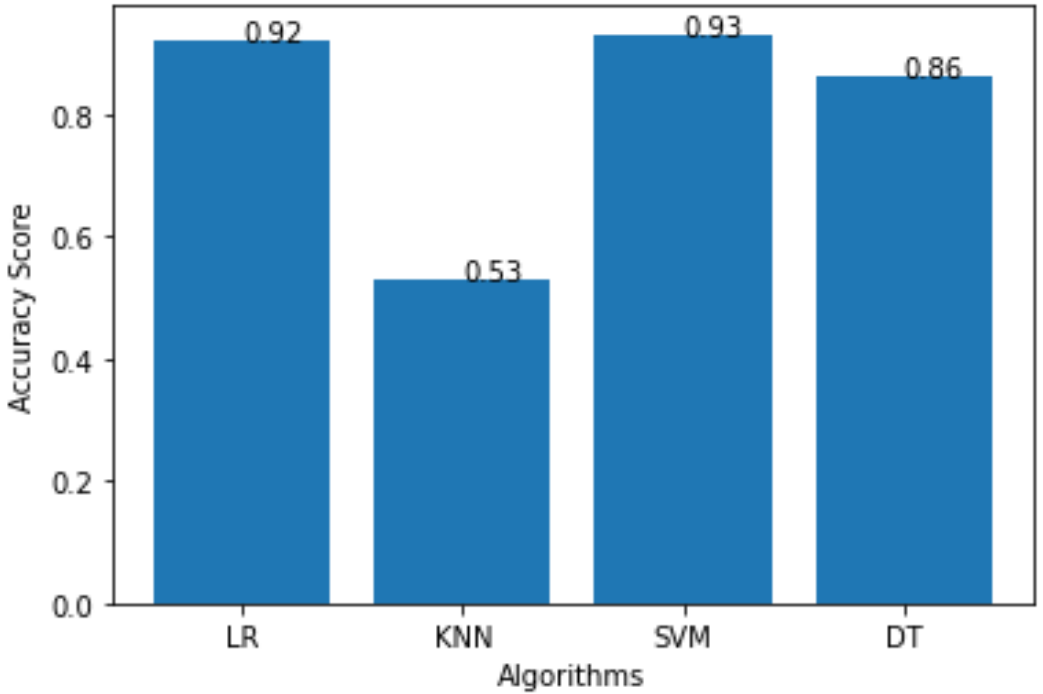








Comparisons of all Models :



Classification Report

	precision	recall	f1-score	support
0	0.93	0.99	0.96	159
1	0.94	0.88	0.91	155
2	0.88	0.92	0.90	130
3	0.97	0.93	0.95	150
accuracy			0.93	594
macro avg	0.93	0.93	0.93	594
weighted avg	0.93	0.93	0.93	594

Logistic regression

	precision	recall	f1-score	support
0	0.93	0.99	0.96	159
1	0.93	0.88	0.90	155
2	0.87	0.90	0.89	130
3	0.97	0.94	0.96	150
accuracy			0.93	594
macro avg	0.93	0.93	0.93	594
weighted avg	0.93	0.93	0.93	594

Support Vector Machine

Conclusion :

- The best algorithm for this dataset is Logistic Regression as compared to the rest of the algorithms.
- Logistic Regression has highest accuracy whereas KNN has the lowest accuracy among all algorithms.
- After tuning the algorithm using Grid Search CV on Logistic Regression and Support Vector Machine the score is not getting much difference compared to previous results.

Challenges :

- Apart from RAM, most of features have very low correlation so feature selection was challengeable.
- There is not much improvement in accuracy even after hyperparameter tuning.

Thank you!