Capstone Project

Mobile Price Range Prediction

Chandrashekhar Awate

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

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
Price_range - This is the target variable with value of 0(low cost), 1(medium cost), 2(high cost) and 3(very high cost).

Content

Step 1

Exploratory Data Analysis

1)Data Exploration

2)Data Analysis

3)Finding Some key Insights

4)Correlation

5)Studying the Factors affecting Mobile price

Step 2

Model Building And Evaluation

1)Pre-proccesing

2)Support Vector Machine

3)Logistic Regression

4)Decision Tree Classifier

5)Naïve Bayes Model

6)Evaluation Metrics

7)Hyperparameter Tuning

8)Conclusions

Exploratory Data Analysis

Reading the Data

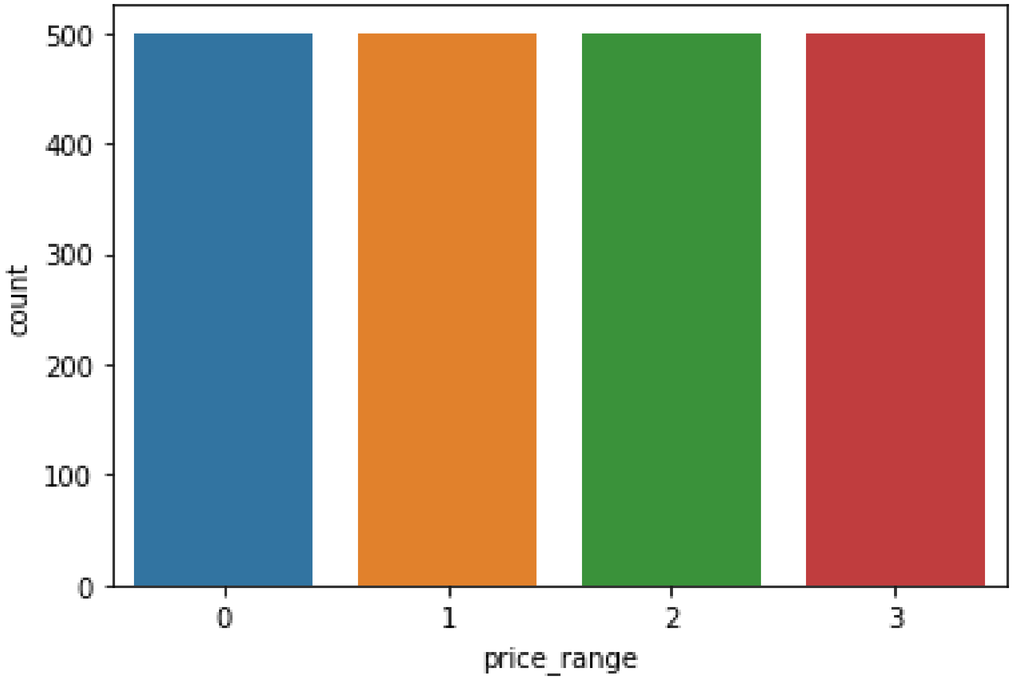
```
[ ] # Lets look at top records
    df.head(5)
```

	battery_power	blue	clock_speed	dual_sim	fc	four_g	int_memory	m_dep	mobile_wt	n_cores	...	px_height	px_width	ram	sc_h	sc_w	talk_time	three_g	touch_
0	842	0	2.2	0	1	0	7	0.6	188	2	...	20	756	2549	9	7	19	0	
1	1021	1	0.5	1	0	1	53	0.7	136	3	...	905	1988	2631	17	3	7	1	
2	563	1	0.5	1	2	1	41	0.9	145	5	...	1263	1716	2603	11	2	9	1	
3	615	1	2.5	0	0	0	10	0.8	131	6	...	1216	1786	2769	16	8	11	1	
4	1821	1	1.2	0	13	1	44	0.6	141	2	...	1208	1212	1411	8	2	15	1	

5 rows × 21 columns

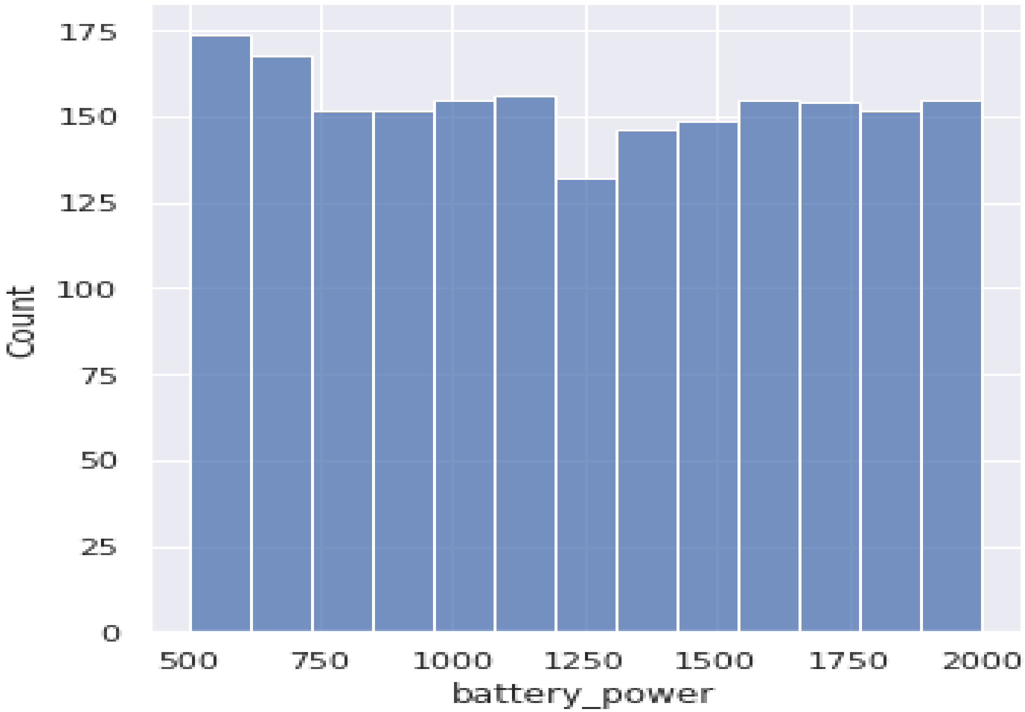
Exploratory Data Analysis

- **Checking class Imbalance for target variable**

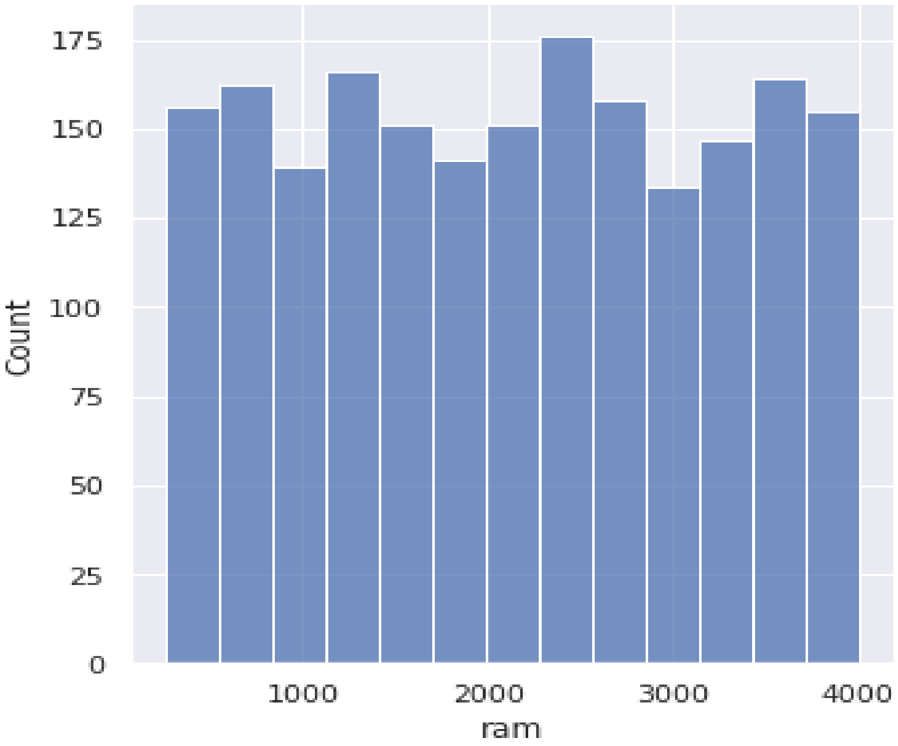


Exploratory Data Analysis

Range of Battery power

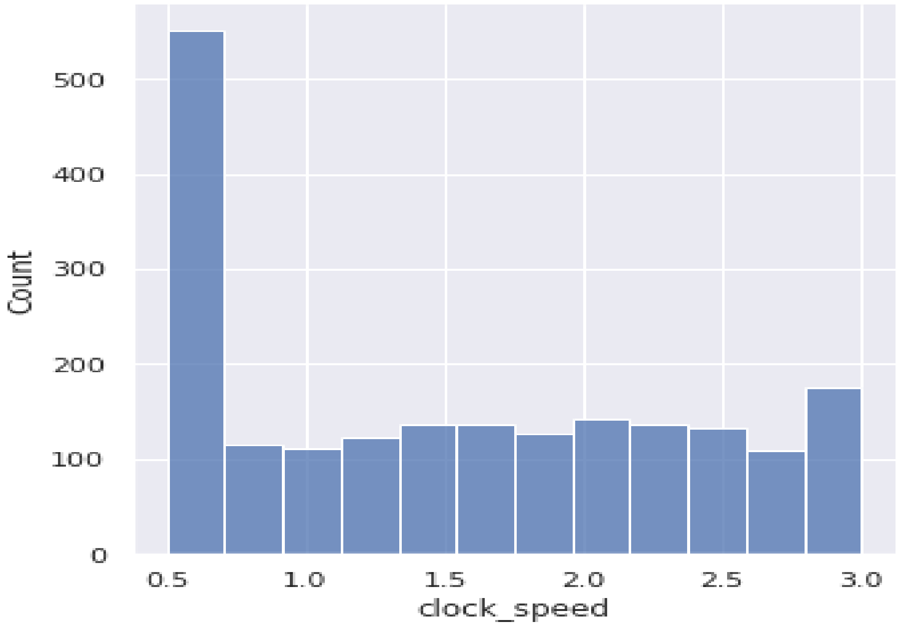


Exploratory Data Analysis



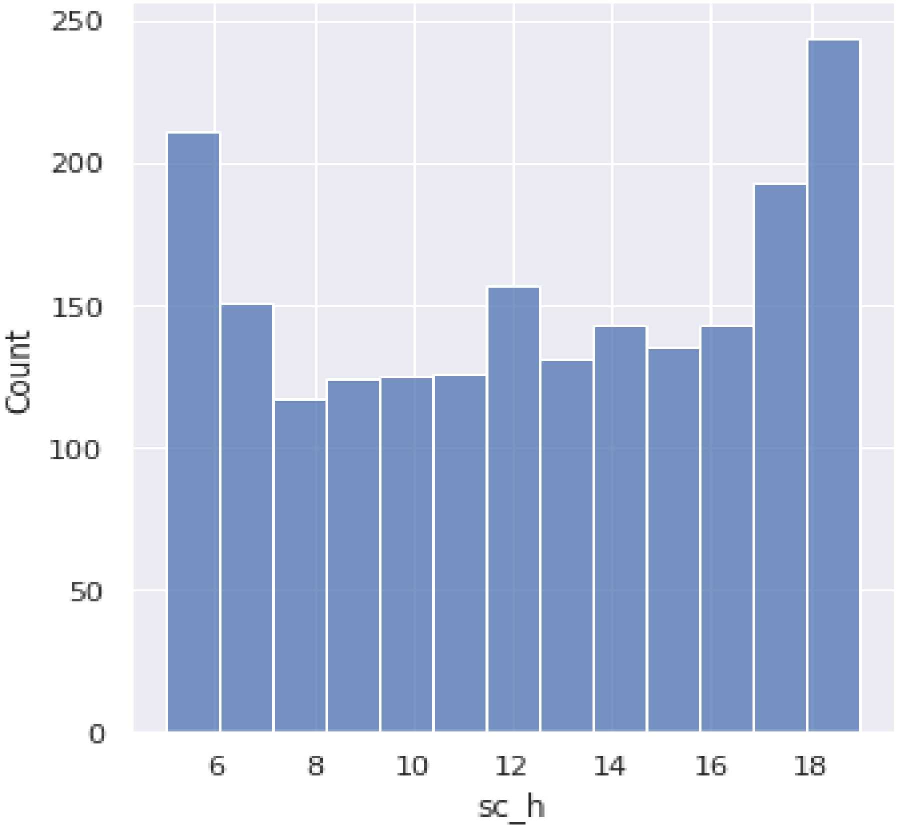
Exploratory Data Analysis

Clock speed



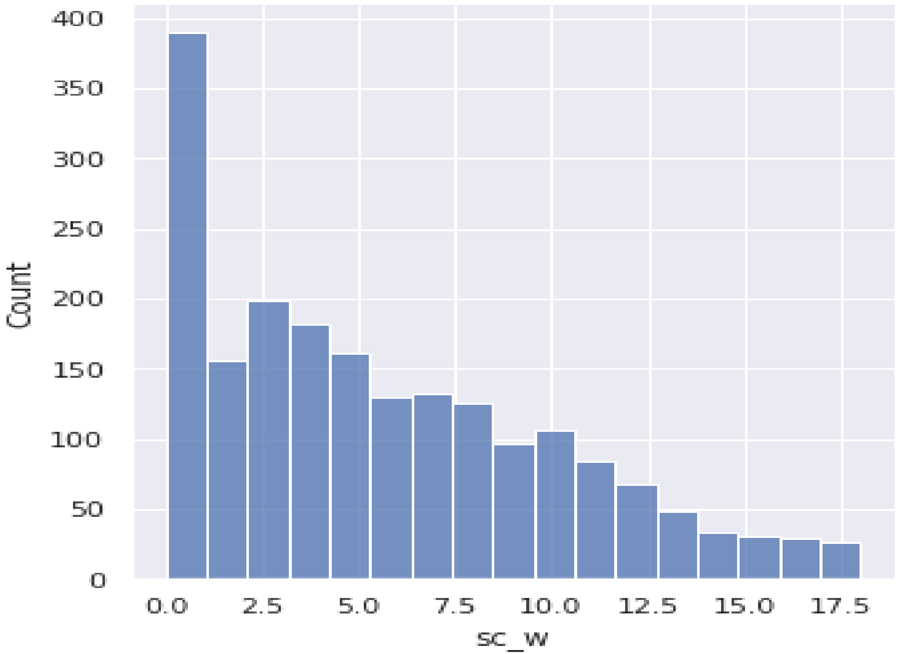
Exploratory Data Analysis

- Screen Hight



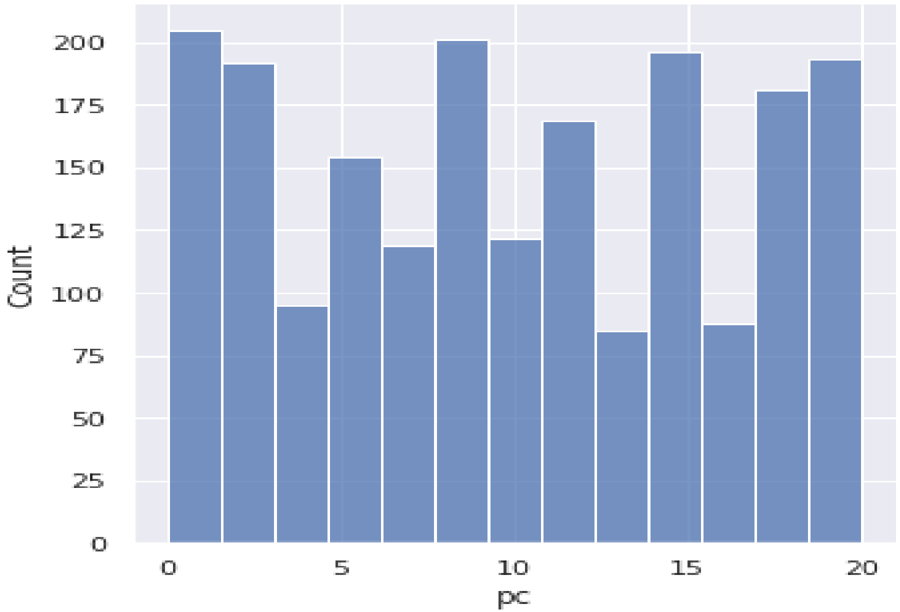
Exploratory Data Analysis

- Screen width



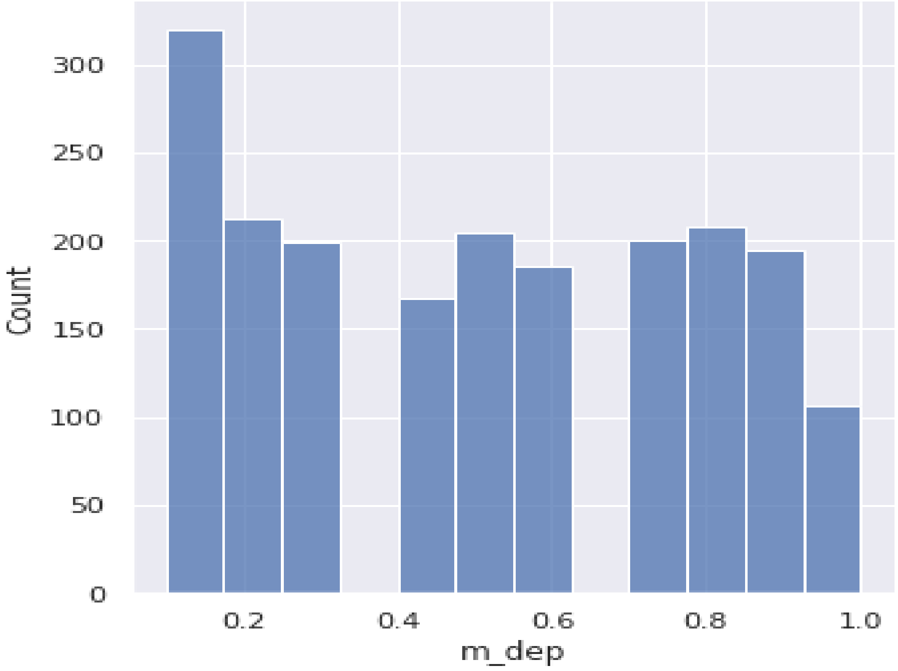
Exploratory Data Analysis

- Primary camera in Megapixels



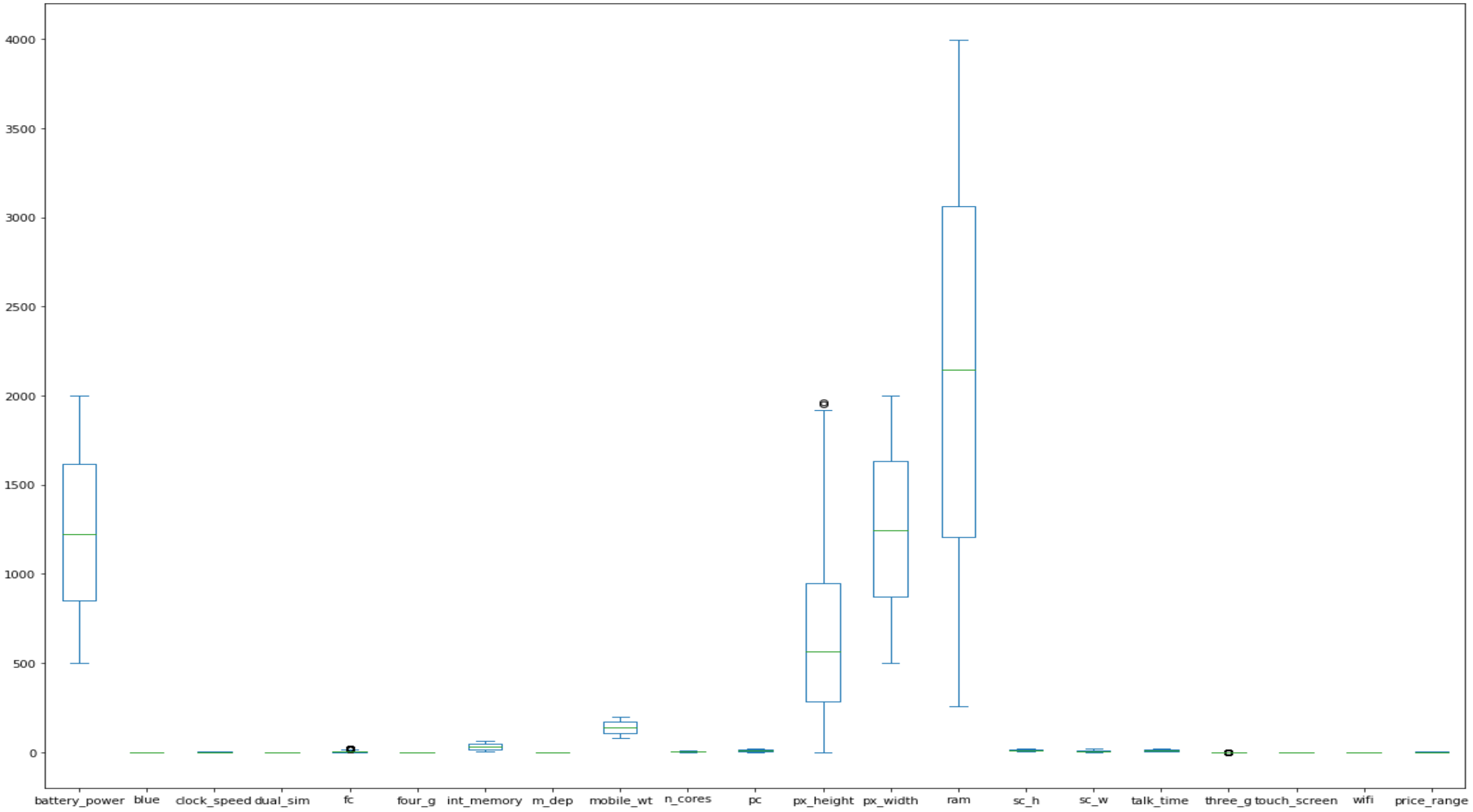
Exploratory Data Analysis

- Mobile Thickness

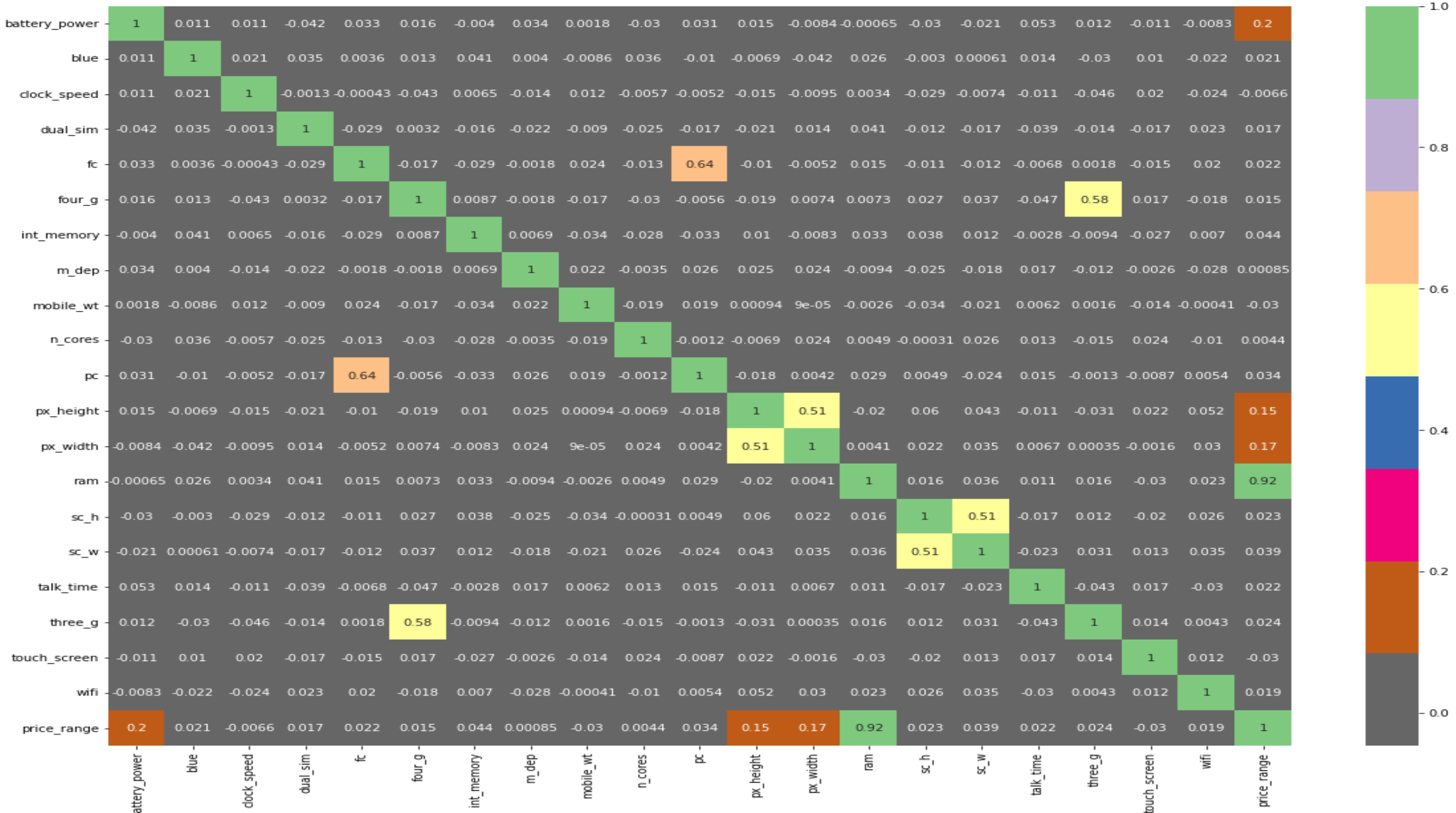


Exploratory Data Analysis

- Outliers



Correlation

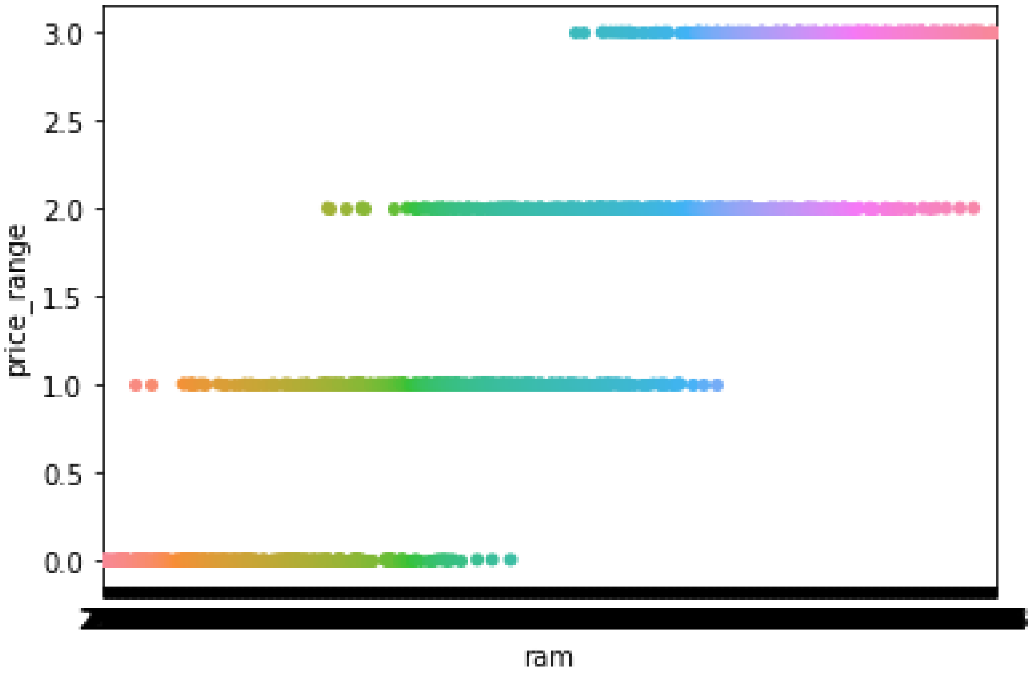


Correlation of Target Variable

```
[59] df.corr()["price_range"]

battery_power     0.200723
blue              0.020573
clock_speed      -0.006606
dual_sim          0.017444
fc                0.021998
four_g            0.014772
int_memory        0.044435
m_dep             0.000853
mobile_wt        -0.030302
n_cores           0.004399
pc                0.033599
px_height         0.148858
px_width          0.165818
ram               0.917046
sc_h              0.022986
sc_w              0.038711
talk_time         0.021859
three_g           0.023611
touch_screen     -0.030411
wifi              0.018785
price_range       1.000000
Name: price_range, dtype: float64
```

Ram and Price of mobile



Modelling And Evaluation

1)Support Vector Machine

Support Vector Machines Work by constructing a hyperplane that separates points between two classes. The hyperplane is determined using the maximal margin hyperplane, which is the hyperplane that is the maximum distance from the training observations. This distance is called the margin. Points that fall on one side of the hyperplane are classified as -1 and the other +1.

```
[ ] from sklearn.metrics import accuracy_score,confusion_matrix
    svc_acc = accuracy_score(pred1,Y_test)
    print(svc_acc)
    print(confusion_matrix(pred1,Y_test))
```

```
0.878
[[125   4   0   0]
 [ 11  98  18   0]
 [  0   8 106  13]
 [  0   0   7 110]]
```

```
[ ] # Get the accuracy scores
    train_class_preds = svc.predict(X_train)
    test_class_preds = svc.predict(X_test)
    train_accuracy = accuracy_score(train_class_preds,Y_train)
    test_accuracy = accuracy_score(test_class_preds,Y_test)

    print("The accuracy on train data is ", train_accuracy)
    print("The accuracy on test data is ", test_accuracy)
```

```
The accuracy on train data is  0.9866666666666667
The accuracy on test data is  0.878
```

Some sort of overfitting is seen in our svc model.

Modelling And Evaluation

2)Logistic Regression

Logistic regression is used for classification, where the response variable is categorical rather than numerical. The model works by predicting the probability that Y belongs to a particular category by first fitting the data to a linear regression model, which is then passed to the logistic function (below). The logistic function will always produce a S-shaped curve, so regardless of X, we can always obtain a sensible answer (between 0 and 1).

```
[ ] lr_acc = accuracy_score(pred2,Y_test)
    print(lr_acc)
    print(confusion_matrix(pred2,Y_test))
```

```
0.962
[[130   3   0   0]
 [  6 105   6   0]
 [  0   2 124   1]
 [  0   0   1 122]]
```

```
[ ] # Get the accuracy scores
    train_class_preds = lr.predict(X_train)
    test_class_preds = lr.predict(X_test)
    train_accuracy = accuracy_score(train_class_preds,Y_train)
    test_accuracy = accuracy_score(test_class_preds,Y_test)

    print("The accuracy on train data is ", train_accuracy)
    print("The accuracy on test data is ", test_accuracy)
```

```
The accuracy on train data is  0.9746666666666667
The accuracy on test data is  0.962
```

Yes,we got optimal model for our problem.

Modelling And Evaluation

3) Decision Tree Classifier

Binary branching structure used to classify an arbitrary input vector X. Each node in the tree contains a simple feature comparison against some field (xi > 42?). Result of each comparison is either true or false, which determines if we should proceed along to the left or right child of the given node. Also known as sometimes called classification and regression trees (CART).

```
[ ]  from sklearn.metrics import accuracy_score,confusion_matrix
     dtc_acc = accuracy_score(pred3,Y_test)
     print(dtc_acc)
     print(confusion_matrix(pred3,Y_test))
```

```
0.84
[[124  11   0   0]
 [ 11  88  13   0]
 [  1  11  97  12]
 [  0   0  21 111]]
```

```
[ ]  # Get the accuracy scores
     train_class_preds = dtc.predict(X_train)
     test_class_preds = dtc.predict(X_test)
     train_accuracy = accuracy_score(train_class_preds,Y_train)
     test_accuracy = accuracy_score(test_class_preds,Y_test)

     print("The accuracy on train data is ", train_accuracy)
     print("The accuracy on test data is ", test_accuracy)
```

```
The accuracy on train data is  1.0
The accuracy on test data is  0.84
```

There is some sort of overfitting seen in Decision tree model also.

Modelling And Evaluation

- 4)Naive Bayes Model
- It is a Classification Technique based on Bayes' Theorem with an assumption of independence among predictors. In simple terms, a Naive Bayes classifier assumes that the presence of a particular feature in a class is unrelated to the presence of any other feature.

```
[57] from sklearn.metrics import accuracy_score,confusion_matrix
     classifier_acc = accuracy_score(pred4,Y_test)
     print(classifier_acc)
     print(confusion_matrix(pred4,Y_test))

     0.804
     [[122   5   0   0]
      [ 14  79  27   0]
      [  0  26  94  16]
      [  0   0  10 107]]
```

```
[55] # Get the accuracy scores
     train_class_preds = classifier.predict(X_train)
     test_class_preds = classifier.predict(X_test)
     train_accuracy = accuracy_score(train_class_preds,Y_train)
     test_accuracy = accuracy_score(test_class_preds,Y_test)

     print("The accuracy on train data is ", train_accuracy)
     print("The accuracy on test data is ", test_accuracy)

     The accuracy on train data is  0.8186666666666667
     The accuracy on test data is  0.804
```

After logistic regression this model working fine on both test and train data with eual accuracy on both so we can say that this is the best model after logistic regression

Confusion matrics

		Condition (as determined by "Gold standard")		
		Condition Positive	Condition Negative	
Test Outcome	Test Outcome Positive	**True Positive**	**False Positive** (Type I error)	Positive predictive value = $\frac{\Sigma \text{ True Positive}}{\Sigma \text{ Test Outcome Positive}}$
	Test Outcome Negative	**False Negative** (Type II error)	**True Negative**	Negative predictive value = $\frac{\Sigma \text{ True Negative}}{\Sigma \text{ Test Outcome Negative}}$
		Sensitivity = $\frac{\Sigma \text{ True Positive}}{\Sigma \text{ Condition Positive}}$	**Specificity** = $\frac{\Sigma \text{ True Negative}}{\Sigma \text{ Condition Negative}}$	

Evaluation Metrics

- **Metrics that can provide better insight are:**
- **Confusion Matrix**: a table showing correct predictions and types of incorrect predictions.
- **Precision**: the number of true positives divided by all positive predictions. Precision is also called Positive Predictive Value. It is a measure of a classifier’s exactness. Low precision indicates a high number of false positives.
- **Recall**: the number of true positives divided by the number of positive values in the test data. The recall is also called Sensitivity or the True Positive Rate. It is a measure of a classifier’s completeness. Low recall indicates a high number of false negatives.
- **F1 Score**: the weighted average of precision and recall.

1)Support vector Machine

```
[ ]  from sklearn.metrics import classification_report
     print(classification_report(Y_test, pred1))
```

```
              precision    recall  f1-score   support

           0       0.97      0.92      0.94       136
           1       0.77      0.89      0.83       110
           2       0.83      0.81      0.82       131
           3       0.94      0.89      0.92       123

    accuracy                           0.88       500
   macro avg       0.88      0.88      0.88       500
weighted avg       0.88      0.88      0.88       500
```

2)Logistic Regression

```
[ ]  from sklearn.metrics import classification_report
     print(classification_report(Y_test, pred2))
```

```
              precision    recall  f1-score   support

           0       0.98      0.96      0.97       136
           1       0.90      0.95      0.93       110
           2       0.98      0.95      0.96       131
           3       0.99      0.99      0.99       123

    accuracy                           0.96       500
   macro avg       0.96      0.96      0.96       500
weighted avg       0.96      0.96      0.96       500
```

3)Decision Tree Classifier

```
[ ]  from sklearn.metrics import classification_report
     print(classification_report(Y_test, pred3))
```

```
              precision    recall  f1-score   support

           0       0.93      0.90      0.92       136
           1       0.78      0.81      0.79       110
           2       0.78      0.73      0.76       131
           3       0.83      0.89      0.86       123

    accuracy                           0.83       500
   macro avg       0.83      0.83      0.83       500
weighted avg       0.83      0.83      0.83       500
```

4) Naive Bayes

```
[ ]  from sklearn.metrics import classification_report
     print(classification_report(Y_test, pred4))
```

```
              precision    recall  f1-score   support

           0       0.96      0.90      0.93       136
           1       0.66      0.72      0.69       110
           2       0.69      0.72      0.70       131
           3       0.91      0.87      0.89       123

    accuracy                           0.80       500
   macro avg       0.81      0.80      0.80       500
weighted avg       0.81      0.80      0.81       500
```

Hyperparameter Tuning

Hyperparameter tuning for logistic regression

```
[ ] import numpy as np
    from sklearn.model_selection import GridSearchCV
    import warnings
    warnings.filterwarnings('ignore')
    # parameter grid
    parameters = {
        'penalty' : ['l1','l2'],
        'C'       : np.logspace(-3,3,7),
        'solver'  : ['newton-cg', 'lbfgs', 'liblinear'],
    }
```

```
[ ] logreg = LogisticRegression()
    clf = GridSearchCV(logreg,                    # model
                       param_grid = parameters,   # hyperparameters
                       scoring='accuracy',        # metric for scoring
                       cv=10)                     # number of folds
```

```
[ ] clf.fit(X_train,Y_train)
```

```
GridSearchCV(cv=10, estimator=LogisticRegression(),
             param_grid={'C': array([1.e-03, 1.e-02, 1.e-01, 1.e+00, 1.e+01, 1.e+02, 1.e+03]),
                         'penalty': ['l1', 'l2'],
                         'solver': ['newton-cg', 'lbfgs', 'liblinear']},
             scoring='accuracy')
```

```
[ ] print("Tuned Hyperparameters :", clf.best_params_)
    print("Accuracy :",clf.best_score_)
```

```
Tuned Hyperparameters : {'C': 100.0, 'penalty': 'l2', 'solver': 'newton-cg'}
Accuracy : 0.9640000000000001
```

After Hyperparameter tuning of logistic regression the accuracy Remains same.So we will accept the logistic regression model.

Hyperparameter Tuning for decision tree

```
[ ] params_1 = {'criterion': 'gini', 'splitter': 'best', 'max_depth': 50}
    model_1 = DecisionTreeClassifier(**params_1)
    model_1.fit(X_train, Y_train)
    # Prediction sets
    preds_1 = model_1.predict(X_test)
    print(f'Accuracy on Model 1: {round(accuracy_score(Y_test, preds_1), 3)}')
```

```
Accuracy on Model 1: 0.824
```

```
[ ] # Get the accuracy scores
    train_class_preds = model_1.predict(X_train)
    test_class_preds = model_1.predict(X_test)
    train_accuracy = accuracy_score(train_class_preds,Y_train)
    test_accuracy = accuracy_score(test_class_preds,Y_test)

    print("The accuracy on train data is ", train_accuracy)
    print("The accuracy on test data is ", test_accuracy)
```

```
The accuracy on train data is  1.0
The accuracy on test data is  0.824
```

After hyperparameter tuning also there is no outstanding performance seen by model.

Conclusions:

From Exploratory Data Analysis

1)Majorly impacting properties on price of phone are Ram, Battery power and pixel resolution. while clock speed ,mobile weight and whether it is screen touch phone or not impacting price of mobile negatively.

2) The battery power ranges between 600-2000.

3) Clock speed ranges between 0.6 to 3.

4) The random access memory of the phone ranges between 250 megabytes to 4000 megabytes.

5) The hight of screen of mobile phones ranges between 3cm to 18cm.

6) The width of screen of mobile phones ranges between 2cm to 17cm.

7) The depth or thickness of mobile ranges between something around 0.2cm to 1cm.

8) The camera megapixels ranges between 1 to 20.

9) The front camera of mobile phones ranges between 1 mp to 17mp.

10)There is no class imbalance in Target Variable.

Conclusion

From Models

1)The best performing model is Logistic Regression as it stands outstanding in all Four algorithms with respect to Accuracy, Precision,Recall and F-1 Score.

2)After Logistic Regression the support vector classifier is the best model according to all evaluation Scores.

3)After hyperparameter tuning on logistic and Decision Tree model no outstanding performance noticed.

4)So we will accept the both models that is Logistic Regression Model and Support Vector Machine.