Capstone Project-3
Mobile Price Range Prediction

Team Members

Sachin S Panchal
Shreedarsh M
Ranjith K
Rohan A G
Kshipra Parihar

Contents :

Mobile Price Range Prediction

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.

Attribute Information

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).

Exploring the dataset

	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_screen	wifi	price_range
0	842	0	2.2	0	1	0	7	0.6	188	2	...	20	756	2549	9	7	19	0	0	1	1
1	1021	1	0.5	1	0	1	53	0.7	136	3	...	905	1988	2631	17	3	7	1	1	0	2
2	563	1	0.5	1	2	1	41	0.9	145	5	...	1263	1716	2603	11	2	9	1	1	0	2
3	615	1	2.5	0	0	0	10	0.8	131	6	...	1216	1786	2769	16	8	11	1	0	0	2
4	1821	1	1.2	0	13	1	44	0.6	141	2	...	1208	1212	1411	8	2	15	1	1	0	1

5 rows × 21 columns

```
df.tail()
```

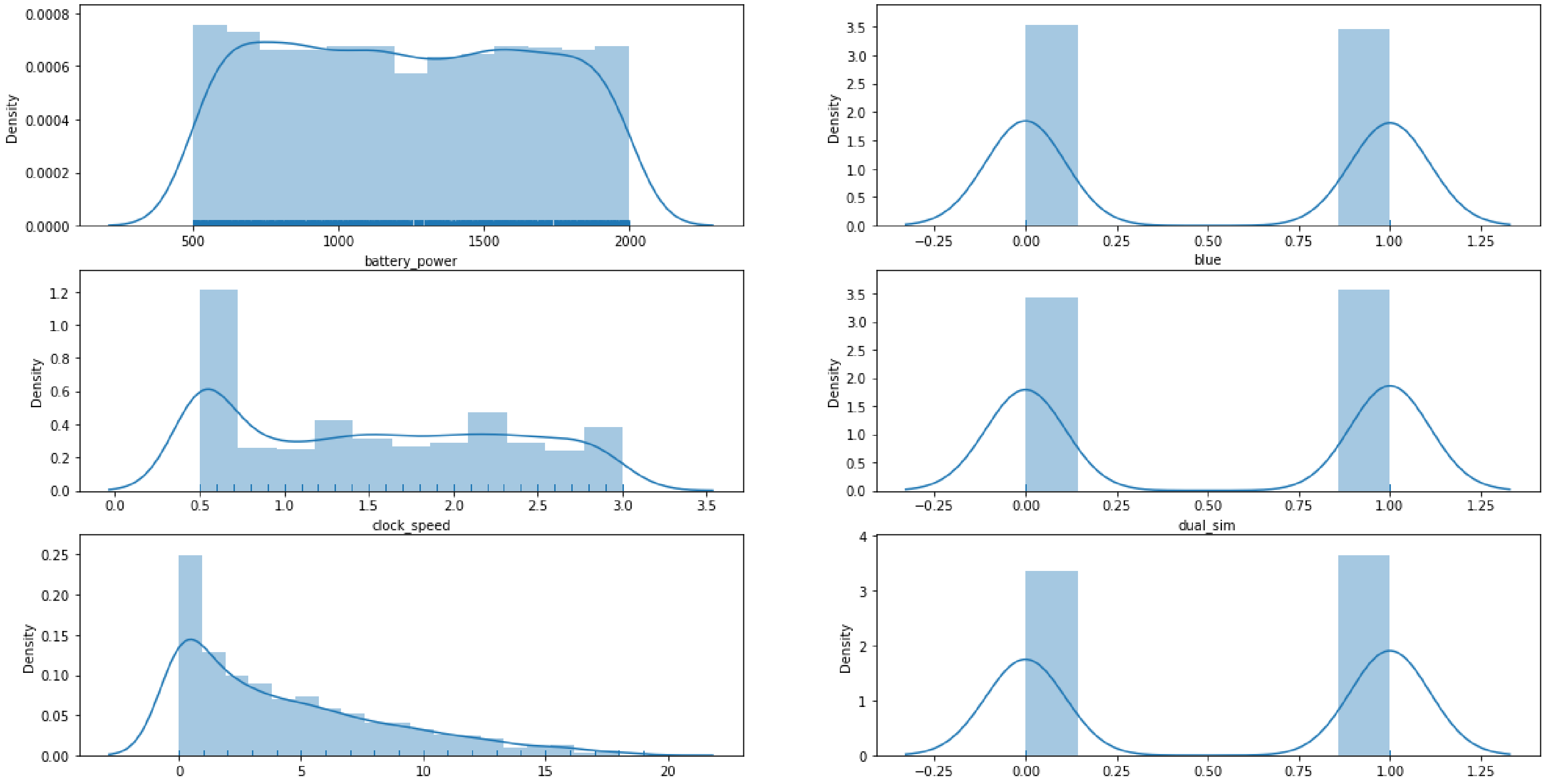
	battery_power	blue	clock_speed	dual_sim	fc	four_g	int_memory	m_dep	mobile_wt	n_cores	...	px_height	px_width	ram	sc_h	sc_w	talk_time	three_g	touch_screen	wifi	price_range
1995	794	1	0.5	1	0	1	2	0.8	106	6	...	1222	1890	668	13	4	19	1	1	0	0
1996	1965	1	2.6	1	0	0	39	0.2	187	4	...	915	1965	2032	11	10	16	1	1	1	2
1997	1911	0	0.9	1	1	1	36	0.7	108	8	...	868	1632	3057	9	1	5	1	1	0	3
1998	1512	0	0.9	0	4	1	46	0.1	145	5	...	336	670	869	18	10	19	1	1	1	0
1999	510	1	2.0	1	5	1	45	0.9	168	6	...	483	754	3919	19	4	2	1	1	1	3

5 rows × 21 columns

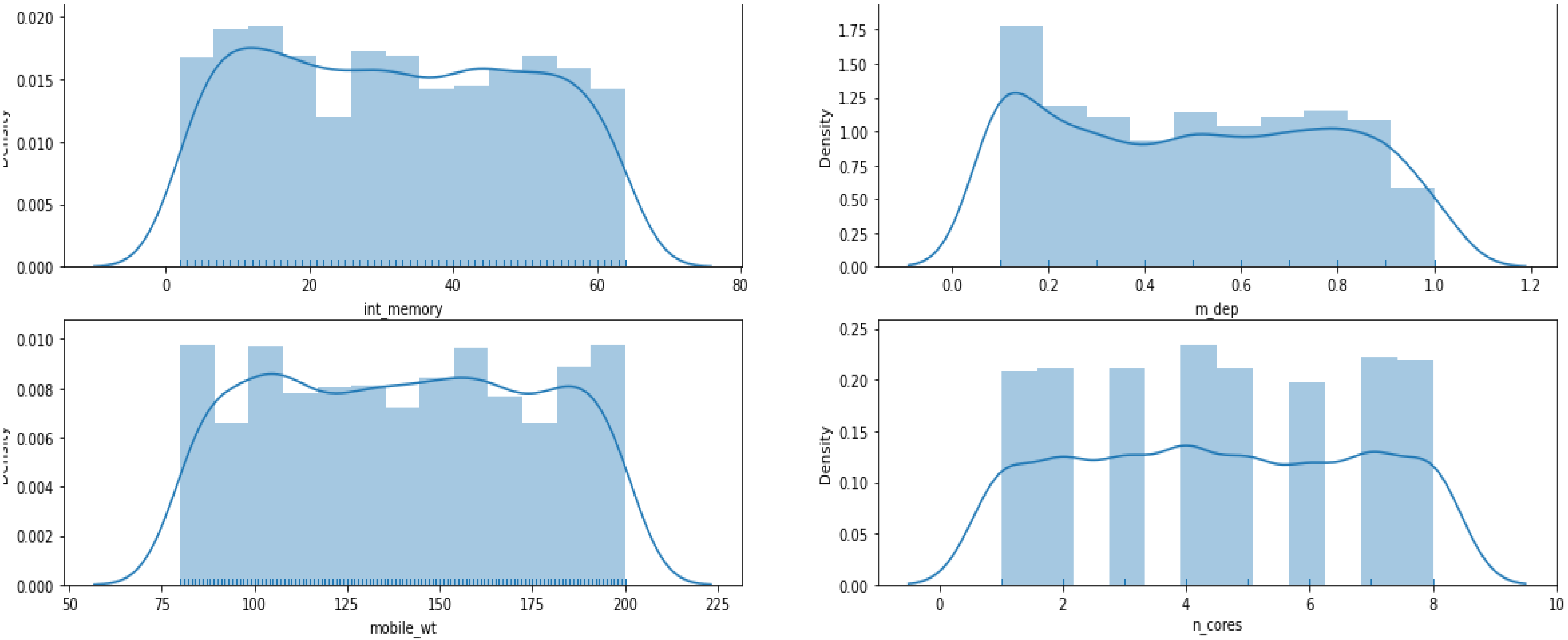
[45]
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 2000 entries, 0 to 1999
Data columns (total 21 columns):
 #   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  
```

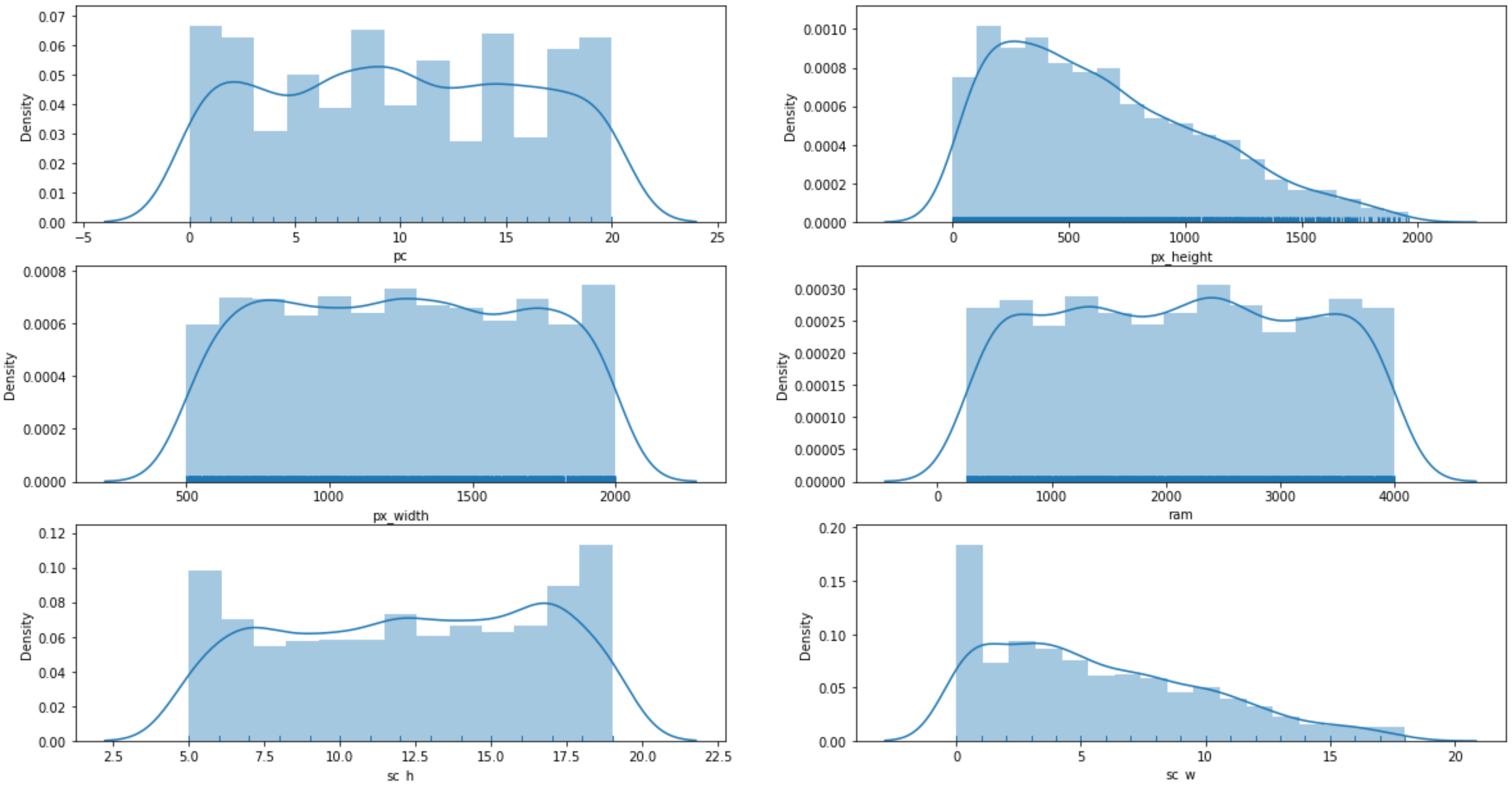
Exploratory Data Analysis



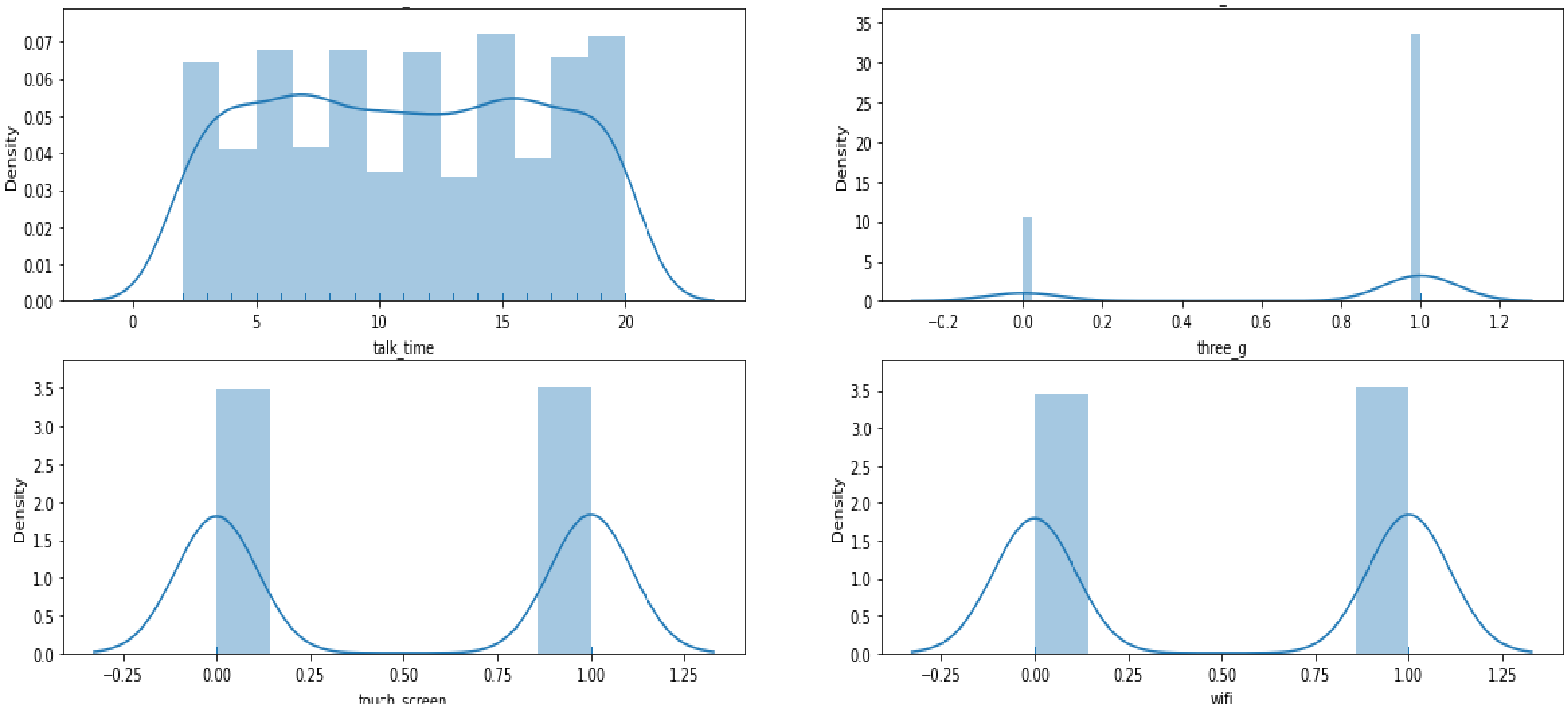
Exploratory Data Analysis



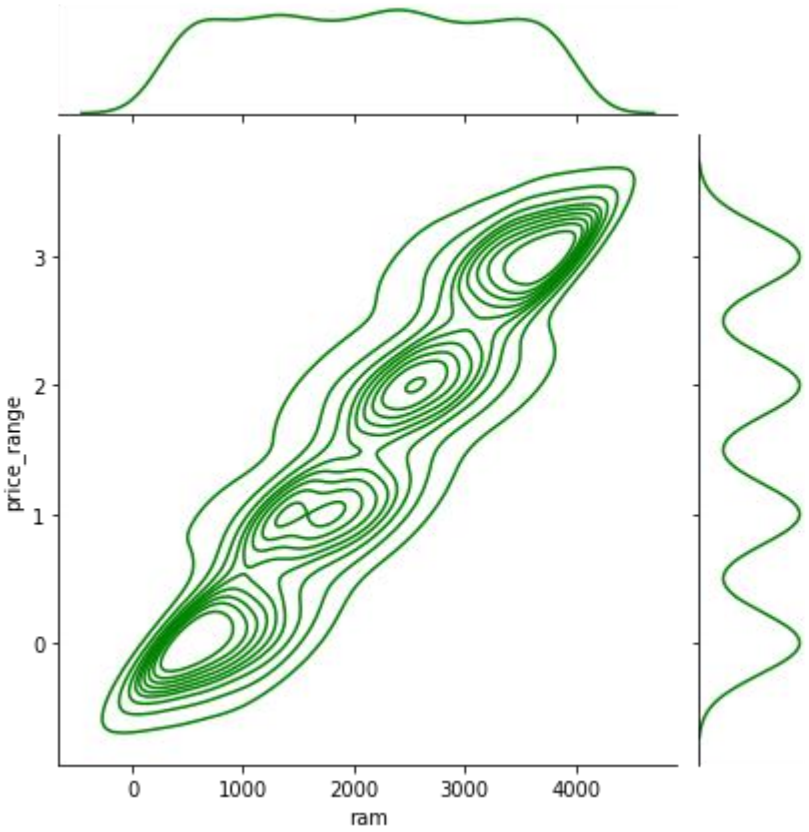
Exploratory Data Analysis

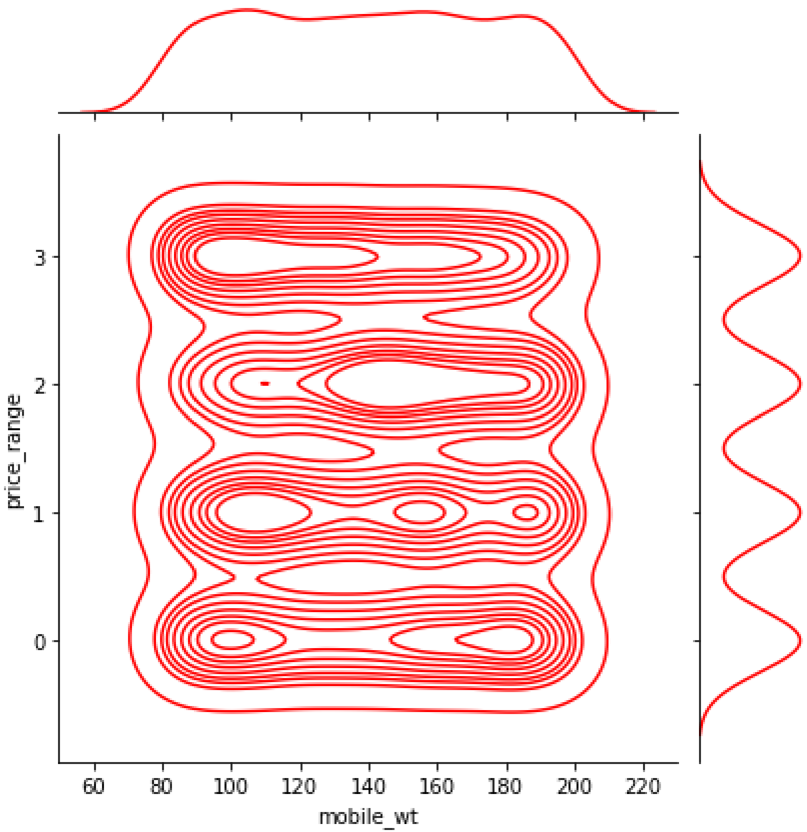


Exploratory Data Analysis

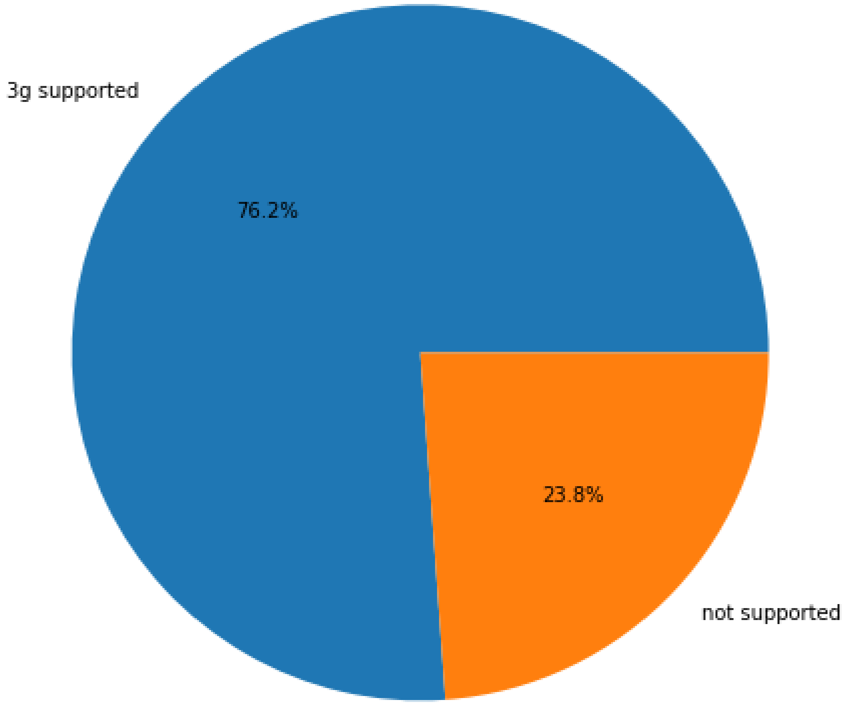


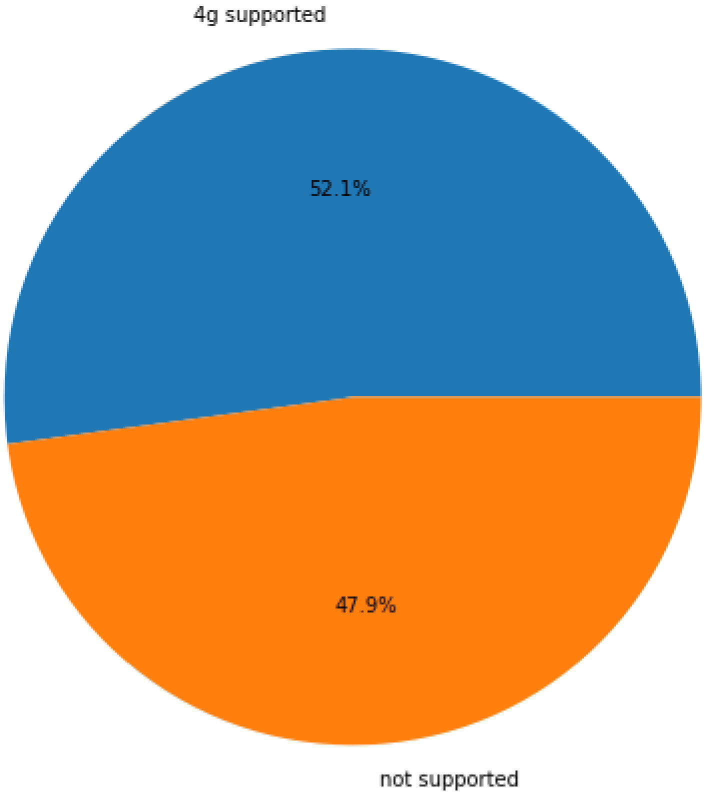
Exploratory Data Analysis



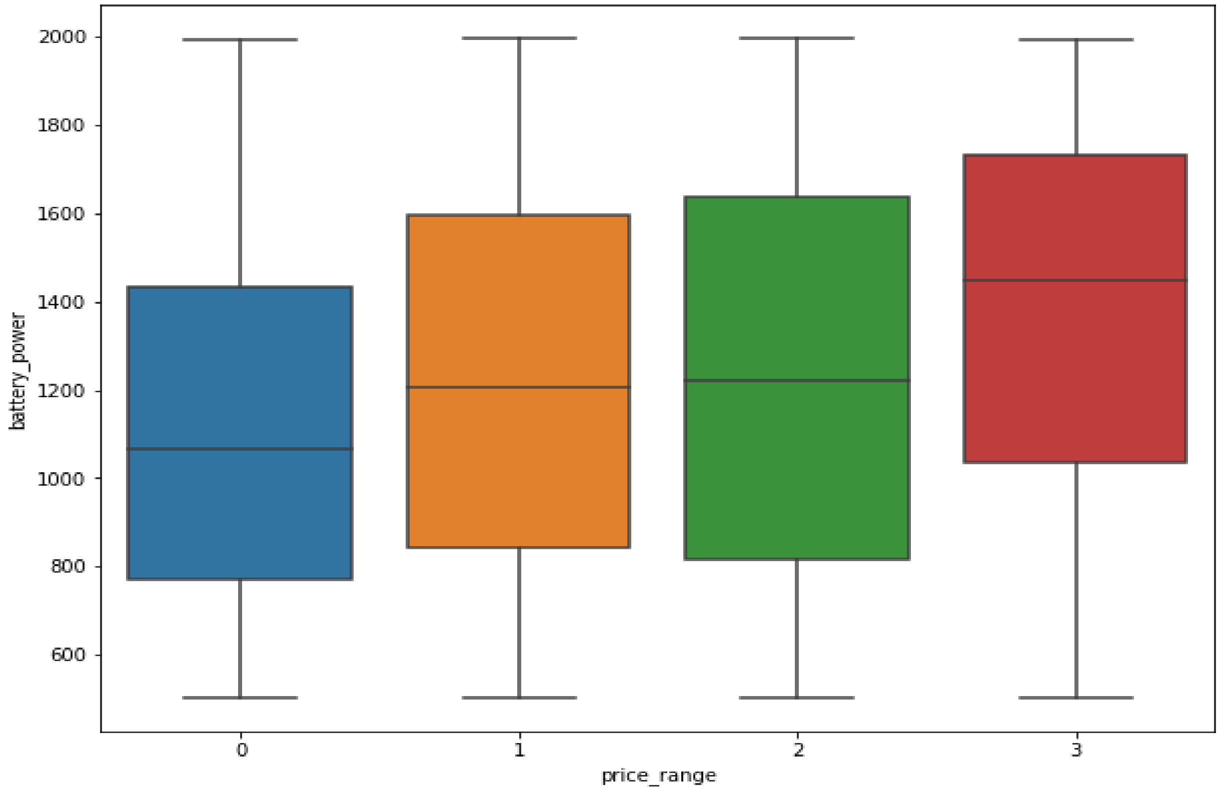


Exploratory Data Analysis

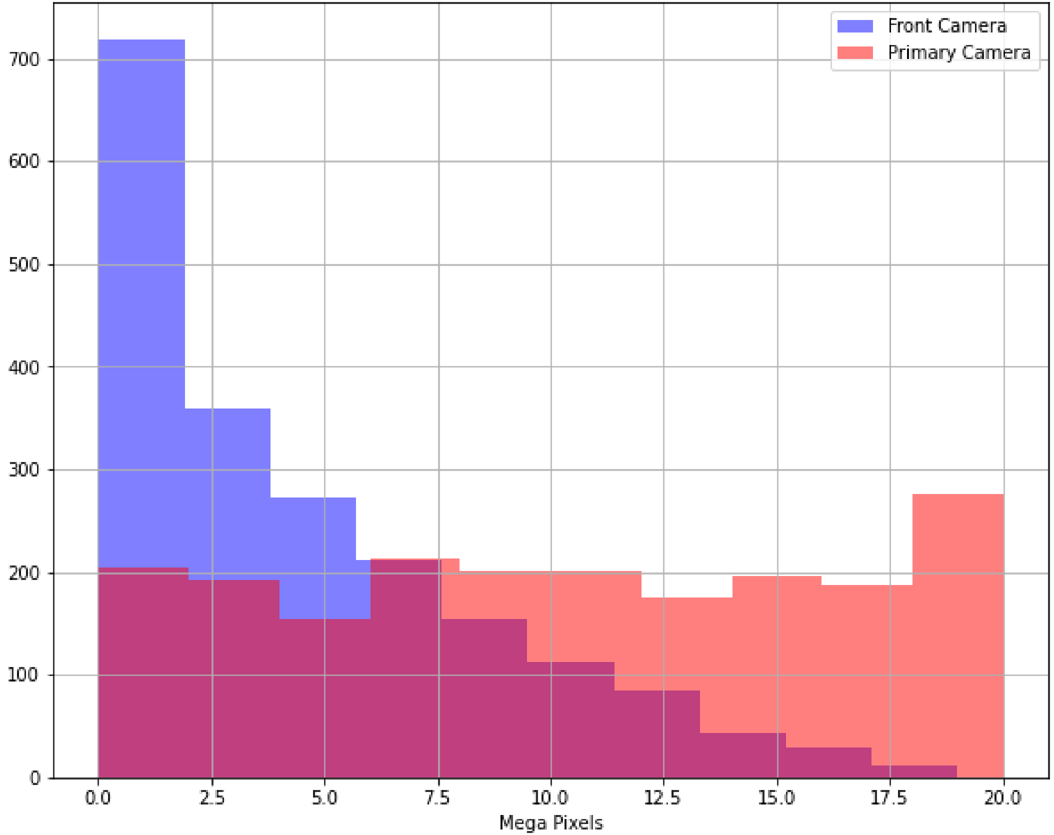




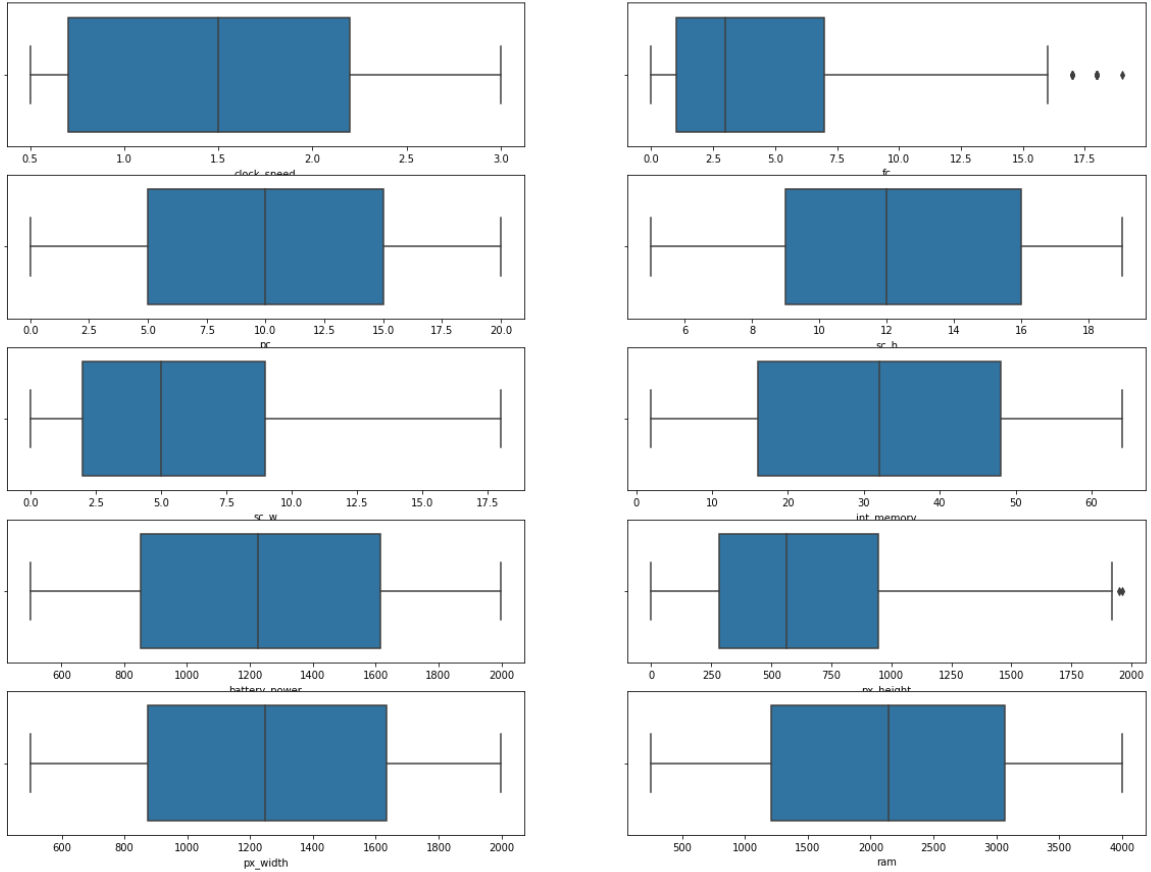
Exploratory Data Analysis



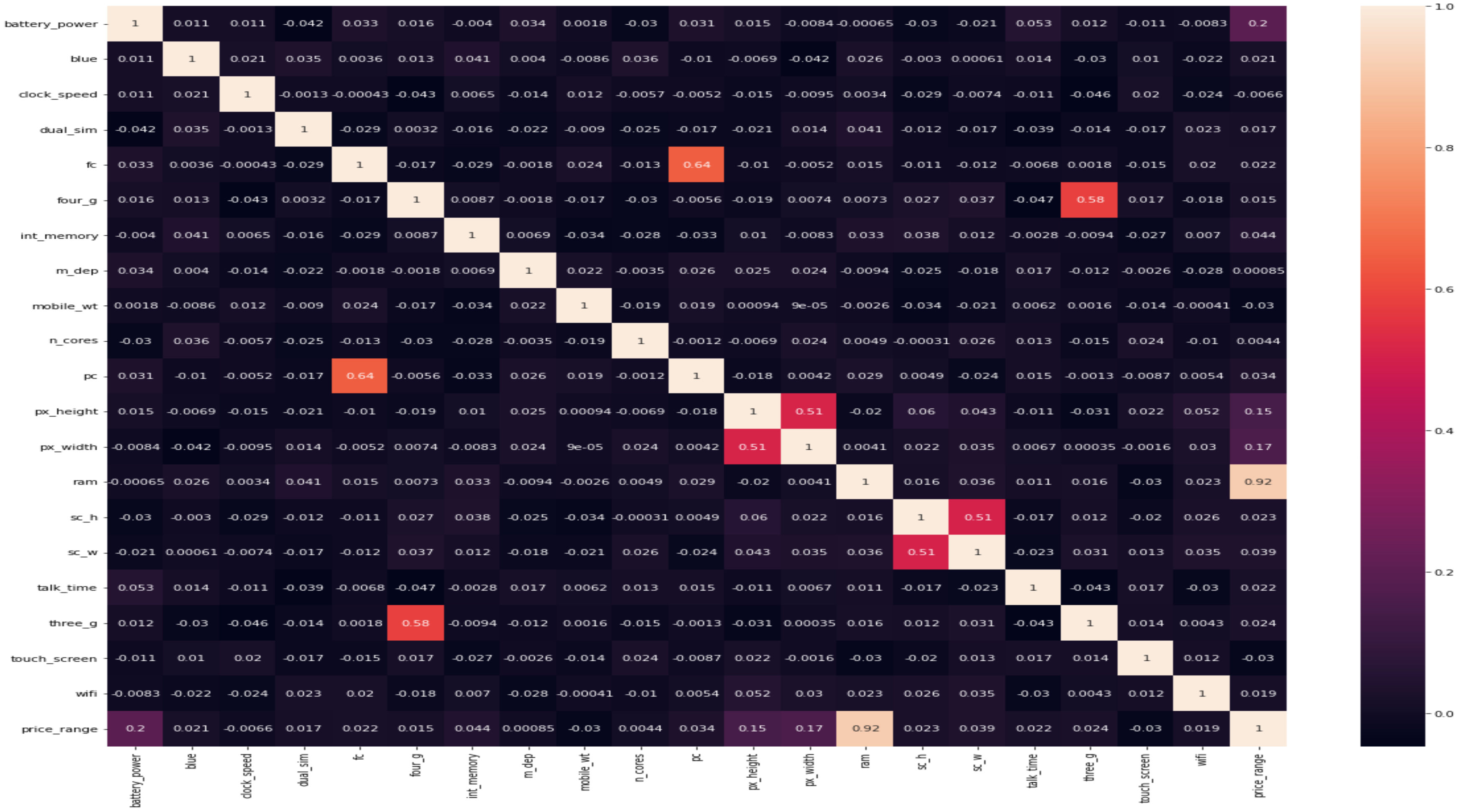
Exploratory Data Analysis



Outliers detection and treatment



Correlation Heatmap



Standardization

Standardization is an important technique that is mostly performed as a pre-processing step before many Machine Learning models, to standardize the range of features of input data set.

Satandardization

```
[68] scaler = StandardScaler()
     X_train = scaler.fit_transform(X_train)
     X_test = scaler.fit_transform(X_test)
```

0s

Machine Learning Modelling

1. K Nearest Neighbours

- K-Nearest Neighbor is one of the simplest Machine Learning algorithms based on Supervised Learning technique.
- K-NN algorithm assumes the similarity between the new case/data and available cases and put the new case into the category that is most similar to the available categories.
- K-NN algorithm stores all the available data and classifies a new data point based on the similarity. This means when new data appears then it can be easily classified into a well suite category by using K- NN algorithm.
- K-NN algorithm can be used for Regression as well as for Classification but mostly it is used for the Classification problems.
- K-NN is a **non-parametric algorithm**, which means it does not make any assumption on underlying data.

1.K Nearest Neighbors

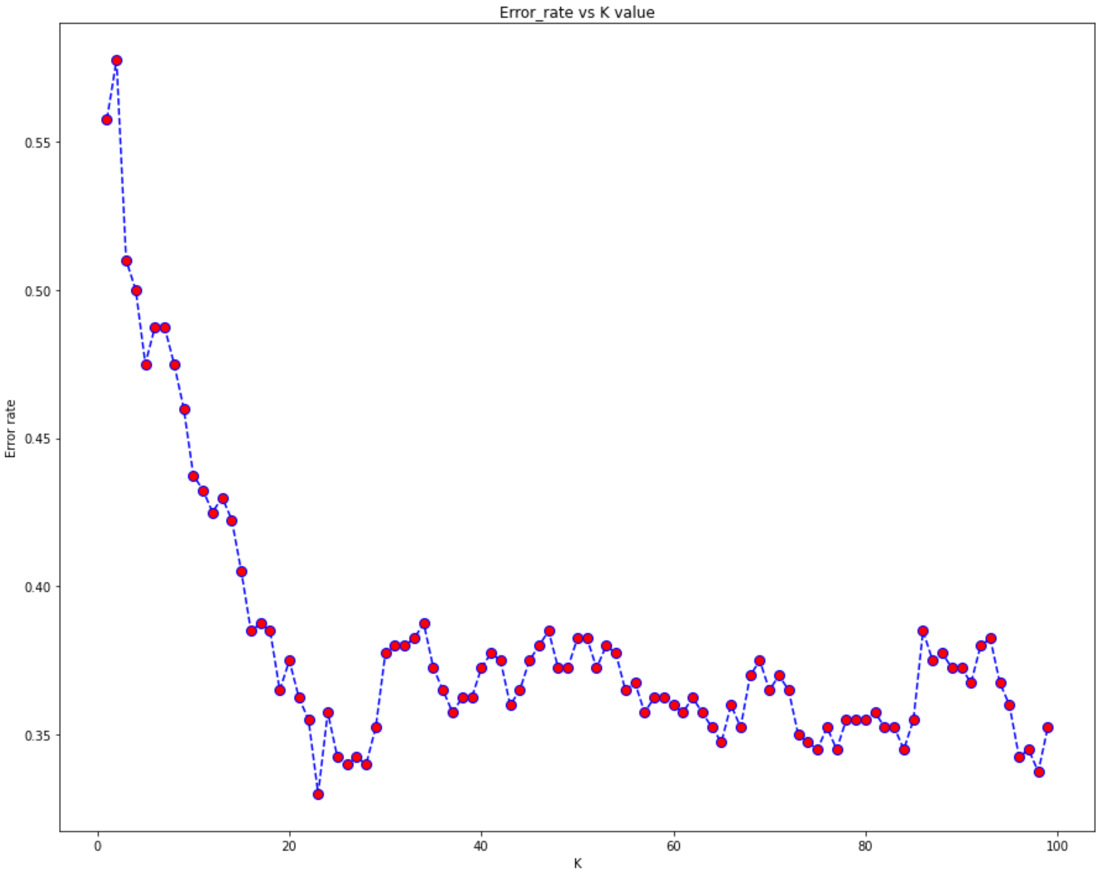
```
from sklearn.neighbors import KNeighborsClassifier
knn = KNeighborsClassifier(n_neighbors=10)
knn.fit(X_train,y_train)
```

```
KNeighborsClassifier(n_neighbors=10)
```

```
knn.score(X_test,y_test)
```

```
0.5625
```

Elbow method for least error rate



Machine Learning Modelling

AI

2. Logistic Regression

- Logistic regression is one of the most popular Machine Learning algorithms, which comes under the Supervised Learning technique. It is used for predicting the categorical dependent variable using a given set of independent variables.
- Logistic regression predicts the output of a categorical dependent variable. Therefore the outcome must be a categorical or discrete value. It can be either Yes or No, 0 or 1, true or False, etc. but instead of giving the exact value as 0 and 1, **it gives the probabilistic values which lie between 0 and 1**.
- Logistic Regression is much similar to the Linear Regression except that how they are used. Linear Regression is used for solving Regression problems, whereas **Logistic regression is used for solving the classification problems**.
- In Logistic regression, instead of fitting a regression line, we fit an "S" shaped logistic function, which predicts two maximum values (0 or 1).

```
[ ] from sklearn.linear_model import LogisticRegression
    logmodel = LogisticRegression()
    logmodel.fit(X_train, y_train)

    LogisticRegression()

[ ] logmodel.score(X_test,y_test)

    0.9125

[ ] y_pred_train2 = logmodel.predict(X_train)
    y_pred_test2 = logmodel.predict(X_test)
```

Machine Learning Modelling

3.XGBoost Classifier

- XGBoost is a decision-tree-based ensemble Machine Learning algorithm that uses a gradient boosting framework. In prediction problems involving unstructured data (images, text, etc.) artificial neural networks tend to outperform all other algorithms or frameworks.

XGBoost classifier

```
[ ]  from xgboost import XGBClassifier
     xgbmodel = XGBClassifier(random_state=0)
     xgbmodel.fit(X_train,y_train)

     XGBClassifier(objective='multi:softprob')
```

```
[ ]  y_pred_train = xgbmodel.predict(X_train)
     y_pred_test = xgbmodel.predict(X_test)
```

```
[ ]  xgbmodel.score(X_test,y_test)

     0.8925
```

Machine Learning Modelling

Hyper-parameter Tuning

Hperparameter tuning for xgboost model

```
[ ]  n_estimators = list(np.arange(5,20,2,dtype='int64'))    #Number of Trees
     max_depth = list(np.arange(10,25,1,dtype='int64'))      #Max depth of trees
     learning_rate = list(np.arange(0.03,0.20,0.01))         #Learning rate
     gamma = list(np.arange(10,20,1,dtype='int64'))          #gamma
     subsample = [0.3,0.4,0.5,0.6]                           #subsamples
```

```
[ ]  #Hyperparameters assigning
     param_dict = {'n_estimators' : n_estimators,
                   'max_depth' : max_depth,
                   'learning_rate' : learning_rate,
                   'gamma':gamma,
                   'subsample':subsample}
```

```
[ ]  #Randomized Search CV
     xgb_randomized = RandomizedSearchCV(estimator=xgbmodel, param_distributions=param_dict, cv=5,scoring = 'accuracy', random_state = 0)
     xgb_randomized.fit(X_train,y_train)
```

```
RandomizedSearchCV(cv=5, estimator=XGBClassifier(objective='multi:softprob'),
                   param_distributions={'gamma': [10, 11, 12, 13, 14, 15, 16,
                                                  17, 18, 19],
                                        'learning_rate': [0.03, 0.04, 0.05,
                                                          0.06000000000000005,
```

Machine Learning Modelling

Hyper-parameter Tuning

```
[ ] xgb_randomized.best_params_
```

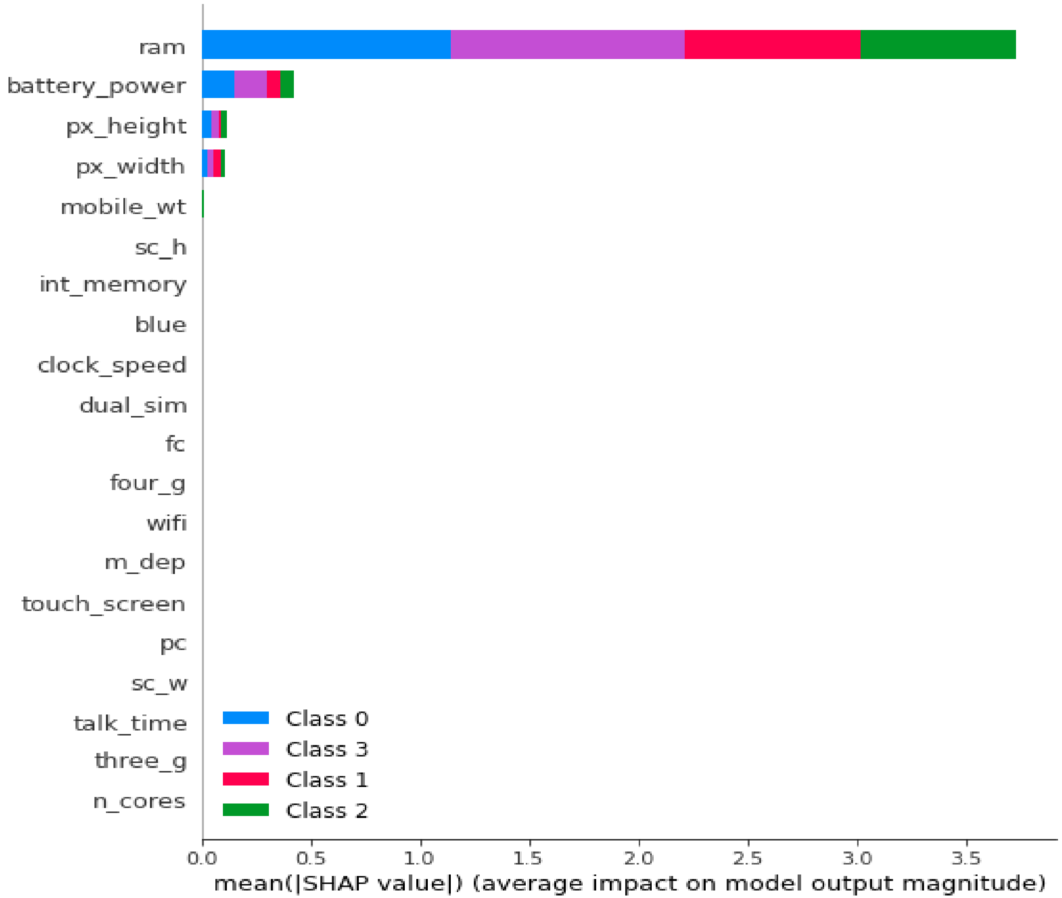
```
{'gamma': 12,
 'learning_rate': 0.13,
 'max_depth': 13,
 'n_estimators': 17,
 'subsample': 0.6}
```

```
[ ] xgb_final_model=xgb_randomized.best_estimator_
```

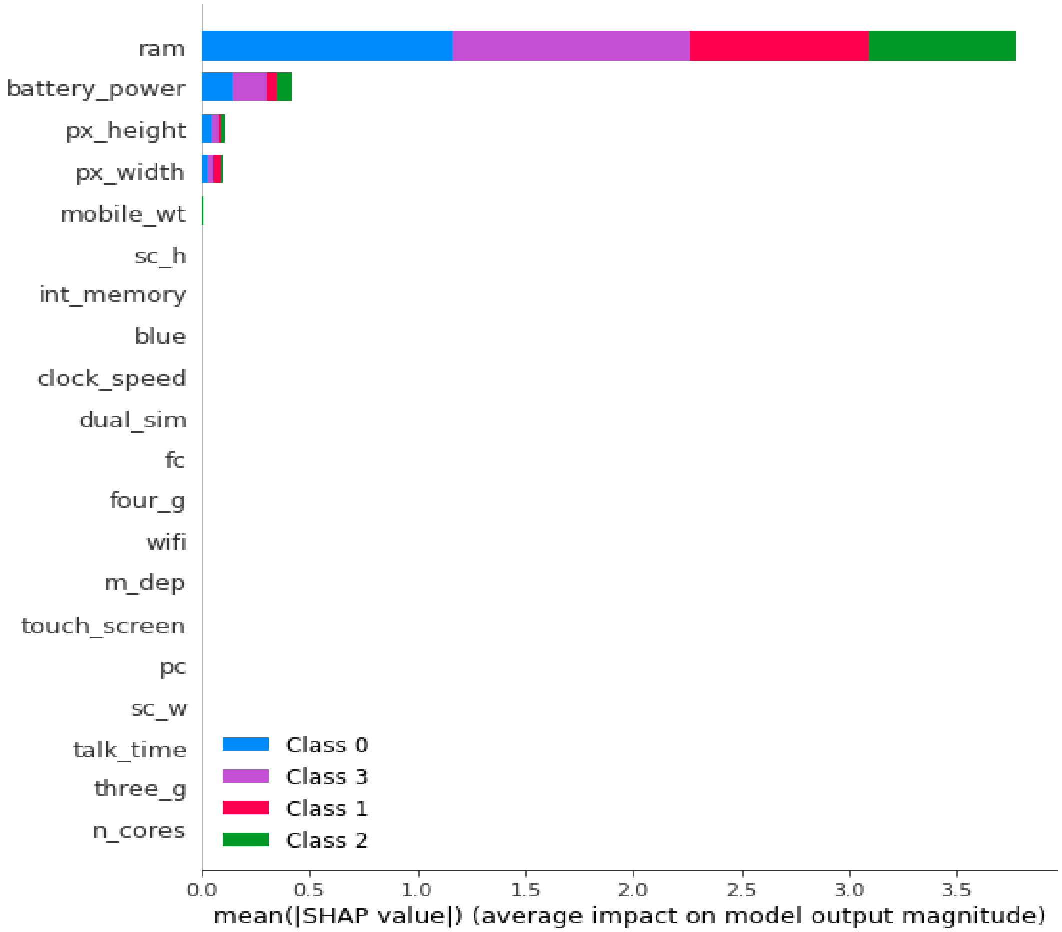
```
[ ] # predicting on both train and test

    y_pred_train3=xgb_final_model.predict(X_train)
    y_pred_test3=xgb_final_model.predict(X_test)
```

Model Explainability(Training Set)



Model Explainability(Test Set)



Conclusion

1. The 'price range' of the given dataset has equal distribution of the total number of phones in each of the price range with 500 nos.
2. It isobserved that 76.2 percent are 3g supported and 27.8 percent are not supp orted.
3. It is observed that 52.1 percent are 4g supported and 47.9 percent are not supported.
4. There is only few number of outlier in 'fc' column in feature engineering and we can neglect it as it has negligible amount.
5. During Multivariate analysis, in correlation heatmap, we get to see that 'ram is highly correlated with 'price range' thus inferring that 'ram' has high impact on p rice prediction.
6. In K nearest Neighbors classification model, we have got the knn score as 56.25%, accuracy score of 67% for training set and 65% for test set.

Conclusion

7. During 'elbow method' we have got the insight that the optimum value of k is 22 with least error rate.
8. In Logistic Regression Model, we have got the log score as 91%.
accuracy score of 98% for training set and 91% for test set.
9. In XGBoost model the score was 89% before hyper parameter tuning.
10. RandomizedSearchCV is used for hyperparameter tuning in XGBoost classifier and the accuracy obtained after hyper parameter tuning was 86% for training set and 80% for test set.
11. Finally, in the model explaininabilty we have used shap and we got the insight that 'ram', 'battery power', and phone dimensions are the features which is deciding as key factor for the price range prediction.

References

1). https://pandas.pydata.org/

2). https://matplotlib.org/

3). https://seaborn.pydata.org/

4). Geek for geeks

5). Analytics Vindhya

6).XGBoost Documentation

Thank You