Mobile Price Range Prediction

Rudrajit Bhattacharyya
Cohort Zanskar
AlmaBetter

Abstract:

Mobile phones come in all sorts of prices, features, specifications etc. Price estimation and prediction is an important part of consumer strategy. Deciding on the correct price of a product is very important for its market success. In the competitive mobile phone market companies want to understand sales data of mobile phones and factors which drive the prices. The objective of this work is to find out some relation between features of a mobile phone and its selling price. We didn't need to predict the actual prices but a price range indicating how high the price is. We will be using some classification models to accurately classify the data in correct price ranges. This kind of prediction will help businesses estimate the price of mobile phones accurately to give good competition to other manufacturers. We will be using a confusion matrix to calculate the different metrics and compare the models with each other to see which classification model performs better.

Problem Statement:

In the competitive mobile phone market companies want to understand sales data of mobile phones and factors which drive the prices. Mobile phones come in all sorts of prices, features, specifications etc and estimating the price of mobile phones is an important part of consumer strategy.

The objective of this project is to find out some relation between features of a mobile phone and its selling price. In this problem, we do not have to predict the actual prices but a price range indicating how high the price is. The dataset contains 2000 records of mobile phone information with 20 features which is a mix of categorical features and numerical features.

Data Description:

The dataset contains 2000 records and 21 features which consists of:

- **Battery_power:** Total energy a battery can store in one time measured in mAh.
- **Blue:** Has bluetooth or not
- **Clock_speed:** Speed at which the microprocessor executes instructions
- **Dual_sim:** Has dual sim support or not
- **Fc:** Front camera megapixels
- **Four_g:** Has 4G access 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 megapixels
- **Px_height:** Pixel resolution height
- **Px_width:** Pixel resolution width
- **Ram:** Random Access Memory in megabytes
- **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 talking over phone
- **Three_g:** Has 3G access or not
- **Touch_screen:** Has touch screen or not
- **Wifi:** Has wifi or not
- **Price_range:** This is the target variable with values of **0(low cost), 1(medium cost), 2(high cost) and 3(very high cost)**.

Introduction:

Mobile phone, cellular phone, sometimes shortened to simply mobile, cell or just phone is a portable telephone that can make and receive calls over a radio frequency link while the user is moving within a telephone service area. Mobile phones, particularly the smartphones that have become our inseparable companions today, are relatively new.

The very first mobile phones were not really mobile phones at all. They were two-way radios that allowed people like taxi drivers and the emergency services to communicate. Instead of relying on base stations with separate cells, the first mobile phone networks involved one very powerful base station covering a much wider area.

Mobile phones are the best selling electronic devices as people keep updating their cell phones whenever they find new features in a new device. Thousands of mobiles are sold daily, in such a situation it is a very difficult task for someone who is planning to set up their own mobile phone business to decide what the price of the mobile should be. Price estimation and prediction is an important part of consumer strategy as a new product that has to be launched, must have the correct price so that the consumers find it appropriate to buy the product. During the purchase of mobile phones, people fail to make correct decisions due to the non-availability of necessary resources to cross validate the price. To address this issue we developed different classification models using the data related to different features of a mobile phone. The developed model is then used to predict the price range of the new mobile phones.

Classification is a task that requires the use of machine learning algorithms that learn how to assign a class label to examples from the problem domain. As we have more than two classes in our target variable, this is a case of multi-class classification. Unlike binary classification, multi-class classification does not have the notion of normal and abnormal outcomes. Instead, examples are classified as belonging to one among a range of known classes. Six classification models have been tried and tested to compare and find out which algorithm performs better in predicting the price ranges. This kind of prediction will help businesses estimate the price of mobile phones accurately to give good competition to other manufacturers and place their own product in the correct segment.

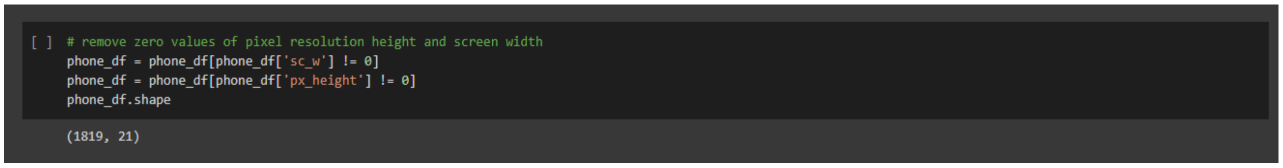
Exploratory Data Analysis:

- **Data Cleaning**:

  Data cleaning is one of the most time consuming aspects of data analysis. Formatting issues, missing values, duplicated rows, spelling mistakes, and so on could all be present in a dataset. These difficulties make data analysis difficult, resulting in inappropriate results. Missing or duplicate data may exist in a dataset for a number of different reasons. Sometimes, missing or duplicate data is introduced as we perform cleaning and transformation tasks such as combining data, reindexing data, and reshaping data. Other times, it exists in the original dataset for reasons such as user input error or data storage or conversion issues.

We need to fill in the missing values because most of the machine learning models that we want to use will provide an error if we pass NaN values into it. The easiest way is to just fill them up with 0, but this can reduce our model accuracy significantly. There are many methods available for filling up missing values but for choosing the best method, we need to understand the type of missing value and its significance.

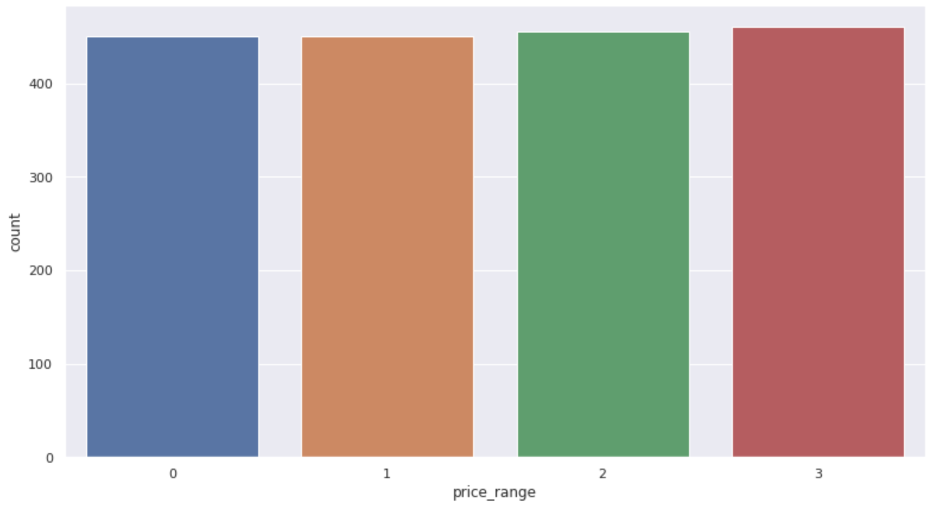
The dataset we have in our hand is almost cleaned with no null values present or duplicate records found. Two features had some zero values which we had to remove as these values cannot be zero in real life. Pixel resolution and screen width has to be some numerical value above zero.



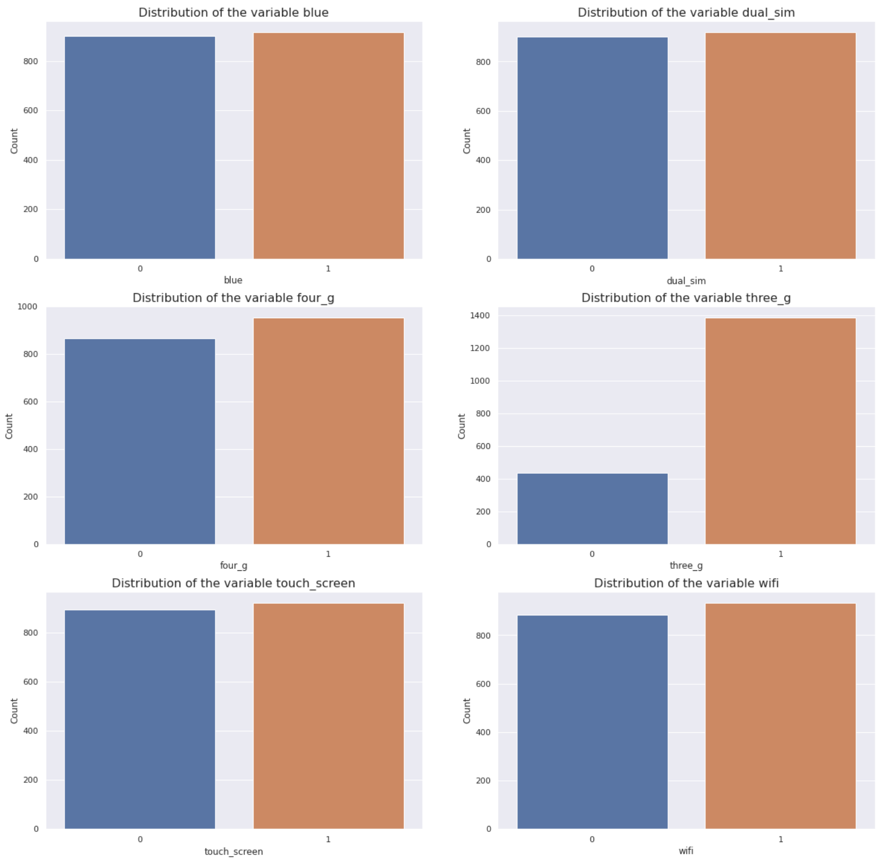
```
# remove zero values of pixel resolution height and screen width
phone_df = phone_df[phone_df['sc_w'] != 0]
phone_df = phone_df[phone_df['px_height'] != 0]
phone_df.shape
```

```
(1819, 21)
```

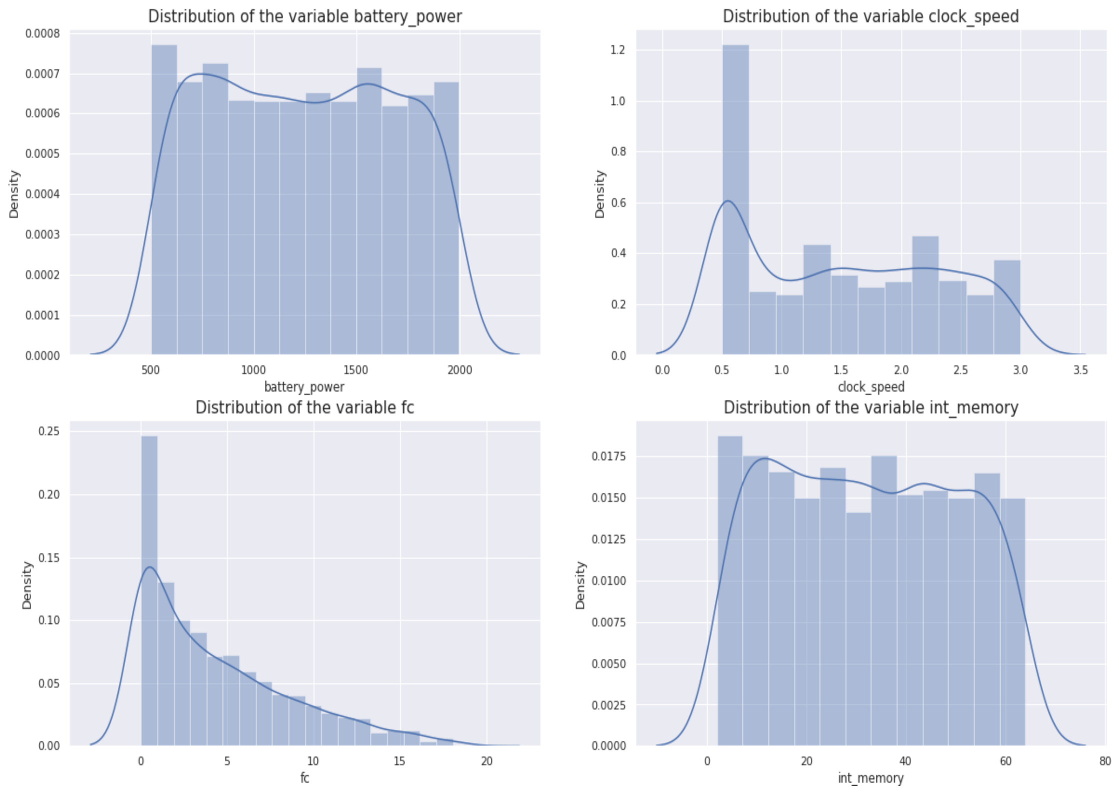
- **Data Visualization/Exploration**:
  - The distribution of the target variable shows us that the classes are almost balanced as the difference between the classes is minimal. Thus, we know that there's no class imbalance problem here. All the classes have around 450 records.

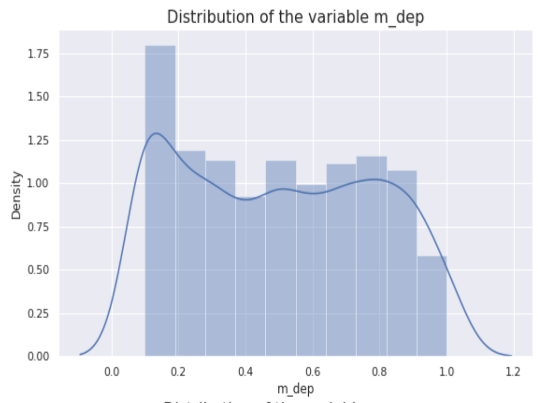


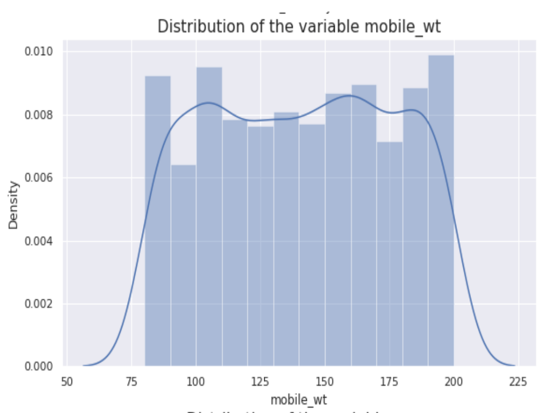
- The distribution of the categorical features are almost similar to each other except the feature ‘three_g’ which contains very few mobile phones which do not have 3G access. We can infer that almost all phones have 3G access if not 4G.

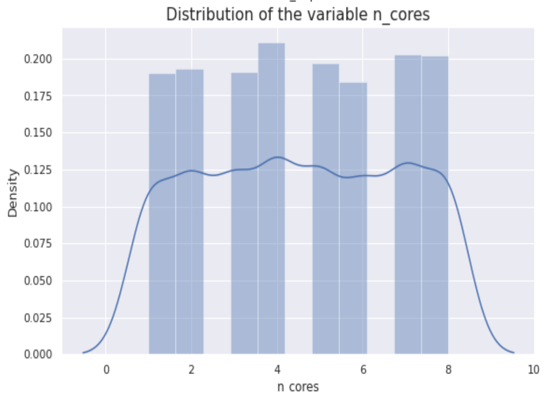


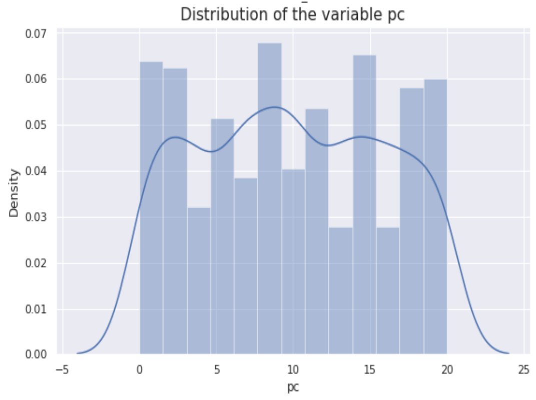
- Most of the numerical features follow an uniform distribution except a few features like fc, px_height, and sc_w follow a right skewed distribution.

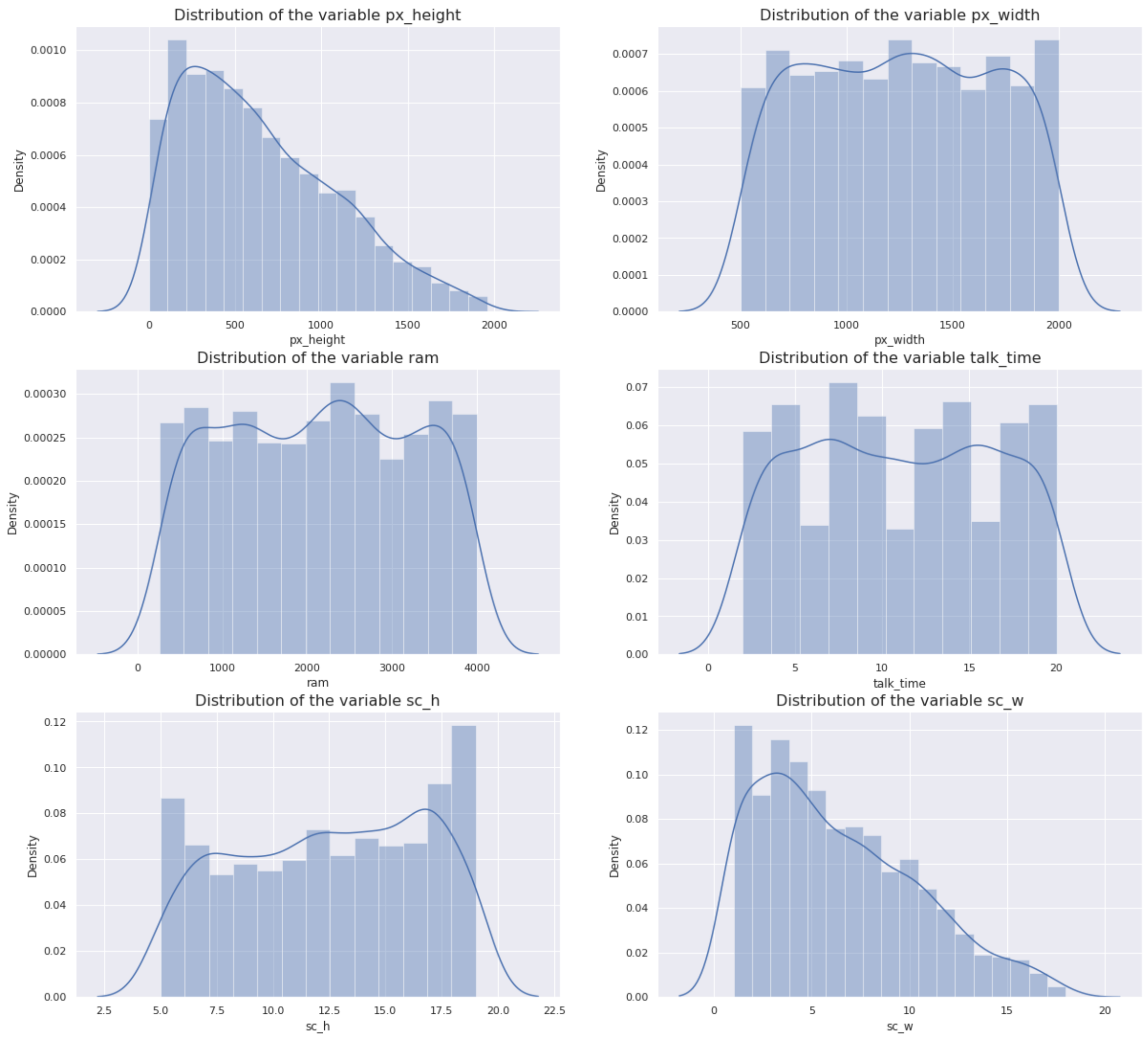



Distribution of the variable m_dep
Density
m_dep

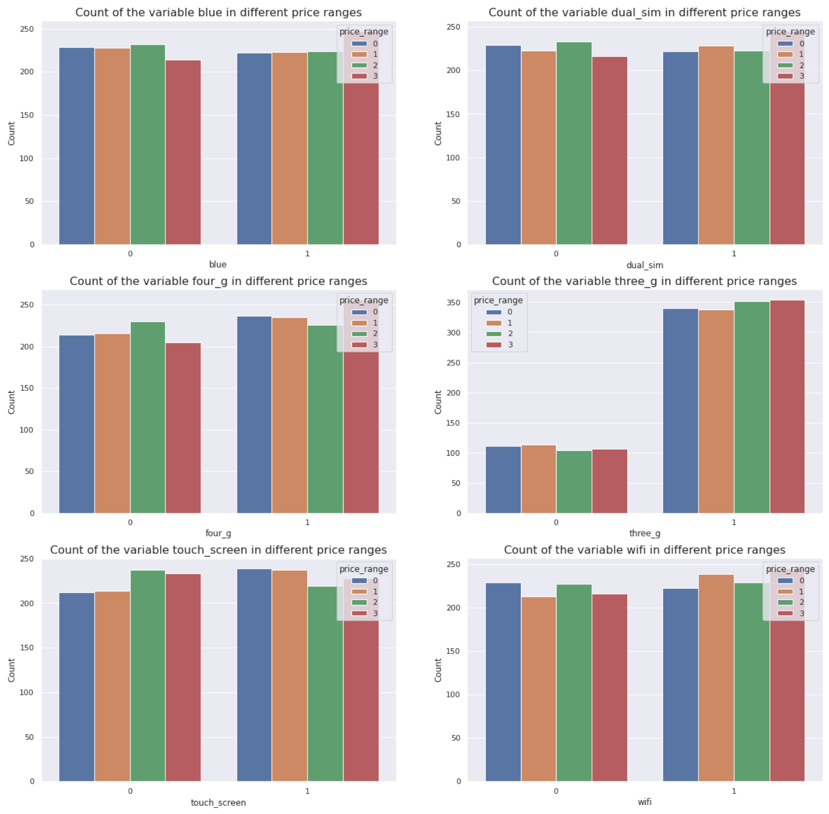

Distribution of the variable mobile_wt
Density
mobile_wt


Distribution of the variable n_cores
Density
n_cores

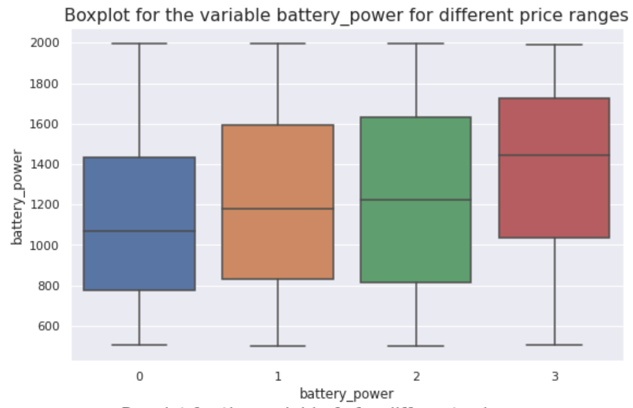

Distribution of the variable pc
Density
pc

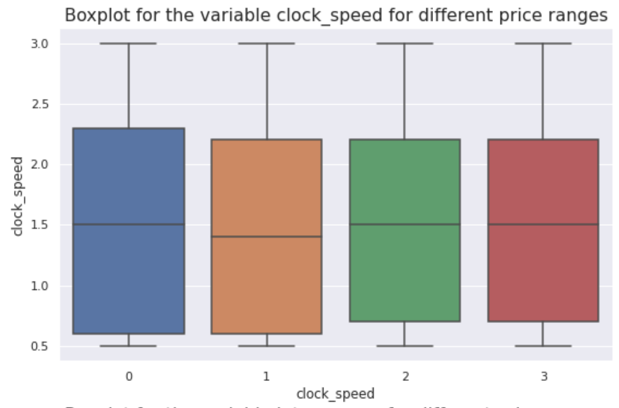


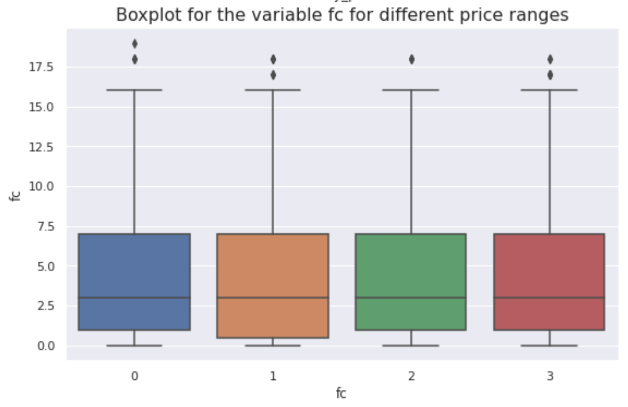
- The distribution of categorical features across different price ranges stays the same. There is a slight increase in the count of mobile phones in the very high cost category for each categorical feature. Thus we can infer that the more we pay, the more choices we can get.

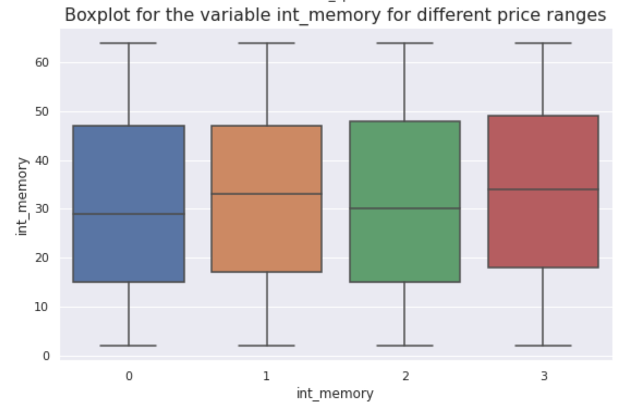


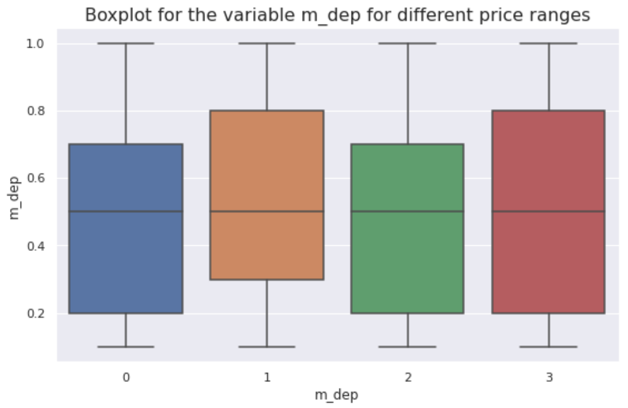
- The distribution of the numerical features across different price ranges shows us that only the features RAM, battery power, px_height and px_width increase with an increase in price. These features are the most influential in determining the price ranges.

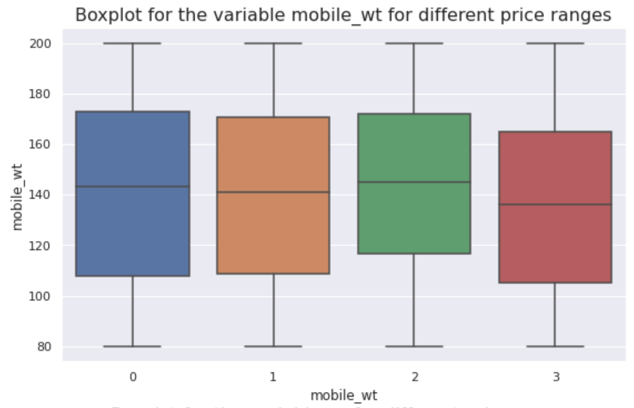

Boxplot for the variable battery_power for different price ranges
battery_power
2000
1800
1600
1400
1200
1000
800
600
0
1
2
3
battery_power

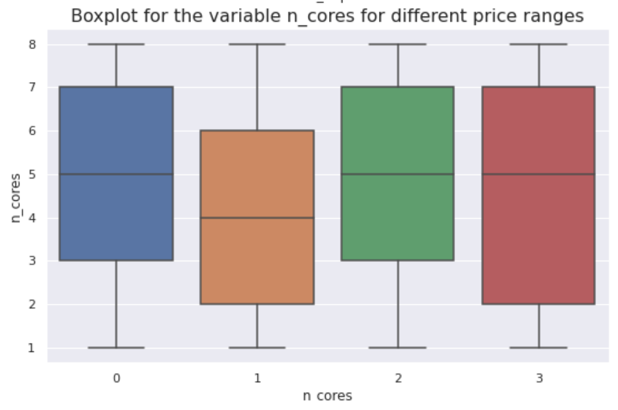

Boxplot for the variable clock_speed for different price ranges
clock_speed
3.0
2.5
2.0
1.5
1.0
0.5
0
1
2
3
clock_speed

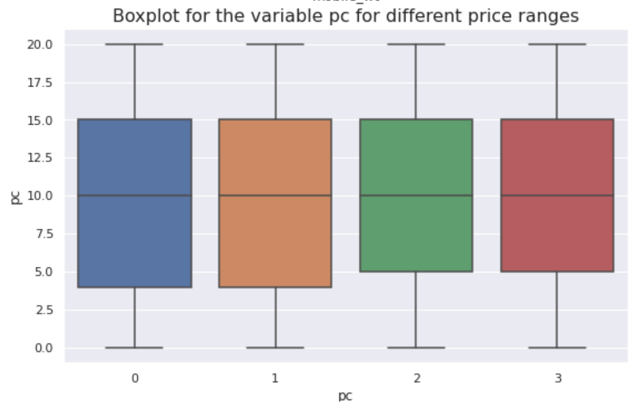

Boxplot for the variable fc for different price ranges
fc
17.5
15.0
12.5
10.0
7.5
5.0
2.5
0.0
0
1
2
3
fc

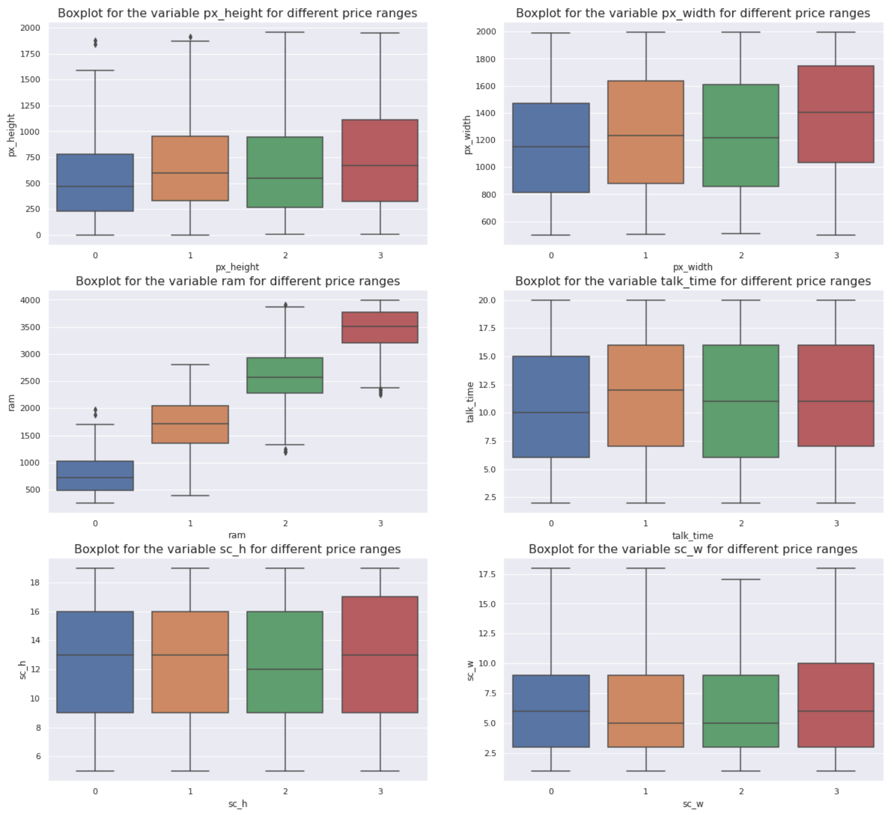

Boxplot for the variable int_memory for different price ranges
int_memory
60
50
40
30
20
10
0
0
1
2
3
int_memory


Boxplot for the variable m_dep for different price ranges
m_dep
1.0
0.8
0.6
0.4
0.2
0
1
2
3
m_dep

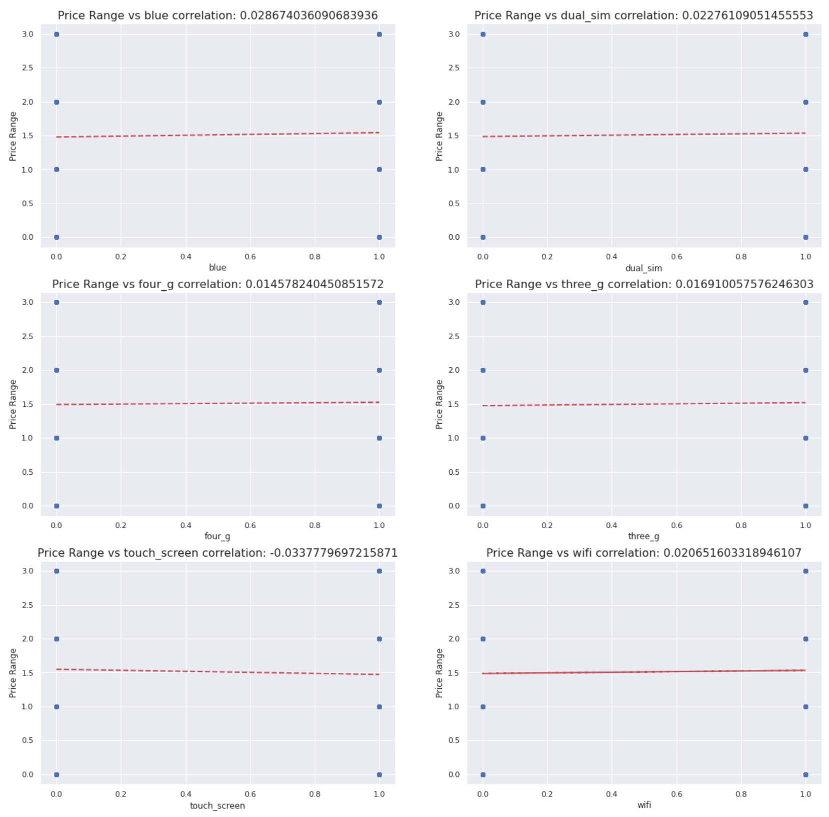

Boxplot for the variable mobile_wt for different price ranges
mobile_wt
200
180
160
140
120
100
80
0
1
2
3
mobile_wt


Boxplot for the variable n_cores for different price ranges
n_cores
8
7
6
5
4
3
2
1
0
1
2
3
n_cores

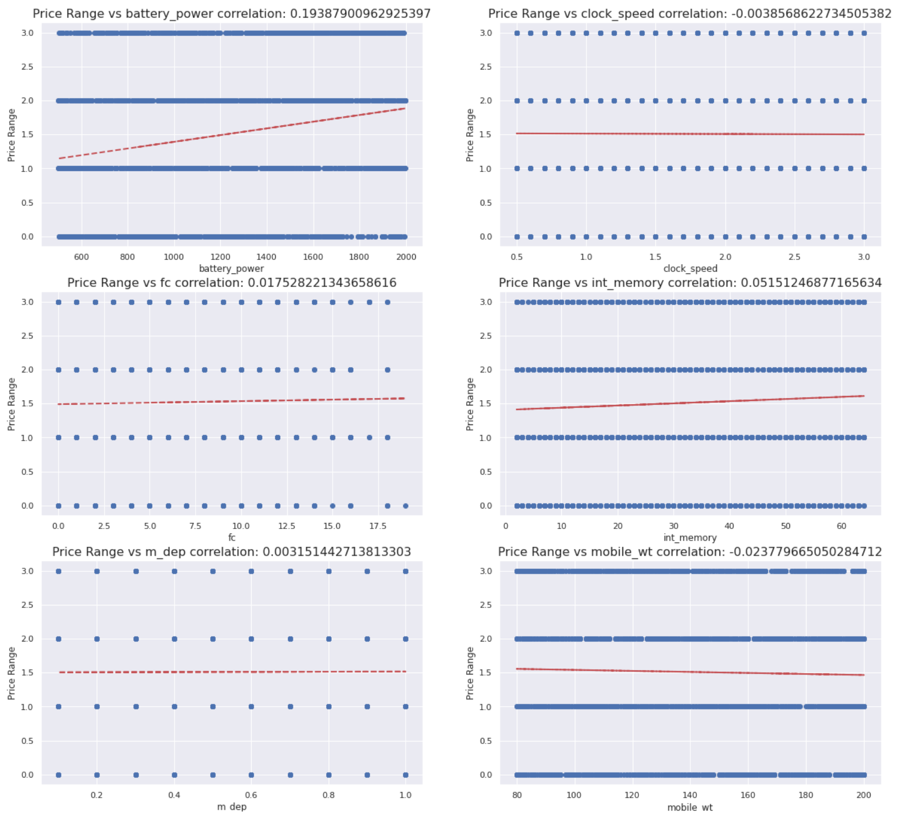

Boxplot for the variable pc for different price ranges
pc
20.0
17.5
15.0
12.5
10.0
7.5
5.0
2.5
0.0
0
1
2
3
pc

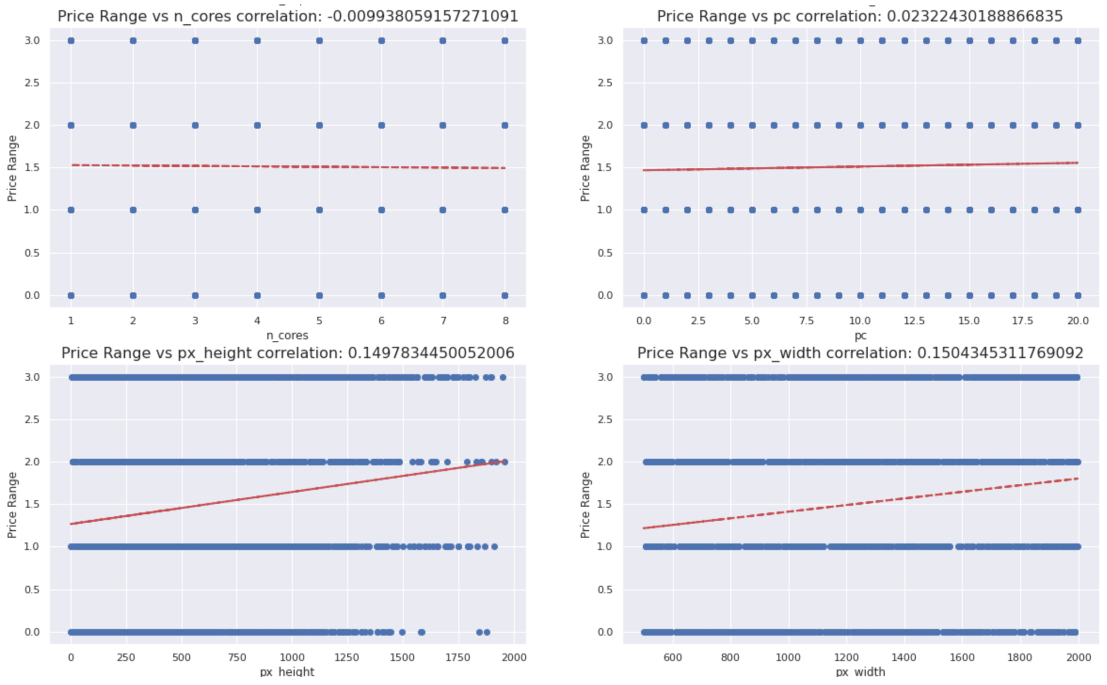


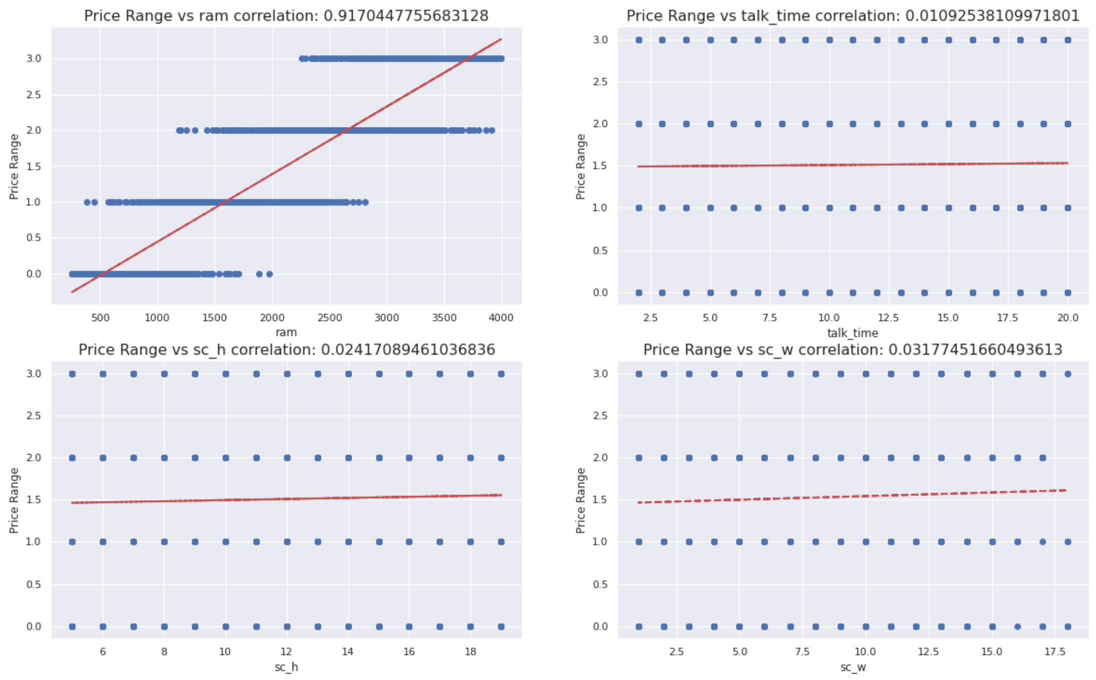
- No categorical feature has a strong correlation with the target variable.



- RAM has the strongest correlation with the target variable followed by battery power, px_height and px_width. Rest of the numerical features have a very low correlation with the target variable.


Price Range vs battery_power correlation: 0.19387900962925397
Price Range
battery_power
Price Range vs clock_speed correlation: -0.0038568622734505382
Price Range
clock_speed
Price Range vs fc correlation: 0.017528221343658616
Price Range
fc
Price Range vs int_memory correlation: 0.05151246877165634
Price Range
int_memory
Price Range vs m_dep correlation: 0.003151442713813303
Price Range
m_dep
Price Range vs mobile_wt correlation: -0.023779665050284712
Price Range
mobile_wt


Price Range vs n_cores correlation: -0.009938059157271091
Price Range
n_cores
Price Range vs pc correlation: 0.02322430188866835
Price Range
pc
Price Range vs px_height correlation: 0.1497834450052006
Price Range
px_height
Price Range vs px_width correlation: 0.1504345311769092
Price Range
px_width

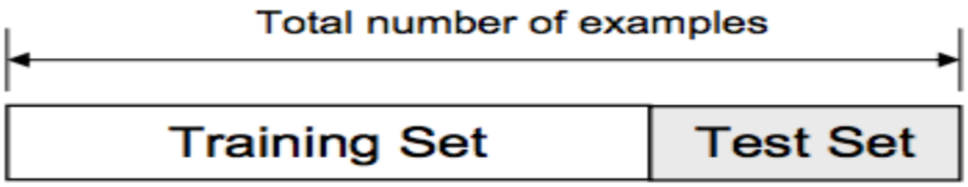


Data Preparation:

Before diving into modeling the data and predicting the mobile price range, we will be doing the following steps:

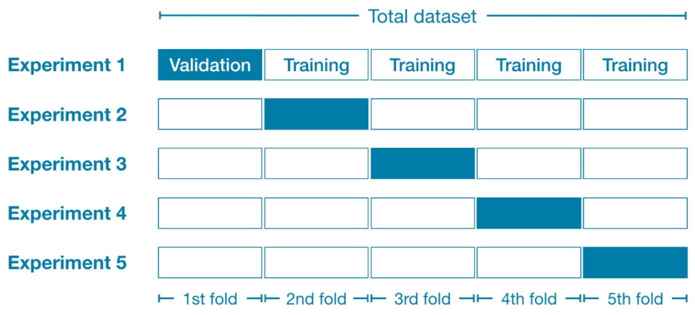
- **Train-Test Split**:

  The train-test split procedure is used to estimate the performance of machine learning algorithms when they are used to make predictions on data not used to train the model. It is a fast and easy procedure to perform, the results of which will allow us to compare the performance of machine learning algorithms for our prediction. Although it is simple to use and interpret, there are times when this procedure should not be used, such as when we have a small dataset. The train-test split procedure is appropriate when we have a large dataset, a costly model to train, or require a good estimate of model performance quickly.



The train-test procedure is not appropriate when the dataset is small. The reason is that when the dataset is split into train and test sets, there will not be enough data in the train set to train the model well and there will not be enough data in the test set to effectively evaluate the model performance. The estimated performance could be overly optimistic or overly pessimistic. A suitable alternate model evaluation procedure would be the k-fold cross validation.

Cross validation is a resampling procedure used to evaluate machine learning models on a limited data sample. The procedure has a single parameter called k that refers to the number of groups that a given data sample is to be split into. As such the procedure is often called k-fold cross validation. When a specific value for k is chosen, such as k=10 it becomes 10-fold cross validation. Cross validation is primarily used in machine learning to estimate the skill of a machine learning model on unseen data i.e, to use a limited sample in order to estimate how the model is expected to perform in general when used to make predictions on data not used to train the model. It is a popular method because it is simple to understand and generally results in a less biased or less optimistic estimate of the model than a simple train-test split. We have used cross validation to evaluate model performance for all other models except logistic regression.



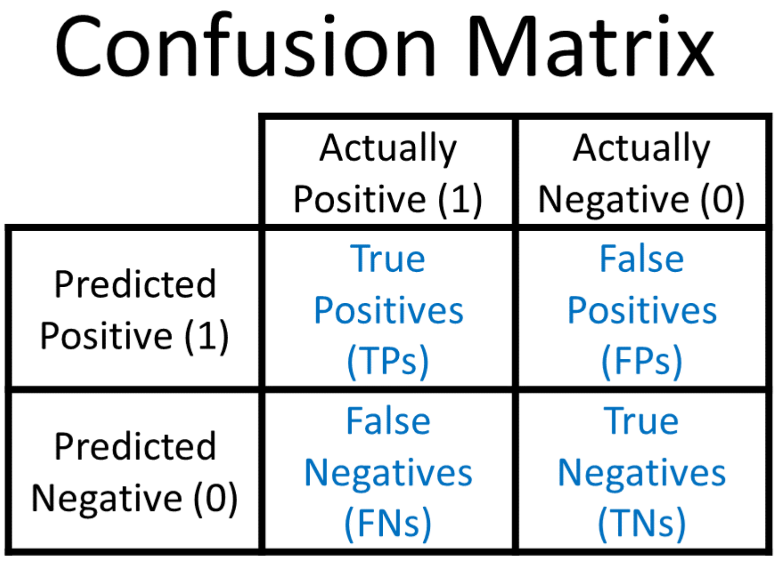
- **Feature Scaling/Standardization**:

Some machine learning algorithms are sensitive to feature scaling while others are not. Machine learning algorithms like linear regression, logistic regression, neural networks etc, that use gradient descent as an optimization technique require data to be scaled. The presence of feature value X in the formula will affect the step size of the gradient descent. The difference in ranges of features will cause different step sizes for each feature. To ensure that the gradient descent moves smoothly towards the minima and that the steps for gradient descent are updated at the same rate for all the features, we need to scale the data before feeding it to the model. We have chosen standardization as a scaling technique where the values are centered around the mean with a unit standard deviation.

$$x_{\text{stand}} = \frac{x - \text{mean}(x)}{\text{standard deviation }(x)}$$

Evaluation Metrics:

Different performance metrics are available for evaluating different machine learning algorithms. The metrics chosen influence how the performance of a machine learning algorithm is measured and compared. The performance for our classification purpose is measured or evaluated by a confusion matrix. The confusion matrix is one of the most intuitive and easiest metrics used for determining the accuracy of a model. It is used for classification problems where the output can be of two or more classes. The confusion matrix is a table with two dimensions “Actual” and “Predicted” and sets of “Classes” in both dimensions. The confusion matrix in itself is not a performance measure as such, but almost all performance metrics are based on the confusion matrix.



Confusion Matrix

	Actually Positive (1)	Actually Negative (0)
Predicted Positive (1)	True Positives (TPs)	False Positives (FPs)
Predicted Negative (0)	False Negatives (FNs)	True Negatives (TNs)

- **Accuracy**:

  The accuracy is measured as the number of correctly predicted classes made by the model over all predicted classes. Accuracy is a good measure when the target variable classes in the data are nearly balanced. A simple equation for finding the accuracy is given below:

$$Accuracy = \frac{TP+TN}{TP+TN+FP+FN}$$

- **Precision:**

  The precision is calculated as the number of correct positive results divided by the number of positive results predicted by the classifier.

$$Precision = \frac{TP}{TP+FP}$$

- **Recall:**

The recall is calculated as the number of correct positive results divided by the number of all samples that should have been identified or classified as positive by the classifier.

$$Recall = \frac{TP}{TP+FN}$$

- **ROC AUC:**

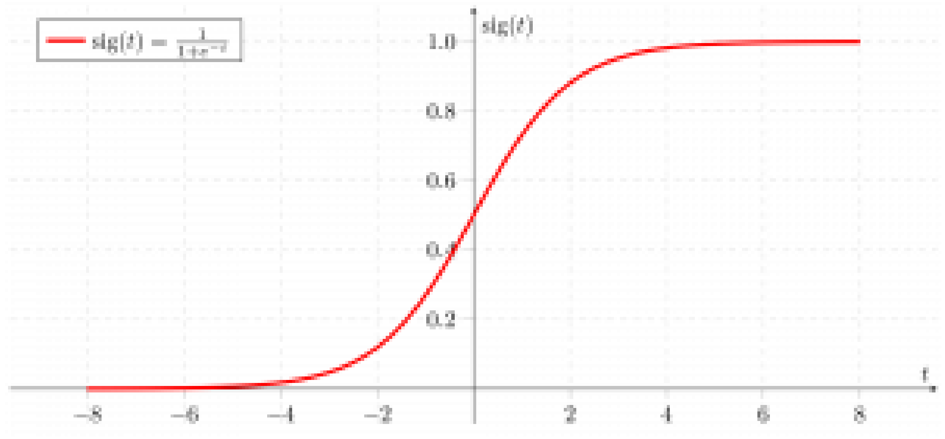
  The ROC curve is a probability curve that plots the true positive rate against false positive rate at various threshold values and essentially separates the signal from the noise. The AUC is the measure of the ability of a classifier to distinguish between the classes and is used as a summary of the ROC curve. The higher the AUC, the better is the performance of the model at distinguishing between the classes.

Data Modeling:

We have used several classification models to predict the mobile price range and compare their performances. Let us see each model in depth:

- **Logistic Regression:**

  Logistic regression is a process of modeling the probability of a discrete outcome given an input variable. The most common logistic regression models a binary outcome, something that can take two values such as true/false, yes/no and so on. The model builds a regression model to predict the probability that a given data entry belongs to the category numbered as 1. Just like linear regression assumes that the data follows a linear function, logistic regression models the data using the sigmoid function.



Logistic regression becomes a classification technique only when a decision threshold is brought into the picture. The setting of the threshold value is a very important aspect of logistic regression and is dependent on the problem itself. Logistic regression does not assume a linear relationship between the dependent variable and the independent variables, but it does assume a linear relationship between the logit of the explanatory variables and the response. It uses maximum likelihood estimation rather than ordinary least squares to estimate the parameters and thus relies on large sample approximations.

The results obtained by implementing logistic regression to predict mobile price range is:

Accuracy	Precision	Recall	ROC AUC
0.9643	0.9645	0.9643	0.9975

- **Random Forest:**

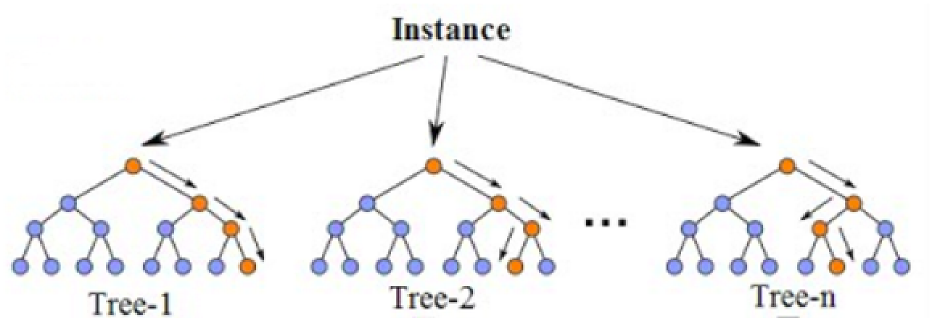
  Random forest is an ensemble of decision trees. Decision trees are great for obtaining non-linear relationships between input features and the target variable.

The inner workings of a decision tree can be thought of as a bunch of if-else conditions.

It starts at the very top with one node which then splits into a left and right node known as decision nodes. These nodes then split into their respective left and right nodes. At the end of the leaf node, the average of the observation that occurs within that area is computed. The most bottom nodes are referred to as leaves or terminal nodes.

Many trees constructed in a certain random way form a Random Forest.

- Each tree is created from a different sample of rows and at each node, a different sample of features is selected for splitting.
- Each of the trees makes its own individual prediction.
- The final output is considered based on Majority Voting for classification.



Random forest is a classifier that contains a number of decision trees on various subsets of the given dataset and takes the average to improve the predictive accuracy of that dataset. Instead of relying on one decision tree, the random forest takes the prediction from each tree and based on the majority votes of predictions, it predicts the final output. The greater number of trees in the forest leads to higher accuracy and prevents the problem of overfitting.

The results obtained by implementing Random Forest with cross validation is:

Accuracy	**Precision**	**Recall**	**ROC AUC**

0.8956	0.8958	0.8956	0.9888

- **Gradient Boosting:**

  Gradient boosting is one of the most powerful techniques for building predictive models. The idea of boosting came out of the idea of whether a weak learner can be modified to become better. Gradient boosting involves three elements:

  - A loss function to be optimized: The loss function depends on the type of problem being solved. For e.g, regression may use a squared error and classification may use logarithmic loss. It must be differentiable.

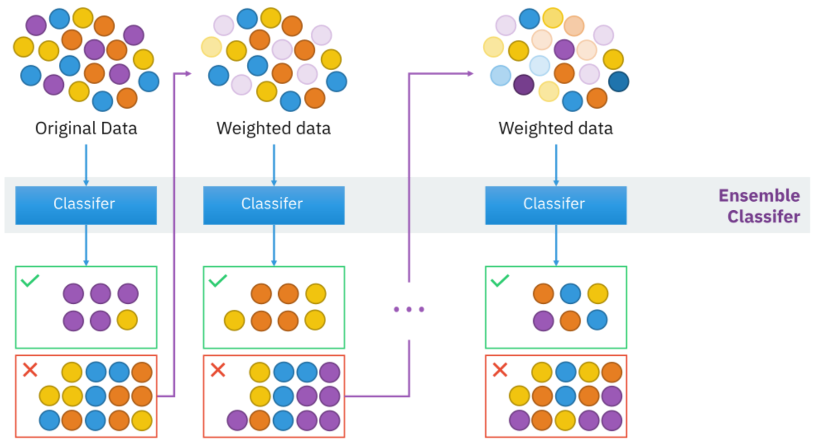
  - A weak learner to make predictions: Decision trees are used as the weak learner in gradient boosting. Specifically regression trees are used that output real values for splits and whose output can be added together, allowing subsequent models outputs to be added and correct the residuals in the predictions. Trees are constructed in a greedy manner, choosing the best split points based on purity scores like Gini or to minimize the loss.

  - An additive model to add weak learners to minimize the loss function: Trees are added one at a time, and existing trees in the model are not changed. A gradient descent procedure is used to minimize the loss when adding trees.

    Traditionally, gradient descent is used to minimize a set of parameters, such as the coefficients in a regression equation or weights in a neural network. After calculating error or loss, the weights are updated to minimize that error.

    Instead of parameters, we have weak learner sub-models or more specifically decision trees. After calculating the loss, to perform gradient descent procedure, we must add a tree to the model that reduces the loss. We do this by parameterizing the tree, then modify the parameters of the tree and move in the right direction by reducing the loss.

The output for the new tree is then added to the output of the existing sequence of trees in an effort to correct or improve the final output of the model.



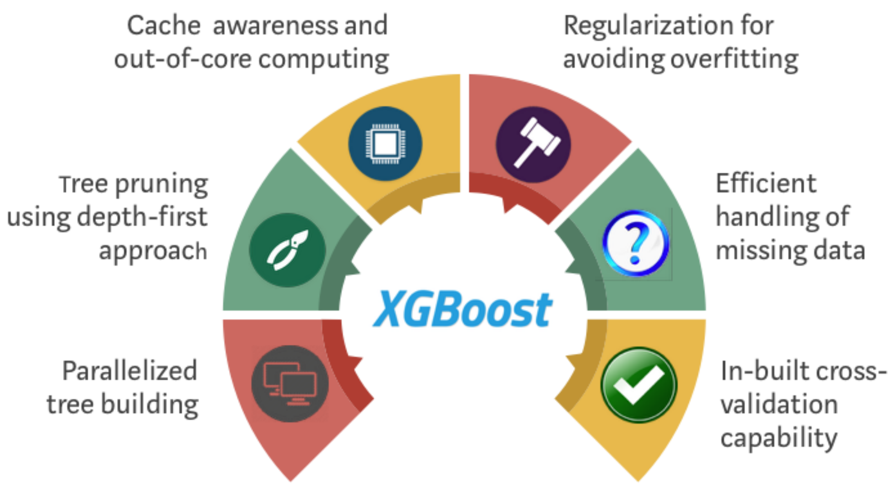
The results obtained by implementing Gradient Boosting with cross validation is:

Accuracy	**Precision**	**Recall**	**ROC AUC**
0.8928	0.8924	0.8928	0.9879

- **XGBoost:**

XGBoost is an implementation of gradient boosted decision trees designed for speed and performance. XGBoost stands for Extreme Gradient Boosting. Boosting is an ensemble modeling technique that attempts to build a strong classifier from the number of weak classifiers. It is done by building a model by using weak models in series. Firstly a model is built from the training data and then the second model is built which tries to correct the errors present in the first model. This procedure is continued and models are added until either the complete training data is predicted correctly or the maximum number of models are added.

In XGBoost, decision trees are created sequentially. Weights play an important role in XGBoost. Weights are assigned to all the independent variables which are then fed into the decision tree which predicts results. The weight of variables predicted wrong by the tree is increased and these variables are then fed to the second decision tree. These individual classifiers then ensemble to give a strong and more precise model.


Cache awareness and out-of-core computing
Regularization for avoiding overfitting
Tree pruning using depth-first approach
Efficient handling of missing data
XGBoost
Parallelized tree building
In-built cross-validation capability

The results obtained by implementing XGBoost to predict mobile price range is:

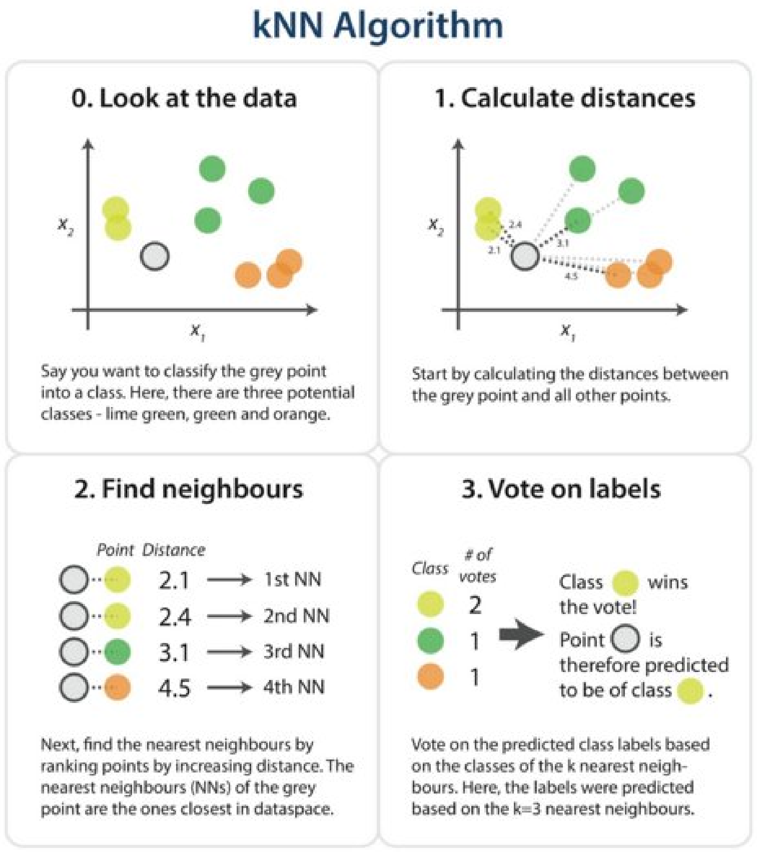
Accuracy	Precision	Recall	ROC AUC
0.9066	0.9075	0.9066	0.9909

- **K Nearest Neighbors:**

KNN algorithm falls under supervised learning category and is used for classification mostly. It is a versatile algorithm, also used for imputing missing values and resampling datasets. The algorithm’s learning is:

- Instance-based learning: It does not learn weights from training data to predict output but uses entire training instances to predict output for unseen data.
- Lazy learning: Model is not learned using training data prior and the learning process is postponed to a time when prediction is requested on the new instance.
- Non-parametric: There is no predefined form of the mapping function.

KNN assumes the similarity between the new cases and available cases and puts the new data into the category that is most similar to the available categories.


kNN Algorithm
0. Look at the data
x_2
x_1
Say you want to classify the grey point into a class. Here, there are three potential classes - lime green, green and orange.
1. Calculate distances
2.4
2.1
3.1
4.5
Start by calculating the distances between the grey point and all other points.
2. Find neighbours
Point Distance
2.1 → 1st NN
2.4 → 2nd NN
3.1 → 3rd NN
4.5 → 4th NN
Next, find the nearest neighbours by ranking points by increasing distance. The nearest neighbours (NNs) of the grey point are the ones closest in dataspace.
3. Vote on labels
Class # of votes
2
1
1
Class wins the vote!
Point is therefore predicted to be of class .
Vote on the predicted class labels based on the classes of the k nearest neighbours. Here, the labels were predicted based on the k=3 nearest neighbours.

The results obtained by implementing KNN to predict the mobile price range is:

Accuracy	**Precision**	**Recall**	**ROC AUC**

0.9176	0.9176	0.9176	0.9775

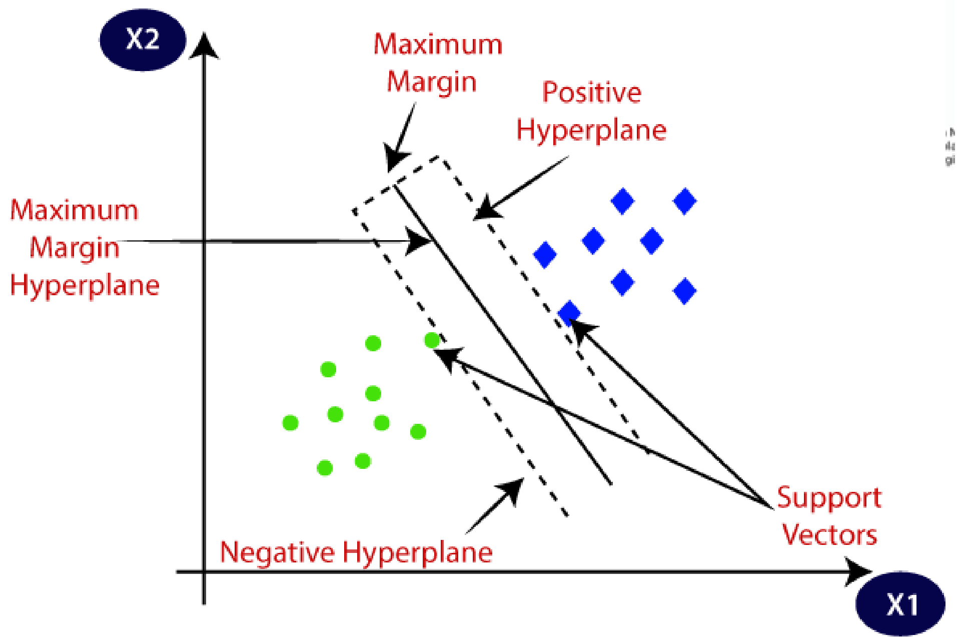
- **Support Vector Machine:**

  Support vector machines are a set of supervised learning methods used for classification, regression and outlier detection. SVMs are different from the other classification algorithms because of the way they choose the decision boundary that maximizes the distance from the nearest data points of all the classes. The decision boundary created by SVMs is called the maximum margin classifier or the maximum margin hyperplane.

  A simple linear SVM classifier works by making a straight line between two classes. This means there can be an infinite number of lines to choose from. What makes the linear SVM better than some of the other algorithms is that it chooses the best line to classify the data points. It chooses the line that separates the data and is the furthest away from the closest data points as possible. The decision boundary doesn't have to be a line. It is also referred to as a hyperplane because we can find the decision boundary with any number of features, not just two.

  There are two types of SVMs, each used for different things:

  - Simple SVM: Typically used for linear regression and classification problems.
  - Kernel SVM: Has more flexibility for non-linear data because we can add more features to fit a hyperplane instead of a two dimensional space.



The results obtained by implementing SVM to predict mobile price range is:

Accuracy	Precision	Recall	ROC AUC
0.9615	0.9616	0.9615	0.9987

Model Comparison and Performance:

The final matrix of results after applying all the algorithms:

Models	Accuracy	Precision	Recall	ROC AUC
Logistic Regression	0.9643	0.9645	0.9643	0.9975

SVM	0.9615	0.9616	0.9615	0.9987
KNN	0.9176	0.9176	0.9176	0.9775
XGBoost	0.9066	0.9075	0.9066	0.9909
Random Forest	0.8956	0.8958	0.8956	0.9888
Gradient Boosting	0.8928	0.8924	0.8928	0.9879

All the algorithms have provided us with a good result for predicting the mobile price range. As our target classes are balanced, we can consider accuracy as an important metric to measure and compare our models. The accuracy achieved by most of the models is above 90% which proves that the models have classified the data well.

Logistic Regression along with SVM has performed the best out of all the models as they have achieved an accuracy of 96% and also scored well in precision, recall and roc auc. The tree based methods have performed poorly in comparison to the other classification models implemented.

Challenges:

- There were few records with zero values for some features which is impossible in real life scenarios.
- Only a few features showed good correlation with the target variable, thus difficult to achieve very good accuracy.
- No feature followed a normal distribution.

Conclusion:

We have reached the end of our mobile price range prediction project and have achieved a fairly good result for all the models implemented. We have discovered a lot of insights from the data through EDA which helped us in determining which features will have a strong influence on the price range prediction.

This kind of prediction is very important for businesses to understand which factors drive the price of a mobile phone and estimate the price of mobile phones accurately to give good competition to other manufacturers.

Let us now summarize the whole work in few points below:

- The dataset contained 2000 records which were a mix of categorical features and numerical features.
- The dataset was almost cleaned as there were no null values present or duplicate records found.
- Few features had zero values which had to be removed before proceeding further.
- The target classes were almost balanced so we didn't need to worry about class imbalance problem.
- Most of the categorical features had a similar distribution except 'three_g' where there were very few records for mobile phones not having 3G access.
- The story remains the same for categorical features when we break down the distribution across different price ranges.
- Most of the numerical features follow an uniform distribution except few features which follow a right skewed distribution.
- There is a slight increase in the count of each feature for the very high cost category. Thus we can infer that the more we pay, the more choices we get to buy a mobile phone with all the features present.
- RAM, battery power, px_height and px_width increase with increase in price. Rest of the numerical features don't show a significant change with the price range.
- Through correlation plots, we found out that RAM, battery power, px_height and px_width have the strongest correlation with the target variable. These features will be the most influential in determining the price ranges.
- No categorical feature has a strong correlation with the target variable.

With the help of EDA we found out which features are related to each other, which features might drive the prices. There was no multicollinearity problem so we didn't need to eliminate any feature.

We implemented six classification models to evaluate and compare their performances for our mobile price range prediction.
The six classification models were:

- Logistic Regression
- Random Forest
- Gradient Boosting
- XGBoost
- K Nearest Neighbors
- Support Vector Machine

All the models have done a good job in classifying our data as most of the models have achieved an accuracy over 90% which is a very good measure when the target classes are balanced. Apart from accuracy as a metric we have used precision, recall and roc auc score to compare and cross validate the performance of our models.

Logistic Regression along with SVM has performed the best in our case and have achieved an accuracy of 96% which is way above the other models. Logistic Regression with standardization/feature scaling has surpassed our expectations for the mobile price range prediction. The tree based methods in general have performed poorly in comparison to the other classification models. We have also calculated the feature importances from the tree based models and have seen that RAM, battery power, px_height and px_width are the only important features to determine the price range. These models can be further used in production to predict the price range of unseen data accurately.

References:

- Analytics Vidhya
- Statistics How To
- Machine Learning Mastery
- Javatpoint
- Wikipedia
- Towards Data Science
- freecodecamp