Mobile Price Range Prediction

Ankit Rai
Data science trainee
AlmaBetter, Bangalore

ABSTRACT

We are in an era of mobile phone revolution, where the changes in a span of less than a decade have been so dramatic that many have come to realize that they can't think of a life without a mobile phone. Smartphones have become such a necessity that not owning one is questioned, not the other way around. The dependability of people on their phone created a huge demand in the phone market, due to which many new players (phone Brands) entered the market. Today, phone market is considered as a highly competitive market, and one must do adequate research before entering or launching their new product.

Harvard studies have found that a 1% improvement in your pricing can add up to 11% to your profits. With bad pricing, you're missing out on profits in every transaction that you make, not to mention the deals that you completely miss out on.

Our job is to perform data analysis and to develop a predictive classification model on the given data, which can identify the important factors/features driving mobile prices and predict the suitable price range for the mobile.

PROBLEM STATEMENT

Price could be said as the most influential factor affecting the purchase of a new mobile phone. The RIGHT Pricing is also a big factor in company's branding and reputation. Mobile phone customers have perceived price as a significant indicator of product quality, whereby high price indicates advanced technology, design, and improved features.

In the competitive mobile phone market companies want to understand sales data of mobile phones and factors which drive the prices. The goal is to find out some relation between mobile selling price and its features. These features are those attributes which are related directly to the phone itself and plays an important role in mobile price decision making.

Our main objective is to build a predictive classification model, which could identify the important factors which drives mobile price and predict the suitable price range depending on the given phone attributes. This would in turn help the company to make RIGHT pricing decisions.

ABOUT THE DATA

We have been provided sales data of mobile phones for the analysis. The dataset contains following 21 number features, which are classified as below:

- Dependent Variable:
    - ***Price_range -*** This is the target variable with value of 0(low cost), 1(medium cost), 2(high cost) and 3(very high cost).
- Binary Independent Variables:
    - ***Blue –*** tells whether phone has bluetooth or not
    - ***Dual_sim –*** tells whether phone has dual sim support or not
    - ***Four_g -*** tells whether phone has 4G or not
    - ***Three_g*** - tells whether phone has 3G or not
    - ***Touch_screen -*** tells whether phone has touch screen or not
    - ***Wifi -*** tells whether phone has wifi or not
- Numeric Independent Variables:
    - ***Battery_power -*** Total energy a battery can store in mAh
    - ***Clock_speed -*** speed at which microprocessor executes instructions
    - ***Fc -*** Front Camera mega pixels
    - ***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

STEPS INVOLVED

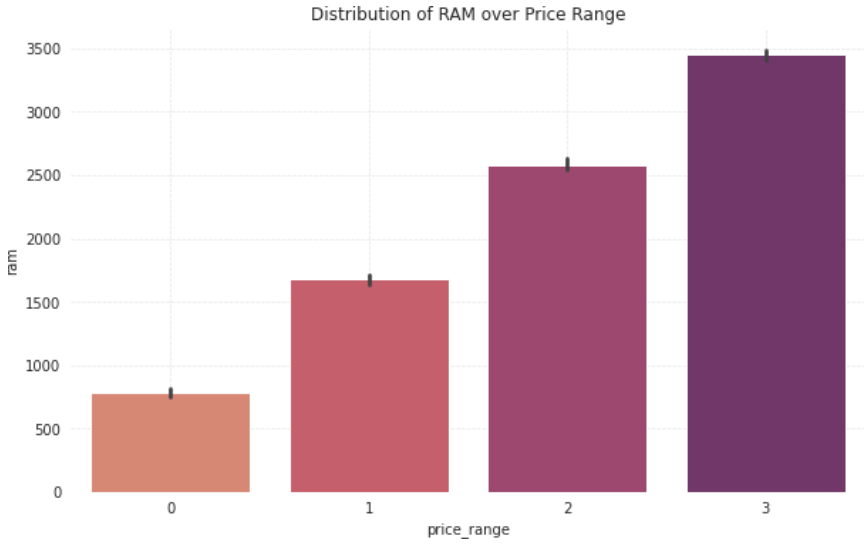
1. Data Pre-processing

We started with checking for null values and duplicate values and ensured that none of them were present. Then we looked for outliers in our data and found that there were none/minimal outliers present which were not irrelevant at all. Then we derived new variables (sc_area and px_size) from the existing variables and dropped the variables which were not further required.

2. Discovering Information – EDA

We performed EDA on our data by comparing our dependent variable ("price_range") with other independent variables. This process helped us figuring out various aspects and relationships which impacts on mobile phone prices. We also used Bar plots and line graphs to represent the effects of change in independent variables on mobile price. Some of the discovered aspects were:

1. Almost 66% of mobiles in the market support 3g and the ratio of phones supporting 3g to those which do not support 3g is almost same.
2. Very cost phones have high battery power, while low cost phones have lowest battery power.
3. RAM feature is very strongly and linearly proportional to the phone price. That is price increase with increase in Random Access Memory of a phone.
4. Pixel Size is maximum for very costly phones while it is lowest for low-cost phones.



3. Data Preparation for Modelling –

- **Splitting Data:** First we assigned our dependent and independent variable values to new variables X and y respectively. Then we split 20% data for testing and rest for training.

- **Rescaling:** To ensure that the variables are on same scale, we standardized the variables using' StandardScaler'. The idea was to scale our data into a uniform format that would allow us to utilize the data in a better way while performing fitting and applying.

4. Model Fitting & Model Evaluation -

After splitting and rescaling our values, our data was ready to be applied on classification models. A classification model tries to draw some conclusion from the input values given for training. It will predict the class labels/categories for the new data.

The following are the steps involved in building a classification model:

- **Initialize:** It is to assign the classifier to be used for classification. Considering the requirement of multiclass classification, we are going to use four classifiers for this project : Decision Tree, KNeighbors, Random Forest and Gausian Naive Bayes classifier.

- **Train the classifier:** All classifiers in scikit-learn uses a fit(X, y) method to fit the model(training) for the given train data X and train label y.

- **Predict the target:** Given an unlabeled observation X, the predict(X) returns the predicted label y.

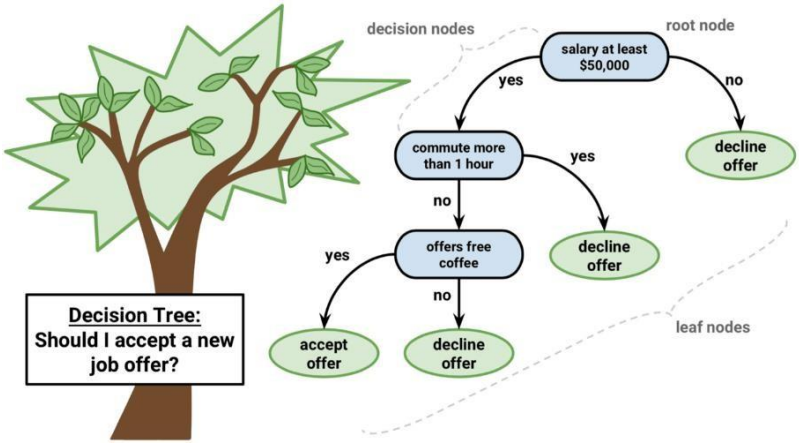
- **Evaluate the classifier model:** For evaluation we will be using Accuracy score.

- **Cross Validation:** Cross-validation is a technique for evaluating ML models by training several ML models on subsets of the available input data and evaluating them on the complementary subset of the data.

ALGORITHMS

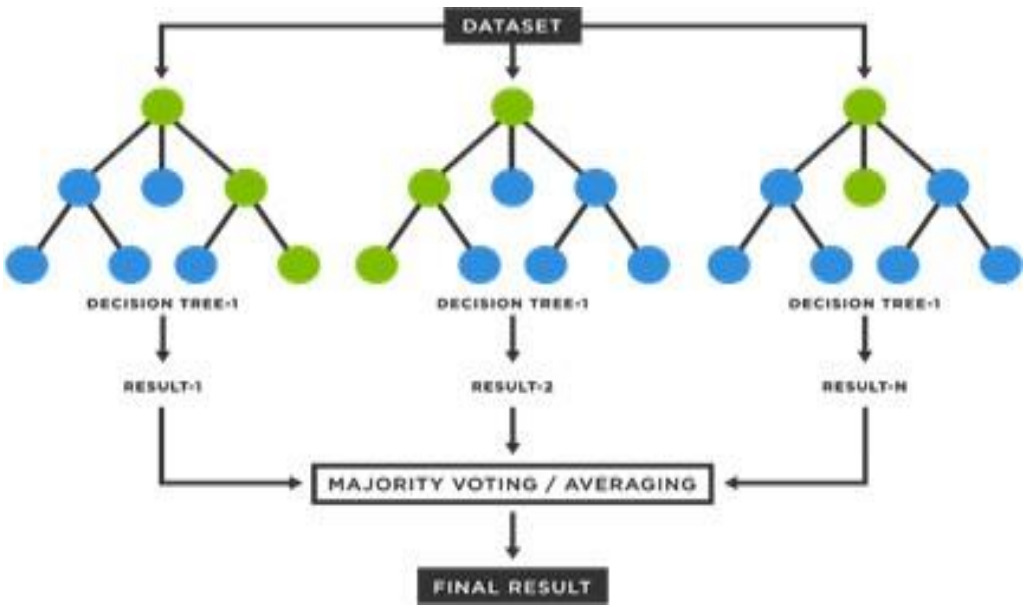
a. Decision Tree:

A decision tree is a supervised learning algorithm that is perfect for classification problems, as it's able to order classes on a precise level. It works like a flow chart, separating data points into two similar categories at a time from the "tree trunk" to "branches," to "leaves," where the categories become more finitely similar. This creates categories within categories, allowing for organic classification with limited human supervision.



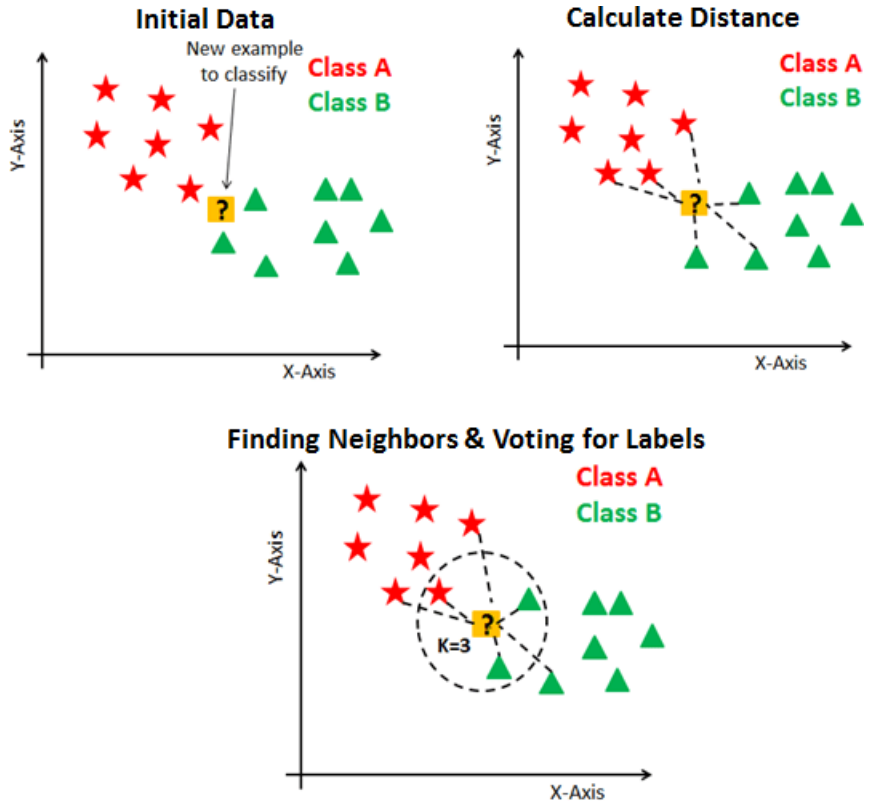
b. Random Forest:

A Random Forest is a reliable ensemble of multiple Decision Trees (or CARTs); though more popular for classification. Here, the individual trees are built via bagging (i.e. aggregation of bootstraps which are nothing but multiple train datasets created via sampling of records with replacement) and split using fewer features.



c. K-nearest Neighbors:

K-nearest neighbors (k-NN) is a pattern recognition algorithm that uses training datasets to find the k closest relatives in future examples. When k-NN is used in classification, you calculate to place data within the category of its nearest neighbor. If k = 1, then it would be placed in the class nearest 1. K is classified by a plurality poll of its neighbors.



d. Naive Bayes:

Naive Bayes calculates the possibility of whether a data point belongs within a certain category or not. To decide whether or not a a data point belong to a certain category Naïve Bayes uses following calculation-

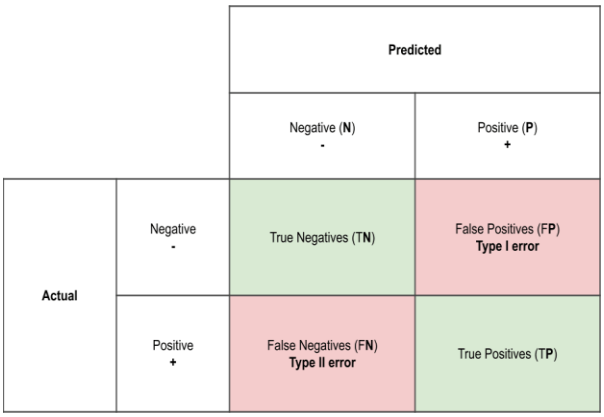
$$P(A|B) = \frac{P(B|A) \times P(A)}{P(B)}$$

MODEL PERFORMANCE METRICS

Model can be evaluated by various metrics such as:

a. **Confusion Matrix**:

A confusion matrix is a table that is often used to describe the performance of a classification model (or "classifier") on a set of test data for which the true values are known. Each row of the matrix represents the instances in an actual class while each column represents the instances in a predicted class, or vice versa – both variants are found in the literature.



b. Precision/Recall:

Precision is a measure of result relevancy, while recall is a measure of how many truly relevant results are returned.

Precision (P) is defined as the number of true positives (Tp) over the number of true positives plus the number of false positives (Fp).

$$\square = \frac{\square_{\square}}{\square_{\square} + \square_{\square}}$$

Recall (R) is defined as the number of true positives (Tp) over the number of true positives plus the number of false negatives (Fn).

$$\square = \frac{\square_{\square}}{\square_{\square} + \square_{\square}}$$

c. Accuracy Score:

Accuracy is given by the number of correctly classified examples divided by the total number of classified examples. In terms of the confusion matrix, it is given by:

$$\text{Accuracy} = TP+TN/TP+TN+FP+FN$$

d. AUC ROC Score:

The Area Under the Curve (AUC) is the measure of the ability of a classifier to distinguish between classes and is used as a summary of the ROC curve. The higher the AUC, the better the performance of the model at distinguishing between the positive and negative classes.

e. CV Score:

Cross-validation is primarily used in applied machine learning to estimate the skill of a machine learning model on unseen data. This involves simply repeating the cross-validation procedure multiple times and reporting the mean result across all folds from all runs.

We got following results after applying performance evaluation metrics:

	Model	CV Score	Test Accuracy	Train Accuracy	Test Precision	Train Precision	Test Recall	Train Recall	ROC AUC Score
0	Decision Tree	0.848785	0.8475	1.000000	0.850420	1.000000	0.8475	1.000000	0.899799
1	K-Nearest Neighbors	0.498530	0.4925	0.706875	0.511893	0.716247	0.4925	0.706875	0.740154
2	Random_Forest_Classification	0.871019	0.8500	1.000000	0.853321	1.000000	0.8500	1.000000	0.970705
3	Naive Bayes	0.799961	0.7550	0.816875	0.760865	0.817445	0.7550	0.816875	0.933301

HYPERPARAMETER TUNING

Hyperparameter tuning is choosing a set of optimal hyperparameters for a learning algorithm. A hyperparameter is a model argument whose value is set before the learning process begins. We used GridSearch CV method to optimize our model. It helps to loop through predefined hyperparameters and fit your estimator (model) on the training set. So, in the end, we can select the best parameters from the listed hyperparameters.

We had chosen Random Forest Classifier for our prediction and the best Hyperparameters obtained are as below.

- criterion = ‘entropy’
- max_depth = 8
- max_features = ‘log2’
- N_estimators = 200

CONCLUSION

We have reached to the conclusion of our exercise !! We started this project with the intention to identify the useful variables/factors which drives phone price and to build a predictive model which can give phone price range depending on its feature. For this, we performed exploratory data analysis on our data after cleaning and making it easy to analyse. This analysis helped us to identify variables which directly impacts Mobile Phone Prices. We found that 'RAM' of a phone linearly affects the phone prices. Other variables like battery life and ' px_size' also shows linearity (up to extent only) with the phone prices. We found that most *very high-cost* phones have low handset weight, high Internal Memory, high pixel and screen size. Our next job was to make a price range predictive model. For this, we processed our data, split it for training and testing and finally applied four different models. Both Random Forest and Decision tree classifiers gave some good accuracy scores, however on cross validation scores Random Forest performed much better. This gave us confidence to perform hyperparameter tuning on our Random Forest model. Finally, we optimized our mode, which increased the accuracy score of our model to 0.86, which is pretty decent.

With this we achieved our objectives of the project.