Mobile price range prediction

Karan Tiwari

Abstract

Various aspects like as memory, display, battery, camera, and so on are examined while purchasing a mobile phone. People make incorrect judgments because the required resources to cross-validate the pricing are unavailable. To overcome this issue, a machine learning model is created utilizing data from the mobile phone's main attributes. The model that has been constructed is then utilized to anticipate the price range of the new mobile phone. To train the model and forecast the output as low, medium, high, or very high, machine learning methods such as Logistic Regression, Decision Tree Classifier, Random Forest Classifier, XGBoost Classifier, K-Nearest Neighbor Classifier, and hyper parameter tuning using GridsearchCV are employed.

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.

Objective

The project's primary objective is to implement a Classifier model that can predict the prices of mobiles phones .

Data Description

The dataset has 2000 rows and 21 columns

- Dataset does not have any null values
- No outliers found in our dataset.
- No duplicate rows

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 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)**.

Challenges

We did not face any issues in tackling null values and outliers because the dataset was perfect and no transformation was required in the dataset.

Tools Used

This project is performed on Google collaboratory and Python was used throughout the entire project. The following libraries were imported for data analysis and visualization:.

- Pandas: Extensively used to load and wrangle with the dataset.
- Matplotlib: Used for visualization.
- Seaborn: Used for visualization.
- .Numpy: For some math operations in predictions.
- Sklearn: For the purpose of analysis and prediction.

Steps involved

The following steps are involved in the project

- **Exploratory Data Analysis:** After loading and reading the dataset in notebook, we performed EDA.
  Exploratory Data Analysis is the crucial process of doing preliminary investigations on data in order to uncover patterns, spot anomalies, test hypotheses, and validate assumptions using summary statistics and graphical representations.

- **DATA MODELLING**

  In this we assign independent and dependent features.

  Independent variable: An independent variable is exactly what it sounds like. It is a variable that cannot be influenced by the other elements you are seeking to evaluate.

  Dependent variable:: A dependent variable is exactly what it sounds like: something on which other components are reliant.

- **Train test split**

  In train test split we take ‘x’ as dependent variables and ‘y’ take as independent variable then train the model.

- **Model Implemented**

  We uses 5 ML Models to train the data and for predicting the accuracy,
  - Logistic Regression
  - Decision Tree
  - Random Forest Classifier
  - XGBoost Classifier
  - K-Nearest neighbor

- **Algorithms**

  - **Logistic Regression**

  Logistic Regression is actually a classification algorithm that was given the name regression due to the fact that the mathematical formulation is very similar to linear regression.

The function used in Logistic Regression is sigmoid function or the logistic function given by:

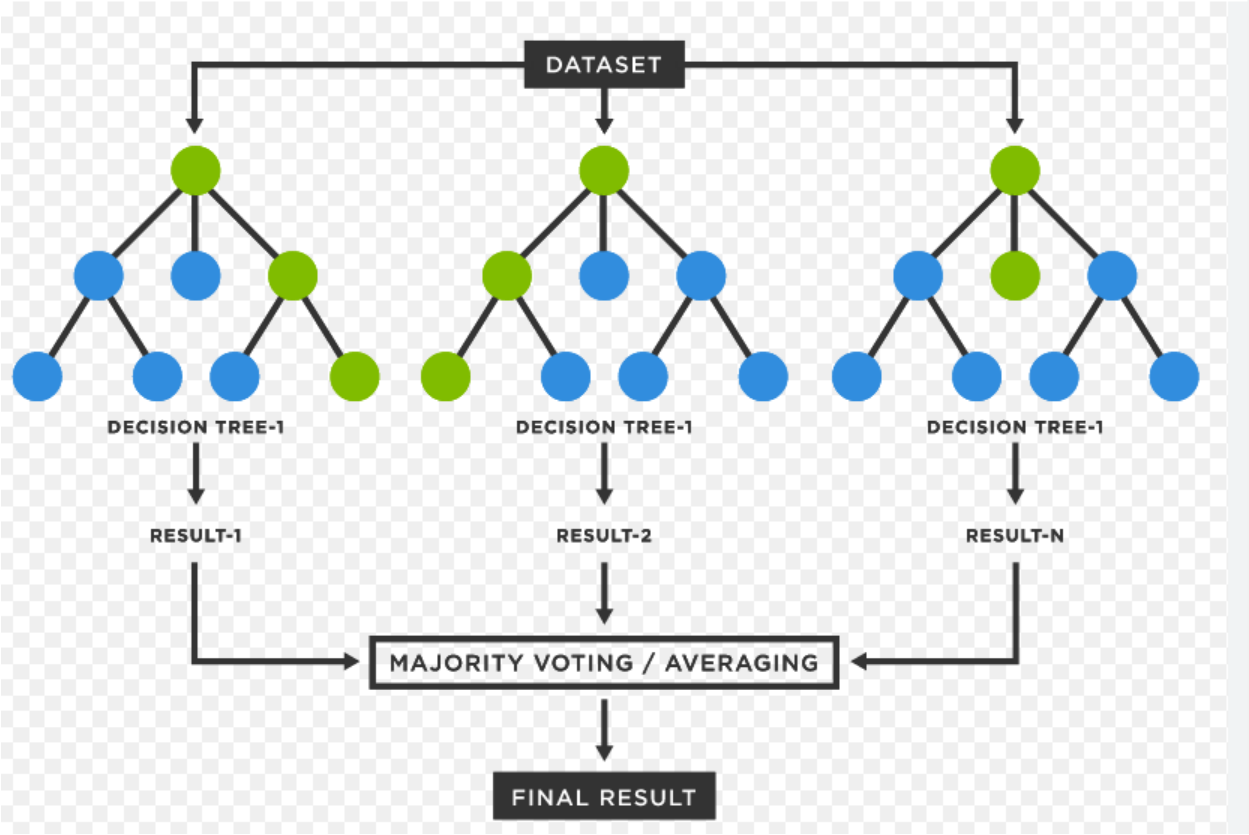
$$f(x)= 1/1+e ^(-x)$$

- **Decision Tree**

Decision Tree is the most powerful and popular tool for classification and prediction. A Decision tree is a flowchart-like tree structure, where each internal node denotes a test on an attribute, each branch represents an outcome of the test, and each leaf node (terminal node) holds a class label.

- **Random Forest Classifier**

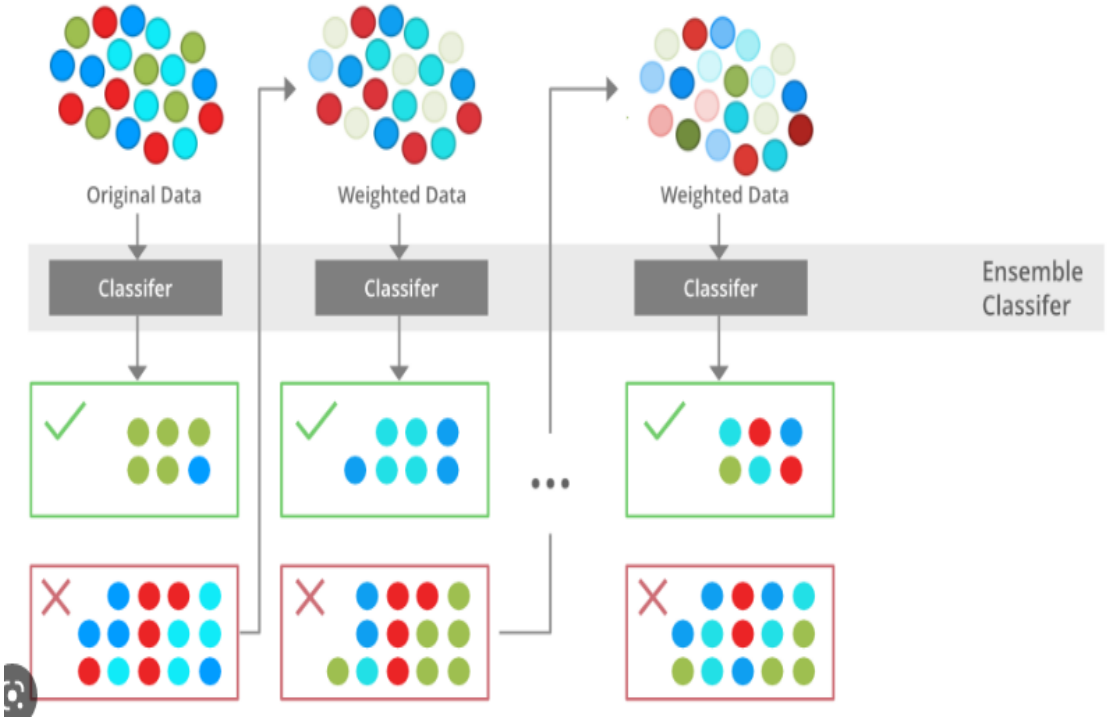
Random Forest is a bagging type of Decision Tree Algorithm that creates a number of decision trees from a randomly selected subset of the training set, collects the labels from these subsets and then averages the final prediction depending on the most number of times a label has been predicted out of all.



- **XGBoost Classifier**

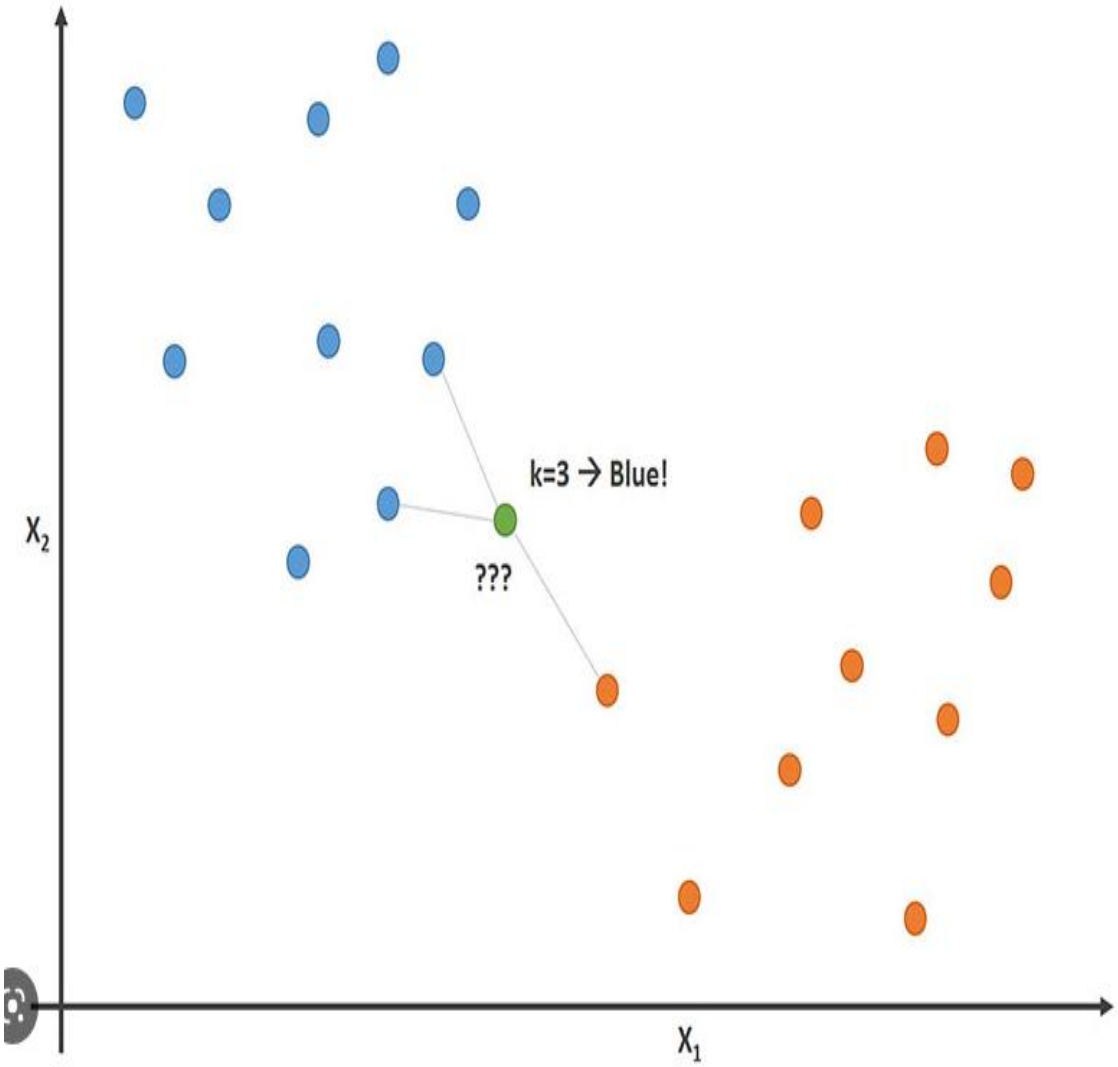
  XGBoost is an optimized distributed gradient boosting library designed to be highly efficient, flexible and portable. It implements machine learning algorithms under the Gradient Boosting framework. XGBoost provides a parallel tree boosting that solve many data science problems in a fast and accurate way.

XGBoost Classifier



- **K Nearest Neighbor**

The k-nearest neighbors algorithm, also known as KNN , is a non-parametric, supervised learning classifier, which uses proximity to make classifications or predictions about the grouping of an individual data point.



Model performance

Model can be evaluated by various metrics such as:

- **Confusion matrix**

The confusion matrix is a table that summarizes how successful the classification model is at predicting examples elonging to various classes. One axis of the

confusion matrix is the label that the model predicted, and the other axis is the actual label.

- **Precision/Recall**-

Precision is the ratio of correct positive predictions to the overall number of positive predictions: TP/TP+FP
Recall is the ratio of correct positive predictions to the overall number of positive examples in the set: TP/FN+TP

- **Accuracy**-

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: TP+TN/TP+TN+FP+FN

Hyper parameter tuning:

Finding the optimal mix of hyperparameters to enhance the model's performance is known as hyperparameter tweaking. It operates by doing several trials within a single training procedure. Every trial entails the full execution of your training application with the values of the selected

hyperparameters set within the predetermined bounds. Once complete, this procedure will provide you with the set of hyperparameter values that are ideal for the model to produce the best outcomes.

Grid search CV

Grid Search combines a set of hyperparameters chosen by the scientist and goes through them all to assess the model's performance. It has the benefit of being a straightforward procedure that will go through all of the preset combinations.

8. Conclusion:

That's all! We had completed our exercise.
So far, we have done EDA, data modeling, and model construction, beginning with data loading.
In all of these models, our accuracy ranges from 76 to 93%.
Even after hyperparameter adjustment, there is no improvement in accuracy score. So our best model's accuracy is 93%, which is considered decent.