|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | | | | | | | | |
|  | | PROJECT REPORT | | | | |  | |
|  |  | | | | | | |  |
|  | | | |  | |  | | |
|  | | | | EDA ON MOBILE SALES DATASET | |  | | |
|  | | | | September 7th, 2024—Machine Learning using python—Rishav BairagyaSouma DasSarthak Dutta | |  | | |
|  | | |  | |  | | | |



**Index**

1. **Introduction** 
   1. What is data analytics
   2. Libraries used in this project
   3. Implemented procedure
2. **Project Explanation**
   1. About dataset
   2. Data Analytics
   3. Encoding data
3. **Machine learning**
   1. Libraries imported
   2. Terms used in machine learning
   3. Machine learning methods used summary
   4. Which model had the best accuracy
4. **Conclusion**
5. Introduction

In this Project, we have aimed to predict the selling price of mobile phones from different Brands using various machine learning algorithms, to find the best suited algorithm for our dataset, performance comparison, and an exploratory data analysis on the mobile sales dataset.

**1.1 WHAT IS DATA ANALYTICS**

Data analytics is the process of examining, cleaning, transforming, and modelling data to discover useful information, draw conclusions, and support decision-making. It involves using statistical techniques, algorithms, and software tools to analyse raw data and extract meaningful insights that can inform business strategies, improve processes, and predict future trends.

Key aspects of data analytics include:

* Data Collection: Gathering relevant data from various sources, such as databases, spreadsheets, or external sources like social media.
* Data Cleaning: Removing errors, duplicates, and inconsistencies from the data to ensure accuracy and reliability.
* Data Exploration: Using descriptive statistics and visualisation tools to understand the data, identify patterns, and detect anomalies.
* Data Modelling: Applying statistical models or machine learning algorithms to the data to identify relationships, predict outcomes, or classify information.
* Data Interpretation: Analysing the results to draw meaningful conclusions and provide actionable insights.
* Data Visualization: Presenting the findings in a clear and understandable way, often through charts, graphs, or dashboards.

Applications of Data Analytics:

* Business Intelligence: Improving decision-making by analysing sales data, customer behaviour, and market trends.
* Healthcare: Enhancing patient care through analysis of medical records and treatment outcomes.
* Finance: Detecting fraud, managing risk, and optimising investment strategies.
* Marketing: Personalising marketing campaigns by analysing consumer data and preferences.

**1.2 LIBRARIES USED IN THIS PROJECT**

Python is widely used in machine learning due to its simplicity, readability, and the extensive ecosystem of libraries an frameworks that facilitate the development and deployment of machine learning models.

Here are some Python Libraries that we used for our project:

1. Data Preprocessing:

Libraries: Pandas, NumPy, Sklearn-Ordinal Encoder.

Dataset cleaning, transforming, and organizing raw data into a suitable format for analysis. Libraries like Pandas and NumPy are commonly used for handling data structures and performing mathematical operations. Encoding and Scaling are done with the help of Sklearn Ordinal encoder.

1. Exploratory Data Analysis (EDA):

Libraries: Matplotlib, Seaborn.

Python's visualization libraries like Matplotlib and Seaborn allow for creating plots, charts, and graphs to understand the underlying patterns, trends, and relationships in the data. EDA is crucial for selecting features and understanding data distribution.

1. Model Training and Evaluation:

Libraries: Scikit-learn, XGBoost.

These libraries allow for the efficient training of models using various algorithms and evaluating their performance using metrics like accuracy, precision, recall, R2 score, etc. Cross-validation and hyperparameter tuning can also be done using these.

**1.3 PROCEDURE IMPLEMENTED**

Steps taken to complete this project:

1. Data Preprocessing: first we have begun by loading the dataset and performing data preprocessing tasks such as removing the duplicate values, null values, and cleaning the dataset by removing some unwanted white spaces in certain features, additionally we have converted the data irregularities under storage, and memory which will be very helpful while implementing ML algorithms. We have also performed Ordinal encoding for categorical values and then scaling them appropriately, this part will be explained in detail later on.
2. Exploratory Data Analysis (EDA): We conducted EDA to understand the distribution of variables, identify outliers, and gain insights into the relationships between features and the target variable.
3. Implementation of machine learning algorithms to our dataset  
   1. Regression Modelling: We have implemented various Machine Learning algorithms which include Linear Regression, Random Forest Regression, Support Vector Machine Regression, and K-Nearest Neighbours Regression, ADA-Boost regressor, Polynomial regressor.
   2. Model Evaluation: For each regression model, we evaluated its performance using Root mean square, Standard deviation, R² score, Absolute mean error to assess how well each model predicted smartphone prices.
4. Conclusion: After comparing the performance of various regression models, we have come up with a statement that the random forest regressor worked best for our dataset and was able to predict the selling price with accuracy score higher than other regressor models.

2. Project explanation

**2.1 About the Dataset**

The dataset that we have selected is based on Mobile Models Data up to the year 2021, which includes Mobile phone models from 17 different companies, and 914 unique models.

The dataset contains:

* 3114 values
* 12 features

The following table represents the categorical and numeric features present in our dataset.

Table 2.1.1 Features description

|  |  |  |
| --- | --- | --- |
| ***Features*** | ***Description*** | ***type*** |
| *Brand* | Company that manufactured the mobile model | object |
| *Model* | The unique ID with which the model is released in market | object |
| *colour* | The colour of the mobile model chassis | object |
| *Memory* | RAM capacity of that mobile model | object |
| *Storage* | Storage capacity of that mobile model | object |
| *Camera* | Camera present or not | object |
| *Rating* | Customer Feedback in a scale of 5 | Float |
| *Selling Price* | Actual price for which the model is sold in market | int |
| *Original Price* | MRP assigned to that mobile phone model | Int |
| *Mobile* | The Global Model Name of that Mobile | Object |
| *Discount* | Discount amount offered by the company on Original Price | int |
| *Discount Percentage* | Discount Amount in Percent | float |

**2.2 Explanation of each procedure with data**

1. **Loading the dataset:**

We load the dataset using df **=** pd**.**read\_csv(“filename.csv”)We can import the csv file using pandas library.

1. **Data Preprocessing:**
   1. **Data Cleaning:** Data cleaning is a crucial step in data preprocessing. It involves detecting and correcting or removing inaccuracies, inconsistencies, and errors in the dataset to ensure that it is accurate, reliable, and ready for analysis or model building. Properly cleaned data leads to better insights and more accurate machine learning models, as poor-quality data can introduce bias or reduce the performance of the analysis.  
        
      In our project we ensured that we check for irregularities in our dataset and clean them accordingly.Checking for null values, after running the function: dataframe.isna().sum()**,** which gave the respective values:

Table 2.1.2 Null values of the dataset

|  |  |
| --- | --- |
| Memory | 43 |
| Storage | 39 |
| Rating | 144 |

Since our dataset contains 3114 values in total, we decided to drop the null values by using dataframe.dropna(inplace=True), instead of replacing them with mean values, as the number of null values were far less than the total number of values to cause any significant change.

* 1. **Checking for duplicates values**: using dataframe.duplicated().sum():  
     We find that there are **91** total duplicate values, which were dropped using dataframe.drop\_duplicates(keep="first", inplace=True)

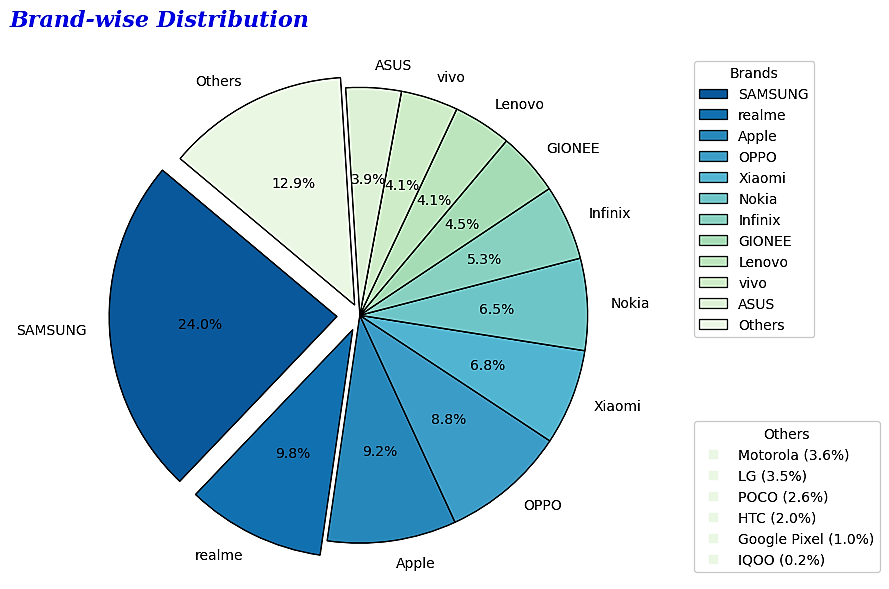
1. Additionally we removed all the anonymous whitespaces which were causing irregularities in analysis by applying dataframe.apply(lambda x: x.str.strip() if x.dtype == "object" else x).
2. **Data Encoding:** Data encoding is the process of converting categorical or textual data into numerical format so that machine learning models can process it. Most machine learning algorithms require numerical inputs, and data encoding allows categorical features (like names, colours, or labels) to be represented in a way that algorithms can understand, which will be covered in details later on.

**2.3 Exploratory Data Analytics**

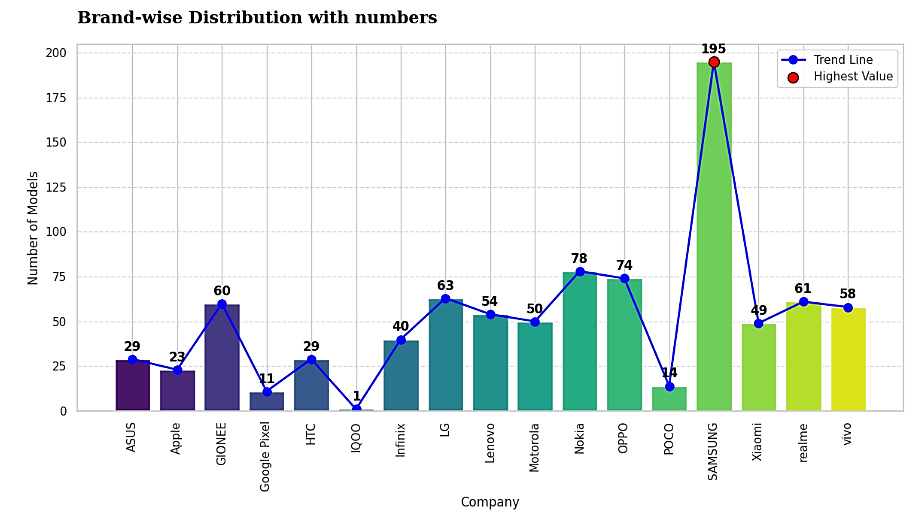
**Exploratory Data Analysis**

The visualisations was made using matplotlib and seaborn libraries

1. **Brand wise distribution:**

Brand wise distribution gives us an insight about the number of mobile models released per Brand throughout the years.Fig 2.3.1 Brand wise Piechart

The representation in the pie chart is done by counting the number of times the brand appears in the dataset thereby taking its value\_counts and grouping the brands under a threshold of 110 counts under others category for sake of convenience and proper representation.

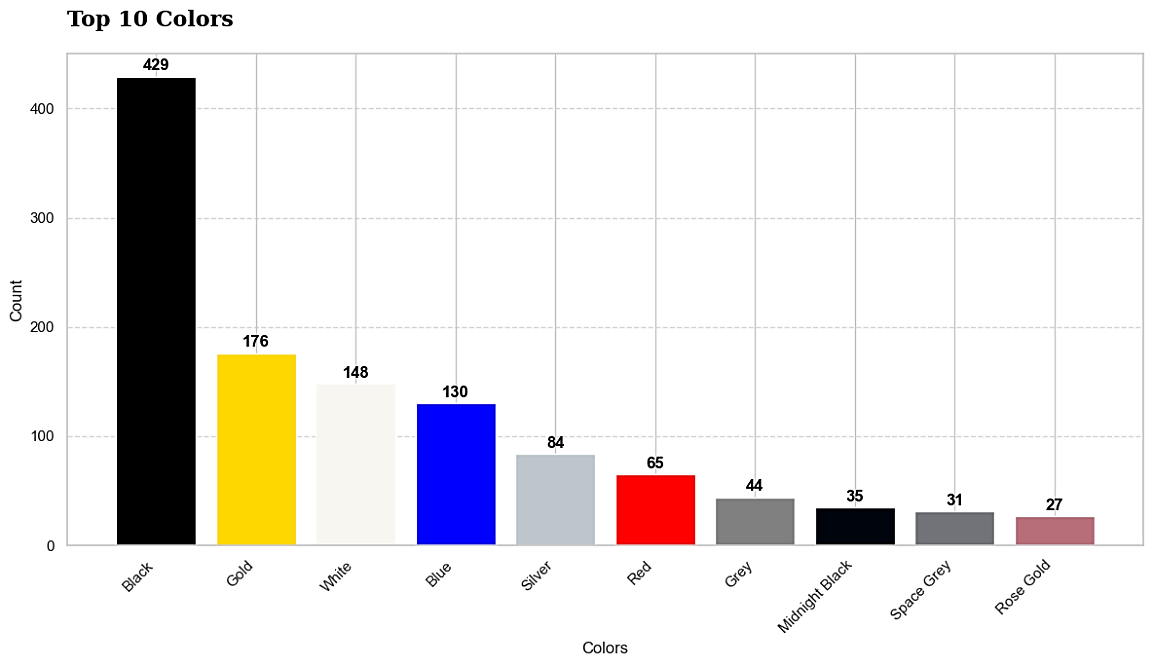
  
Fig 2.3.2 Brand wise bar graph

The bar graph representation is based on number of unique models per Brand and is achieved by grouping the brands and models feature and find the value counts

We can see the majority of the products are from the brand "Samsung" followed by "Apple" and "Realme", but over the years **Samsung** has the greatest number of models in the dataset followed by Nokia and OPPO. Hence, we can conclude that Samsung is the most popular brand in the dataset based on dominating the market solely on a variety of unique models released every year.

1. **Colorwise Distribution**

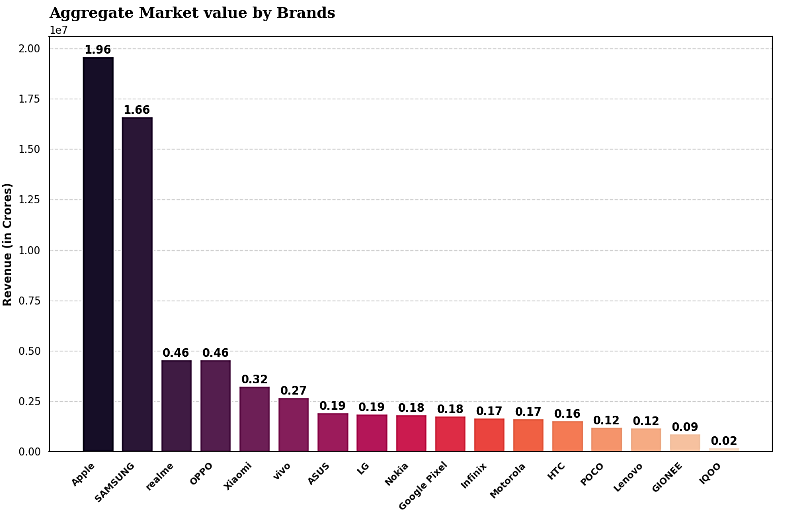
In this Distribution, we calculated the value counts of colour (feature) in the dataset to find the most popular colour preferred by brands and customers over the years, again for convenience factor we only selected top 10 colours by implementing head (10) so have a popular overview of the distribution.

  
Fig 2.3.3 Top 10 colours

As we can see, Black is the most popular colour and is widely used as a medium for mobile body panels.

1. **Market value of each brand**

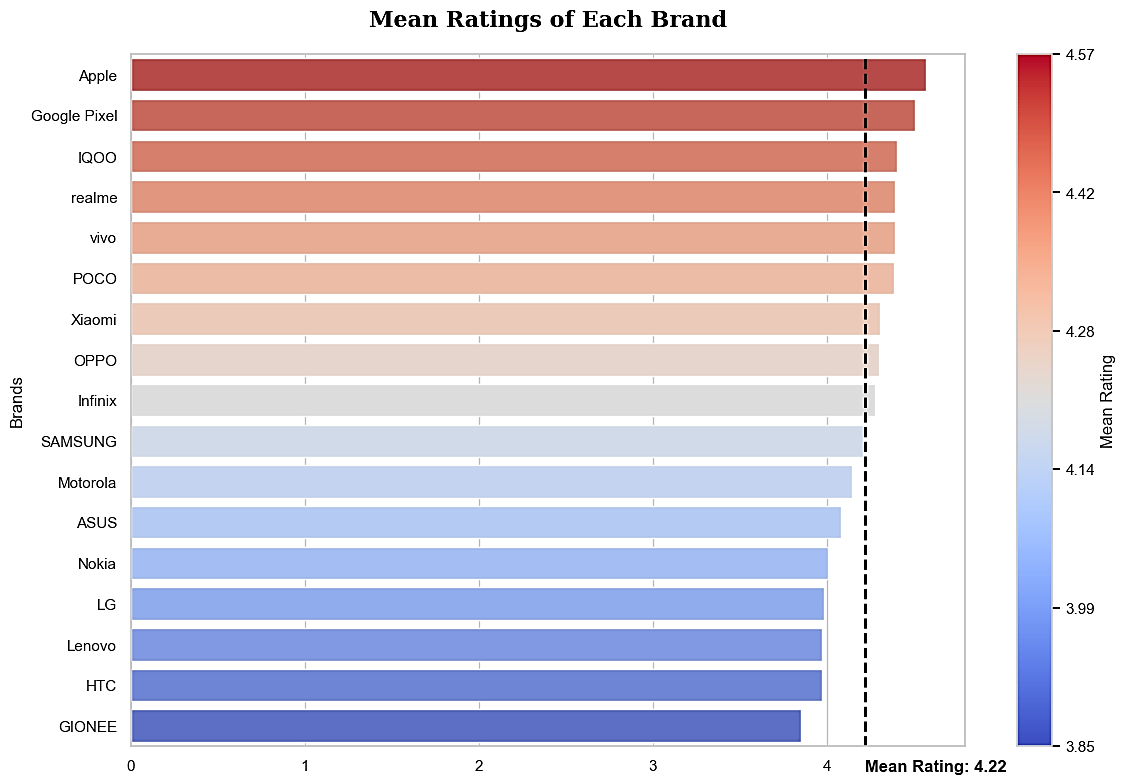
This was done to check the Market value of each brand and was achieved by counting the values after grouping the selling price and brand together and finding their sum and sorting them to have a clear plot of the Aggregate Market Value of Each Brand in Descending order

Fig 2.3.4 Market value of each brand in crores

This shows that Apple has the largest Market value amongst all other brands and Samsung just falling short by a few margins, which clearly shows the competition between these 2 brands to take the top spot. Market Values are represented in Crores.

1. **Mean rating of each brand**

Rating is measured in a scale of 5 of each brand. We have made a bar chart showing the rating of each brand.

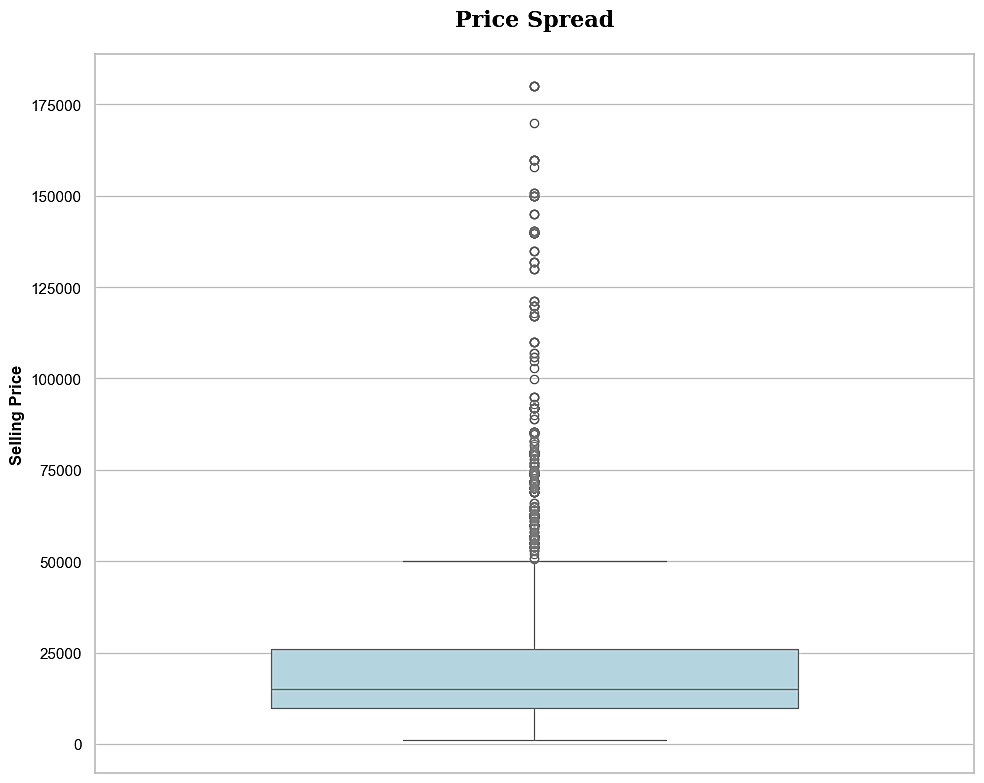
  
fig 2.3.5 Mean rating of Each Brand

We can see apple is the most rated brand in the dataset. This signifies that apple has the greatest number of models with higher ratings.

The reasons could be that apple has the best pricing strategy or the best quality of the products, or it is reaping the benefits of the brand name.

We will see this in the upcoming price discount analysis.

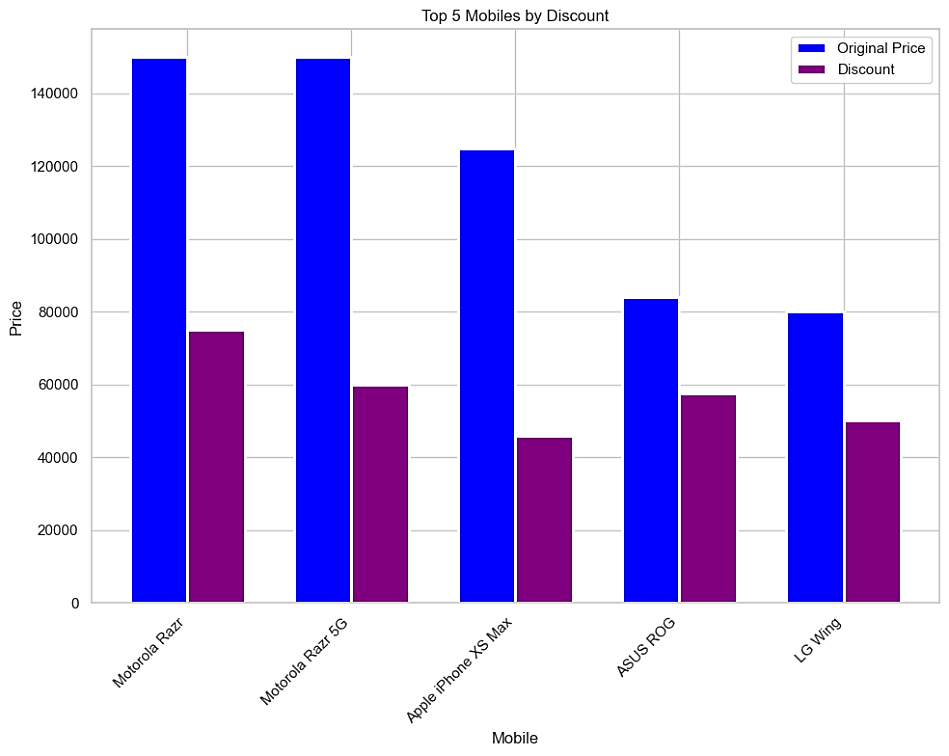
1. **Price spread of dataset**

****Fig 2.3.6 Price Spread of our dataset

We can see most of the models are clustered around the budget price range. But are many premium mobiles above INR 50000. We cannot remove the outliers since there is a significant chunk of data in the premium market section and also some brands only focus on the premium customers. This we will verify in our upcoming observations.

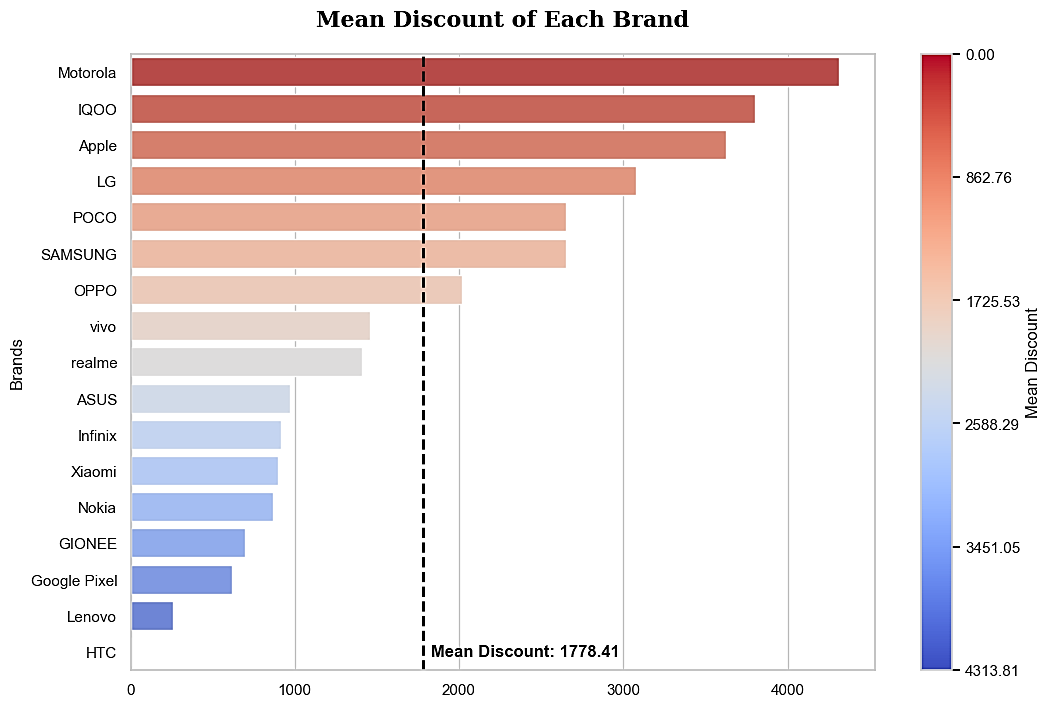
1. **Top 5 brands with highest discount with their selling price**

The aim of this distribution was to check which brand was offering high discounts and on which model. The bar chart consist of 2 bars, blue states the original price and purple is the discount offered.

Fig 2.3.7 Top 5 mobile with respect to their selling price and discount

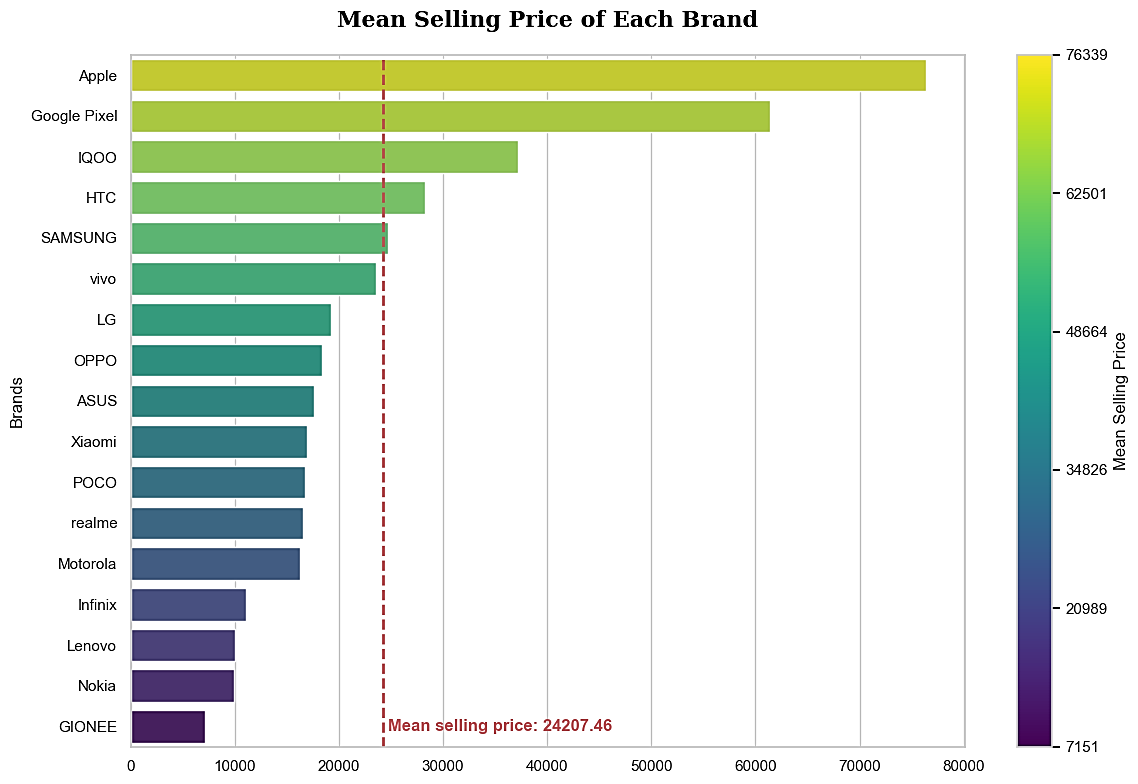
From the Graph we can see that Motorola offered the highest discounts which was on Razr and Razr 5G models along with the highest discount rate from ASUS and LG. Does the brands give this kind of discount for every model or this is just for specific models only? We will discover next.

1. **Brand wise discount distribution Sou**

****

We can see that the most customer friendly brand is maybe **Moto** as it has the highest discount rate but does it have a reasonable selling price?

1. **Mean selling price of each brand**

****Fig 2.3.9 Mean selling price of each brand

We can see in terms of selling price **Motorola** is very low in the list making it very consumer friendly, along with **Gionee** and **Nokia**

Although Gionee, Nokia, Lenovo have the least selling price, their ratings are also lower compared to other brands.

On contrary to this, Apple has the highest selling price but also has the highest ratings. Hence indeed Apple reaps the benefit of being a luxury brand.

1. **Samsung vs Apple price spread**

Now we know that Samsung and Apple have the highest distribution in our dataset. Let us see how the selling price of their models are spread in the market.

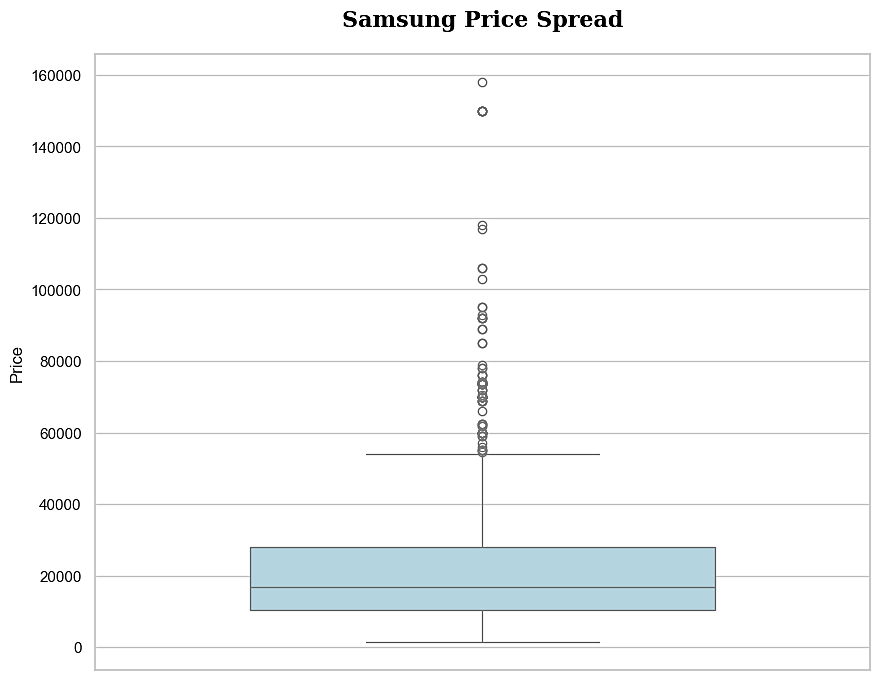
 

Fig 2.3.10 Samsung vs Apple Price spread

We can see that although much of the products of samsung are clustered in the lower price range, it also have quite a few products in the higher price range. Apple on the other hand has most of its products in the higher price range, i.e above 50000. That means removing the outliers from our previous 2.3.6 price spread graph would remove Apple completely.

1. **Price vs Rating**

Are Rating and Selling price related? Lets observe the given graph.

****Fig 2.3.11 Selling price vs Rating

For premium model there is a general trend that the ratings are higher for higher priced models. (Which may also be biased due to the high price a customer pays for the product).

The lower to mid-range models have a wide spread of ratings which means people are review more bluntly and are dissatisfied more with this price range category compared to the premium price range.

**Storage Distribution**

Here we have shown the distribution of storage (ROM) of the phones after calculating.

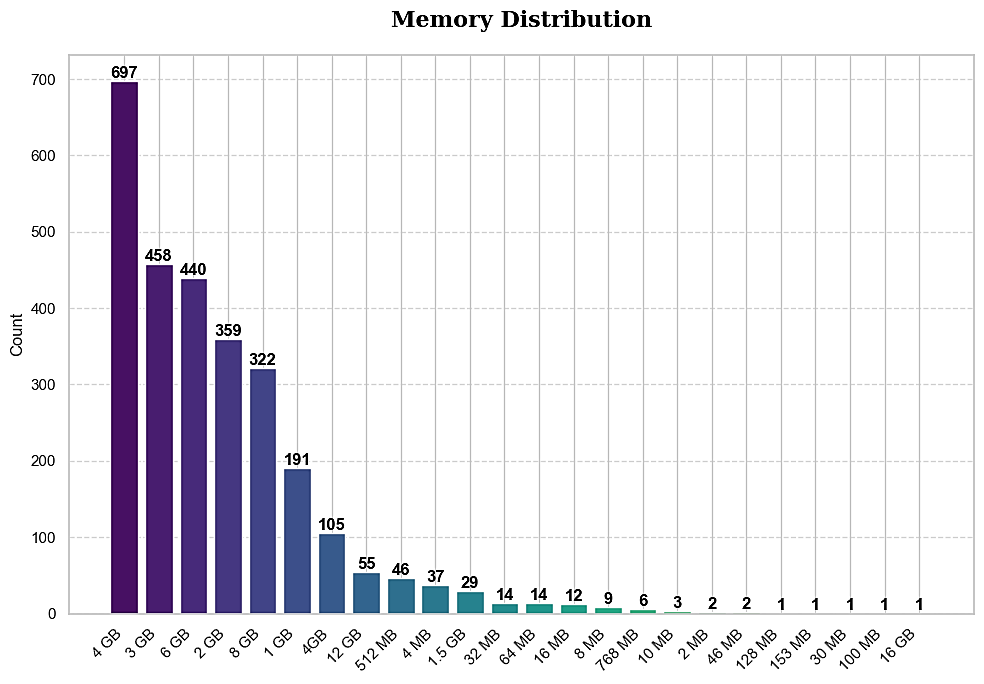


Fig 2.3.12 Storage wise distribution

The fact that companies are focusing on more 64GB and 128GB models is very good, which shows how demands are increasing for more storage.

**Memory Distribution**

Here we have shown the distribution of memory (RAM) of the phones.

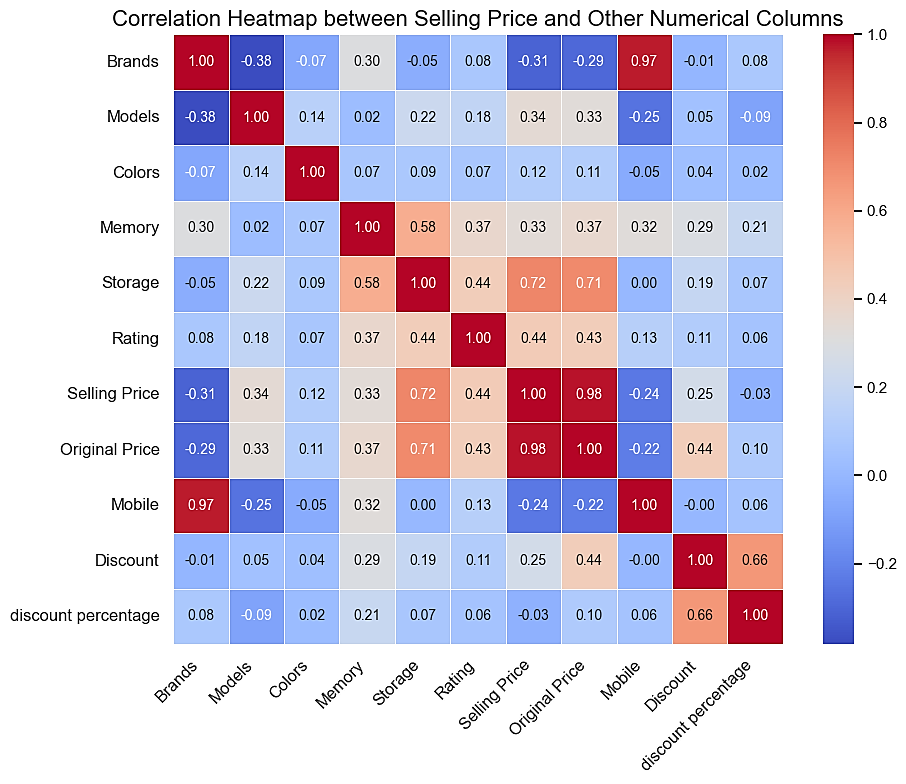
  
Fig 2.3.13 Memory Wise distribution.

By far the most popular memory size is 4GB, which is obvious as most of the phones are in the mid-range category. There is one 16 GB model which means companies do not target the gaming industry much.

1. **Heatmap of all features**

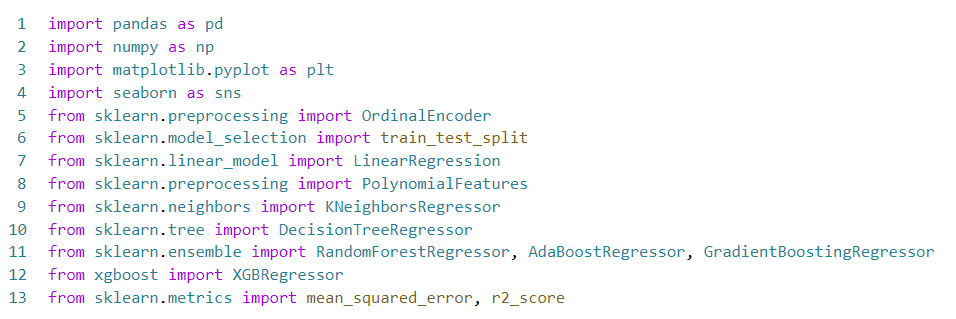
Heatmap is a usually used to see the correlation among each features vs feature. This can be used to select the features or just check the relationship between each variable.  
The value of correlation coefficient ranges from -1 to 1 where

* 1 means the two feature x and y are highly correlated, i.e. if one increases other increases too
* 0 means the two features do not have any correlation or pattern of increment decrement
* -1 means the two features are inversely correlated, i.e. if one increases other decreases.

  
Fig 2.3.14 Heatmap of all features based on correlation coefficient.

3. Machine Learning

**3.1 Libraries used**

****

Brief summary of the libraries

1. pandas: For data manipulation and analysis, particularly with dataframes.
2. numpy: For numerical operations, especially arrays and mathematical functions.
3. matplotlib: To create visualizations like plots and graphs.
4. seaborn: A statistical plotting library, used for more advanced and visually appealing graphs.
5. sklearn.preprocessing.OrdinalEncoder: Encodes categorical variables into numerical values for machine learning.
6. sklearn.model\_selection.train\_test\_split: Splits data into training and testing sets for model evaluation.
7. sklearn.linear\_model.LinearRegression: A basic regression model for predicting continuous variables.
8. sklearn.preprocessing.PolynomialFeatures: Transforms features to include polynomial terms for nonlinear regression.
9. sklearn.neighbors.KNeighborsRegressor: A regression model based on the k-nearest neighbors algorithm.
10. sklearn.tree.DecisionTreeRegressor: A tree-based regression model that splits data into decision branches.
11. sklearn.ensemble.RandomForestRegressor: An ensemble method using multiple decision trees for better accuracy.
12. sklearn.ensemble.AdaBoostRegressor: Boosting algorithm that improves weak learners by focusing on errors.
13. sklearn.ensemble.GradientBoostingRegressor: A boosting algorithm that corrects errors from previous models.
14. xgboost.XGBRegressor: A fast, optimized gradient boosting algorithm.
15. sklearn.metrics.mean\_squared\_error, r2\_score: Metrics to evaluate model performance, measuring error and fit.

**3.2 Terms used in machine learning implementation**

**Fitting:** It refers to the process of training a model by adjusting its parameters to match the patterns found in the training data. The goal is to make the model capable of accurately predicting or classifying new, unseen data based on what it has learned from the training set.

**Mean Squared Error (MSE)** is a common metric used to evaluate the performance of regression models in machine learning. It measures the average of the squares of the errors, where the error is the difference between the actual value and the predicted value by the model.

where is the actual value, is the predicted value, and n is the number of observations.

**Root Mean Square Error (RMSE):** is a measure of the differences between the predicted values generated by a model and the actual values observed. It is the square root of the average of the squared differences between predicted and actual values.

RMSE **=**

where is the actual value, is the predicted value, and n is the number of observations.

RMSE gives a sense of how far the predictions are from the actual values in the same units as the target variable. A lower RMSE indicates a better fit of the model to the data. RMSE is used since it is more comparable with another metric standard deviation.

**Standard Deviation**: It measures the amount of variation or dispersion in a set of values. In machine learning, it can describe the spread of the data points around the mean, or the variation of the errors (residuals) in predictions.

SD **=**

where represents individual data points and is the mean of the data points.

Since standard deviation is a metric which takes all values and find the net dispersion of the values with respect to the mean, if we compare it with RMSE of our machine learning model, we can find if our model prediction is more inclined based on features or it is simply returning comparing with the mean or its predictions are actually based on features.

Standard deviation for selling price of our dataset is ≈ 24424.65

**R2 (R-squared):** R2 is a statistical measure that indicates the proportion of the variance in the dependent variable that is predictable from the independent variables. It is also known as the coefficient of determination.

R2 **=**

where SSres is the sum of squares of residuals, and SStot is the total sum of squares (proportional to the variance of the data).

In simpler terms it is used to “measure the goodness-of-fit”

An R2 value of

* 0 indicates that the model does show any correlation with the given features vs label
* 1 indicates that the model explains high correlation with the given features vs label

A higher R2 value indicates a better fit of the model to the data.

**3.3 Machine learning method summary**

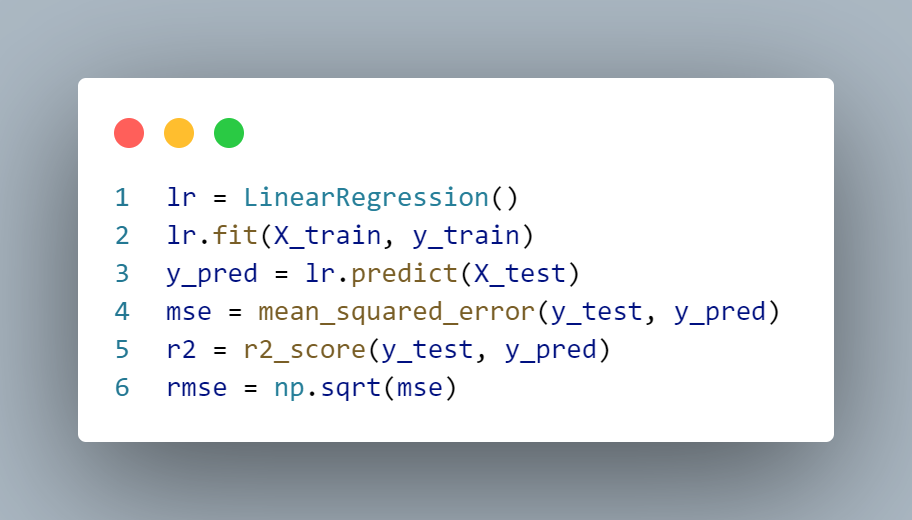
The features for our model are = 'Memory' ,'Storage' ,'Rating' ,'Discount' ,'Brands' , 'Models', 'Discount', 'discount percentage'

The label is the “Selling price”

The methods implemented are as follows:

1. Linear Regression
2. Polynomial Regression
3. KNN
4. Decision tree
5. Random forest
6. Ada boost
7. Gradient boosting
8. XG boost

**Linear Regression:**



It is a type of machine-learning algorithm more specifically a supervised machine-learning algorithm that learns from the labelled datasets and maps the data points to the most optimized linear functions. which can be used for prediction on new datasets.

This model is imported from sklearn.linear\_model

In statement 1- we have created an object from LinearRegression class named lr

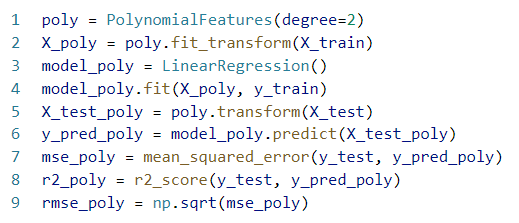
Lr.fit(x\_train, y\_train) here fit() method trains the linear regression model on the training data (x\_train and y\_train). This process involves finding the best-fitting line.

y\_predict is a numpy array whose value is populated by giving the test dataset as input to the model trained inside the object lr

Rmse is the root mean squared error of the actual value(label: y\_test) vs the predicted value(y\_predict) which is 13,795.35 which compared to standard deviation, which is 24424.65, is little bit less only, and the error is quite high.

R2 score of 0.681 shows that the model accuracy is not great.

**Polynomial Regression**

****

It is a type of regression analysis in machine learning that models the relationship between the independent variable(s) and the dependent variable as an nnn-th degree polynomial. Unlike linear regression, which assumes a linear relationship between the features and the target variable, polynomial regression allows for more complex, nonlinear relationships.

This model is imported from sklearn.preprocessing

In statement 1, we have created an object called poly of the class PolynomialFeatures().   
Here degree = 2 means that it will be create fartues up to second degree (quadratic features).

After that the method poly.fit\_transform(x\_train) transforms the original features in x\_train into a new feature matrix that include the original features, their squares and their interaction terms.

Then, we have created an object of the class LinearRegressor() called model\_poly and then in the 4th line we use the fit() method, which trains the linear regression model using the transformed training data (x\_poly) and target variable (y\_train).

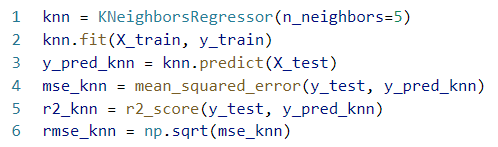
poly.transform(x\_test) transform the x\_test data into same polynomial feature space as the training data.

After training the model, we used predict() method to make predictions on the transformed test data (x\_test\_poly) and we stored the values in y\_pred\_poly.

Rmse is the root mean squared error of the actual value(label: y\_test) vs the predicted value(y\_predict) which is 12602.75 which compared to standard deviation, 24424.65, is also little bit less, and the error is still quite high.

This also has a R2 score of 0.677 which is not up to the mark

**KNN**



It is a simple, non-parametric, and instance-based machine learning algorithm used for classification and regression tasks. The basic idea behind KNN is that it classifies or predicts the value of a data point based on how its nearest neighbors (in terms of distance in the feature space) are classified or valued. The "K" in KNN refers to the number of nearest neighbors the algorithm considers when making a decision.

This model is imported from sklearn.neighbors

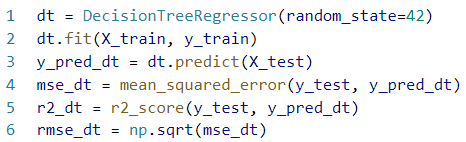
In statement 1, we have created an object called knn\_ob of the class KNeighborsRegressor().

After that, knn\_ob(x\_train, y\_train) here the fit() method trains the KNN regressor on the training data (x\_train and y\_train).

After the model is trained, the predict() method is used to make predictions on the test dataset (x\_test). The result, y\_predict, contains the predicted values for the target variable

An interesting observation was found that, KNN and linear Regression produced the same result. KNN is usually used in classification algorithms and was not expected to perform well for regression task. It also had a R² score of 0.681

**Decision Tree**

****

A Decision Tree is a simple machine learning model used for classification and regression. It works by splitting data into branches based on feature values, creating a tree-like structure. At each step, the algorithm selects the feature that best separates the data, and this process continues until it reaches a final prediction at the leaf nodes.

Decision trees are easy to understand because each path from the root to a leaf represents a clear set of decision rules. However, they can overfit the data, meaning they might not perform well on new data unless techniques like pruning are applied.

This model is imported from sklearn.tree

In the 1st line, we have created an object called dt of the class DecisionTreeRegressor()

After creation of object we have trained the model using the “fit” function and stored the predicted values in y\_pred\_dt.

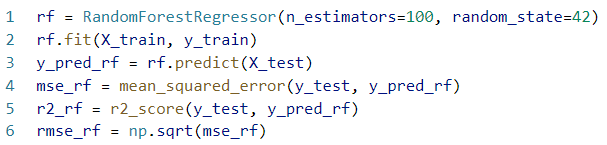
Random\_state = 42 sets a random seed ensures that the random processes (such as selecting data subsets for each tree) are reproducible, so you get the same results every time you run the code.

For this algorithm RMSE value is 7714.78 which is quite less and also the difference from Standard deviation (24424.65) is very high, hence this algorithm is working very well with our dataset.

The R2 score is 0.90 which is good but we have better algorithms.

Random forest classifier is an ensemble method that is used by implementing several decision trees. This will be discussed next.

**Random Forest Classifier (Bagging Ensemble method)**

****

The Random Forest Classifier is a popular machine learning algorithm that operates using the Bagging ensemble method. Bagging, or bootstrap aggregating, involves creating multiple subsets of the original dataset by randomly sampling with replacement. For each subset, a decision tree is trained independently. Random Forest improves upon this by introducing an additional layer of randomness: not only is each tree trained on a different random subset of the data, but each tree is also trained using a random selection of features. When making predictions, the algorithm aggregates the results from all the trees (usually by majority vote) to arrive at a final prediction. This ensemble approach helps reduce overfitting, improves accuracy, and makes Random Forest robust to noisy data and outliers.

This model is imported from sklearn.ensemble

In statement 1, we have created an object called rf of the class RandomForestRegressor()

Here n\_estimators = 100 sets the number of decision trees in the forest to 100. The final prediction will be the average of the predictions from all these trees.

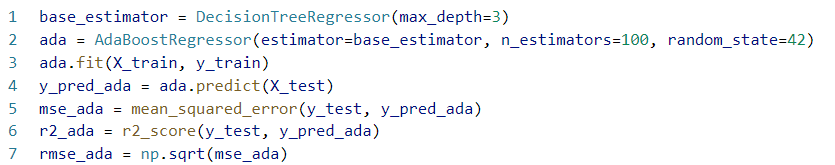
Random\_state = 42 sets a random seed ensures that the random processes (such as selecting data subsets for each tree) are reproducible, so you get the same results every time you run the code.

After creation of object we have trained the model using the “fit” function and stored the predicted values in y\_pred\_rf as previous methods.

For this algorithm RMSE value is 5800.42 which is very less and also the difference from Standard deviation (24424.65) is very high, hence this algorithm is working very well with our dataset.

The R2 score is 0.943 which is very high and very favourable.

**Adaptive Boosting (ADA Boost)**



Adaptive Boosting (AdaBoost) is a sequential ensemble learning method designed to improve the performance of weak classifiers by focusing on the mistakes of previous models. The algorithm starts by training a weak learner, often a decision stump (a shallow decision tree), on the entire dataset. After the first model is trained, AdaBoost assigns higher weights to the data points that were misclassified, making them more important in the next iteration. A new weak learner is then trained on this reweighted dataset, with the goal of correctly classifying the previously misclassified points. This process continues for a specified number of iterations, with each subsequent model focusing more on the difficult cases. Finally, the predictions from all the weak learners are combined through weighted voting, where models with better performance contribute more to the final prediction. This sequential reweighting helps to create a strong classifier by progressively improving the model's accuracy.

The model is imported from sklearn.ensemble and sklearn.tree

In statement 1, we have created an object called base\_estimater of the class DecisionTreeRegressor() which created a model that Adaboost used. And the depth of the tree is 3.

AdaBoostRegressor is initialized with the decision tree regresor as the base estimator.

n\_estimators=100 means that the AdaBoost model will create an ensemble of 100 weak learners (100 decision trees).

random\_state=42 ensures that the results are reproducible, providing consistency in splitting data and random number generation.

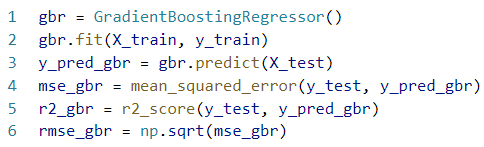
ada.fit(x\_train, y\_t) trains the adaboost model using training data, here x\_train is the features of the training data, and y\_t is the target values (actual outcome) corresponding to x\_train.

After the model is trained, ada.predict(x\_test) is used to make predictions on the test data (x\_test), and the predictions are stored in y\_pred\_daata.

This algorithm has a RMSE of 17912.20 which is very high (highest among all the algorithms we have used) and has a R2 score of 0.462 which is very poor

Ada boost is generally used for classification purposes.

**Gradient Boosting**



Gradient Boosting is a machine learning algorithm used to create a strong predictive model by combining the power of many weak models, usually decision trees. The process works by building trees one at a time, where each new tree tries to correct the mistakes made by the previous trees. Instead of training all the trees independently, gradient boosting trains them sequentially, so each new tree focuses more on the data points that previous trees had trouble predicting.

The name "gradient" comes from how the algorithm uses gradient descent, a mathematical method to minimize the errors of the model. After each tree is added, the algorithm calculates the errors (also called residuals) and fits the next tree to these errors, effectively making the model better at predicting the tricky parts of the data. This process continues for a set number of iterations or until the errors stop improving.

Gradient boosting is very flexible and can produce highly accurate models, but it can also be prone to overfitting if not carefully tuned. However, with proper regularization techniques, it is one of the most powerful algorithms for classification and regression tasks.

This model is imported from sklearn.ensemble

In statement 1, we have created an object called gbr of the class GradientBoostRegressor()

After that we are using the fit() function to train the x\_train and y\_train which builds an ensemble of decision trees.

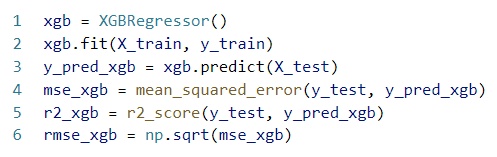
After training, the model predicts the target values for the test set (x\_train). These predicted values (y\_pred\_gbr) represent the model’s estimation of the true values for the test data.

The root mean square error and it is 6998.35 and if we compare it with the standard deviation which is 24424.65, we can tell that the prediction errors are also low compared to the spread of the data.

R2\_gbr is the r^2 score which is 0.92 which indicates that 92% predictions are near accurate after training the model.

But we also have a improved version of gradient boosting which may improve the accuracy score even further, this will be discussed next.

**Extreme Gradient Boosting (XG Boost)**

****

Extreme Gradient Boosting (XGBoost) is an improved version of the gradient boosting algorithm, designed to be faster and more accurate. It works by building a series of small decision trees, where each new tree tries to fix the mistakes made by the previous ones. The idea is to improve the model step by step, learning from its errors.

XGBoost adds several features to make this process more effective. It uses techniques to prevent the model from overfitting, meaning it avoids becoming too tailored to the training data and works better on new data. It also builds trees more efficiently by using only parts of the data at a time, making it faster and less likely to overfit. Another advantage is that it can handle missing data on its own and can build trees in parallel, speeding up the process even further. Because of these optimizations, XGBoost is widely used in tasks where both speed and accuracy are important, such as large-scale data problems.

we have created an object called xgb of the class XGBRegressor()

Then we are training the model on x\_train and y\_train using fit() function.

Once the model is trained, it predicts target values (y\_pred\_xgb) for the test data (x\_train) using the predict() method.

The root mean square error and it is 5503.36 and if we compare it with the standard deviation which is 24424.65, we can tell that the prediction errors are very low compared to the spread of the data.

R2\_xgb is the R2 score and it is 0.95 which indicates that 95% predictions are near accurate after training the model. This is the highest score among all other algorithms.

**3.4 Model accuracy comparison**

**Root mean squared error**

We can see that ADA Boost leads the root mean squared error compared to all other algorithms hence it has the most error.

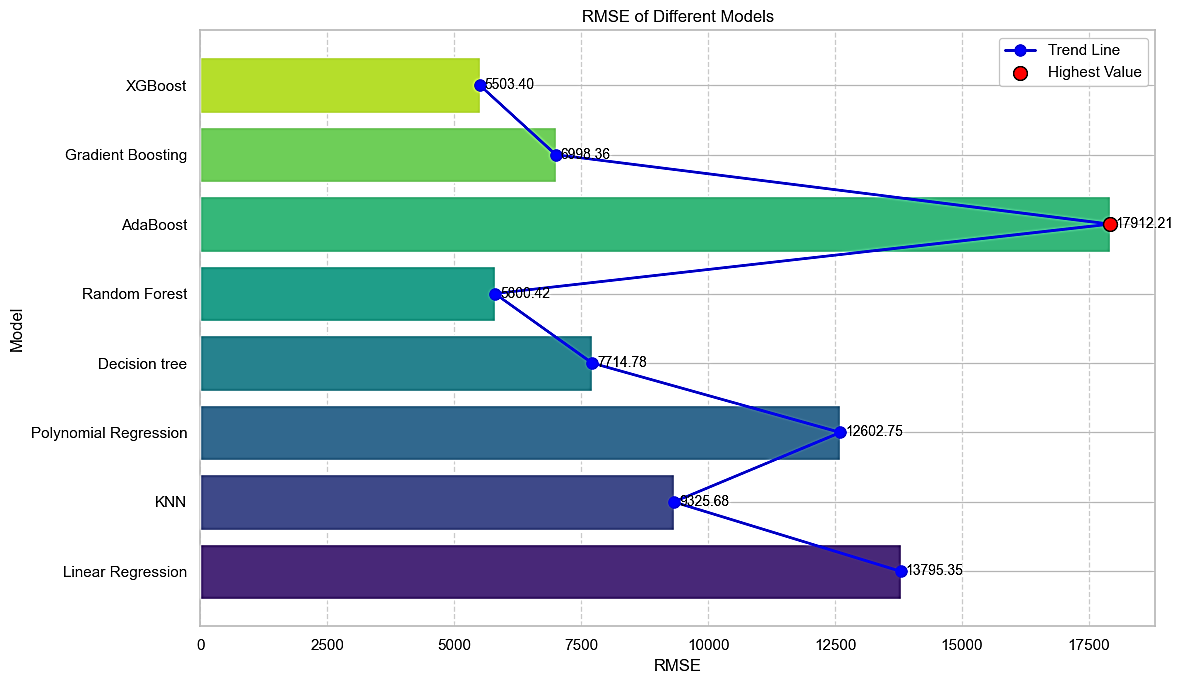
****

Fig 3.4.1 RMSE of different models

**R2 score of each algorithm**

From this metric we have can see that the best performing algorithm is Extreme Gradient Boosting with a score of 0.95 out of 1.

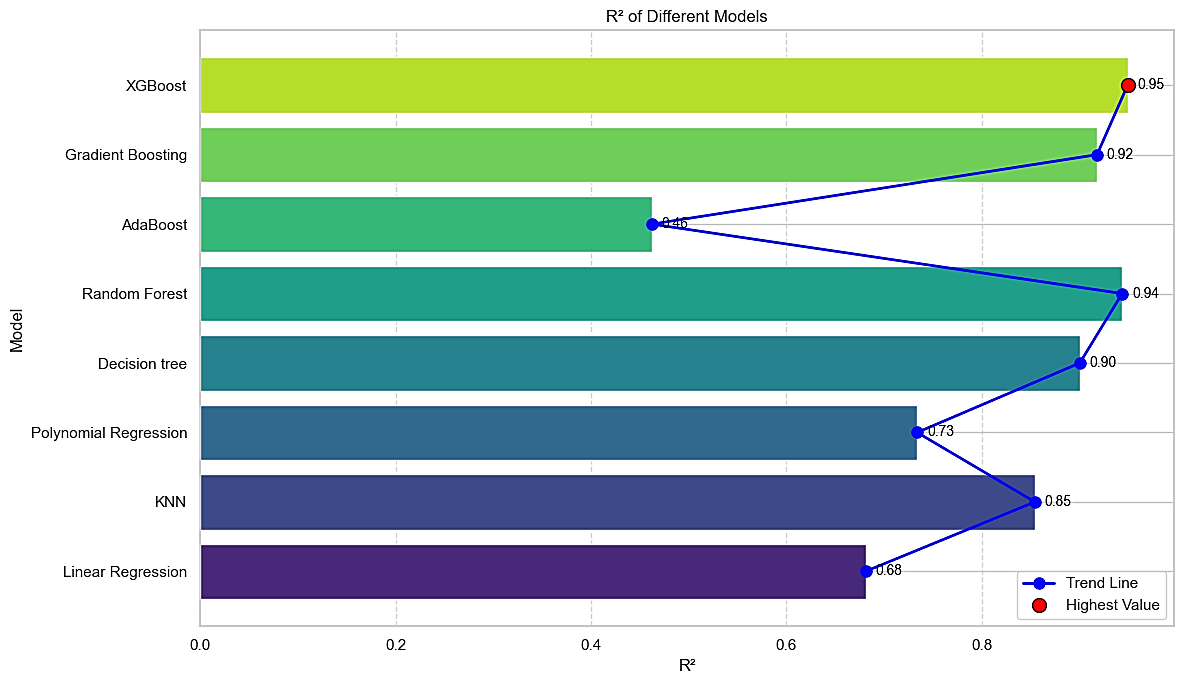


Fig 3.4.2 R2 of Different Models

4. Conclusion

Data analytics plays a crucial role in visualising and extracting key insights from data. From this visualisation we have found that how different mobile brands cater to different customer segment. We have seen how:

* Previously famous brands like Nokia, Gionee have faded away in the inflated competition of budget phones against Motorola.
* Samsung in pursuit of appealing to all customers have lost their average rating to Apple and Google pixel who have significantly high rating as well as high selling price models only.
* Google although has a smaller number of models performs quite well compared to other brands in the premium section.
* Motorola being the most consumer-friendly mobile has the highest discount and less selling price.
* How brands are incorporating high end specification to most of their models to meet the technological demand
* Drastically increase in rating in premium category of mobile phones

Finally, we have applied several machine learning models taking features as label as the “selling price”. We found out that Extreme Gradient boosting algorithm performed best for our given dataset with a R2 score of 0.95 which is quite high and did not have a problem of overfitting.