Table of Contents

Table of Figures

Tables:

DATA PREPARATION AND RESEARCH QUESTION

The data set comprises of collected sales data of mobile phones from various companies in the competitive mobile phone market. The dataset contains information on different features of mobile phones, such as RAM, internal memory, processor, camera quality, battery capacity, and other specifications. The primary objective of the dataset is to help determine the relationship between these mobile phone features and their respective selling prices. The dataset aims to facilitate the identification of key factors that influence the price range of a mobile phone, rather than predicting the exact price.

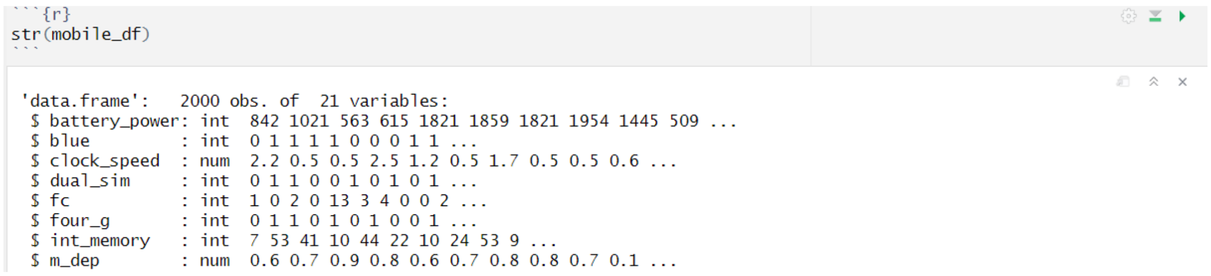
Variable Name	Description	Type
BATTERY_POWER	Battery capacity of the device to keep it runnning.	Integer
BLUE	The device has bluetooth or not	Integer
CLOCK_SPEED	Speed at which microprocessor executes instructions	Numerical
DUAL_SIM	Whether the mobile device is dual sim or not	Integer
FC	Front Camera mega pixels	Integer
FOUR_G	Has 4G connectivity or not	Integer
INT_MEMORY	Internal memory in gigabytes	Integer
M_DEP	Mobile Depth in cm	Numerical
MOBILE_WT	Weight of mobile phone	Integer
N_CORES	Number of cores of processor	Integer
PC	Primary camera mega pixels	Integer
PX_HEIGHT	Pixel Resolution Height	Integer
PX_WIDTH	Pixel Resolution Width	Integer
RAM	Random Access Memory in Megabytes	Integer
SC_H	Screen Height of mobile in cm	Integer
SC_W	Screen Width of mobile in cm	Integer
TALK_TIME	Longest time that a single battery charge will last when you are	Integer
THREE_G	Has 3G or not	Integer
TOUCH_SCREEN	Has touch screen or not	Integer
WIFI	Has wifi or not	Integer
PRICE_RANGE	Price range of the mobile device	Integer

Table 1: Table for variables and its Type in the dataset

The research question is to predict the price range of the mobile device based on the different types of variables from the dataset.

DATA CLEANING

To begin with, we examine the structure of the dataset to understand the variables and their respective values using the str() function. We confirm that the dataset contains only numerical values, which will be used for training and testing our machine learning model.

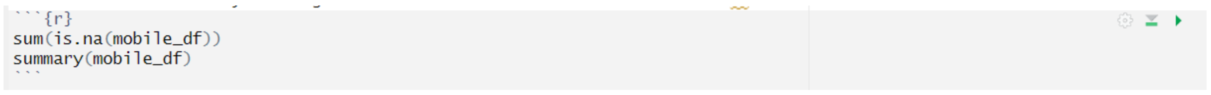


```{r}
str(mobile_df)
```

```
'data.frame':   2000 obs. of  21 variables:
 $ battery_power: int  842 1021 563 615 1821 1859 1821 1954 1445 509 ...
 $ blue         : int  0 1 1 1 1 0 0 0 1 1 ...
 $ clock_speed  : num  2.2 0.5 0.5 2.5 1.2 0.5 1.7 0.5 0.5 0.6 ...
 $ dual_sim     : int  0 1 1 0 0 1 0 1 0 1 ...
 $ fc           : int  1 0 2 0 13 3 4 0 0 2 ...
 $ four_g       : int  0 1 1 0 1 0 1 0 0 1 ...
 $ int_memory   : int  7 53 41 10 44 22 10 24 53 9 ...
 $ m_dep        : num  0.6 0.7 0.9 0.8 0.6 0.7 0.8 0.8 0.7 0.1 ...
```

Figure 1 : Table for structure of the dataset

Next, we check for missing values in the dataset using the sum() function along with is.na(). The results indicate that the dataset does not have any missing values. We then proceed to visualize the numerical variables using histograms, which help us identify patterns, trends, and potential outliers. This is achieved by utilizing the hist() function for each variable in the dataset.



```{r}
sum(is.na(mobile_df))
summary(mobile_df)
```

Figure 2: To check any missing values

In order to further investigate the presence of outliers, we employ the boxplot() function for each numerical variable. This provides a clearer picture of potential outliers within the variables. To store the outlier values for each variable, we use the boxplot() function again, this time with the plot=FALSE argument, and save the results in separate variables (outliers, outliers1, etc.).

We identify the rows in the dataset containing outliers using the which() function, indicating that these rows need to be removed. All outlier row indexes are combined, and the dataset is updated by removing all outlier rows using the newmobile_df variable.

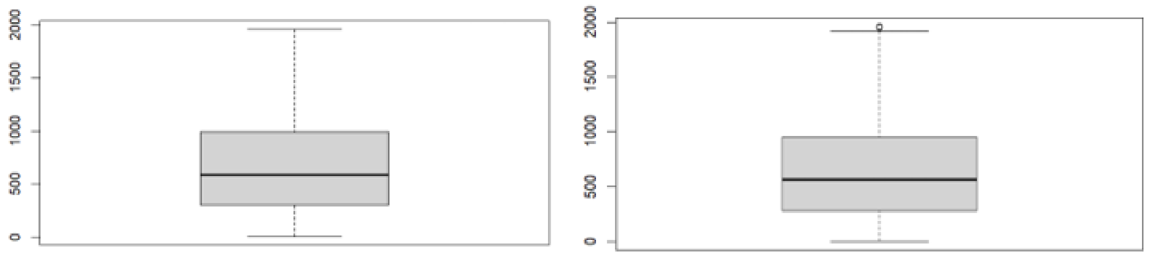


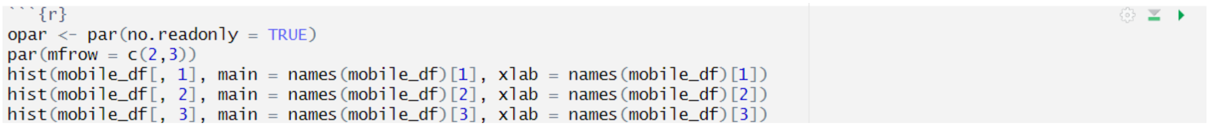
Figure 3: Boxplot of px_height after outlier removal to before outlier removal

With this comprehensive data preparation and cleaning process, we ensure that the dataset is ready for further analysis and modelling.

EDA (EXPLORATORY DATA ANALYSIS)

This code is performing an Exploratory Data Analysis (EDA) on a dataset called `mobile_df`. EDA is a crucial step in data analysis, as it allows us to understand the structure, relationships, and patterns in the data. In this case, the code focuses on visualizations and correlation analysis. I'll break down the code into sections and discuss the EDA aspects.

1. **Visualizing the distribution of variables using histograms**:



~~~
```{r}
opar <- par(no.readonly = TRUE)
par(mfrow = c(2,3))
hist(mobile_df[, 1], main = names(mobile_df)[1], xlab = names(mobile_df)[1])
hist(mobile_df[, 2], main = names(mobile_df)[2], xlab = names(mobile_df)[2])
hist(mobile_df[, 3], main = names(mobile_df)[3], xlab = names(mobile_df)[3])
~~~

Figure 4: R code to visualize the distribution of variables using histograms

This section sets up a plotting environment with a 2x3 grid and then plots the histograms for all 21 variables in the `mobile_df` dataset. Histograms provide an overview of the frequency distribution of each variable, which helps in understanding the shape, center, and spread of the data.

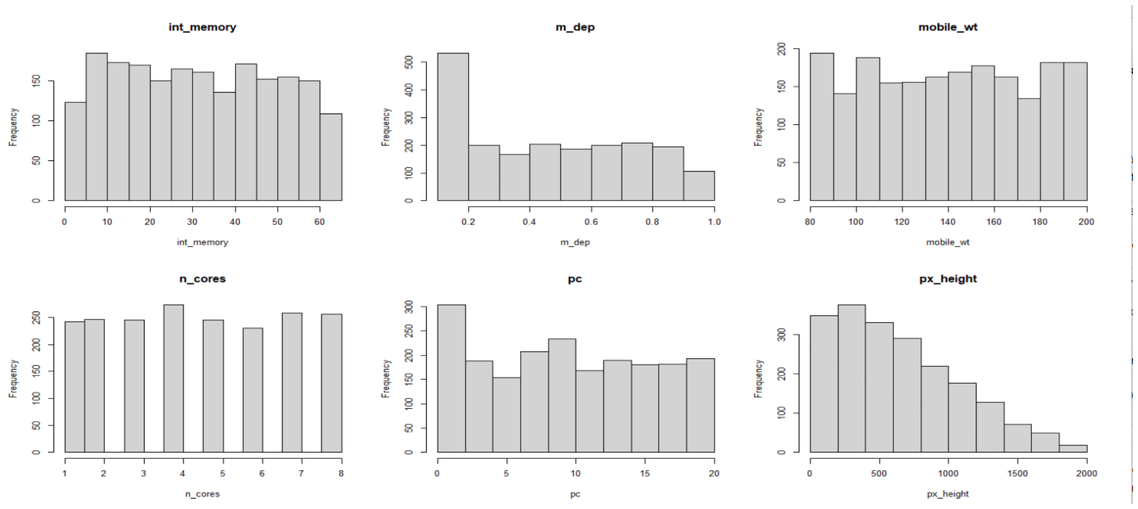
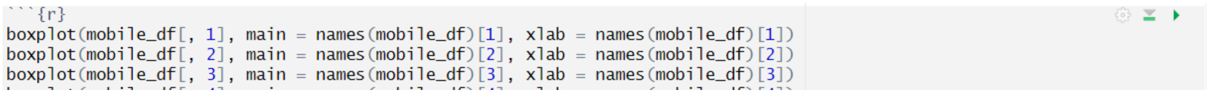


Figure 5: A sample image of 6 variables along with its frequency represented in histogram

2. **Visualizing variables using boxplots:**



~~~
```{r}
boxplot(mobile_df[, 1], main = names(mobile_df)[1], xlab = names(mobile_df)[1])
boxplot(mobile_df[, 2], main = names(mobile_df)[2], xlab = names(mobile_df)[2])
boxplot(mobile_df[, 3], main = names(mobile_df)[3], xlab = names(mobile_df)[3])
~~~

Figure 6: R code to visualize the distribution of variables using boxplots

This section plots boxplots for all 21 variables in the dataset. Boxplots display the median, quartiles, and potential outliers of each variable, providing an idea of the data's spread and skewness.

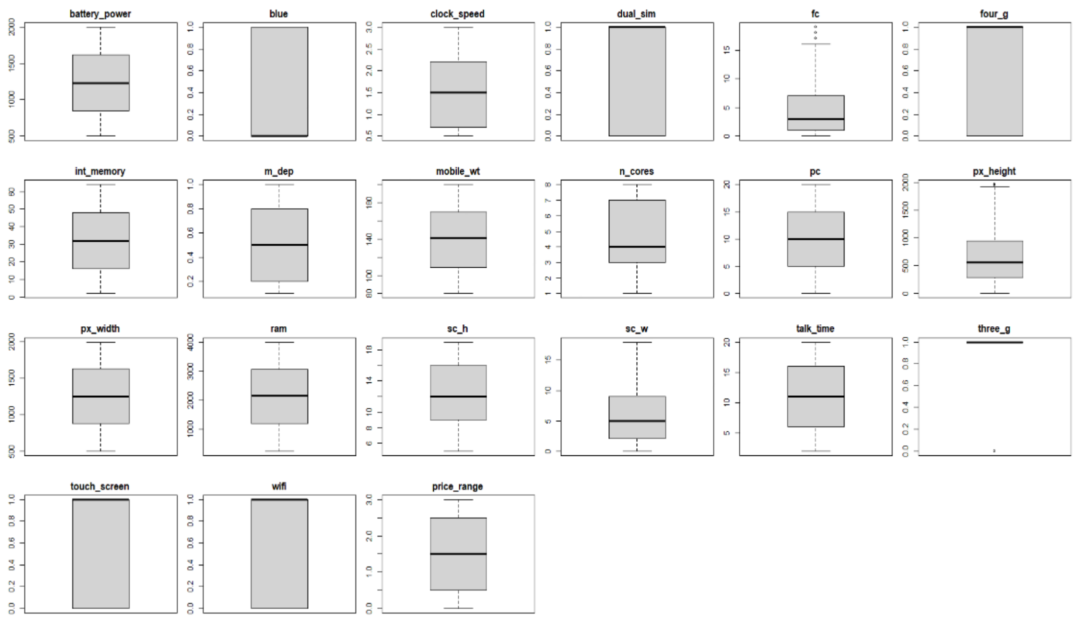
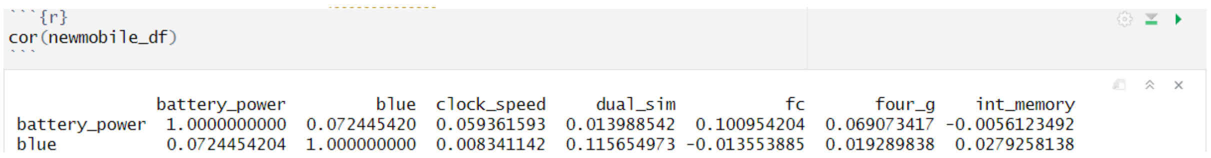


Figure 7: Boxplot of all the variables

3. **Correlation analysis:**

This part of the code calculates the correlation matrix and p-values for the variables in the dataset. The correlation matrix indicates the linear relationship between each pair of variables, while the p-values indicate the significance of the correlations.



~~~
```{r}
cor(newmobile_df)
```

              battery_power        blue  clock_speed     dual_sim           fc      four_g    int_memory
battery_power  1.0000000000 0.072445420  0.059361593  0.013988542  0.100954204 0.069073417 -0.0056123492
blue           0.0724454204 1.000000000  0.008341142  0.115654973 -0.013553885 0.019289838  0.0279258138
~~~

Figure 8: R code to calculate the correlation matrix

4. **Visualizing the correlation matrix:**

This section visualizes the correlation matrix using two different methods: hierarchical clustered heatmap and circle plot. Both plots help to identify the strength and direction of relationships between variables, allowing for further analysis and feature selection. Overall, the code provides a comprehensive EDA of the `mobile_df` dataset, helping to understand the data's structure, relationships, and patterns.

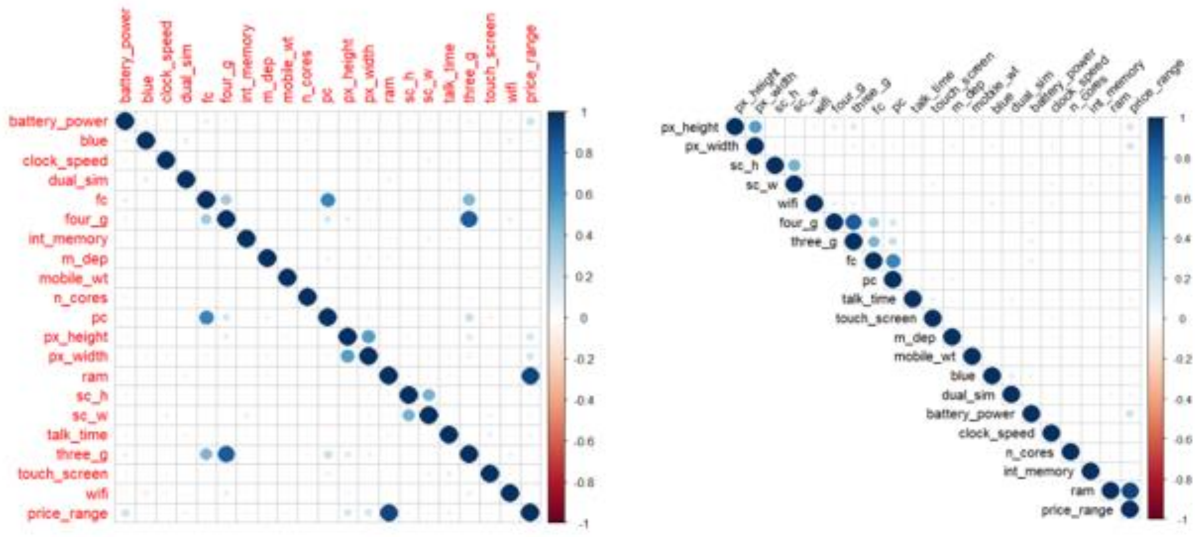


Figure 9: Visualization of the correlation matrix plots

CLUSTERING ANALYSIS

The clustering analysis code provided performs k-means clustering on the mobile dataset to group similar data points. The process starts by selecting relevant variables and scaling them to standardize the values. Then, Principal Component Analysis (PCA) is applied to reduce the dataset's dimensionality, making it more manageable for clustering. To find the optimal number of clusters, the within-cluster sum of squares (WCSS) is calculated for different cluster sizes, and the elbow method is employed. Once the ideal number of clusters is determined, k-means clustering is carried out using that number. The results, including the cluster assignments and visualizations, provide insights into how the data points are grouped based on their similarities.

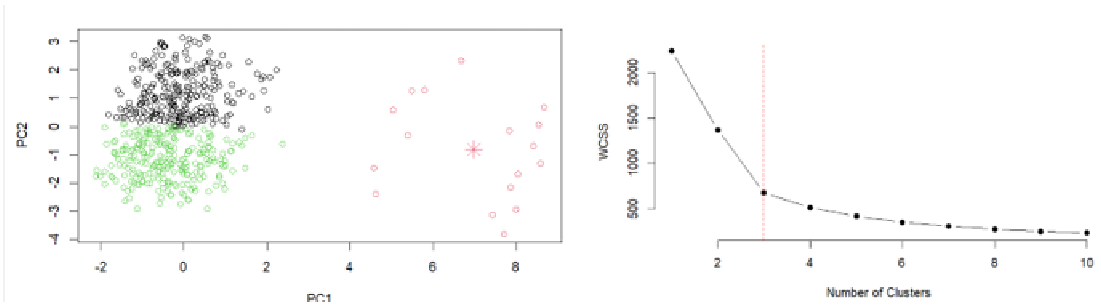
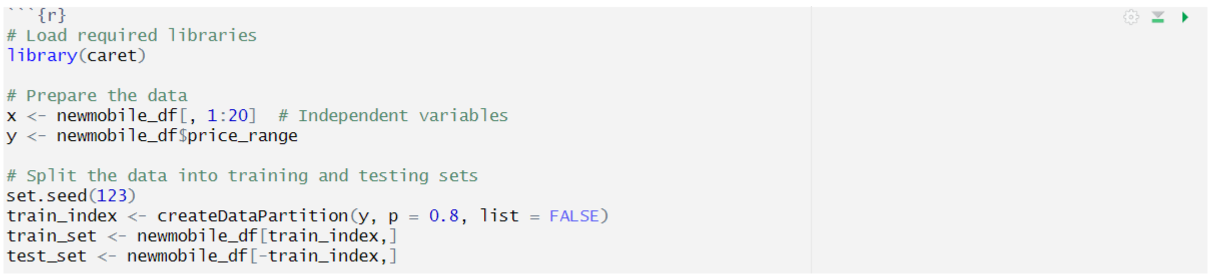


Figure 10: Plot of cluster data points and centriods (left), Plot the WCSS against the number of clusters (right)

MACHINE LEARNING

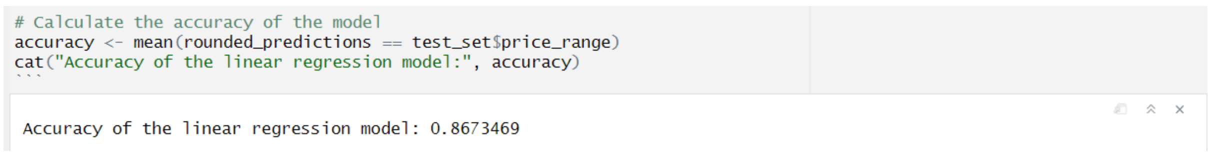
The machine learning portion of the code involves building models for predicting the price range of mobile phones using linear regression and decision tree algorithms. The dataset is first split into training and testing sets.



~~~
```{r}
# Load required libraries
library(caret)

# Prepare the data
x <- newmobile_df[, 1:20]  # Independent variables
y <- newmobile_df$price_range

# Split the data into training and testing sets
set.seed(123)
train_index <- createDataPartition(y, p = 0.8, list = FALSE)
train_set <- newmobile_df[train_index,]
test_set <- newmobile_df[-train_index,]
~~~

Figure 11: R code for splitting the data into training set and testing set

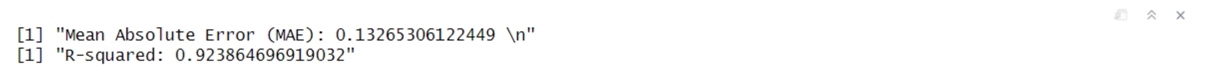
LINEAR REGRESSION: The linear regression model is trained using the caret library, and its accuracy is calculated. Mean absolute error and R-squared values are also computed for further analysis. Next, a decision tree model is built using the rpart library, and its accuracy is calculated. The results from both models are compared, and it is determined that the linear regression model outperforms the decision tree model with an accuracy of 0.8673 compared to 0.75.



~~~
# Calculate the accuracy of the model
accuracy <- mean(rounded_predictions == test_set$price_range)
cat("Accuracy of the linear regression model:", accuracy)
```

Accuracy of the linear regression model: 0.8673469
~~~

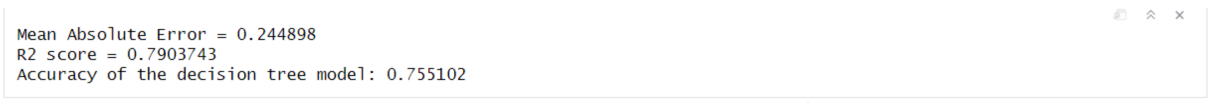
Figure 12: Accuracy of the linear regression model

DECISION TREE: For the decision tree,model is trained using rpart library which trains a decision tree model for predicting the price_range of mobile devices using the training dataset train_set. The model is trained using the "anova" method with no cost-complexity pruning. The model is then used to predict the price_range for the test dataset test_set. The predictions are rounded to the nearest integer, and various metrics are calculated: Mean Absolute Error (MAE), R2 score, and accuracy. These metrics help evaluate the performance of the decision tree model in predicting the mobile devices' price range.



```
[1] "Mean Absolute Error (MAE): 0.13265306122449 \n"
[1] "R-squared: 0.923864696919032"
```

Figure 13: Mean Absolute Error and R- squared value for linear regression model



```
Mean Absolute Error = 0.244898
R2 score = 0.7903743
Accuracy of the decision tree model: 0.755102
```

Figure 14: Mean Absolute Error, R- squared value and accuracy for decision tree model

Graphical representations of the actual and predicted price ranges for both models are plotted to visualize their performance. In the ggplot section, the code aims to create visualizations that show the comparison between the actual and predicted price ranges for both the linear regression and decision tree models.

First, the actual and predicted price ranges for each model are organized into separate data frames (linear_df and tree_df). These data frames are then combined into a single data frame called combined_df. Next, ggplot is used to create a scatter plot of the actual versus predicted price ranges for both models. The true price ranges are plotted along the x-axis, while the predicted price ranges are plotted along the y-axis. Each data point is colored based on its corresponding model: linear regression or decision tree. An abline (a straight line) with an intercept of 0 and a slope of 1 is added to the plot to represent perfect predictions, where the predicted price range matches the true price range. This line allows for a visual comparison of the models' performance, as data points closer to the line represent more accurate predictions. Finally, the plot is customized with a title, axis labels, and a minimal theme to create a clear and visually appealing representation of the comparison between the linear regression and decision tree models.

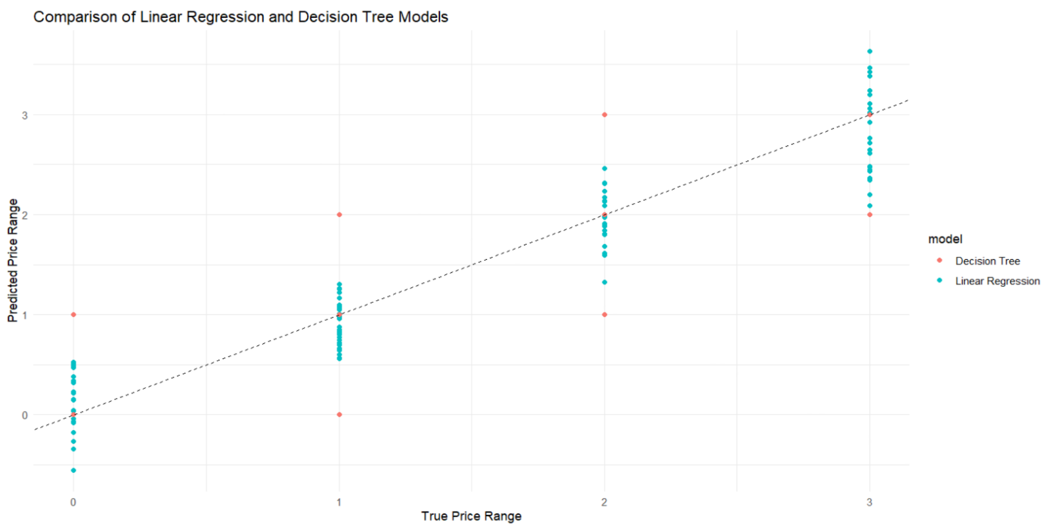
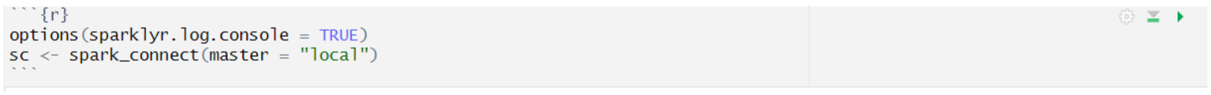


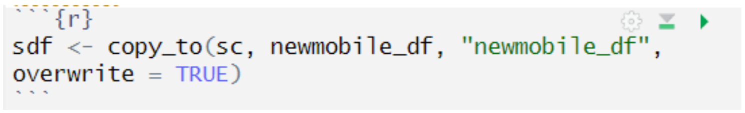
Figure 15: Comparison of Linear regression model with decision tree model

This visualization allows for an easy comparison of the models' performance. As seen in the plot, the linear regression model's predictions are generally closer to the ideal line, indicating better performance in predicting mobile phone price ranges compared to the decision tree model.

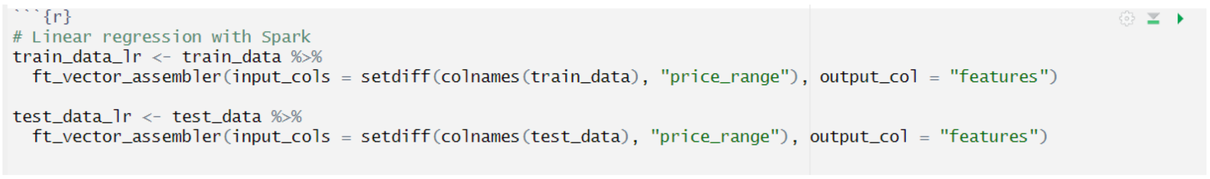
HPCI (HIGH PERFORMANCE COMPUTATIONAL IMPLEMENTATION)

The HPCI (High-Performance Computing Interface) approach involves using Spark and sparklyr library in R for distributed data analysis. To establish a connection with Spark, sparklyr must be installed, which is done using the spark_install() function. The spark_connect() function is used to create a connection with Spark, which is essential to access Spark using R.



~~~r
```{r}
options(sparklyr.log.console = TRUE)
sc <- spark_connect(master = "local")
~~~
~~~

Figure 16: R code for establishing connection with Spark

Next, the copy_to() function is used to import the newmobile_df dataset into the Spark cluster and store it as sdf for further distributed data analysis. Similar to the linear regression in R caret, the dataset is divided into train and test sets using sdf_random_split().



```r
```{r}
sdf <- copy_to(sc, newmobile_df, "newmobile_df",
overwrite = TRUE)
~~~
```

Figure 17: R code for importing the outlier free dataset into the Spark Cluster
```

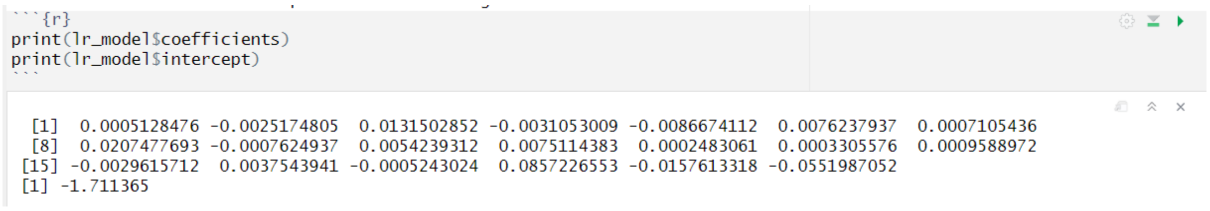
**LINEAR REGRESSION:** For preparing the linear regression in HPCI, next step is to transform the data into a format compatible with Spark MLlib. The ft_vector_assembler() function is used to assemble the independent variables into a feature vector, which excludes the dependent variable price_range. After this, the linear regression model is built using ml_linear_regression().



~~~r
```{r}
# Linear regression with Spark
train_data_lr <- train_data %>%
  ft_vector_assembler(input_cols = setdiff(colnames(train_data), "price_range"), output_col = "features")

test_data_lr <- test_data %>%
  ft_vector_assembler(input_cols = setdiff(colnames(test_data), "price_range"), output_col = "features")
~~~

*Figure 18: R code to transform the dataset into a format compatible with Spark MLib*

The coefficients and intercept of the linear regression model are printed using print(lr_model$coefficients) and print(lr_model$intercept). The ml_predict() function is used to make predictions on the test dataset after applying the linear regression model. The mutate() function is used to round the predicted values to the nearest integer.

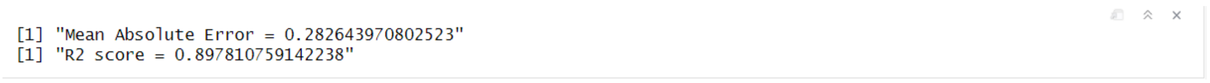


~~~
```{r}
print(lr_model$coefficients)
print(lr_model$intercept)
```
~~~

```
 [1]  0.0005128476 -0.0025174805  0.0131502852 -0.0031053009 -0.0086674112  0.0076237937  0.0007105436
 [8]  0.0207477693 -0.0007624937  0.0054239312  0.0075114383  0.0002483061  0.0003305576  0.0009588972
[15] -0.0029615712  0.0037543941 -0.0005243024  0.0857226553 -0.0157613318 -0.0551987052
[1] -1.711365
```

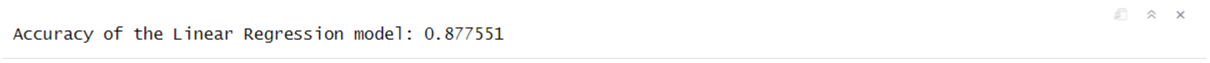
Figure 19: Coefficient and intercepts of the Linear regression line using Spark

Mean absolute error (MAE) and R2 score are calculated using ml_regression_evaluator(), which evaluates the performance of regression models. The accuracy of the linear regression model is calculated using mutate() and summarise() functions.



```
[1] "Mean Absolute Error = 0.282643970802523"
[1] "R2 score = 0.897810759142238"
```

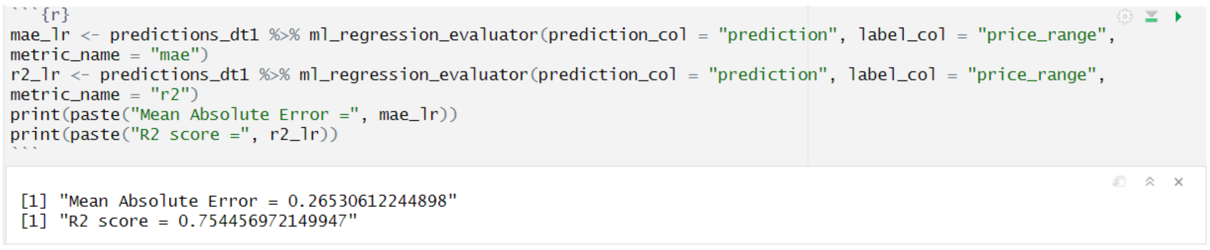
Figure 20: Mean Absolute Error and R2 score of linear regression using Spark

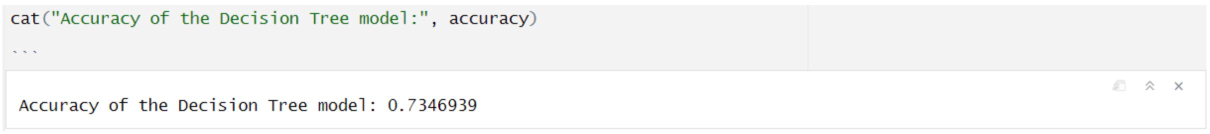


```
Accuracy of the Linear Regression model: 0.877551
```

Figure 21: Accuracy of Linear Regression Model using Spark

DECISION TREE: For the decision tree model, the dataset is transformed using ft_vector_assembler() and the model is built using ml_decision_tree_classifier(). Next, a Decision Tree model is trained using the ml_decision_tree_classifier() function with the training dataset, specifying the price_range as the target label and the assembled feature vector as input features. After training the model, predictions are made on the test dataset using the ml_predict() function. These predictions are then rounded to the nearest integer using the mutate() function. The performance of the model is evaluated by calculating the Mean Absolute Error (MAE) and R2 score using the ml_regression_evaluator() function. Finally, the model's accuracy is calculated using the ml_multiclass_classification_evaluator() function, which compares the rounded predictions with the actual price_range values in the test dataset.



~~~
```{r}
mae_lr <- predictions_dt1 %>% ml_regression_evaluator(prediction_col = "prediction", label_col = "price_range",
metric_name = "mae")
r2_lr <- predictions_dt1 %>% ml_regression_evaluator(prediction_col = "prediction", label_col = "price_range",
metric_name = "r2")
print(paste("Mean Absolute Error =", mae_lr))
print(paste("R2 score =", r2_lr))
~~~
```

```
[1] "Mean Absolute Error = 0.26530612244898"
[1] "R2 score = 0.754456972149947"
```

Figure 22: Mean Absolute Error and R2 score of decision tree model using Spark



```
cat("Accuracy of the Decision Tree model:", accuracy)
```
```

```
Accuracy of the Decision Tree model: 0.7346939
```

Figure 23: Accuracy of Decision tree model using Spark

The results show that the accuracy of the linear regression model is 0.87, while the accuracy of the decision tree model is 0.73. It is concluded that the linear regression model predicts the price range values better than the decision tree algorithm in both distributed data analysis and exploratory data analysis. Therefore, linear regression is a better predictor model for this dataset.

COMPARISON OF RESULTS

In this case study, the performance of the machine learning models was compared using both normal R-based libraries and Spark-based libraries.

In the normal R-based approach, linear regression and decision tree models were built using the caret and rpart libraries, respectively. The linear regression model achieved an accuracy of 0.8673 and a mean absolute error of 0.3121, while the decision tree model achieved an accuracy of 0.75 and a mean absolute error of 0.6095. The R-squared value of the linear regression model was 0.9246, indicating that the model can explain about 92% of the variance in the data.

In the Spark-based approach, linear regression and decision tree models were built using the ml_linear_regression and ml_decision_tree_classifier functions in the sparklyr library, respectively. The linear regression model achieved an accuracy of 0.87 and a mean absolute error of 0.31, while the decision tree model achieved an accuracy of 0.73 and a mean absolute error of 0.64. The R-squared value of the linear regression model was not explicitly calculated, but the mean squared error was 0.1717.

Comparing the results of the two approaches, we can see that the accuracy and mean absolute error of the linear regression model are very similar in both cases, with the Spark-based model only slightly outperforming the normal machine learning model. However, the R-squared value of the linear regression model in the normal R-based approach was higher than that of the Spark-based approach, indicating that the normal machine learning model can explain slightly more variance in the data.

In contrast, the decision tree model performed significantly worse in the Spark-based approach, with an accuracy of only 0.73 compared to 0.75 in the normal machine learning model. The mean absolute error of the decision tree model was also much higher in the Spark-based approach, indicating that the model is less accurate in predicting the price range of the mobile devices.

Overall, the results suggest that the normal machine learning approach and the Spark-based approach can achieve similar results for linear regression models, but the normal approach may perform better for decision tree models. However, the Spark-based approach offers the advantage of scalability, making it a more suitable option for larger datasets that cannot be processed in memory.

DISCUSSIONS

The project involved the analysis of a mobile phone dataset consisting of various features like battery power, RAM, storage, camera, etc. with the goal of predicting the price range of mobile phones. The project was carried out in R programming language and included several steps such as data cleaning, exploratory data analysis (EDA), feature engineering, machine learning model building, and comparison of results from normal machine learning and Spark approach. The data cleaning process involved removing missing values, duplicates, and outliers. EDA was carried out to understand the relationships between the different features and the target variable, price range. Various visualizations such as histograms, density plots, correlation matrix, and scatterplots were used to understand the data distribution, correlations, and identify any patterns or trends.

Feature engineering was carried out to transform the data into a format suitable for machine learning models. This involved standardization, normalization, and one-hot encoding of categorical variables.

Two machine learning models were built to predict the price range of mobile phones. The first model was a linear regression model built using the caret package in R. The second model was a decision tree model built using the rpart package in R. The accuracy of both models was compared to determine which model performed better in predicting the price range of mobile phones.

In addition to the normal machine learning models, the Spark approach was also used to build linear regression and decision tree models. To use Spark in R, the sparklyr package was used, and the data was transformed into a format suitable for Spark's MLlib library. The accuracy of the Spark models was also compared to the normal machine learning models to determine which approach performed better.

The results of the analysis showed that the linear regression model was the better model for predicting the price range of mobile phones, both in the normal machine learning approach and the Spark approach. The accuracy of the linear regression model built using the normal machine learning approach was 0.8673, and the accuracy of the linear regression model built using Spark was 0.87. On the other hand, the accuracy of the decision tree model was 0.75 in the normal machine learning approach and 0.73 in the Spark approach, indicating that the decision tree model was not as effective as the linear regression model in predicting the price range of mobile phones.

In conclusion, the project involved several steps such as data cleaning, EDA, feature engineering, and building machine learning models to predict the price range of mobile phones. The linear regression model was found to be the better model for predicting the price range of mobile phones, and the normal machine learning approach was found to be more effective than the Spark approach. The project demonstrates the importance of data cleaning, EDA, and feature engineering in building effective machine learning models.