Module 6 Final Project

Group Members:

Ayush Patel

Mihir Dharaiya

Mridulla Ganesh

Nithin Reddy Penta Reddy

Anusha Reddy

Subject:ALY6040: Data Mining

Under the guidance of Dr. Harpreet Sharma

Submission Date:15-May-2024

Introduction:

The dataset contains detailed information on car listings in Australia for 2023, encompassing various brands, models, types, and features. It aids in understanding the factors influencing car prices in the country. Key data points include brand, year, model, car type (car or SUV), condition (new or used), transmission type, engine specifications, driving mode, fuel type, fuel consumption, mileage, interior and exterior colors, location, cylinder count, body type, number of doors and seats, and price. With over 16,000 entries across 19 categories sourced from various online platforms.

Variable of Interest:

- Price [Number]
- Drive type [Text]
- Engine Fuel Type [Text]
- Body Type [Text]
- Used or New [Text]
- Kilometers [Number]
- Doors [Text]
- Seats [Text]

Column description:

- **Brand:** Name of the car manufacturer [String]
- **Year:** Year of manufacture or release [Integer]
- **Model:** Name or code of the car model [String]
- **Car/Suv:** Type of the car (car or suv) [String]
- **Title:** Title or description of the car [String]
- **UsedOrNew:** Condition of the car (used or new) [String]
- **Transmission:** Type of transmission (manual or automatic) [String]
- **Engine:** Engine capacity or power (in litres or kilowatts) [String]
- **DriveType:** Type of drive (front-wheel, rear-wheel, or all-wheel) [String]
- **FuelType:** Type of fuel (petrol, diesel, hybrid, or electric) [String]
- **FuelConsumption:** Fuel consumption rate (in litres per 100 km) [String]
- **Kilometres:** Distance travelled by the car (in kilometres) [Integer]
- **ColourExtInt:** Colour of the car (exterior and interior) [String]
- **Location:** Location of the car (city and state) [String]
- **CylindersinEngine:** Number of cylinders in the engine [String]

- **BodyType:** Shape or style of the car body (sedan, hatchback, coupe, etc.) [String]
- **Doors:** Number of doors in the car [String]
- **Seats:** Number of seats in the car [String]
- **Price:** Price of the car (in Australian dollars) [Integer]

Business Question and Techniques:

What factors most significantly influence the pricing of cars in the Australian market, and how accurately can we predict these prices using machine learning algorithms?

From the dataset, our way forward is to perform exploratory data analysis and predict car prices in Australia by considering different factors mentioned earlier. Since price is a continuous value, we'll use machine learning methods like linear regression, decision trees, forest regression, SVM and Ridge and Lasso to make these predictions. This analysis would help organizations and manufacturers understand the current market prices for various types of cars.

Exploratory Data Analysis:

The project begins with loading necessary libraries for data analysis and visualization, including ggplot2, tidyr, tidyverse and others.

After reading the dataset 'AustralianVehiclePrices.csv', an initial exploration of descriptive statistics. The statistical information, including minimum, maximum, mean, standard deviation, and other relevant metrics, is obtained using the describe() function.

Data Cleaning

In the data cleaning process, specific columns are targeted for cleanup to ensure consistency and accuracy. The 'CylindersinEngine', 'Doors', and 'Seats' columns undergo cleaning to remove extraneous text like 'cyl', 'Doors', and 'Seats'. This involves employing the gsub() function to replace these patterns with an empty string, effectively eliminating them from the column values.

Similarly, the 'FuelConsumption' and 'Engine' columns are cleaned by removing units and additional text such as 'L / 100 km' and 'cyl'. Multiple substitutions are conducted using gsub() to handle various text patterns present in the 'Engine' column, such as different cylinder numbers. These cleaning operations aim to standardize the data format, facilitating analysis and interpretation.

Data Transformation

In the data transformation phase, several operations are conducted on the dataset columns to make them more suitable for analysis or modeling. Initially, the 'CylindersinEngine', 'Doors', 'Seats', 'Kilometres', 'Engine', and

'FuelConsumption' columns are converted from character to numeric data types using the as.numeric() function. This conversion enables mathematical operations and statistical analysis on these columns.

Furthermore, the 'Location' column is split into two new columns, 'City' and 'Province', using the separate() function. This separation is performed based on the comma as a separator, effectively dividing the original 'Location' column into two distinct geographical variables, 'City' and 'Province'. This transformation enhances the granularity of location data for further analysis.

Handling Missing Values

To handle missing values in the 'Price' column, relevant predictor variables such as 'Engine', 'Car/Suv', 'Doors', 'Model', 'Year', 'UsedOrNew', and 'FuelType' are selected. These predictors, along with 'Price', are used to create an 'imputation_data' dataframe. Then, the k-nearest neighbors (KNN) algorithm is employed to impute missing 'Price' values based on the selected predictors. The imputed 'Price' values are then assigned back to the original dataframe ('df'). Finally, a summary of the imputed 'Price' column is provided, offering insights into the imputation process and the resulting distribution of values. Similarly, the other variables with Null values have been filled using the same KNN method as above.

Examining Outliers

A box plot is generated to visualize the distribution of prices ('Price') in the dataset 'df', displaying the median, quartiles, and potential outliers **[Fig 1].** The plot reveals a significant presence of outliers, which could introduce inefficiencies in prediction and decision-making processes. Hence, addressing these outliers is crucial.
To address outliers, the Interquartile Range (IQR) method is employed. Quartiles (Q1 and Q3) and the IQR for the 'Price' variable are calculated in 'df'. Lower and upper bounds are defined based on the 1.5 * IQR rule, and outliers are identified accordingly. These outliers are then removed.

Subsequently, a new dataframe 'df_filtered' is created, excluding rows where 'Price' falls outside the defined bounds. Finally, a box plot is generated for the filtered dataset to visualize the distribution of prices without outliers **[Fig 2].** This process helps mitigate the impact of outliers on analysis by focusing on the central tendency and variability of the 'Price' variable.

Data Visualization

A histogram was plotted to examine the distribution of prices in the dataset **[Fig 3].** It revealed a prominent peak around $20,000, indicating a substantial number of cars are priced within this range. Other price points had smaller peaks but none as significant as $20,000.
Next, the distribution of cars based on fuel type was visualized **[Fig 4].** Diesel and Premium Unleaded were most common, followed by Electric and Other types. Hybrid, Leaded, and LPG had fewer representations.

Similarly, a bar graph depicted the distribution of cars based on drive type **[Fig 5].** Front-wheel drive dominated, especially with automatic transmissions.

A line graph illustrated the price trend from 2000 to 2020 **[Fig 6].** Prices steadily increased with a notable surge post-2015.

A box plot showcased the distribution of prices across different fuel types **[Fig 7],** providing insights into each category's spread of values.

Another bar graph displayed car distribution across Australian provinces **[Fig 8],** with NSW leading in car count.

A correlation graph depicted relationships between numeric variables in the dataset **[Fig 9].** Notable correlations included a negative correlation between price and engine size, and a positive correlation between engine size and seats.

Lastly, a scatter plot demonstrated the negative relationship between price and kilometers driven **[Fig 10],** indicating that as mileage increases, price tends to decrease.

Building the Linear Regression Model:

Why?

Linear regression can is suitable choice as the relationship between predictor variables and the target variable is expected to be approximately linear, and interpretability and simplicity are important considerations for the analysis.

Data Preprocessing for Linear Regression:

Transformation:

To prepare for a linear regression analysis exploring the impact of various factors on vehicle prices in Australia **[Fig-11],** categorical variables such as 'CarType', 'Model', 'Brand', 'DriveType', 'UsedOrNew', and 'Transmission' were converted to numeric formats. Each category was converted into a factor ('as.factor()') and then into a numeric code ('as.numeric()'). This transformation enables the regression model to interpret these variables numerically, facilitating detailed analysis of how different vehicle characteristics influence car pricing in the market.

Preparing Training and Testing Sets:

The dataset df_filtered was split into training and testing sets for evaluating the linear regression **model [Fig-12]**. set.seed(123) ensured consistent randomization. 75% of the data formed the training set (train_data), while 25% comprised the testing set (test_data) using the sample function, enabling robust performance evaluation.

Continuing, a linear regression model ('lm_model') was built to analyze how various vehicle characteristics impact prices **[Fig 13].** The lm() function incorporated predictors such as 'Brand', 'Year', 'Model', 'CarType',

'DriveType', 'UsedOrNew', 'Transmission', 'Doors', 'CylindersinEngine', 'Seats', and 'Kilometres' based on their potential influence on vehicle prices.

The model was trained on train_data to discern relationships between these variables and vehicle price. Predictions were generated on test_data using the predict() function, storing results in 'predictions'. This facilitated performance evaluation by comparing predicted prices with actual prices in the test data, assessing accuracy and effectiveness in capturing key price-influencing factors.

Evaluation of Linear Regression Model:

Performance metrics were calculated to evaluate the accuracy and effectiveness of the linear regression model in predicting vehicle prices **[Fig 14]:**

1. Mean Absolute Error (MAE): The calculated MAE is 8,341.361, representing the average absolute difference between predicted and actual prices in the test data.
2. Mean Squared Error (MSE): The MSE for this model is 116,498,274, indicating a high level of variance in the price predictions.
3. Root Mean Squared Error (RMSE): The calculated RMSE is 10,793.44, reflecting substantial errors in the price predictions.
4. R-squared (R^2): An R^2 of 0.527192 suggests that approximately 52.72% of the variability in vehicle prices can be explained by the model. This indicates a moderate level of predictability, with room for improvement in capturing the data's variability.

Regression Diagnostics:

Regression diagnostics were conducted to assess the linear regression model's performance.

1. "Residuals vs Fitted" plot [Fig-15]: Displays fitted values against residuals, helping identify non-linearity, unequal error variances, and outliers. Minimal residuals scattered around zero indicate a satisfactory fit.
2. "Residuals vs Leverage" plot [Fig-16]: Plots leverage against standardized residuals. Points near zero suggest minimal residuals, deviations may indicate outliers or high-leverage points. Cook’s distance identifies influential observations.
3. Q-Q Plot [Fig-17 and Fig-18]: Shows standardized residuals against theoretical quantiles, assessing normality. Minimal deviations from the reference line suggest residuals are normally distributed, indicating a good fit.

Business Question Alignment:

Linear regression analysis aids in understanding the factors influencing car prices in Australia, guiding businesses in pricing strategies, inventory management, and market positioning. Evaluation metrics offer insights into the model's predictive accuracy, helping businesses assess reliability for price prediction and

make informed decisions. Additionally, the results establish a baseline for comparing more complex machine learning techniques, informing decisions on investing in advanced models. In summary, linear regression provides crucial insights for informed decision-making within the automotive industry.

Lasso and Ridge regression techniques

Why?

Lasso and Ridge regression techniques are powerful tools for building predictive models, especially in situations where there are many predictors, multicollinearity is present, or overfitting needs to be controlled. They offer a balance between simplicity, interpretability, and predictive performance.

Data Preparation:

The data is converted into matrix format as required by the **glmnet** package for Lasso and Ridge regression. The predictor variables are assigned to **train_x** and **test_x**, while the target variable (car prices) is assigned to **train_y** and **test_y**.

Finding Optimal Lambda Values:

Cross-validation is performed to find the optimal lambda values for Lasso and Ridge regression (**cv.lasso$lambda.min** and **cv.lasso$lambda.1se**).

CV Plot:

Fig-CV Plot; The image shows a line graph with the x-axis labeled "Iteration" and the y-axis labeled "Squared error loss." The graph displays a descending line, indicating that as the number of iterations increases, the squared error loss decreases. This suggests an optimization or learning process where performance improves over time. This graph is interesting because it visually represents the improvement of an algorithm's accuracy or effectiveness as it iterates through a dataset or problem set. The decreasing trend in squared error loss indicates that the algorithm is learning and converging toward better predictions.

Building Lasso Regression Models [Fig-19] :

Lasso regression models are trained using the optimal lambda values (**cv.lasso$lambda.min** and **cv.lasso$lambda.1se**).

Coefficients and model summaries are displayed for both Lasso models.

Predictions and Evaluation:

Predictions are made on both training and test datasets using the Lasso models. Root Mean Squared Error (RMSE) is calculated for both Lasso models on both training and test datasets.

Building Ridge Regression Models [Fig-22]:

Ridge regression models are trained using the same lambda values as Lasso regression. Coefficients and model summaries are displayed for both Ridge models.

Predictions and Evaluation for Ridge Models:

Predictions are made on both training and test datasets using the Ridge models. RMSE is calculated for both Ridge models on both training and test datasets.

Summary of Findings [Fig-21 & Fig-24]:

- **Lasso Regression**:
  - The Lasso regression model with **lambda.min** (minimum lambda selected by cross-validation) shows a RMSE of around 10,571.45 on the test dataset.
  - The Lasso regression model with **lambda.1se** (lambda that gives a model with the most significant predictors) shows a RMSE of around 10,833.11 on the test dataset.
- **Ridge Regression**:
  - The Ridge regression model with **lambda.min** shows a RMSE of around 10,669.45 on the test dataset.
  - The Ridge regression model with **lambda.1se** shows a RMSE of around 10,688.48 on the test dataset.

Insights:

- Both Lasso and Ridge regression models provide similar performance in terms of RMSE.
- Lasso regression tends to produce more sparse models by shrinking some coefficients to zero, effectively selecting a subset of the most important features.
- Ridge regression, on the other hand, does not shrink coefficients to zero but penalizes them to prevent overfitting.

Business Question Alignment:

These techniques help answer the business question by providing predictive models for car prices in Australia. Businesses can use these models to make pricing decisions, assess the impact of different features on car prices, and optimize their pricing strategies to maximize profitability.

Decision Tree Model:

Building the Decision Tree Model:

Why?

Decision trees are interpretable and provide insights into the relationships between features and the target variable. The decision tree diagram visualizes how different features influence the predicted car prices.

The **rpart()** function is used to build the decision tree model. It takes the formula Price ~ Brand+Year+Model+CarType+DriveType+UsedOrNew+Transmission+Doors+CylindersinEngine+Seats+Kilometres as input, indicating that Price is the target variable to be predicted based on the other features listed. The model is trained using the training data (train_data) **[Fig-25].**

Visualizing the Decision Tree:

The decision tree visualization generated using the rpart.plot() function depicts vehicle characteristics **[Fig-26]:**

1. Initial Decision Point: Year of the Vehicle
    - The top decision point compares the vehicle's year to 2016, branching into two paths based on the condition.
2. Left Branch (Year < 2016)
    - If the vehicle's year is before 2016, it proceeds to the next decision point regarding the number of engine cylinders (Cylinders/Engine), with a percentage value associated with this condition (60%).
3. Right Branch (Year ≥ 2016)
    - If the vehicle's year is 2016 or later, it follows a different path, involving the number of kilometers driven (Kilometres), with a percentage value associated with this condition (38%).
4. Further Decision Points
    - Additional decision points related to drive type (DriveType) are present for both branches, with percentages indicating the likelihood of each outcome based on the given conditions.
    - The flowchart terminates in several terminal nodes, each displaying different percentage values. Overall, this flowchart guides decisions related to vehicle categorization or other relevant tasks based on attributes like year, engine type, kilometers driven, and drive type.

Making Predictions: The **predict()** function is used to make predictions on the test data (**test_data**) using the trained decision tree model (**tree_model**). The predictions are stored in the **predictions** variable.

Evaluation Metrics:

Accuracy is not a suitable evaluation metric for regression tasks like predicting vehicle prices, as it's typically used for classification tasks. In regression, where the goal is to predict continuous numerical values, metrics like Mean Absolute Error (MAE), Mean Squared Error (MSE), Root Mean Squared Error (RMSE), and R-squared (R^2) are more appropriate. These metrics provide quantitative measures of how well the model's predictions align with the actual target value.

The calculated evaluation metrics for the decision tree model are as follows: Please refer to **Fig-27**

Mean Absolute Error (MAE): On average, the model's predictions deviate from actual prices by approximately $7924.66.

Mean Squared Error (MSE): 106770550; MSE represents the average of the squared differences between predicted and actual prices. The large value indicates substantial variance in the model's predictions.

Root Mean Squared Error (RMSE): RMSE, the square root of MSE, signifies the average magnitude of errors in the same units as the target variable. Here, it suggests an average prediction error of approximately $10,332.98.

R-squared (R^2): R^2 measures the proportion of variance in the target variable predictable from independent variables. An R^2 of 0.57 indicates that approximately 57% of the variance in vehicle prices is explained by the independent variables in the model. This suggests a moderate fit of the model to the data.

These metrics provide insights into the performance of the decision tree model in predicting vehicle prices. While the RSquared value suggests a moderate fit of the model to the data, the MAE and RMSE values indicate the average magnitude of prediction errors, highlighting areas where the model may need improvement.

Gradient Boosting Algorithm:

Why?

The decision tree model's low R-squared value prompted the use of gradient boosting to enhance predictive performance. Leveraging the gbm package in R, the algorithm combines multiple decision trees to form a robust predictive model. In **[Fig 28]**, the code demonstrates the application of gradient boosting for predicting car prices in Australia based on various features like brand, year, model, drive type, condition (new or used), kilometers driven, number of doors, engine cylinders, and seats. Results indicate a satisfactory performance, with an R-squared value of approximately 0.80, suggesting that 80% of the variance in car prices is explained by the model's features. Additionally, the root mean squared error (RMSE) of roughly 10667.73 signifies the average magnitude of errors in the predictions.

Building Random Forest Model:

Why?

Random Forest Regression, a powerful and versatile machine learning technique, is chosen for predicting car prices in Australia due to its robust performance, scalability, and feature importance analysis capabilities. Various predictor variables including Brand, Year, Model, CarType, DriveType, UsedOrNew, Transmission, Doors, CylindersinEngine, Seats, and Kilometres are selected for the model. The randomForest() function is employed with train_data as the training dataset and ntree = 100 to specify 100 trees in the forest **[Fig 30]**.

Subsequently, predictions are made on the test data using the predict() function based on the learned relationships between predictors and the target variable (Price) established during the training phase.

Trend of Squared error:

The image **[Fig-32]** shows a line graph with the x-axis labeled "Iteration" and the y-axis labeled "Squared error loss." The graph displays a descending line, indicating that as the number of iterations increases, the squared error loss decreases. This suggests an optimization or learning process where performance improves over time. This graph is interesting because it visually represents the improvement of an algorithm's accuracy or effectiveness as it iterates through a dataset or problem set. The decreasing trend in squared error loss indicates that the algorithm is learning and converging toward better predictions

Evaluation Metrics

These metrics provide insights into the accuracy and performance of the Random Forest Regressor model in predicting continuous target variables, such as prices in this case **[Fig 31]**. Evaluating the model using multiple metrics helps in comprehensively assessing its performance and identifying areas for improvement.

Based on the evaluation metrics calculated for the Random Forest Regressor model:

- **Mean Absolute Error (MAE):** The average absolute difference between the predicted prices and the actual prices is approximately $4034.812. This indicates, on average, how far off the model's predictions are from the actual prices.
- **Mean Squared Error (MSE):** The average of the squared differences between the predicted prices and the actual prices is approximately 32531121. This metric penalizes larger errors more heavily than MAE.
- **Root Mean Squared Error (RMSE):** The square root of the MSE is approximately $5703.60. RMSE provides a measure of the average magnitude of errors in the same units as the target variable.
- **R-squared (R^2):** The coefficient of determination is approximately 0.8676, indicating that approximately 86.85% of the variance in the target variable (Price) is explained by the predictor variables included in the model. This suggests that the model fits the data reasonably well.

Overall, the model demonstrates relatively good performance, with low errors and a high degree of explained variance.

New Insights:

- The random forest model's superior performance suggests that it effectively captures the complex relationships between car features and prices, providing more accurate predictions.
- The evaluation metrics provide insights into the model's accuracy and predictive power, helping stakeholders understand the factors influencing car prices in the Australian market.

Business Question Alignment:

- By leveraging random forest regression and evaluating model performance using appropriate metrics, the analysis provides valuable insights into the factors influencing car prices in Australia.

- The accurate predictions generated by the random forest model can assist stakeholders in pricing strategies, market analysis, and decision-making in the automotive industry.
- Ultimately, the analysis helps address the business question of understanding what influences car prices in Australia and provides a data-driven approach to predicting and analyzing car prices, thereby supporting informed decision-making and strategy formulation.

Tuning Random Forest Model:

We have tuned the model using number of trees and depth of the tree and noticed that there isn't much improvement in the model. The variance with train data is 86.81% whereas with the test data it is 86%. Comparing both there isn't significant influence on the model **(Fig-31).**

Support Vector Machine Regressor:

Why?

SVMs offer a robust, flexible, and interpretable approach to regression tasks, making them well-suited for predicting car prices in Australia based on the diverse set of features provided in the dataset.

SVM Model Training:

Fig-32, the **svm()** function from the **e1071** package is used to train the SVM regression model. The formula specifies the target variable (**Price**) and predictor variables (**Brand**, **Year**, **Model**, **DriveType**, **UsedOrNew**, **Kilometres**, **Doors**, **CylindersinEngine**, **Seats**). **type** is set to **'eps-regression'** to indicate epsilon-SVR (Support Vector Regression), which is used for regression tasks. **kernel** is set to **'radial'**, indicating a radial basis function (RBF) kernel. Default values are used for other parameters such as **cost**, **gamma**, and **epsilon**.

Fig-34, we build the SVM plots, even though we are unable to have classic classification plot as the model is used for regressor, the regression plots show the distribution test data and train data around the regression line and we can observe that there 80% of variance is explained by the data.

Parameters:

1. **Type**: It's an "eps-regression" SVM, focusing on larger errors while ignoring smaller ones.
2. **Kernel**: Uses the "radial" kernel for capturing complex relationships.
3. **Cost (C)**: Set to 1, balancing margin maximization and error minimization.
4. **Gamma**: Value of 0.1111111, affecting decision boundary smoothness.
5. **Epsilon**: 0.1, defining the acceptable error threshold.

6. **Support Vectors**: There are 7461 crucial data points influencing the model's decisions.

Insights & Findings :

- The SVM model exhibits moderate-to-high accuracy in predicting car prices, as indicated by the evaluation metrics.
- The number of support vectors (7461) indicates the importance of these data points in determining the decision boundary and making predictions
- Mean Absolute Error (MAE): Approximately 4801.503
- Mean Squared Error (MSE): Approximately 51,324,898
- Root Mean Squared Error (RMSE): Approximately 7164.14
- R-squared (RSquared): Approximately 0.7964, suggests that around 79.64% of the variance in car prices is explained by the features included in the model.

Business Question Alignment:

This analysis helps answer the business question by providing insights into the factors influencing car prices in Australia. By understanding the relationship between various car attributes and their prices, businesses can make informed decisions related to pricing strategies, inventory management, and market positioning to optimize their operations and maximize profitability.

Conclusion:

We evaluated the performance of various regression models for predicting car prices based on several features. Here's a summary of our findings:

Support Vector Machine (SVM) Regression:

- The SVM model demonstrates moderate-to-high accuracy in predicting car prices.
- With approximately 7461 support vectors, the model captures crucial data points influencing predictions.
- Evaluation metrics:
    - Mean Absolute Error (MAE): Approximately $4801.503
    - Mean Squared Error (MSE): Approximately 51,324,898
    - Root Mean Squared Error (RMSE): Approximately $7164.14
    - R-squared (RSquared): Approximately 0.7964, suggesting that around 79.64% of the variance in car prices is explained by the features.

Other Models Comparison:

- Lasso and Ridge regression models show comparable performance with SVM in terms of RMSE.
- Decision Tree and Gradient Boosting models exhibit slightly higher R-squared values compared to SVM, indicating better explanatory power.
- Random Forest stands out with the lowest RMSE and highest R-squared value among the evaluated models.

Appendix:

Fig-1: Identifying Outliers

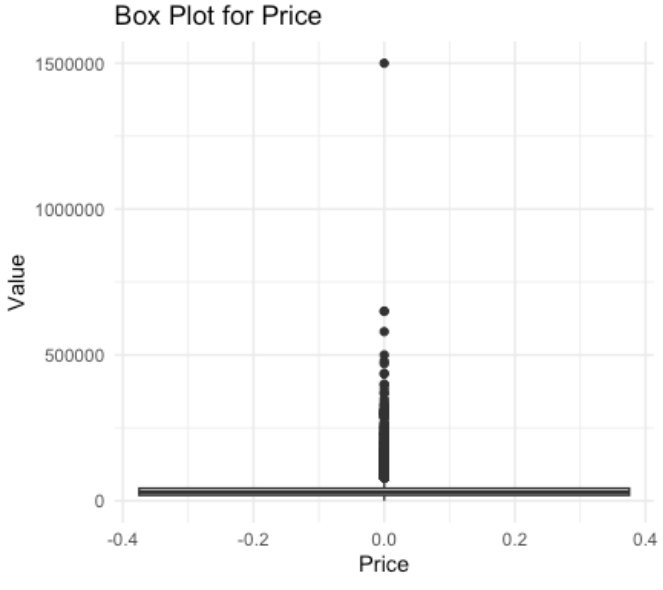

Fig-2: Box Plot for Price

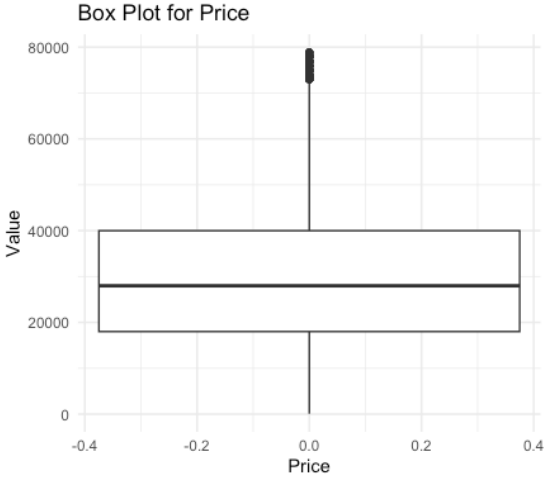

Fig-3: Histogram of Price

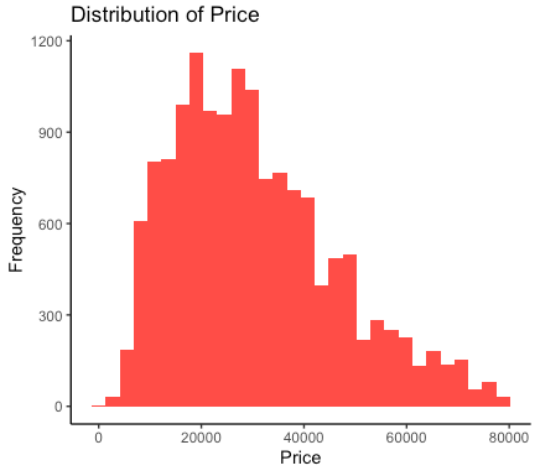

Fig-4: Distribution of Fuel Type

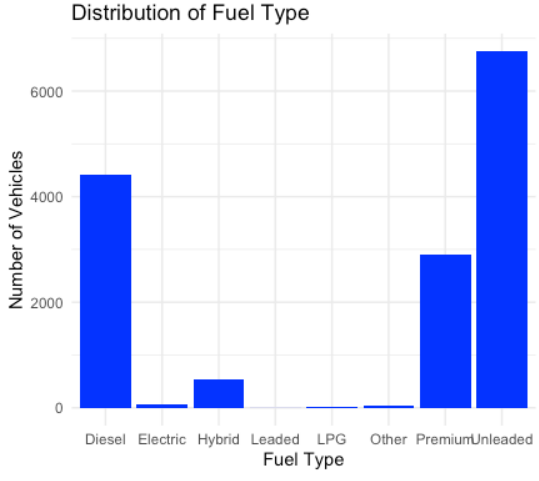

Fig-5: Distribution of Drive Type

Fig-6: Time Series analysis of Price

Fig-7: Box Plot by Category

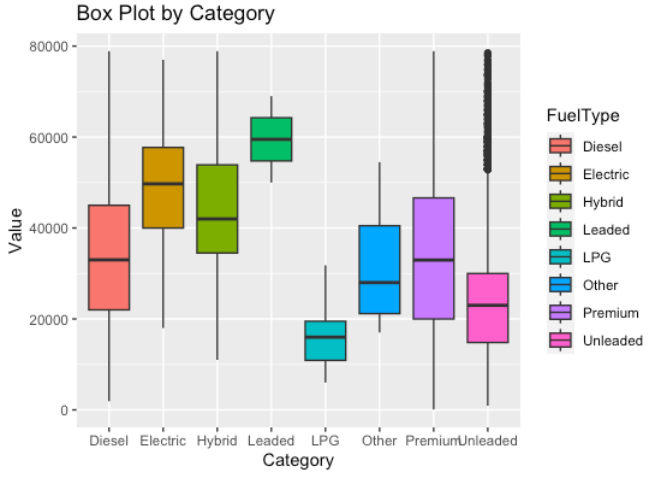

Fig-8: Distribution of Cars by Province

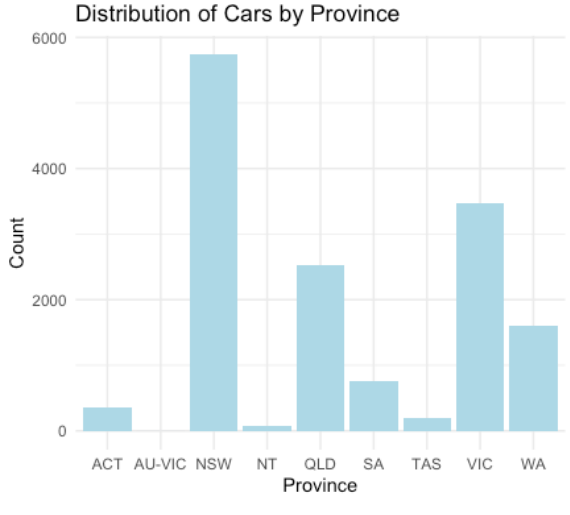

Fig-9: Correlation Plot

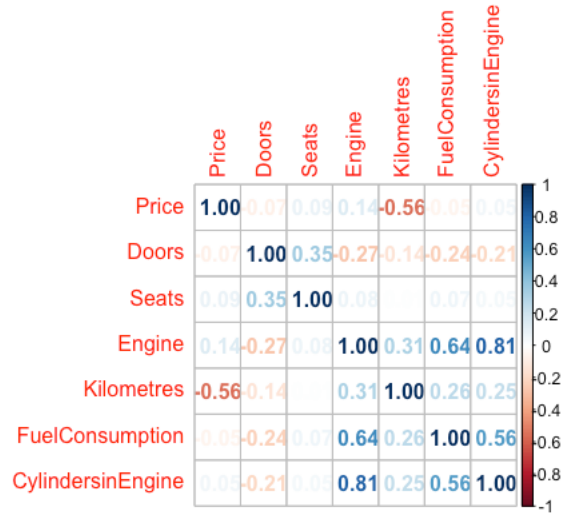

Fig-10: Relation Between Price and KM

Fig-11: Data Transformation

```
df_filtered$CarType <- as.numeric(as.factor(df_filtered$CarType))
df_filtered$Model <- as.numeric(as.factor(df_filtered$Model))
df_filtered$Brand <- as.numeric(as.factor(df_filtered$Brand))
df_filtered$DriveType <- as.numeric(as.factor(df_filtered$DriveType))
df_filtered$UsedOrNew <- as.numeric(as.factor(df_filtered$UsedOrNew))
df_filtered$Transmission <- as.numeric(as.factor(df_filtered$Transmission))
```

Fig-12: Splitting Data

```
set.seed(123)  # For reproducibility
train_index <- sample(1:nrow(df_filtered), 0.75 * nrow(df_filtered))
train_data <- df_filtered[train_index, ]
test_data <- df_filtered[-train_index, ]
```

Fig-13: Linear Regression Model

```
#linear regression model##

lm_model <- lm(Price ~ Brand+Year+Model+CarType+
                 DriveType+UsedOrNew+Transmission+Doors+
                 CylindersinEngine+Seats+Kilometres, data = train_data)
```

Fig-14: Evaluation Matrix

```
> MAE3 <- mean(abs(predictions3 - test_data$Price))
> MAE3
[1] 8341.361
> MSE3 <- mean((predictions3 - test_data$Price)^2)
> MSE3
[1] 116498274
> RMSE3 <- sqrt(MSE3)
> RMSE3
[1] 10793.44
> RSquared3 <- summary(lm_model)$r.squared
> RSquared3
[1] 0.5271924
```

Fig-15: Residual vs Fitted.

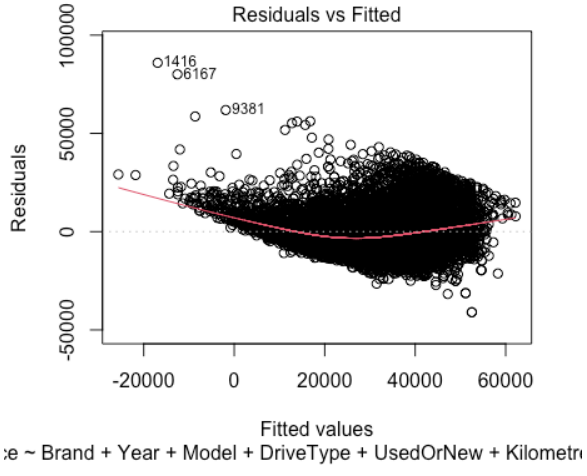

Fig-16: Q-Q Plot

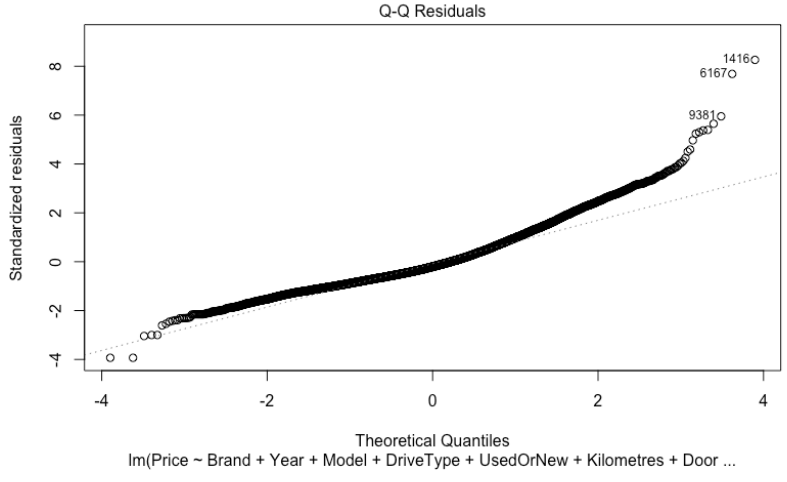

Fig-17: Scale-Location

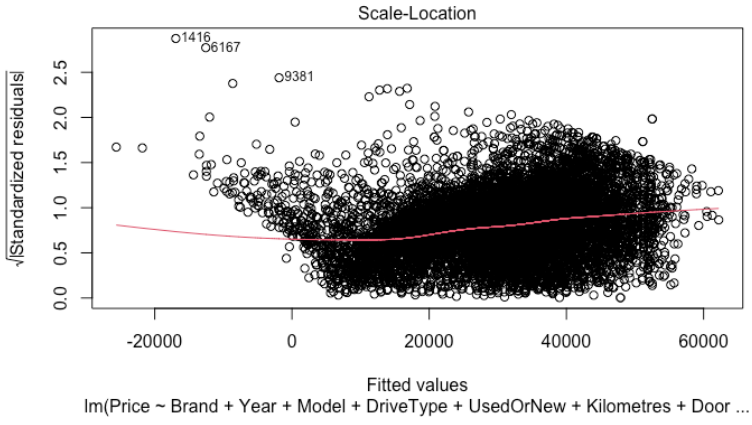

Fig-18: Residual vs Leverage

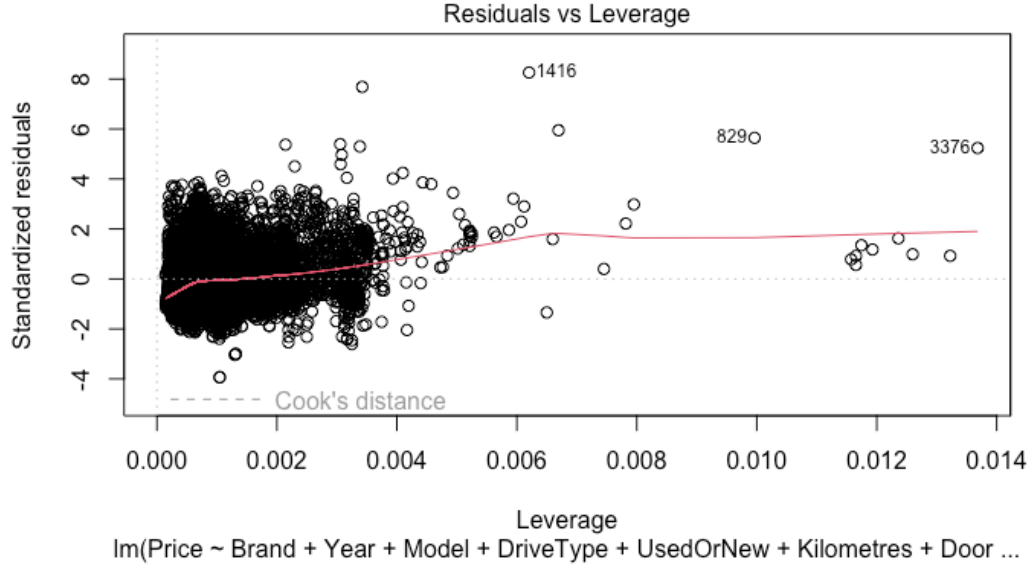

Fig: CV Plot

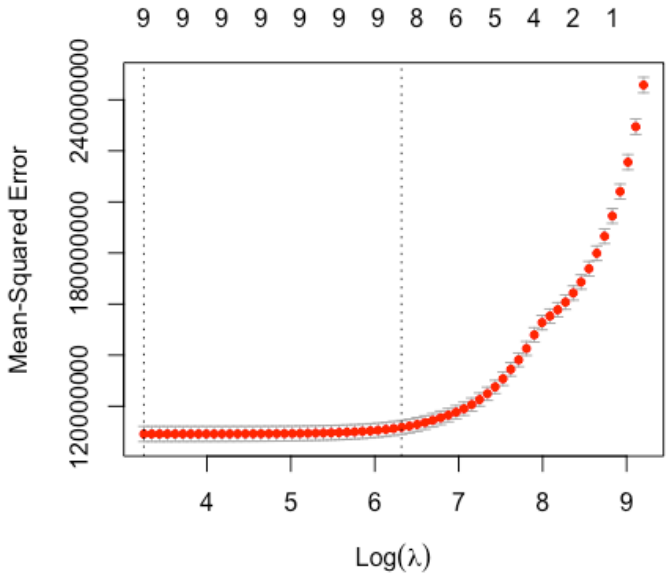

Fig-19: Lasso Regression

```
> ##Fitting models based in Lambda##
>
> #building model on training data using lambda.min
>
> #alpha =1 <we specify that we are running a Lasso model>
> #alpha=0 <we specify that we are running Ridge model>
>
> ##Lasso Regression##
> lasso_train_model_min <- glmnet(train_x, train_y, alpha=1, lambda = cv.lasso$lambda.min)
> lasso_train_model_min

Call:  glmnet(x = train_x, y = train_y, alpha = 1, lambda = cv.lasso$lambda.min)

  Df  %Dev Lambda
1  9 55.79  25.74
>
> #display coefficients
> coef(lasso_train_model_min )
10 x 1 sparse Matrix of class "dgCMatrix"
                                s0
(Intercept)       -3154912.31107489
Brand                    58.70212905
Year                   1583.48683364
Model                     2.19295663
DriveType             -2307.74398772
UsedOrNew             -2610.38784779
Kilometres               -0.05311992
Doors                 -3853.99886850
CylindersinEngine      4589.28684985
Seats                  1007.19407356
```

Fig-20: training data using lambda.1se

```
> #building model on training data using lambda.1se
> lasso_train_model_1se <- glmnet(train_x, train_y, alpha=1, lambda = cv.lasso$lambda.1se)
> lasso_train_model_1se

Call:  glmnet(x = train_x, y = train_y, alpha = 1, lambda = cv.lasso$lambda.1se)

  Df  %Dev Lambda
1  9 54.74  554.6
>
> #display coefficients
> coef(lasso_train_model_1se)
10 x 1 sparse Matrix of class "dgCMatrix"
                                    s0
(Intercept)       -3053610.11687223
Brand                   20.85521347
Year                  1532.88369068
Model                    0.33339530
DriveType            -2005.50593938
UsedOrNew            -1716.80972432
Kilometres              -0.04626771
Doors                -2665.05253483
CylindersinEngine     3802.72252644
Seats                  442.00243795
```

Fig-21: RMSE of lambda

```
> # Predicting based on train data using lambda.1se
> predit_train_lasso_1se <- predict(lasso_train_model_1se, newx = train_x)
> rmse_train_lasso_1se <- rmse(train_y, predit_train_lasso_1se)
> rmse_train_lasso_1se
[1] 10553.45
>
> # Predicting based on test data using lambda.1se
> predit_test_lasso_1se <- predict(lasso_train_model_1se, newx = test_x)
> rmse_test_lasso_1se <- rmse(test_y, predit_test_lasso_1se)
> rmse_test_lasso_1se
[1] 10833.11
>
> ## Predicting based on train data using lambda.min
> predit_train <- predict(lasso_train_model_min ,newx=train_x)
> rmse_train <- rmse(train_y,predit_train )
> rmse_train
[1] 10430.08
>
> ## Predicting based on test data using lambda.min
>
> predit_test <- predict(lasso_train_model_min ,newx=test_x)
> rmse_test <- rmse(test_y,predit_test)
> rmse_test
[1] 10671.31
```

Fig-22: Ridge Regression

```
> ##Ridge Regression###
>
> #building model on training data using lambda.1se
>
> regid_train_model_1se <- glmnet(train_x, train_y, alpha=0, lambda = cv.lasso$lambda.1se)
> regid_train_model_1se

Call:  glmnet(x = train_x, y = train_y, alpha = 0, lambda = cv.lasso$lambda.1se)

  Df  %Dev Lambda
1  9 55.73  554.6
>
> #display coefficients
>
> coef(regid_train_model_1se)
10 x 1 sparse Matrix of class "dgCMatrix"
                                s0
(Intercept)       -2992415.29223128
Brand                    57.31512208
Year                   1503.35824917
Model                     2.61833935
DriveType             -2244.22686729
UsedOrNew             -2777.58181067
Kilometres               -0.05419841
Doors                 -3759.13458325
CylindersinEngine      4356.80055977
Seats                  1009.67250161
```

Fig-23: training data using lambda.min

```
> #building model on training data using lambda.min
>
> regid_train_model_min <- glmnet(train_x, train_y, alpha=0, lambda = cv.lasso$lambda.min)
> regid_train_model_min

Call:  glmnet(x = train_x, y = train_y, alpha = 0, lambda = cv.lasso$lambda.min)

  Df  %Dev Lambda
1  9 55.79  25.74
>
> #display coefficients
>
> coef(regid_train_model_1se)
10 x 1 sparse Matrix of class "dgCMatrix"
                                s0
(Intercept)       -2992415.29223128
Brand                    57.31512208
Year                   1503.35824917
Model                     2.61833935
DriveType             -2244.22686729
UsedOrNew             -2777.58181067
Kilometres               -0.05419841
Doors                 -3759.13458325
CylindersinEngine      4356.80055977
Seats                  1009.67250161
```

Fig-24: RMSE

```
> ## Predicting based on train data using lambda.1se
> predit_train2 <- predict(regid_train_model_1se ,newx=train_x)
> rmse_train2 <- rmse(train_y,predit_train2)
> rmse_train2
[1] 10437.14
> ## Predicting based on test data using lambda.1se
> predit_test2 <- predict(regid_train_model_1se ,newx=test_x)
> rmse_test2 <- rmse(test_y,predit_test2)
> rmse_test2
[1] 10688.48
> ## Predicting based on train data using lambda.min
> predit_train3 <- predict(regid_train_model_min ,newx=train_x)
> rmse_train3 <- rmse(train_y,predit_train3)
> rmse_train3
[1] 10429.83
> ## Predicting based on test data using lambda.min
> predit_test3 <- predict(regid_train_model_min ,newx=test_x)
> rmse_test3 <- rmse(test_y,predit_test3)
> rmse_test3
[1] 10669.45
```

Fig-25: Decision Tree Model

```
tree_model <- rpart(Price ~ Brand+Year+Model+CarType+
                      DriveType+UsedOrNew+Transmission+Doors+
                      CylindersinEngine+Seats+Kilometres, data = train_data)
rpart.plot(tree_model)
predictions <- predict(tree_model, newdata = test_data)
```

Fig-26: Decision Tree

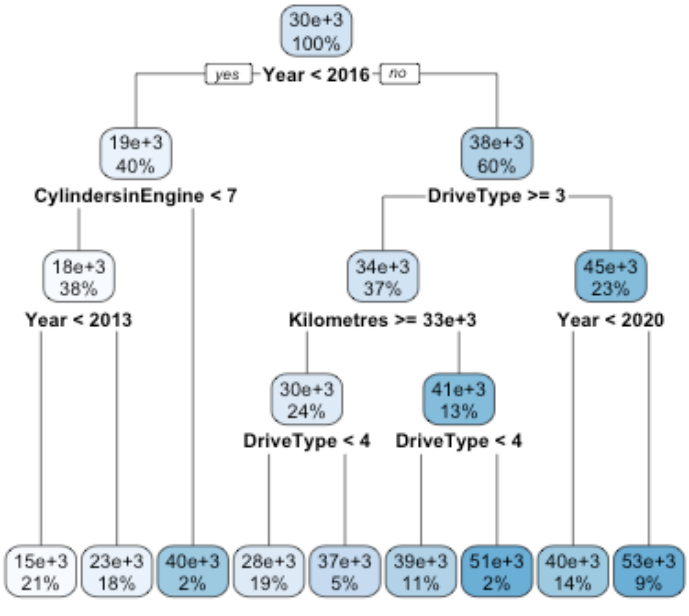

Fig-27: Evaluation Metrix

```
> # Calculate evaluation metrics
> MAE <- mean(abs(predictions - test_data$Price)) #Mean Abs Error
> MAE
[1] 7924.661
> MSE <- mean((predictions - test_data$Price)^2) ##Mean Sqr Error
> MSE
[1] 106770550
> RMSE <- sqrt(MSE) #Root Mean Sq Error
> RMSE
[1] 10332.98
> RSquared <- 1 - sum((test_data$Price - predictions)^2) / sum((test_data$Price - mean(test_data$Price))^2)
> RSquared
[1] 0.5686751
```

Fig-28: Gradient Boosting Method

```
> MAE_gbm <- mean(abs(predictions_gbm - test_data$Price)) #Mean Abs Error
> MAE_gbm
[1] 5089.131
> MSE_gbm <- mean((predictions_gbm - test_data$Price)^2) ##Mean Sqr Error
> MSE_gbm
[1] 49472718
> RMSE_gbm <- sqrt(MSE) #Root Mean Sq Error
> RMSE_gbm
[1] 10667.73
> RSquared_gbm <- 1 - sum((test_data$Price - predictions_gbm)^2) / sum((test_data$Price - mean(test_data$Price))^2)
> RSquared_gbm
[1] 0.8037656
```

Fig 30: Random Forest Model

```
rf_model <- randomForest(Price ~ Brand+Year+Model+CarType+
                           DriveType+UsedOrNew+Transmission+Doors+
                           CylindersinEngine+Seats+Kilometres,
                         data = train_data, ntree = 100)

# Make predictions on test data
predictions2 <- predict(rf_model, newdata = test_data)
```

Fig 31: RF Output

```
> # Calculate evaluation metrics
> MAE2 <- mean(abs(predictions2 - test_data$Price))
> MAE2
[1] 4034.812
> MSE2 <- mean((predictions2 - test_data$Price)^2)
> MSE2
[1] 32531121
> RMSE2 <- sqrt(MSE2)
> RMSE2
[1] 5703.606
> RSquared2 <- 1 - sum((test_data$Price - predictions2)^2) / sum((test_data$Price - mean(test_data$Price))^2)
> RSquared2
[1] 0.8685828
```

Tuned RF Output:

```
> predictions_rf_2 <- predict(rf_model_2, newdata = test_data)
> MAE_RF_2 <- mean(abs(predictions_rf_2 - test_data$Price))
> MAE_RF_2
[1] 4092.411
> MSE_RF_2 <- mean((predictions_rf_2 - test_data$Price)^2)
> MSE_RF_2
[1] 35253428
> RMSE_RF_2 <- sqrt(MSE_RF_2)
> RMSE_RF_2
[1] 5937.46
> RSquared_RF_2 <- 1 - sum((test_data$Price - predictions_rf_2)^2) / sum((test_data
$Price - mean(test_data$Price))^2)
> RSquared_RF_2
[1] 0.8601667
```

Fig 32: Trend of Squared error based on number of iterations

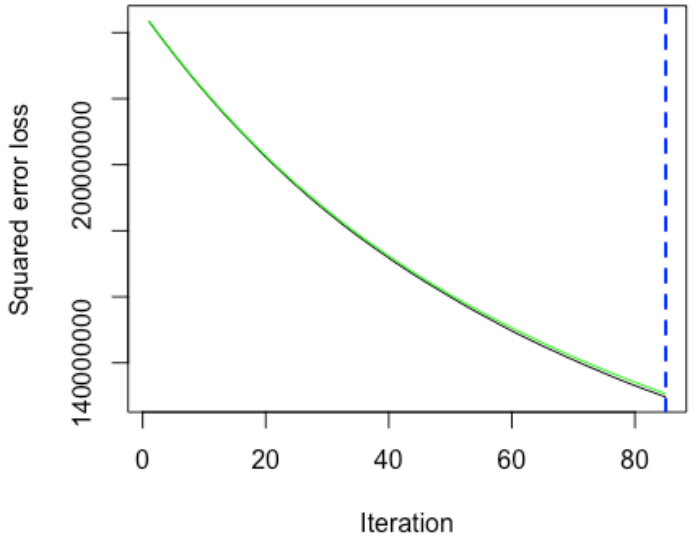

Fig 33: SVM Model

```
> print(svm_model)

Call:
svm(formula = Price ~ Brand + Year + Model + DriveType + UsedOrNew + Kilometres +
    Doors + CylindersinEngine + Seats, data = train_data, type = "eps-regression",
    kernel = "radial")


Parameters:
   SVM-Type:  eps-regression
 SVM-Kernel:  radial
       cost:  1
      gamma:  0.1111111
    epsilon:  0.1


Number of Support Vectors:  7461
```

Fig 34: SVM Evaluation

```
> # Predict and evaluate the model
> predictions_svm <- predict(svm_model, test_data)
> MAE_svm <- mean(abs(predictions_svm - test_data$Price))
> MAE_svm
[1] 4801.503
> MSE_svm <- mean((predictions_svm - test_data$Price)^2)
> MSE_svm
[1] 51324898
> RMSE_svm <- sqrt(MSE_svm)
> RMSE_svm
[1] 7164.14
> RSquared_svm <- 1 - sum((test_data$Price - predictions_svm)^2) / sum((test_data$Price - mean(test_data$Price))^2)
> RSquared_svm
[1] 0.7964189
```

Fig 35: SVM Plots

References:

Australian Vehicle Prices. (n.d.). Www.kaggle.com. https://www.kaggle.com/datasets/nelgiriyewithana/australian-vehicle-prices/data

Learn Linear Regression in R: Linear Regression In R Cheatsheet. (n.d.). Codecademy. https://www.codecademy.com/learn/learn-linear-regression-in-r/modules/linear-regression-in-r/cheatsheet

Split function - RDocumentation. (n.d.). Www.rdocumentation.org. https://www.rdocumentation.org/packages/base/versions/3.6.2/topics/split

Australian Vehicle Prices. (n.d.). Www.kaggle.com. https://www.kaggle.com/datasets/nelgiriyewithana/australian-vehicle-prices/data

Wei, T., & Simko, V. (2021, November 18). An Introduction to corrplot Package. Cran.r-Project.org. https://cran.r-project.org/web/packages/corrplot/vignettes/corrplot-intro.html

Holtz, Y. (n.d.). Boxplot | the R Graph Gallery. R-Graph-Gallery.com. https://r-graph-gallery.com/boxplot.html

Daniel Johnson (9 March 2024): Decision Tree in R: Classification Tree with Example; [url: https://www.guru99.com/r-decision-trees.html?gpp&gpp_sid]

Dario Radecic (22 August 2022): Machine Learning with R: A Complete Guide to Decision Trees; [url: https://www.appsilon.com/post/r-decision-treees]

Jake Hoare: How is Splitting Decided for Decision Trees?; [online]: https://www.displayr.com/how-is-splitting-decided-for-decision-trees/

Abhishek Sharma (16 Feb 2024): 4 Simple Ways to Split a Decision Tree in Machine Learning' [online]: https://www.analyticsvidhya.com/blog/2020/06/4-ways-split-decision-tree/

svm function - RDocumentation. (n.d.). Www.rdocumentation.org. https://www.rdocumentation.org/packages/e1071/versions/1.7-14/topics/svm