

# (Established under the Presidency University Act, 2013 of the Karnataka Act 41 of 2013)

**Mini Project CSE3035 -**

**R Programming for Data Science**

**(2024 – 2025)**

**Project Title** **: Car Price Prediction**

**Student Name** **: Prem Je Kalister**

**Student Roll Number: 20211CAI0187**

**Section:** **: 7CAI/CST-01**

# Abstract:

The prediction of car resale prices is essential for buyers and sellers in the automobile industry. This project uses machine learning models to predict the selling price of cars based on features like fuel type, seller type, and transmission. Models including K-Nearest Neighbors (KNN), Linear Regression, Random Forest, Decision Tree, and XGBoost are trained and evaluated. Recursive Feature Elimination (RFE) is used for feature selection, and the models are tested at different training set sizes (20%, 50%, and 70%). Metrics like Root Mean Squared Error (RMSE) and Mean Absolute Error (MAE) are used to compare performance, with Random Forest and XGBoost emerging as the best-performing models.

# Introduction

The used car market heavily relies on accurate pricing to ensure fairness and transparency. This project addresses the challenge of predicting car resale prices using machine learning models. The dataset comprises details such as car age, fuel type, seller type, and transmission. By employing feature selection, hyperparameter tuning, and performance evaluation across various training set sizes, the project identifies the most effective model for price prediction.

# Dataset Details

* The dataset used in this project includes information about various used cars, with attributes such as:
* **Car\_Name**: The name of the car (not used in modeling).
* **Year**: The year the car was purchased.
* **Selling\_Price**: The selling price of the car (target variable).
* **Present\_Price**: The current ex-showroom price of the car.
* **Kms\_Driven**: The total kilometers driven by the car.
* **Fuel\_Type**: The type of fuel used (Petrol, Diesel, or CNG).
* **Seller\_Type**: Whether the seller is an individual or a dealer.
* **Transmission**: The transmission type (Manual or Automatic).
* **Owner**: Number of previous owners of the car.

# Pre-processing Techniques

Several preprocessing steps were carried out to prepare the dataset for machine learning models:

1. **Handling Irrelevant Features**: The column Car\_Name was dropped as it doesn't contribute to the prediction.
2. **Dealing with Categorical Variables**: Categorical features such as Fuel\_Type, Seller\_Type, and Transmission were converted to factors to allow models to interpret them appropriately.
3. **Handling Missing Data**: Any missing data (though not explicitly mentioned, it's implied by na.omit() usage) was removed to avoid errors during model training.
4. **Data Splitting**: The dataset was split into an 80% training set and a 20% test set using the createDataPartition() function to allow for training and testing of the models.
5. **Encoding Categorical Variables:** Columns such as Fuel\_Type, Seller\_Type, and Transmission were converted into factors using the mutate function for compatibility with machine learning models.
6. **Stratified Sampling** ensures that **each factor level (like Fuel\_Type = "CNG") is represented proportionally** in both the **training** and **test** sets.

# Visualization Techniques

**1. Correlation Matrix**

The correlation matrix is used to visualize relationships between numeric variables. It helps identify which features are strongly associated with the target variable (Selling\_Price) and whether there are redundant features that could be removed or adjusted.

**Code:**

correlation\_matrix <- cor(data1 %>% select\_if(is.numeric)) corrplot(correlation\_matrix, method = "color", type = "lower", tl.cex = 0.8)

**2. Distribution of Selling Price**

The histogram is used to analyze the spread and skewness of selling prices in the dataset. ars are sold in lower price brackets or if there are any outliers.

**Code:**

ggplot(data1, aes(x = Selling\_Price)) +

geom\_histogram(binwidth = 1, fill = "blue", color = "black") +

labs(title = "Distribution of Selling Prices", x = "Price", y = "Count")

**3. Performance Metrics Visualization**

Bar charts are used to compare model performance across different training sizes for key metrics like RMSE (error magnitude) and MAE (average absolute error). These visualizations make it easy to identify which model performs best and how training size affects accuracy.

**Code:**

ggplot(final\_results, aes(x = Train\_Size, y = RMSE, fill = Model)) + geom\_bar(stat = "identity", position = "dodge") + labs(title = "RMSE Comparison Across Training Sizes", x = "Training Set Size", y = "RMSE") + theme\_minimal()

# Machine Learning Models

1. **Model Implementations**
   1. **KNN**: Predicts prices based on the nearest neighbors in the dataset.
   2. **Linear Regression**:

Linear regression fits a linear equation to predict Selling\_Price based on the features. It assumes a linear relationship between the dependent and independent variables.

**Code:**

linear\_model <- lm(Selling\_Price ~ ., data = train\_data) predictions\_linear <- predict(linear\_model, test\_data) results$Linear <- list( RMSE = RMSE(predictions\_linear, test\_data$Selling\_Price), MAE = MAE(predictions\_linear, test\_data$Selling\_Price) )

* 1. **Random Forest**:

Random Forest uses an ensemble of decision trees to make predictions. It introduces randomness by using a subset of features (mtry) and data for each tree, which reduces overfitting and improves accuracy.

**Code:**

rf\_grid <- expand.grid(.mtry = c(2, 3, 4)) rf\_model <- train( Selling\_Price ~ ., data = train\_data, method = "rf", trControl = trainControl(method = "cv", number = 5), tuneGrid = rf\_grid ) predictions\_rf <- predict(rf\_model, test\_data) results$Random\_Forest <- list( RMSE = RMSE(predictions\_rf, test\_data$Selling\_Price), MAE = MAE(predictions\_rf, test\_data$Selling\_Price) )

* 1. **Decision Tree**:

A Decision Tree creates a tree structure to predict Selling\_Price by splitting the dataset into smaller subsets based on feature thresholds. It continues splitting until a stopping criterion is met (e.g., minimum node size).

**Code:**

decision\_tree\_model <- rpart(Selling\_Price ~ ., data = train\_data) predictions\_tree <- predict(decision\_tree\_model, test\_data) results$Decision\_Tree <- list( RMSE = RMSE(predictions\_tree, test\_data$Selling\_Price), MAE = MAE(predictions\_tree, test\_data$Selling\_Price) )

* 1. **XGBoost**:

XGBoost is an advanced implementation of gradient boosting, which builds trees sequentially, each correcting the errors of the previous tree. It’s highly efficient and achieves high predictive accuracy.

**Code:**

train\_matrix <- as.matrix( train\_data %>% mutate(across(where(is.factor), as.numeric)) %>% select(-Selling\_Price) ) test\_matrix <- as.matrix( test\_data %>% mutate(across(where(is.factor), as.numeric)) %>% select(-Selling\_Price) ) xgb\_model <- xgboost( data = train\_matrix, label = train\_data$Selling\_Price, nrounds = 100, eta = 0.1, max\_depth = 3, objective = "reg:squarederror", verbose = 0 ) predictions\_xgb <- predict(xgb\_model, test\_matrix) results$XGBoost <- list( RMSE = RMSE(predictions\_xgb, test\_data$Selling\_Price), MAE = MAE(predictions\_xgb, test\_data$Selling\_Price) )

1. **Hyperparameter Tuning**
   1. **Random Forest**: Tuned using different values of mtry to optimize model performance.
   2. **XGBoost**: Tuned using parameters like eta and max\_depth.
2. **Training Sizes**

Models were trained and evaluated with 20%, 50%, and 70% of the data as training sets.

# Source Code and Screen shots:

**Pre-processing Technique1: Handling unnecessary columns**

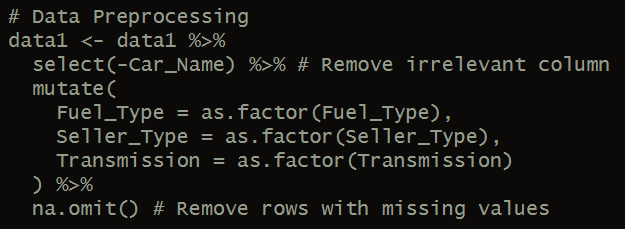
The Car\_Name column is not useful for predictive modeling, so it was removed from the dataset.

**Source code:**

data1 <- data1 %>%

select(-Car\_Name) # Remove irrelevant column

**Screen shot:**



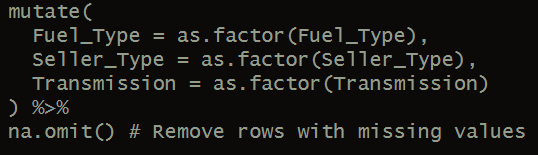
**Pre-processing Technique2: Handling Categorical Variables (converting chars to numeric using mutate function)**

Categorical columns (Fuel\_Type, Seller\_Type, Transmission) were converted into factors using the mutate function to enable compatibility with machine learning algorithms.

**Source code:**

data1 <- data1 %>% mutate( Fuel\_Type = as.factor(Fuel\_Type), Seller\_Type = as.factor(Seller\_Type), Transmission = as.factor(Transmission) )

**Screen shot:**



**Pre-processing Technique 3: Dealing with Missing Values**

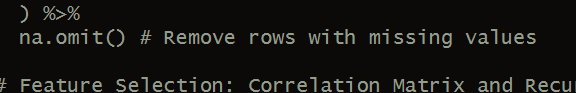
Rows with missing values were removed to ensure the dataset is clean and models are not impacted by missing data.

**Source code:**

data1 <- data1 %>%

na.omit() # Remove rows with missing values

**Screenshot:**



**Pre-processing Technique 4: Feature Selection**

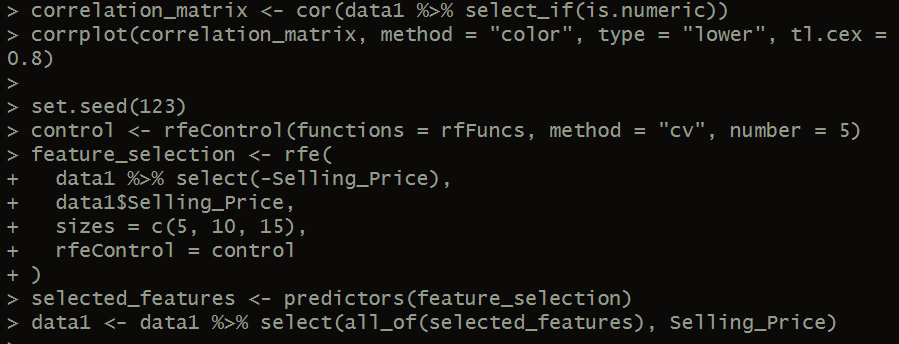
A correlation matrix was generated to identify relationships between numeric variables. Recursive Feature Elimination (RFE) was applied to select the most relevant features for predictive modeling.

**Source Code:**

correlation\_matrix <- cor(data1 %>% select\_if(is.numeric)) corrplot(correlation\_matrix, method = "color", type = "lower", tl.cex = 0.8)

set.seed(123) control <- rfeControl(functions = rfFuncs, method = "cv", number = 5) feature\_selection <- rfe( data1 %>% select(-Selling\_Price), data1$Selling\_Price, sizes = c(5, 10, 15), rfeControl = control ) selected\_features <- predictors(feature\_selection) data1 <- data1 %>% select(all\_of(selected\_features), Selling\_Price)

**Screenshot:**



**Pre-processing Technique 4: Handling Imbalanced Levels**

Stratified sampling was used to ensure that all levels of categorical variables (like Fuel\_Type) are proportionally represented in the training and test datasets.

**Source Code:**

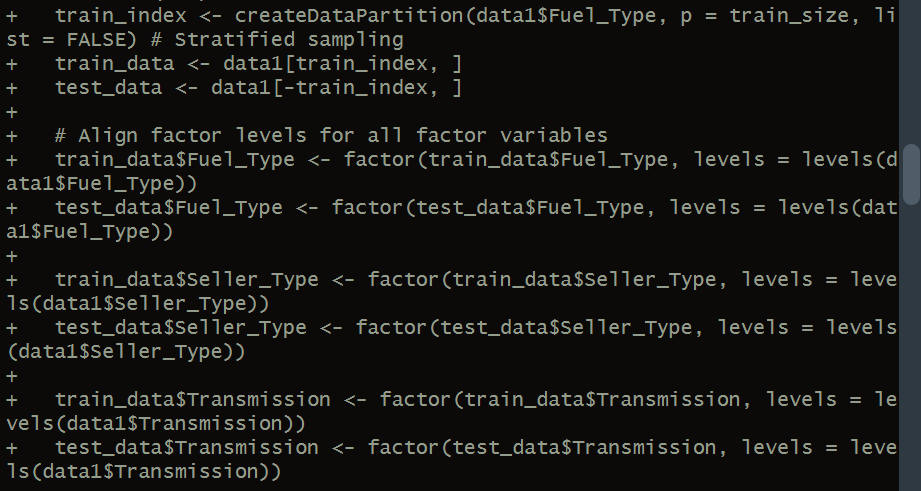
train\_index <- createDataPartition(data1$Fuel\_Type, p = 0.8, list = FALSE) # Stratified sampling train\_data <- data1[train\_index, ] test\_data <- data1[-train\_index, ]

train\_data$Fuel\_Type <- factor(train\_data$Fuel\_Type, levels = levels(data1$Fuel\_Type)) test\_data$Fuel\_Type <- factor(test\_data$Fuel\_Type, levels = levels(data1$Fuel\_Type))

train\_data$Seller\_Type <- factor(train\_data$Seller\_Type, levels = levels(data1$Seller\_Type)) test\_data$Seller\_Type <- factor(test\_data$Seller\_Type, levels = levels(data1$Seller\_Type))

train\_data$Transmission <- factor(train\_data$Transmission, levels = levels(data1$Transmission)) test\_data$Transmission <- factor(test\_data$Transmission, levels = levels(data1$Transmission))

**Screenshot:**



# Results

Performance Metrics:

|  |  |  |  |
| --- | --- | --- | --- |
| **Model** | **Train Size** | **RMSE** | **MAE** |
| KNN | 20% | 5.46 | 3.69 |
| Linear Regression | 20% | 2.20 | 1.33 |
| Random Forest | 20% | 3.47 | 1.33 |
| Decision Tree | 20% | 3.86 | 1.81 |
| XGBoost | 20% | 3.09 | 1.12 |
| KNN | 50% | 5.74 | 3.78 |
| Linear Regression | 50% | 2.51 | 1.44 |
| Random Forest | 50% | 2.72 | 1.11 |
| Decision Tree | 50% | 3.11 | 1.60 |
| XGBoost | 50% | 1.65 | 0.75 |
| KNN | 70% | 4.49 | 3.33 |
| Linear Regression | 70% | 1.49 | 1.11 |
| Random Forest | 70% | 0.95 | 0.60 |
| Decision Tree | 70% | 1.99 | 1.14 |
| XGBoost | 70% | 0.74 | 0.47 |

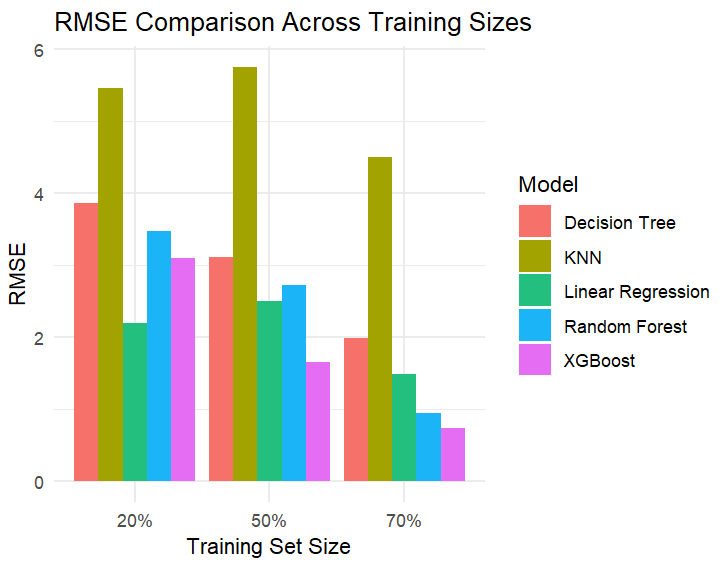
**INFERENCES (Real-World Application)**

For Predicting Car Prices, XGBoost and Random Forest are the most suitable models providing highest predictive accuracy according to the results mentioned.

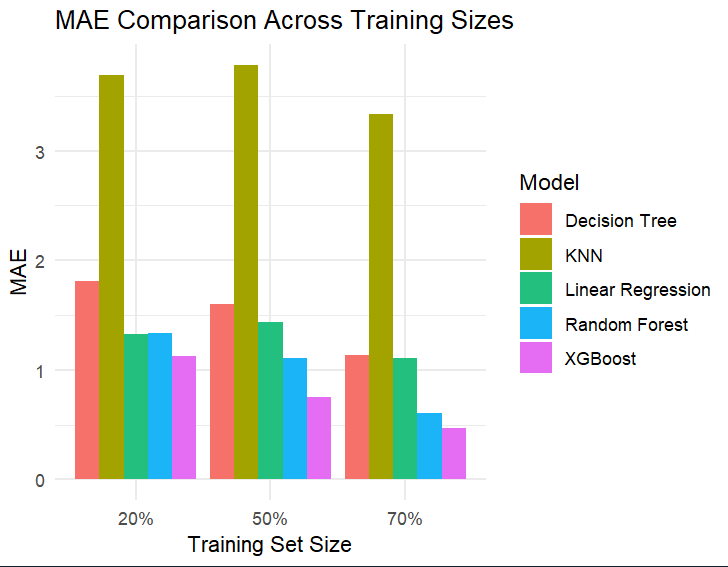
* **Effect on price**: The **higher the original price**, the **higher the resale price**.
* **Older cars have a lower resale value**.
* **Petrol cars** typically have a higher resale value compared to **Diesel** or **CNG cars** due to market demand. This varies regionally.
* **Automatic cars** usually have a higher resale price than **manual cars**.
* **More kilometers driven** leads to a **lower resale value**.

**Bar Plots**

* **RMSE Across Training Sizes**



* **MAE Across Training Sizes**



# Conclusion

The project successfully predicts car resale prices using machine learning. Random Forest and XGBoost delivered the best performance, with XGBoost achieving the lowest RMSE across all training sizes. Future work could include integrating additional features and testing more advanced deep learning techniques.