UNIVERSITY OF CALIFORNIA

Los Angeles

Predicting the Selling Prices of Used Cars in Pakistan Using Various Statistical Learning Models

A thesis submitted in partial satisfaction of the requirements

for the degree

Master of Applied Statistics and Data Science

by

Yulin Song

2024

ABSTRACT OF THE THESIS

Predicting the Selling Prices of Used Cars in Pakistan

Using Various Statistical Learning Models

by

Yulin Song

Master of Applied Statistics and Data Science

University of California, Los Angeles, 2024

Professor Xiaowu Dai, Chair

This thesis is dedicated to exploring the relationship between features of Pakistani used cars and their prices, potentially providing insights to help relevant stakeholders of used cars business make informed decisions. Multiple statistical learning methods involved in regression analysis, namely Linear Regression, Elastic Net, Random Forest, and XGBoost, are utilized to predictively model and quantitatively explain the said relationship.

From a Kaggle dataset of 77,878 rows and 14 columns (including response variable "price"), data preprocessing was done, including removal of outliers, empty values, and missing entries; re-grouping, re-naming, and conversion of certain variables, etc. Then, exploratory data

analyses were conducted, including addressing non-normal or highly-correlated numerical variables and overly homogeneous categorical variables. Next, the processed dataset (70,500 rows and 10 columns) was split into 80% training and 20% testing sets, and model evaluation criteria of Root Mean Squared Error (RMSE) and Coefficient of Determination (R^2) were determined. Following, the four models were applied on the training set, with the 9 main predictors and all 36 pairs of the interactions served as a baseline design matrix, and natural-log transformed "price" served as the response. After fitting models with corresponding methods of dimension reduction method or hyper-parameter tuning, model evaluation criteria testing RMSE and R^2 were calculated using prediction on the testing set. Eventually, XGBoost model stood out as the final model due to optimal performance in both performance scores, and features such as body type, engine volume, age, and transmission type, along with interacting effect between age and the other three factors, were agreed by all four models to be important in predicting Pakistani used car prices.

The thesis of Yulin Song is approved.

Michael Tsiang

Hao Ho

David Anthony Zes

Xiaowu Dai, Committee Chair

University of California, Los Angeles

2024

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

Chapter 1: Introduction

Section 1.1: Background Information and Motivation

The market of automobiles in Pakistan, similar to that of many emerging automobile markets in developing countries around the world, is rapidly growing in size. This increase in popularity in privately owning an automobile in Pakistan is primarily attributed to the contribution of the used vehicle market in comparison with the new vehicle market. According to an article published on Pakistan's largest second-hand automobile selling company's website, *Pakwheels.com*, in Pakistan in 2017, around 750,000 used cars were traded and around 200,000 new cars were purchased. Also, they conducted "... an industry survey in 2017... a total of 19,155 number of responses were recorded... 58% of the respondents said that they bought a second-hand car, while 38% asserted that they bought a brand new car", and that the used vehicles buying percentage have increased slightly compared to previous surveys (Laghari, 2018).

Used cars have many usages in Pakistan, like for personal, commercial, ride-sharing, and vintage car enthusiasts; but Pakistani preference leans from new to used cars is mainly due to affordability. The Pakistani economy is small but quickly expanding. According to World Bank, the GDP per capita for Pakistan in 2022 is 1,588.9 US Dollars which is around one eighth of world average, but still enjoyed a compounded average annual growth rate of 2.54% per year from the 2012 GDP per capita value of 1,236.9 US Dollars ("GDP per capita [current US$] - Pakistan", n.d.). This value is quite impressive, especially considering the impact of the COVID-19 pandemic. In the car-making industry, local manufacturers currently produce little compared to the amount of vehicle buyers. This shortage of local cars has naturally led to bigger portion of imported cars, longer delivery times, higher taxes and import duties, and consequently

significantly higher premiums for new cars. Therefore, used cars have become a more affordable and timely solution for Pakistani customers.

A basic concept from economics states that, in a market of a certain product (for both goods and services), its demand and supply is mostly influenced by its own price. Furthermore, The magnitude of this influence can be determined by several other related factors, such as innovation and production technology levels, the actual or perceived difference from other products of the same category or from the same market, the established or potential size and activeness of the trade activity within a market or between several markets in a system, among others. These economic rules also apply to used car markets in Pakistan, the focus of this paper.

Section 1.2: Literature Review

It is significant to accurately predict the prices of used cars based on their other features and their environments. There exists extensive amount of research papers in predicting the sale price of used cars in various countries or economic entities around the world.

One example of these research papers is the study by Peerun et al. from *The Second International Conference on Data Mining Internet Computing and Big Data* in 2015, which focused on predicting the prices of used cars in Mauritius. Their research aimed to develop four models on 200 cars from various sources and a set of relevant features like engine volume, paint type, transmission type, mileage etc., and found out that Support Vector Regression performs slightly better than Linear Regression and Multi-Layer perceptron (MLP) and much better than K-Nearest Neighbour (KNN) based on the criterion of Mean Absolute Error.

On the other hand, the study by Jin from *IEEE International Conference on Emergency Science and Information Technology (ICESIT)* in 2021 focused on predicting used cars'

reasonable prices in the United Kingdom. This study used data of 13,120 Mercedes vehicles from 100,000 UK used cars' scraped data, which was originally obtained from Kaggle, and contained features of cars like transmission type, fuel type, miles per gallon etc. to fit 5 models, and found out that Random Forest Regression performs better than Decision Tree Regression, Support Vector Regression, Linear Regression and Polynomial regression because Random Forest Regression produces the highest R^2 of around 0.9.

These studies demonstrate the applicability of various statistical learning techniques in predicting used car prices, which is increasingly sophisticated due to factors like global and domestic economic conditions and shift in market trends.

Section 1.3: Research Question and Procedures

This paper focuses on understanding the relationship between features of used cars and their final selling price. This paper aims to provide some suggestions to Pakistani used car dealers to better balance their profit margin and customer counts, and to Pakistani used car buyers and private sellers to be better informed and considerate with their decisions.

In this paper, several statistical learning methods are used to discover the relationship between features of used cars and their prices. Necessary data preprocessing (outlier removal, missing values removal or imputation, transformation of variables etc.), exploratory data analysis and separation of processed data into training and testing sets is first performed. Then, the statistical learning methods are introduced, and models using these methods are fitted on the training set. Next, performance of each fitted result will be accessed and compared, and the result produced by the best model will be used to make conclusions and recommendations about used cars and their prices.

Four statistical learning methods were chosen for this paper, namely Linear Regression, Elastic Net, Random Forest, and XGBoost. The selected ensemble of methods offers a comprehensive approach by combining models with varying strengths and complexities - with Linear Regression and Elastic Net being more straightforward, and Random Forest and XGBoost being more sophisticated. The combination of these methods seeks to improve prediction generalizability, performance and balance between robustness and overfitting for predicting the prices in the dynamic Pakistani used cars market.

Chapter 2: Data Source and Preprocessing

Section 2.1: Data Source

The Pakistani used cars data was obtained from Kaggle, uploaded and improved by Talha Barkaat Ahmad in 2023; this paper uses the dataset "pakwheels_used_car_data_v02.csv", which consists of 77,878 observations and 14 variables. The numerical variable "price" is treated as the response (dependent) variable, and the other 13 are treated as the predictor (independent) variables. Slightly modified variables descriptions are summarized below:

Variable Name	**Variable Type**	**Variable Description**
addref	Numerical	a unique ad reference (equivalent to ID number)
city	Categorical	advertisement city (location where vehicle is sold)
assembly	Categorical	imported or local
body	Categorical	body type of vehicle
make	Categorical	manufacturer of vehicle
model	Categorical	model variant of vehicle
year	Numerical	year of production
engine	Numerical	engine volume of vehicle (in cm^3 or ml)
transmission	Categorical	Automatic/Manual
fuel	Categorical	Petrol/diesel/hybrid

Variable Name	Variable Type	Variable Description
color	Categorical	color of vehicle
registered	Categorical	registration number city/province of vehicle
mileage	Numerical	mileage (in kilometers)
price	Numerical	price of vehicle in PKR (Pakistan Rupee)

Table 2.1: (Data Source) Variables Information from "car_original" (Pg. 4-5)

Section 2.2: Data Preprocessing

To better solve the paper's research question, necessary preprocessing procedures (elimination or modification) were performed on the variables of the original data. These data preprocessing procedures are summarized below.

For the change in response variable "price":

- price (response, leaving with 77,295 rows and 14 columns)
  - The 583 missing ("NA") or empty ("") values (less than 1% of total) were dropped, because imputation of the response would be unreliable for prediction purposes;
  - The highest price (529,000,000 Pakistan Rupee) is approximately 3 times higher than the 2nd highest price, and it is a Toyota Corolla (an inexpensive car) which should be a mistype. By inspecting into Corollas with similar features, a more reasonable price should be 5,290,000 Rupees, so the unreasonable price was changed accordingly.

For the four removed predictor columns:

- addref (left with 77,295 rows and 13 columns)
  - "addref" is an ID column with no missing, empty, or duplicate values, meaning there exist no same cars or unidentifiable cars, therefore it was removed.
- Registered, make, models (left with 77,295 rows and 10 columns)

- The registered city ("registered") is a lot less important and practical than the transacted city ("city"). Also, instead of performing an analysis on models (and corresponding manufacturers), this paper mainly focuses on the car's inherent features, and makes and models serves mainly identification purposes. Therefore, these three were removed.

For the remaining predictor columns:

- body (name changed to "body_group", dimension unchanged)
  - There are 8,857 missing or empty values in "body" (8,857 counts, over 10% but less than 30% of all rows), so the empty values were imputed.
  - Body types were grouped into the following categories:
    - "Sedan" and "Compact sedan" into "Sedan" (30,827 counts);
    - "Hatchback", "Compact hatchback" and "Station Wagon" into "Hatchback" (25,193 counts);
    - "Pick Up", "SUV", "Compact SUV", "Double Cabin", "Single Cabin", "Off-Road Vehicles", "MPV", "Crossover", and "Single Cabin" into "Utility" (9,586 counts);
    - "Micro Van", "Mini Van", and "Mini Vehicles" into "Micro/Mini" (1,853 counts);
    - "Van", "Truck", and "High Roof" into "Commercial" (847 counts);
    - "Convertible" and "Coupe" into "Sports" (132 counts).
  - For each of the 8,857 empty body types, a body type was randomly chosen based on the probability given by the above proportions, with seed setting to 1 for reproducibility, and the chosen body type is imputed on the empty value. For example, every empty type is categorized into "Sedan" with probability 30,827 / 77,295, and so on;
    - After imputation, there are 34859 in "Sedan", 28,330 in "Hatchback", 10,891 in "Utility", 2,111 in "Micro/Mini", 960 in "Commercial" and 144 in "Sports".

- assembly (dimension unchanged)
  - Based on the original dataset description on Kaggle, “assembly” is either imported or local, so it is assumed that an empty value represents a local car, therefore all empty values were replaced with “Local”, and all “Imported” values remained unchanged.
- year (name changed to “age”, left with 72,672 rows and 10 columns)
  - The “year” can be converted to “age” (years old) by subtracting the data source year 2023 by the specific “year”, for example “year” = “2020” is converted to “age” = “3”;
  - The 4,623 (less than 10% of overall) missing (“NA”) or empty (“”) values were dropped and not imputed because age of car is very important in predicting the price of car due to aging of parts, change in fashion trends or technology etc, and imputing on the age of car can be speculative and thus drastically impact the prediction accuracy.
- color (name change to “color_group”, left with 71,287 rows and 10 columns)
  - The 1,385 missing (“NA”) or empty (“”) values, which constitutes less than 10% of all observations, were dropped and not imputed because the unknown color of a car is independent based on the known color of other cars or other features of the car; a empty color can be attributed many factors, like color being too rare, too complicated to be accurately described, or recently modified; Perhaps, the color was simply not reported;
  - After inspection, some colors are very popular like white, silver, black, and grey; and some “colors” are shades of the above, like “Solid White”, “Graphite Grey” etc; others are less popular or perhaps rarely produced (like Purple and Pink);
  - When the color is shades of white (the color value contains “White”, like “White” or “Solid White”), they were converted to “White”, and so on; when the color are not in the above groups, they were converted to “Others”.

- 30,910 “White”, 12,047 “Silver”, 11,979 “Others”, 10,225 “Black”, 6,126 “Gray”.

- engine (left with 71,113 rows and 10 columns)
  - Including only engine between 600ml and 6,600ml (the majority are in this range);
  - Many engines outside of this range were possibly mistyped like 80ml Mercedes and 12,345 Honda Civic engines, so the 162 of them and 2 empty values were removed.

- fuel (left with 70,500 rows and 10 columns)
  - The 613 missing (“NA”) or empty (“”) values (less than 10% of total) were removed and not imputed because over 90% of known values (65,131 out of 70,500) are “Petrol”. Imputing the empty values would include only low extra information and prediction power, and prediction results and model performance would change only a little.

- city (city of car transaction, name changed to “city_popularity”, dimension unchanged)
  - Since there are 291 car transaction cities with many of them have little amount of cars, cities were re-categorized based on their popularity;
    - City with at least 10,000 cars sold (Lahore, Karachi, and Islamabad) were categorized to “high”; between 1,000 and 10,000 (Rawalpindi, Peshawar, Faisalabad, Multan, Gujranwala, Sialkot) to “medium”; and the rest to “low”.

- No changes were made to “transmission” and “mileage”.

In addition to the operations done to manipulate the original dataset, all character columns were individually converted to factors in R. This way, in the summary statistics of the processed dataset in the Exploratory Data Analysis section, after the conversion, the count of each unique factor level will be shown instead of only the total number (70,500) of observations.

The variable descriptions from the resulting “car_processed” dataset is reported below:

Original Variable Name	Updated Variable Name	Variable Type	Updated Description in italics and Original Description in regular test
city	city_popularity	Categorical	*popularity of the transaction city*
assembly		Categorical	imported or local
body	body_group	Categorical	*Body type in combined groups*
year	age	Numerical	*Age of the car (in 2023)*
engine		Numerical	engine volume of vehicle (in cm^3 or ml)
transmission		Categorical	Automatic/Manual
fuel		Categorical	Petrol/diesel/hybrid
color	color_group	Categorical	*color of vehicle in combined groups*
mileage		Numerical	mileage (in kilometers)
price		Numerical	price of vehicle in PKR (Pakistan Rupee)

Table 2.2: (Data Preprocessing) Description of Variables from "car_processed"

Note that deleted variables "addref", "make", "model" and "registered" are excluded here because they were removed during the preprocessing phase.

Chapter 3: Exploratory Data Analysis

The first step in the exploratory data analysis is to produce summary statistics tables of the nine predictors (3 numerical and 6 categorical) and the response numerical "price" of the updated dataset "car_processed".

	age (years)	engine (ml)	mileage (km)	Price (PKR)
Minimum	1	600	1	110,000
1st quarter	4	1,000	39,000	1,500,000
Median	8	1,300	81,000	2,712,500
Mean	10.09	1,395	92,809	3,778,827
3rd quarter	16	1,600	12,3456	4,485,000
Maximum	33	6,600	1,000,000	170,000,000

Table 3.1: (Exploratory Data Analysis) Summary Statistics of Numericals in "car_processed"

	city_popularity		assembly		body_group
low	15,894	**Imported**	21,423	**Sports**	123
medium	16,158	**Local**	49,077	**Commercial**	854
high	38,448			**Micro/Mini**	1,972
				Utility	9,318
				Hatchback	25,887
				Sedan	32,346

	transmission		fuel		color_group
Manual	31,630	**Diesel**	2,717	**Gray**	6,039
Automatic	38,870	**Hybrid**	2,652	**Black**	10,162
		Petrol	65,131	**Silver**	11,930
				Others	11,789
				White	30,580

Table 3.2: (Exploratory Data Analysis) Summary Statistics of Categoricals in "car_processed"

In addition, bar charts of categorical variables with descending ordered frequency were constructed and are provided below.

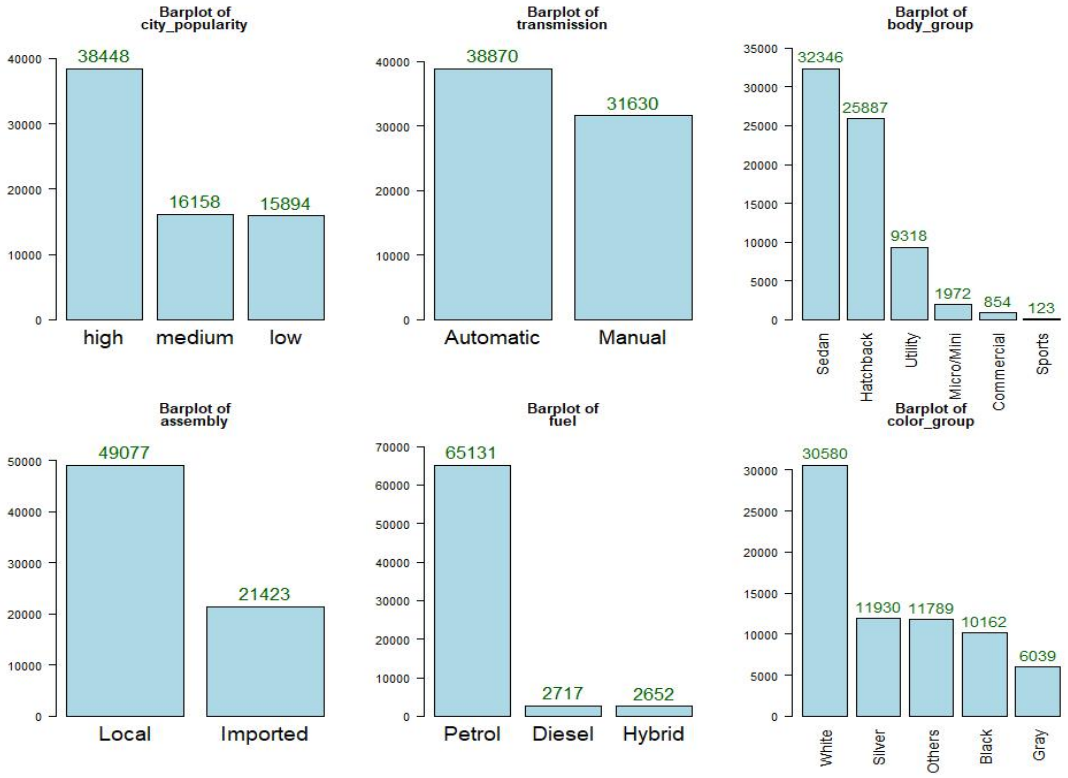


Figure 3.1: (Exploratory Data Analysis) Bar Plots for Categoricals in "car_processed" (Pg. 10)

From the summary statistics and the bar plots, the vast majority of fuel types of cars (over 90%) is petrol. Usually, variables like this where a value has much more presence than any other values often result in low variance and thus less impact on formation of models, so greater care for "fuel" is needed in modeling processes.

Histograms for the numerical variables (age, engine, mileage, price), histograms were constructed to examine the overall distribution. To apply the modeling methods, it is necessary to check if there are distributions with significant departures from approximate normality.

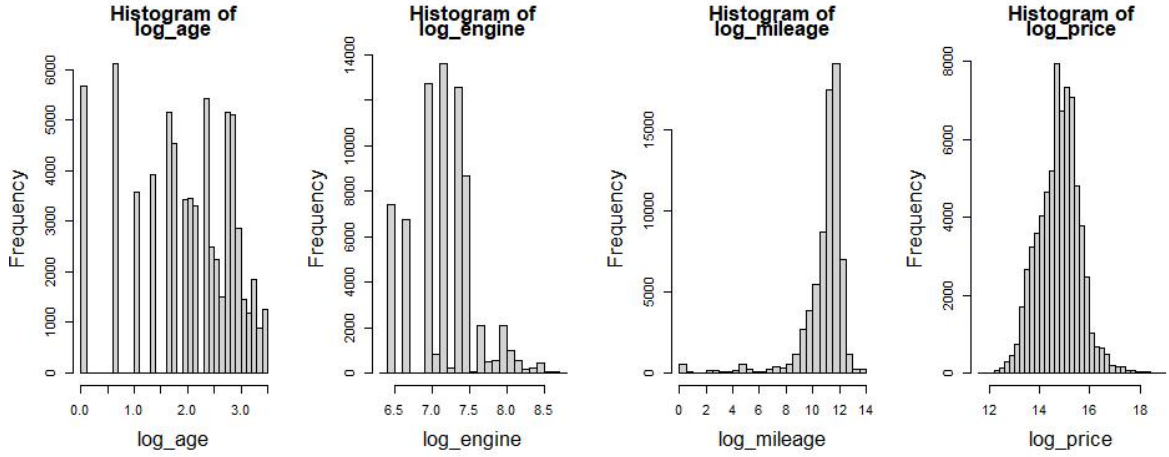


Figure 3.2: (Exploratory Data Analysis) Histograms for Numericals in "car_processed"

From the above histograms, all four numerical predictors are right-skewed with varying extent and thus not approximately normal, so transformations are necessary on these variables before fitting models. In the end, natural logarithm transformations are performed on each of the four numerical variables, and their column names are changed accordingly (from age to log_age, from engine to log_engine, and so on), and the resulting histograms are shown below:

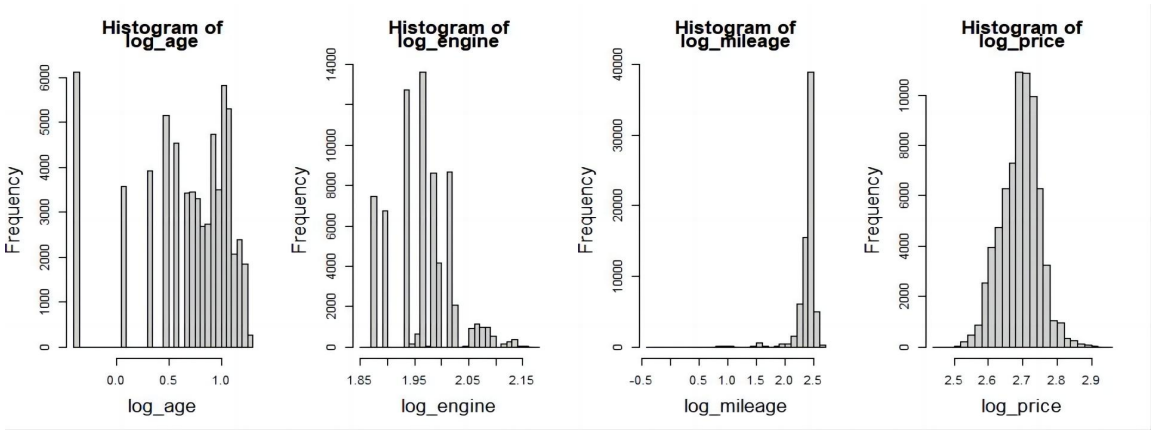


Figure 3.3: (Exploratory Data Analysis) Histograms for Log(Nums) in "car_processed"

Now they better adhere to the approximately-normal requirement.

Next, the results of pairwise correlations between these four numerical variables will be presented; if the correlation plot and scatter plots in the next step show any pairs of extremely high negative or positive correlation between numerical predictors, it might be necessary to avoid including both variables together.

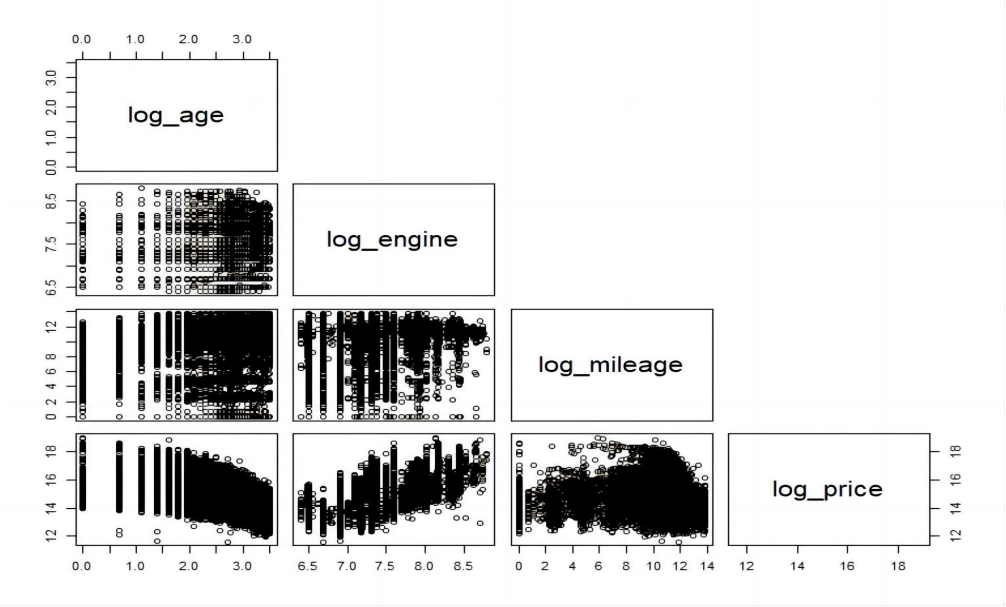


Figure 3.4: (Exploratory Data Analysis) Scatter Plots for Log(Nums) in "car_processed"

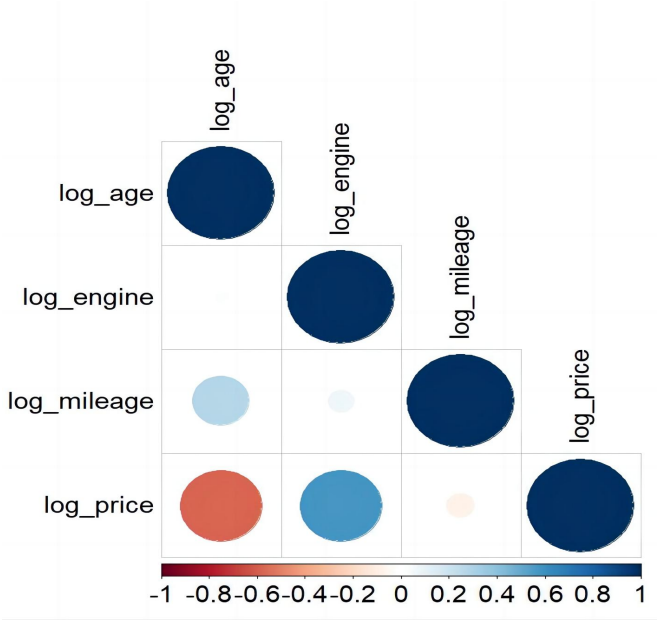


Figure 3.5: (Exploratory Data Analysis) Correlation Plot for Log(Nums) in "car_processed"

From the pairwise scatter plots, there do not seem to be approximately pairwise linear relationships between the three numerical predictors; from the correlation plot, the pairwise correlation between the three numerical predictors are also relatively low, so there is lower chance of pairwise collinearity between numerical predictors when fitting models.

For numerical predictors and the numerical response, from the pairwise scatter plots, there seems to be strong negative linear relationship between "log_age" and "log_price" and strong positive linear relationship between "log_engine" and "log_price". The direction and strength of the relationships of the said two pairs of variables are also visible from the correlation plot. However, there seems to be no approximately linear relationship or correlation between "log_mileage" and "log_price", so greater care might be needed when analyzing the effect of "log_mileage" in further models.

Chapter 4: Data Splitting and Model Evaluation

Section 4.1: Data Splitting

Before using various statistical learning methodologies to fit the models, the "car_processed" dataset was split into training dataset (56,400 rows, 80 percent of total observations) and testing dataset (14,100 rows, 20 percent of total observations). The splitting was done using random sampling without replacement. A seed in R was set using set.seed() function; by setting a seed (an integer), the pseudo-random output can be reproduced. In this paper, the seeds was set to 1. Later, each model will be fitted on the same training dataset and evaluated on the same testing dataset.

Section 4.2: Root Mean Squared Error (RMSE)

Root Mean Squared Error (RMSE) is a widely used metric to evaluate the performance of regression models. It quantifies the standard deviation of the residuals, which represents the distance between the observed (actual) values and the values predicted by the regression model. In other words, RMSE quantifies how much, on average, the model's predictions deviate from the actual values.

The formula for RMSE is: $RMSE = \sqrt{\frac{1}{n}\Sigma_{i=1}^{n}(y_i - \hat{y}_i)^2}$, where y_i represents the actual values, $\hat{y}_i$ represents the predicted values, and n is the number of observations, and a low RMSE means better fit to the testing dataset. RMSE is sensitive to the scale of the data, meaning that its value is expressed in the same units as the outcome variable (in this case, natural log of the price of cars in Pakistan Rupee "PKR").

Section 4.3: Coefficient of determination (R^2)

Coefficient of determination (R-squared, often written as R^2) is also a widely used metric to evaluate the performance of regression models. It is a statistical measurement that represents the proportion of the variance for a dependent variable that is explained by an independent variable (in simple regression model) or independent variables (in multivariate regression model). The formula for R^2 is:

$$R^2 = 1 - \frac{\text{Sum of Squares of Residuals (SSR)}}{\text{Total Sum of Squares (SST)}} = 1 - \frac{\sum_{i=1}^{n}(y_i - \hat{y}_i)^2}{\sum_{i=1}^{n}(y_i - \bar{y})^2}$$
$$= \frac{\text{Sum of Squares Explained (SSE)}}{\text{Total Sum of Squares (SST)}} = \frac{\sum_{i=1}^{n}(\hat{y}_i - \bar{y})^2}{\sum_{i=1}^{n}(y_i - \bar{y})^2}$$

- where y_i represents the actual values, $\hat{y}_i$ represents the predicted values, $\bar{y}$ represents the mean of actual values, and n is the number of observations. Note that Sum of Squares of Residuals plus Sum of Squares Explained equals Total Sum of Squares.
- R^2 is an intuitive statistic that gives information about model's goodness of fit. R^2 takes on the value between 0 and 1 (included); suppose $R^2 = 0.4$, this means that 40% of the variability of the response variable has been accounted for by the model, and the remaining 60% of the variability is unaccounted for by the model. R^2 of 0 means that the model is completely ineffective in predicting the outcome and R^2 of 1 means that the model is completely effective in predicting the outcome.
- Note, R^2 is the squared value of the Pearson coefficient of correlation R between two vectors "x" and "y" , takes on the value between -1 and 1 (inclusive), and the formula is:

$$r = \frac{\text{Covariance between X and Y}}{\sqrt{\text{Variance of X} \cdot \text{Variance of Y}}} = \frac{\sum_{i=1}^{n}(x_i - \bar{x})(y_i - \bar{y})}{\sqrt{\sum_{i=1}^{n}(x_i - \bar{x})^2 \sum_{i=1}^{n}(y_i - \bar{y})^2}}$$

- where y_i represents the y values, $\bar{y}$ represents the mean of y values, x_i represents the x values, and $\bar{x}$ represents the mean of x values.

Chapter 5: Methodologies and their Applications

Section 5.1: Linear Model with Top-5 Significant Interactions

In statistics and data science, linear regression is a fundamental statistical learning method used extensively in a wide spectrum of fields ranging from health science to e-commerce. Linear regression is often used as a baseline for other statistical methods to model and analyze the relationships between a dependent (or response) variable and one or more independent (or predictor) variables. When there is only one independent variable, the process is called simple linear regression; when there are two or more independent variables, the process is called multiple linear regression, which is the case here. This paper is aiming to predict the outcome of the dependent variable (log_price in this case) based on the linear combination of independent variables. Note that multiple linear regression is different from multivariate linear regression, which is to predict multiple dependent variables that are correlated, rather than a single dependent variable.

A linear regression with only the main effects (each of the predictors on their own) assumes the predictors variables independently influence the response, but in real-world scenarios, often there are more complex interactions involved. A linear regression with interactions terms allows for investigating of how the relationship between one predictor and the response changes across the levels of another predictor.

This paper considers the 2-way (pairwise) interaction effects between the predictors in linear regression analysis because many features of a used car do not influence the sale price by themselves. For example, an aging car might not decrease in value if the car is driven very infrequently (mileage grows slow) or if the car is very reliable, or even its popularity suddenly grows. Another example, for the same model, an imported car might not be more expensive compared to a local car, if the imported car is manual and the local car is automatic.

A multiple linear regression, same as simple or multivariate linear regression, is modeled through an error term, often noted as Ɛ, which is an random variable that is unobserved and serves as noise to the linear relationship. Given a dataset $\{y_i, x_{i_1}, \ldots, x_{i_p}\}_{i=1}^{n}$, multiple linear regression equation for a particular observation (row) i with all pairwise i interactions takes the form of:

$$y_i = \beta_0 + \beta_1 \cdot x_{i_1} + \ldots + \beta_p \cdot x_{i_p} + \beta_{p+1} \cdot x_{i_1} \cdot x_{i_2} + \ldots + \beta_{\sum_{j-1}^{p} j} \cdot x_{i_{p-1}} \cdot x_{i_p} + \epsilon$$

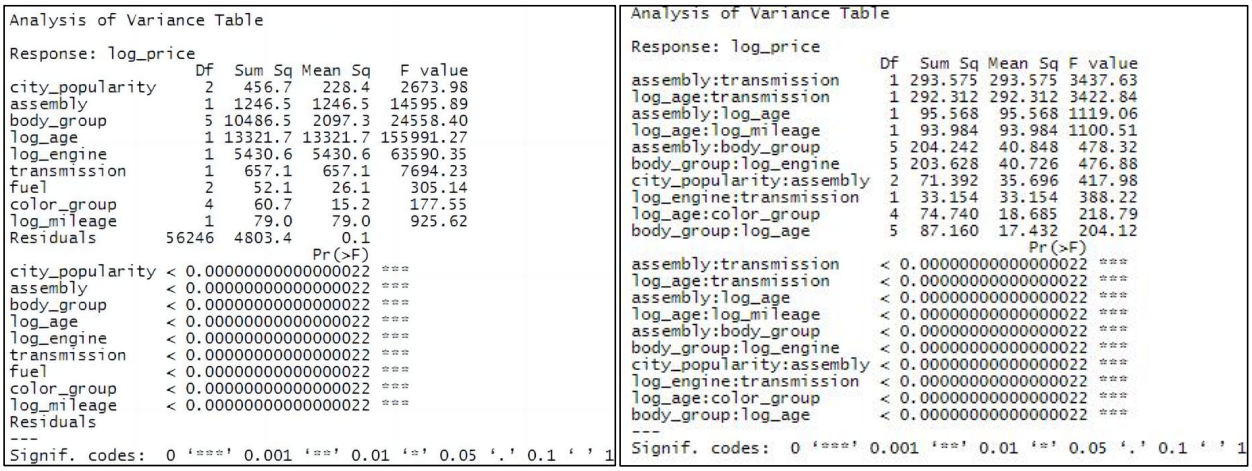
where y_i represents the i[th] response value, x_{i_1} to x_{i_p} represent the i[th] value of the 1[st] to p[th] predictor, β_0 represents the intercept coefficient, β_1 to β_p represent the corresponding coefficient of the p main effects, β_{p+1} to $\beta_{\sum_{j-1}^{p} j}$ represent the corresponding coefficient of each of the $\Sigma_{j=1}^{p-1} j$ interaction terms, and ϵ represents the error term. When fitting a linear regression, the sum of squared error loss function is minimized, namely

$$L(\vec{\beta}) = \Sigma_{i=1}^{n}((\beta_0 + \beta_1 x_{i_1} + \ldots + \beta_p x_{i_p} + \beta_{p+1} x_{i_1} \cdot x_{i_2} + \ldots + \beta_{\Sigma_{j-1}^{p} j} x_{i_{p-1}} \cdot x_{i_p}) - y_i)^2$$

where $\vec{\beta}$ is the vector $(\beta_0, \beta_1, \ldots, \beta_{\Sigma_{j-1}^{p} j})$.

As a baseline for linear regression, a model with all main effects and 2-way interactions are considered. However, since there are 9 predictors (main effects), there are 36 pairwise interactions, which is too computationally expensive for linear model. Therefore, a dimension

reduction method is applied so that only the few interaction terms with the highest statistical significance will be kept for the linear model. Before deciding which interaction terms (and main effects) will be kept, an ANOVA (stands for ANalysis Of VAriance) table is computed. Below, not the whole ANOVA table is printed (since the vast majority of interaction terms p-values are much smaller than 0.01). Instead, the ANOVA tables for main effect and interactions with top 10 F-values are printed below.



```
Analysis of Variance Table

Response: log_price
                Df  Sum Sq Mean Sq   F value
city_popularity  2   456.7   228.4   2673.98
assembly         1  1246.5  1246.5  14595.89
body_group       5 10486.5  2097.3  24558.40
log_age          1 13321.7 13321.7 155991.27
log_engine       1  5430.6  5430.6  63590.35
transmission     1   657.1   657.1   7694.23
fuel             2    52.1    26.1    305.14
color_group      4    60.7    15.2    177.55
log_mileage      1    79.0    79.0    925.62
Residuals    56246  4803.4     0.1
                          Pr(>F)
city_popularity < 0.00000000000000022 ***
assembly        < 0.00000000000000022 ***
body_group      < 0.00000000000000022 ***
log_age         < 0.00000000000000022 ***
log_engine      < 0.00000000000000022 ***
transmission    < 0.00000000000000022 ***
fuel            < 0.00000000000000022 ***
color_group     < 0.00000000000000022 ***
log_mileage     < 0.00000000000000022 ***
Residuals
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1
```

```
Analysis of Variance Table

Response: log_price
                            Df  Sum Sq Mean Sq F value
assembly:transmission        1 293.575 293.575 3437.63
log_age:transmission         1 292.312 292.312 3422.84
assembly:log_age             1  95.568  95.568 1119.06
log_age:log_mileage          1  93.984  93.984 1100.51
assembly:body_group          5 204.242  40.848  478.32
body_group:log_engine        5 203.628  40.726  476.88
city_popularity:assembly     2  71.392  35.696  417.98
log_engine:transmission      1  33.154  33.154  388.22
log_age:color_group          4  74.740  18.685  218.79
body_group:log_age           5  87.160  17.432  204.12
                                        Pr(>F)
assembly:transmission       < 0.00000000000000022 ***
log_age:transmission        < 0.00000000000000022 ***
assembly:log_age            < 0.00000000000000022 ***
log_age:log_mileage         < 0.00000000000000022 ***
assembly:body_group         < 0.00000000000000022 ***
body_group:log_engine       < 0.00000000000000022 ***
city_popularity:assembly    < 0.00000000000000022 ***
log_engine:transmission     < 0.00000000000000022 ***
log_age:color_group         < 0.00000000000000022 ***
body_group:log_age          < 0.00000000000000022 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1
```

Figure 5.1: (Linear) ANOVA Tables for 9 Main Effects (left) and Top-10 Interactions (right)

From the ANOVA tables in Figure 5.1, it seems that all main effects are statistically significant at significance level (alpha) of 0.05, meaning that the probability of each respective predictor having the F-value that big given their respective degrees of freedom, is less than 0.05 (in fact, the probabilities are all very close to zero). These pieces of information mean that the null hypothesis (the predictor's contribution to the "baseline model" is not statistically significant) is rejected for each of the individual main effects. Therefore, for the dimension reduction process, all main effects are kept.

In Figure 5.1, it seems that only including the top-5 significant interactions (out of 10) is a reasonable choice for further reducing dimension, since the top-5 significant interactions all

have F-value of over 1,000 and the percentage reduction of F-value from the 5th to the 6th interaction is the greatest among all. An updated ANOVA table is:

```
Analysis of Variance Table

Response: log_price
                       Df  Sum Sq Mean Sq   F value                Pr(>F)
city_popularity         2   456.7   228.4    2228.20 < 0.00000000000000022 ***
assembly                1  1246.5  1246.5   12162.56 < 0.00000000000000022 ***
body_group              5 10486.5  2097.3   20464.19 < 0.00000000000000022 ***
log_age                 1 13321.7 13321.7  129985.45 < 0.00000000000000022 ***
log_engine              1  5430.6  5430.6   52988.99 < 0.00000000000000022 ***
transmission            1   657.1   657.1    6411.50 < 0.00000000000000022 ***
fuel                    2    52.1    26.1     254.27 < 0.00000000000000022 ***
color_group             4    60.7    15.2     147.95 < 0.00000000000000022 ***
log_mileage             1    79.0    79.0     771.30 < 0.00000000000000022 ***
log_age:transmission    1   224.6   224.6    2191.03 < 0.00000000000000022 ***
assembly:log_age        1   313.7   313.7    3060.51 < 0.00000000000000022 ***
assembly:transmission   1   117.7   117.7    1148.85 < 0.00000000000000022 ***
log_age:log_engine      1    32.9    32.9     321.00 < 0.00000000000000022 ***
log_age:log_mileage     1   118.0   118.0    1151.39 < 0.00000000000000022 ***
Residuals           56376  5777.8     0.1
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Figure 5.2: (Linear) ANOVA for Model with 9 Mains and 5 Interactions

The updated ANOVA table shows that all 9 main- and 5 interaction-effects are statistically significant at alpha level of 0.05. And the summary output of the linear regression is printed below:

```
Call:
lm(formula = log_price ~ . + log_age:transmission + log_age:assembly +
    assembly:transmission + log_age:log_engine + log_age:log_mileage,
    data = train)

Residuals:
     Min       1Q   Median       3Q      Max
-2.85848 -0.17516  0.01867  0.18883  2.49485

Coefficients:
                                    Estimate Std. Error t value             Pr(>|t|)
(Intercept)                         7.590833   0.078856  96.262 < 0.0000000000000002 ***
city_popularitymedium              -0.007662   0.004026  -1.903             0.057024 .
city_popularityhigh                -0.012191   0.003449  -3.534             0.000409 ***
assemblyLocal                      -0.468044   0.013546 -34.553 < 0.0000000000000002 ***
body_groupCommercial               -0.688298   0.033878 -20.317 < 0.0000000000000002 ***
body_groupMicro/Mini               -0.615700   0.032579 -18.899 < 0.0000000000000002 ***
body_groupUtility                  -0.319042   0.031750 -10.049 < 0.0000000000000002 ***
body_groupHatchback                -0.518850   0.031604 -16.417 < 0.0000000000000002 ***
body_groupSedan                    -0.383231   0.031567 -12.140 < 0.0000000000000002 ***
log_age                            -0.027384   0.028774  -0.952             0.341256
log_engine                          1.148256   0.009843 116.660 < 0.0000000000000002 ***
transmissionAutomatic               0.194626   0.012844  15.153 < 0.0000000000000002 ***
fuelHybrid                          0.023890   0.010688   2.235             0.025408 *
fuelPetrol                         -0.016801   0.008053  -2.086             0.036947 *
color_groupBlack                    0.075643   0.005960  12.691 < 0.0000000000000002 ***
color_groupSilver                   0.015976   0.005706   2.800             0.005114 **
color_groupOthers                  -0.017873   0.005737  -3.116             0.001837 **
color_groupWhite                    0.006595   0.005121   1.288             0.197753
log_mileage                         0.085272   0.002175  39.204 < 0.0000000000000002 ***
log_age:transmissionAutomatic       0.168207   0.003904  43.084 < 0.0000000000000002 ***
assemblyLocal:log_age               0.157107   0.004262  36.865 < 0.0000000000000002 ***
assemblyLocal:transmissionAutomatic -0.325424   0.009201 -35.367 < 0.0000000000000002 ***
log_age:log_engine                  -0.062133   0.003988 -15.581 < 0.0000000000000002 ***
log_age:log_mileage                 -0.028909   0.000852 -33.932 < 0.0000000000000002 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 0.3201 on 56376 degrees of freedom
Multiple R-squared:  0.8494,    Adjusted R-squared:  0.8494
F-statistic: 1.383e+04 on 23 and 56376 DF,  p-value: < 0.00000000000000022
```

Figure 5.3: (Linear) Summary Output for Model with 9 Mains and 5 Interactions (Pg.19)

Note that the main effect "log_age" is not statistically significant in the regression output but is statistically significant in the ANOVA table. This result indicates that when combined with its interaction effects, "log_age" contributes significantly to the model; however, when considered on its own, "log_age" as a main effect does not contribute to the model linearly.

Since the price in Pakistani Rupee (PKR) is in natural log (as well as age, engine volume and mileage), meaning of the coefficients are different than when they are not in natural log, and thus the interpretations need to be adjusted accordingly.

- For log_age, log_engine, log_mileage:
  - Keeping all other variables constant, for every 10% increase in age, engine volume, or mileage, the car price on average increases by respectively -0.26%, 10.94%, 0.81%, calculated by ln(1+0.1) times the respective coefficients (-0.0274, 1.1483, and 0.0853).
- For categorical variables:
  - Keeping all other variables constant, compared to "base", which is the first values in each variables in the summary statistics (city_popularity = low, assembly = Imported, body_group = Sports, transmission = Manual, fuel = Diesel, color_group = Gray), each variable on average would result in increase of the car price by $(e^{\beta_i} - 1) \times 100\%$;
  - For example, keeping all others constant, a local car is on average $(e^{-0.4680} - 1) \times 100\% = -37.37\%$ more expensive, or 37.37% cheaper than imported cars, or a black car is on average $(e^{0.0756} - 1) \times 100\% = 7.85\%$ more expensive than gray cars.

Interpretations for coefficients of the interaction terms are included in Appendix E.

Model assumptions of the linear regression model on training are checked in R using their respective popular test methods: Autocorrelation (dependence) of errors using Durbin-Watson Test, heteroskedasticity (non-constant variable) of errors using Breusch-Pagan Test and non-normality of errors using Kolmogorov-Smirnov Test:

Tests	p-value
Autocorrelation using Durbin-Watson Test	0.89
Heteroskedasticity using Breusch-Pagan Test	< 0.00000000000000022
Non-normality using Kolmogorov-Smirnov Test	< 0.00000000000000022

Table 5.1: (Linear) Assumption Testing Results

The test results shows that auto-correlation test is passed but the other tests failed (based on significance level of 0.05), even after attempting to transforming non-normal numerical variables. Nevertheless, the linear model can be used given good performance scores, because the large sample size of the training set (56,400) might still satisfy the normality requirement. After fitting the model onto the training set, the linear model is used to predict on the testing set for calculating performance scores.

The prediction yields the performance scores of testing RMSE of 0.3145 and testing R^2 of 0.8484, which will be compared to the corresponding result of the other models.

Section 5.2: Elastic Net

Elastic net is a regularization and variable selection method that linearly combines the regression regularization (penalty) terms L1 and L2, respectively applied by Lasso (Least Absolute Shrinkage and Selection Operator) and Ridge Regressions, allowing for a more flexible and comprehensive constraint on the coefficients. The elastic net penalized sum of squared error loss function is minimized, namely

$$L(\vec{\beta}) = \frac{\Sigma_{i=1}^{n}(y_i - \beta_0 - \Sigma_{j=1}^{p}\beta_j x_{ij})^2}{2n} + \frac{\lambda}{2}(1-\alpha)\Sigma_{j=1}^{p}(\beta_j)^2 + \lambda\alpha\Sigma_{j=1}^{p}|\beta_j|$$

where λ represents the penalty parameter and α is a parameter that weights the contribution of the L1 and the L2 penalty terms. Notice that when α equals 0, the loss function is equal to that of Ridge, and when α equals 1, the loss function is equal to that of Lasso.

Compared to linear regression, Elastic Net is better in both facilitating the reduction of the complexity of models by performing dimension reduction and stabilizing the prediction model when there is multicollinearity among predictors.

In this paper, Elastic Net serves as a bridge between linear regression and more complex tree-based methods like Random Forest and XGBoost. The essence of applying Elastic Net model on the training set is the detailed process of tuning the parameters, which is important for optimizing the balance between bias and variance, so that the model fitting on unseen data (like the manually separated testing set) can produce relatively accurate and consistent results. The parameter tuning process and the model fitting process are described below, and R package of "glmnet" is utilized with seed setting to 1.

- The elastic net model is fitted using α values between 0 (equivalent to Ridge) and 1 (Lasso), in increments of 0.01.
- For each α , an Elastic Net model using 10-fold cross validation is fitted on the training set; the model fit result is then returned. The result includes a vector of mean cross-validation errors corresponding to each value of λ . Then, the λ which corresponds to the lowest cross-validation error is chosen.

- Eventually, the α value corresponding to the lowest error is 0.77. This α value is used to fit the final elastic net model.

The λ is 0.0003, and the corresponding mean of cross-validation error is 0.0984. The output of coefficients of the elastic model is printed below:

47 x 1 sparse Matrix of class "dgCMatrix"	s1
(Intercept)	6.56218519783
(Intercept)	.
city_popularity	.
assembly	-0.40601500975
body_group	0.13167797777
log_age	-0.30184638726
log_engine	1.15304887447
transmission	1.09758289645
fuel	-0.03742131883
color_group	-0.05028479348
log_mileage	0.02741733828
city_popularity:assembly	-0.04900160243
city_popularity:body_group	-0.00936483801
city_popularity:log_age	-0.01612590320
city_popularity:log_engine	0.01837308536
city_popularity:transmission	0.03183039375
city_popularity:fuel	.
city_popularity:color_group	-0.00502532782
city_popularity:log_mileage	.
assembly:body_group	0.02216272932
assembly:log_age	0.15831982597
assembly:log_engine	0.02495721669
assembly:transmission	-0.18982373936
assembly:fuel	-0.05254487427
assembly:color_group	0.00935949023
assembly:log_mileage	-0.00008310810
body_group:log_age	-0.03996573826
body_group:log_engine	-0.00208926049
body_group:transmission	-0.15403007501
body_group:fuel	0.02213146530
body_group:color_group	-0.00725237689
body_group:log_mileage	0.01620557004
log_age:log_engine	-0.04960948966
log_age:transmission	0.20681469292
log_age:fuel	0.00158615742
log_age:color_group	0.01512989944
log_age:log_mileage	-0.03097608771
log_engine:transmission	-0.00876032637
log_engine:fuel	.
log_engine:color_group	.
log_engine:log_mileage	.
transmission:fuel	0.00018559805
transmission:color_group	0.03118439006
transmission:log_mileage	-0.01686298318
fuel:color_group	.
fuel:log_mileage	.
color_group:log_mileage	-0.00002732774

Figure 5.4: (Elastic Net) Coefficients of Model with 8 Mains and 29 Interactions

From the output, it seems that 8 main effects are included as well as 29 out of 36 2-way interactions. Specifically,"city_popularity" are not significant on its own and 2 of its interactions, as well as "log_engine" and 3 of its interactions, in addition to "fuel" and 2 of its interactions. Next, two plots would be shown to better understand how to select the optimal λ value, and change in coefficients for different predictors as a function of λ .

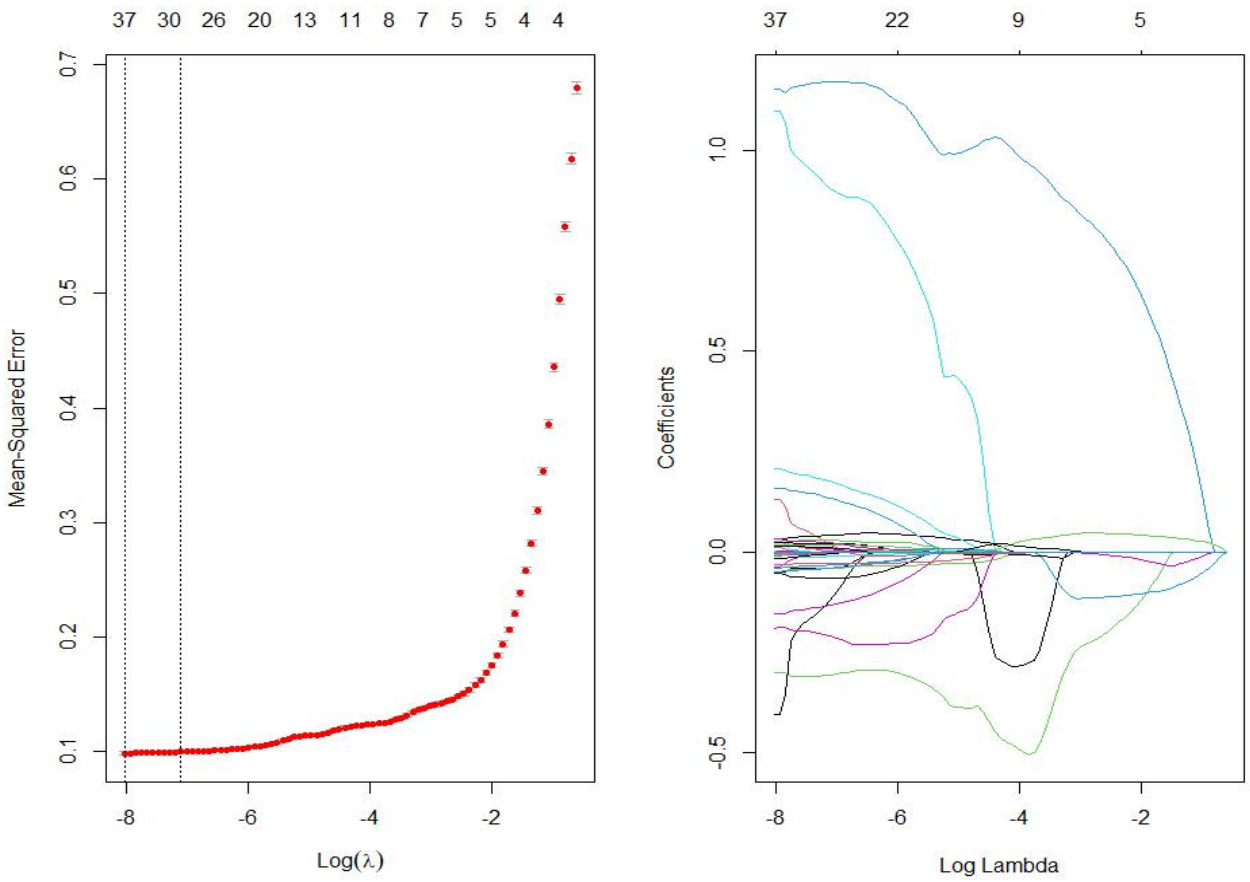


Figure 5.5: (Elastic Net) Cross-Validation Curve (left); Shrinking Coefficients Paths (right)

On the left, the plot shows the cross-validation curve, and is used to assist in selecting the ideal λ value, which is the regularization parameter. Each red dot represents the mean cross-validation error for a specific λ value, and the left of the two vertical dotted lines usually correspond to the λ that minimizes mean cross-validation error. The chosen λ is 0.0003, so the x-axis intercept of that left-side dotted vertical line is approximately ln(0.0005) = -8.11.

On the right, the plot illustrates the paths of the coefficients for different model predictors as λ grows. When λ equals 0.0003 and log(λ) = -8.11, many of the lines (coefficients) have not converge to zero; but as λ grows to a very large value, the vast majority of the coefficients

become zero, indicating that few predictors remains significant when applying a sufficiently large penalty term.

After fitting the elastic net model with chosen α and λ values that minimizes cross-validation error on the training set, the model is used to predict on the testing set for calculating performance scores. For the Elastic Model with α equals 0.77, the testing RMSE of Elastic Net is 0.3083, which is slightly lower than that of the linear model (0.3145), and the testing R^2 of 0.8543 is slightly higher than that of the linear model (0.8484). Based on these pieces of information, The elastic net model with 8 main effects and 29 interaction terms is more ideal compared to the linear model with all main effects and 5 most significant interaction terms.

Next, the Random Forest method will be introduced and model will be fitted and evaluated.

Section 5.3: Random Forest

Random Forest, or Random Decision Forest, is an tree-based ensemble learning method for classification, regression and other tasks that functions by combining the predicting power or multiple decision trees together, which typically has better accuracy achievable by any single tree. Furthermore, combining multiple trees, the risk of overfitting is mitigated - a common risk in complex predictive methods. Random Forest was chosen for the regression task for this paper because it returns the mean or average prediction of the individual trees, thus anomalies in prediction are smoothed out, and the underlying trends of the prices are usually captured with better accuracy.

The creation of a random forest involves constructing a set of decision trees through bagging (bootstrap aggregating), which itself can reduce overfitting and increase stability of the

several decision trees by training and combining multiple models on different subsets of the training set. In addition to bagging, A random forest introduces additional randomness by considering a random subset of features at each split (node) instead of all features, and as a result the forest gives more importance to important features, implicitly performing feature selection.

Just like the Elastic Net model, the Random Forest model can also handle high-dimensional datasets and effortlessly capture non-linear relationships and interactions (just like 2-way interactions previously considered) without the need for explicit engineering of the features. Also, similar to Elastic Net, carefully tuning the Random Forest parameters are of superior significance in optimizing the results. For Random Forest, the parameters usually include the minimum size of terminal nodes ("min.node.size") and the number of features considered as candidates in each split ("mtry"), etc. The model fitting process is described below, R packages of "caret" and "ranger" packages are utilized, and seed is again set to 1.

- The training control method is again 10-fold cross validation, and splitting rule is set to variance. A 3 by 3 tuning grid of possible hyper-parameters (a total of 9 combinations) is created, setting 10, 15, and 20 for both "mtry" and "min.node.size", and will be searched on by the model fitting method later.
  - Random Forest Model is trained using the "ranger" package. The training process searches on the tuning grid, with the total number of trees set to 100, importance calculation method set to "impurity" (the importance based on the impurity decrease contributed by each variable) and performance metric set to RMSE.

The final random forest model selected by the training process is the model with "mtry" equals 10 and "min.node.size" equals 20, as this combination of parameters has the lowest RMSE.

```
Random Forest

56400 samples
   46 predictor

No pre-processing
Resampling: Cross-Validated (10 fold)
Summary of sample sizes: 50761, 50760, 50761, 50760, 50760, 50760, ...
Resampling results across tuning parameters:

  mtry  min.node.size  RMSE       Rsquared   MAE
  10    10            0.2027572  0.9396006  0.1311974
  10    15            0.2018465  0.9401392  0.1304502
  10    20            0.2016186  0.9402769  0.1301073
  15    10            0.2038269  0.9389703  0.1316973
  15    15            0.2025060  0.9397511  0.1307183
  15    20            0.2018295  0.9401549  0.1302643
  20    10            0.2043776  0.9386466  0.1320868
  20    15            0.2027205  0.9396250  0.1309044
  20    20            0.2026260  0.9396790  0.1305903

Tuning parameter 'splitrule' was held constant at a value
 of variance
RMSE was used to select the optimal model using the
 smallest value.
The final values used for the model were mtry = 10,
 splitrule = variance and min.node.size = 20.
```

Figure 5.6: (Random Forest) 100-tree Model Result by Each Combination of Parameters

Furthermore, a graph is shown for the most important features from the random forest model with the most important features shown on the top. Note that the importance ranking did not distinguish between interaction terms and main effects so interpretation of the result should be done with more caution.

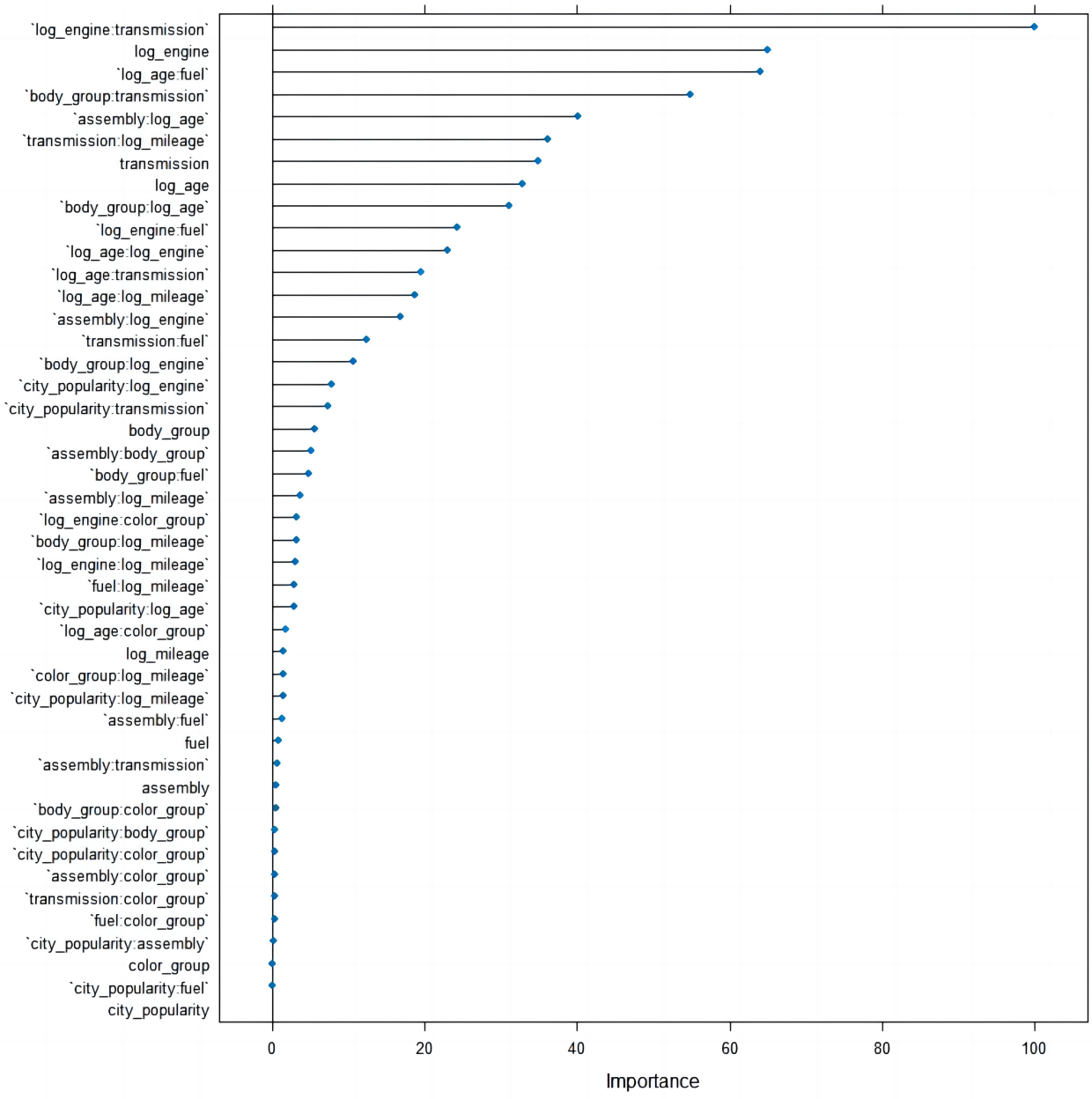


Figure 5.7: (Random Forest) Model Variable Importance Ranking Plot

From the plot, it seems that “log_engine:transmission”, “log_engine”, “log_age:fuel”, “body_group:transmission, “assembly:log_age”, “transmission:log_age”, “transmission”, “log_age”, “body_group:log_age” are some of the most important features according to the

random forest model. Therefore, the Random Forest model shows that features like “log_engine”, “transmission”, “log_age” and “body_group” are important in determining the price of cars.

Then the model is used to predict on the testing set to evaluate performance. For a 100-tree Random Forest model with parameters of “mtry” equals 10 and “min.node.size” equals 20, testing RMSE of 0.1971 is a lot lower than the linear (0.3145) and the elastic net (0.3083) models, and testing R^2 of 0.9405 is also a lot higher than that of the linear (0.8484) and the elastic net (0.8543) models, making random forest more ideal compared to the linear and elastic net models.

Next, a XGBoost model will be fitted, and its performance will decide if Random Forest or XGBoost model will be used as the ideal model.

Section 5.4: XGBoost

XGBoost, abbreviation for eXtreme Gradient Boosting, initially released in the mid-2010s, is an advanced implementation of gradient boosting algorithms, designed to optimize both speed and performance, making it suitable for large datasets. XGBoost has gained immense popularity since then in machine learning competitions and real-world applications due to its capability to deliver high-performance models with a relatively simple set of hyper-parameters compared to traditional gradient boosting methods.

Similar to Random Forest, XGBoost uses an ensemble of decision trees as their base learners and the final prediction is made by aggregating the predictions from all individual trees. However, unlike Random Forest, which builds each decision tree separately and combines their predictions by summing or averaging, XGBoost builds tree sequentially; in fact, it builds trees in a way that each subsequent model attempts to correct the errors made by the previous trees. This

sequential process proceeds until a certain condition has met (like a specific number of tress are constructed, no further improvements can be made).

XGBoost also often performs better than Random Forest models in predictive tasks, especially on datasets where the relationship between features and target is complicated and non-linear; in addition, XGBoost has a wider range of hyper-parameters to be tuned (like for parameters control regularization process and boosting process etc.), thus having more control over the model than Random Forest, but also usually means longer training times despite being a more scalable and efficient model. For this analysis, the parameter tuning process includes learning rate ("eta", takes on values from 0 to 1 which corresponds to gradual increase of regularization strength), maximum depth of tree ("max_depth"), and subsample ratio of number of observations of the training set before growing trees ("subsample").

For the modeling process, R package of "xgboost" is used and seed is again set to 1.

- The learning objective is regression with minimized loss function, and the hyper-parameter grid is defined manually with 3 parameters having 3 values each (27 combinations):
  - "eta" is set to 0.05, 0.1, and 0.15; "max_depth" is set to 5, 10, and 15; and "subsample" is set to 0.5, 0.7, and 0.9.
- For each of the parameter combination, a 10-fold cross-validation is performed with 100 maximum boosting iterations, and training stops if RMSE hasn't decreased for 10 rounds.

The final XGBoost model selected by the training process is the model with "eta" equals 0.1, "max_depth" equals 10 and "subsample" equals 0.9, as this combination of parameters has the lowest RMSE.

Similar to before, a graph is shown for the most important features from the xgboost model, with the most important shown on the top.

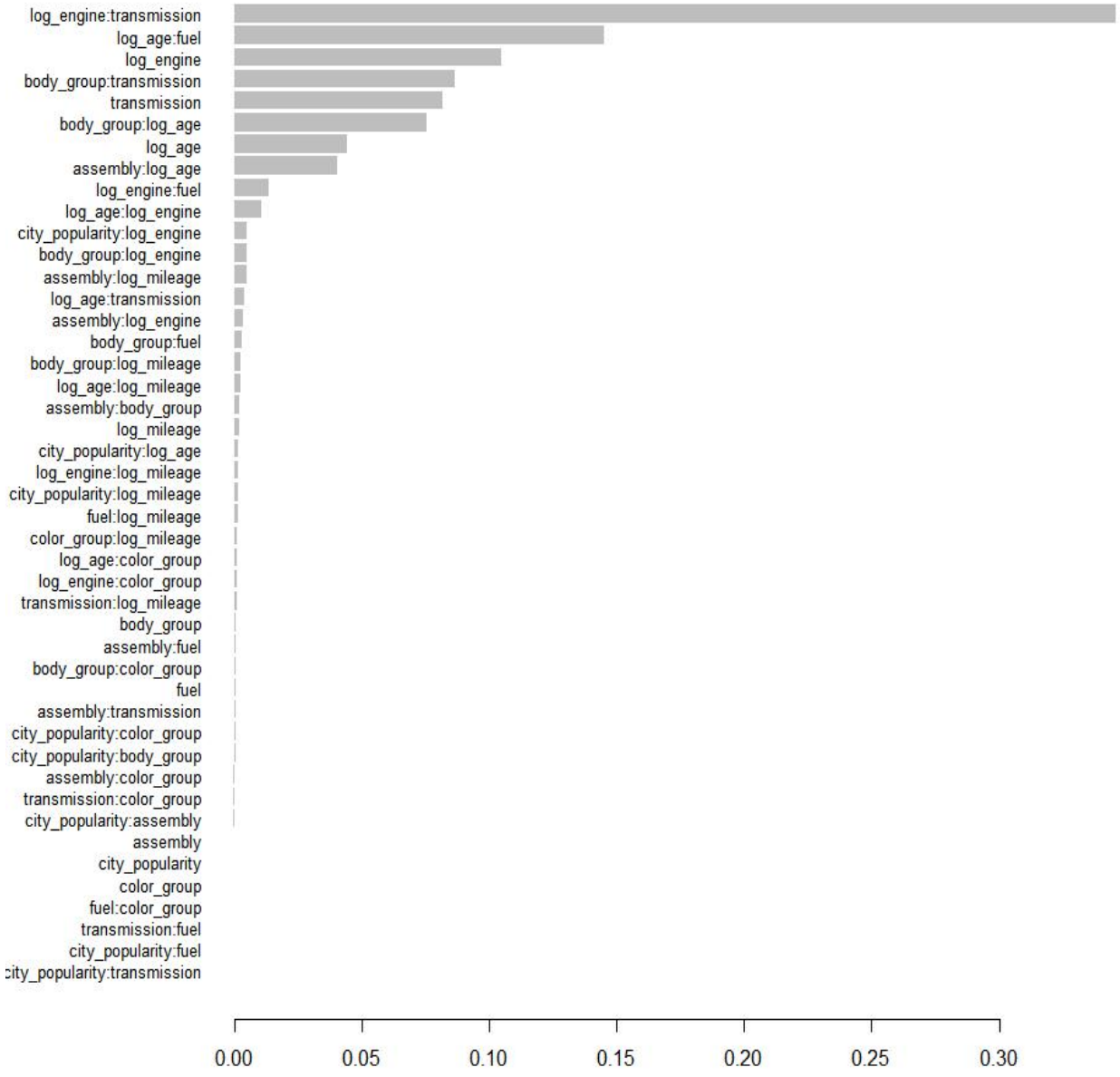


Figure 5.8: (XGBoost) Model Variable Importance Ranking Plot

From the plot, compared to the one for Random Forest model, it seems that "log_engine:transmission" is still the most important but its importance is much more prominent; "log_age:fuel", "log_engine", "body_group:transmission", "transmission",

"body_group:log_age", "log_age", "assembly_log_age" follows right after. It seems that most of the top predictors are same for XGBoost and Random Forest Model. Same as Random Forest, the XGBoost model shows that features like "log_engine", "transmission", "log_age" and "body_group" are important in determining the price of cars.

Then the model is used to predict on the testing set to evaluate performance. For a 100-maximum-iterations XGBoost model with "eta" of 0.1, "max_depth" of 10, and "subsample" of 0.9, the testing RMSE of 0.1965 is slightly lower than the Random Forest (0.1971) and largely lower than the linear (0.3145) and the elastic net (0.3083) models. On the other hand, the testing R^2 of 0.9409 is also slightly higher than that of random forest (0.9405) and largely higher than that of the linear model (0.8484) and the elastic net model (0.8543).

Based on the performance scores of testing RMSE and R^2, XGBoost Model stands out as the best performing model among the four models in this paper: Linear Regression, Elastic Net, Random Forest and XGBoost.

Chapter 6: Conclusion, Limitations and Future Works

Section 6.1: Conclusion

The objective of this paper is to use several statistical learning methods to find the relationship between features of Pakistani used cars and their price. Four methods are used in this paper, and their model performances based on testing RMSE and R^2, along with chosen model parameters after applying 10-fold cross validation on tuning the hyper-parameters, are summarized into a table below.

Model Name	Model Parameters after 10-Fold Cross Validation (In italics if the parameter was tuned)	RMSE (lower = better)	R^2 (higher = better)
Linear	N/A	0.3145	0.8484
Elastic Net	● α *(alpha) = 0.77* ● Penalty term λ (lambda) = 0.0003	0.3083	0.8543
Random Forest	● *feature size at split = 10* ● *minimum node size = 20* ● number of trees = 100	0.1971	0.9405
XGBoost	● *learning rate = 0.01* ● *maximum tree depth = 10* ● *row ratio before trees = 0.9* ● column ratio during trees = 0.7 ● maximum boosting steps = 100 ● training stops if RMSE hasn't decreased for 10 rounds	0.1965	0.9409

Table 6.1: (Conclusion) All Sets of Performance Evaluation Scores (testing RMSE and R^2)

XGBoost is chosen as the final model due to its outstanding model performance evaluation scores. In addition, from the importance ranking of features, "log_engine:transmission" is the most important. Following that feature are "log_age", "log_engine", "fuel" as well as interactions related to these factors. The "body_group" feature is not important alone but is important interacting with "transmission" and "log_age". Therefore, the XGBoost model implies that age, engine size, transmission type and body type of the used car are particularly useful in determining the sale price. These features are agreed by the Random Forest Model, while the Linear and Elastic Model puts more importance on the interaction between age and the other three factors.

It is agreed by the four models age, engine size, transmission type and body type are some of the most influential factors alone, as well as interacting effects between age and the other three factors. Therefore, using these pieces of information, Pakistani used cars buyers and private car sellers can be better informed about their purchase, and used car dealers' business

decision makings can be more reasonable and they can become more financially and popularly successful.

Section 6.2: Limitations and Future works

While contributing valuable insights into predicting Pakistani used car prices, this study has several limitations that are worth to be noticed and some future works that can be done.

Firstly, this dataset contains data only from the year 2023 and mainly sourced from Kaggle and not from primary sources like used car companies or government institutions. Not enough primary (and secondary) sources also means insufficient incorporation of interdisciplinary research involving economics, public transportation or urban design, thus this paper may not capture the full spectrum of the relationships of used car prices and their features and potentially affect the generalization and accuracy of the findings. In the future, collaborating with primary and secondary source providers, including industry stakeholders, government institutions and professionals in the field, as well as expanding ranges of time and regions, could further bridge the gap between theoretical models and practical applicability about understanding the relationships between car features and their final sale prices.

Additionally, imperfect dimension reduction methods were applied and some necessary assumptions were violated when fitting the linear regression model (despite efforts to transform non-normal numerical variables) thus might have caused the choice of final model to be inappropriate. Therefore, more methods of dimension reduction will be considered like Principal Component Regression (PCR) and Partial Least Squares (PLS), so more assumptions might be passed and the prediction power of the model might be stronger.

Likewise, hyper-parameter tuning processes of the Random Forest and XGBoost Models only involve searching grids with limited combinations of parameters due to constrained computation power and insufficient prior knowledge about the effects of the different combinations, possibly oversimplifying complex relationships. As a result, measures will be taken to expand the choices of values of parameters for tuning by including Random Search, which samples a random subset of the potential parameter space, and Sequential Model-Based Optimization (SMBO) methods. These methods might require expanded computation power and a cluster computing system could be utilized in further research.

Furthermore, the re-categorization process for categorical variables (like body type, color and city) might need refining, because some values might fit into more than one group, or that the process itself might obscure the differences between groups. Besides, the decision to drop certain variables from the original dataset, the decision to drop or impute the missing or empty values, and the method of imputation can also be improved. This way, there could be a better balance between preventing the loss of valuable information, preventing the removal of columns with significant relevance, and preventing the imputation of non-representative known values onto the missing values. For missing values, imputing them with mean or median values or using K-Nearest Neighbors (KNN) Imputation might result in better model performances.

Lastly, data splitting (into training and testing sets) can include more options, even though the 80%-20% random split method used in this paper is the most conventional method. Subject to computation power constraints, Leave-p-Out, Monte-Carlo Cross Validation or K-Fold Validation might be potential data splitting candidates that could improve the performance of the model. On the other hand, additional model evaluation methods like Adjusted-R^2 (for drastically different amount of predictors used between each method), Mean Absolute Error or

Median Absolute Deviation (for numerical predictors that are far from being approximately normal even after transformations) could be useful in determining the ideal model for answering the research questions.

Appendix A: R Code Chunks for Data Preprocessing

```{r read_data}
Library(tidyverse)
# read data
car_original <- read.csv("pakwheels_used_car_data_v02.csv")
```

```{r make_copy}
# check the structure of dataset and make a copy of original
str(car_original)
head(car_original)
car_processed <- car_original
```

```{r price_column}
# check if there is any missing or empty values in price
sum(is.na(car_processed["price"]) | car_processed["price"] == "")
# first, drop all observations without price (583 rows, less than 10% of total rows)
# 77878 -> 77295 rows and 14 columns
library(tidyr)
car_processed <- car_processed %>% drop_na(price) %>% filter(price != "")

# by visual inspection, the highest price is a Toyota Corolla
# and is appproximately 3 times the price of the next, so we put 529,000,000 as
5,290,000 Rupees

max_price <- max(car_processed$price)
car_processed[car_processed$price == max_price, ]$price <- 5290000

dim(car_processed)
```

```{r check_missingvalues_predictors}
# check the count of missing or empty observations for all columns
colSums(is.na(car_processed) | car_processed == "")
```

```{r columns_removed}
# check if addref has any missing, empty, or duplicate variables (result: none)
sum(duplicated(car_original$addref))
sum(is.na(car_processed["addref"]) | car_processed["addref"] == "")
# since there is no missing nor duplicate values in "addref" (id), we can delete it,
left with 13 columns
car_processed <- car_processed[, -which(names(car_processed) == "addref")]
dim(car_processed)
```

```
# register city (registered) not too important and practical

# Instead of performing an analysis on models and corresponding manufacturers,
# we are more interested in their features
# so models and manufacturers removed
# we remove those three columns left with 10 columns
car_processed <- car_processed[, -which(names(car_processed) == "registered")]
car_processed <- car_processed[, -which(names(car_processed) == "make")]
car_processed <- car_processed[, -which(names(car_processed) == "model")]

# check dimension of remaining dataframe (n = 77295 k = 10)
dim(car_processed)
# check the count of missing or empty observations for all columns
colSums(is.na(car_processed) | car_processed == "")
```

```{r body_column}
# (missing 8857, little over 10% of total), but body is important, so kept, and
imputation will be on the missing or empty values
library(tidyr)
# change the name of body to body_group
names(car_processed)[3] <- "body_group"

# check the count of body_group from top to bottom
car_processed %>% count(body_group) %>% arrange(desc(n))

# change body to different categories
car_processed = car_processed %>%
  mutate(body_group = case_when(body_group %in%
        c("Sedan", "Compact sedan") ~ "Sedan",
                body_group %in% c("Hatchback", "Compact hatchback", "Station Wagon")
~ "Hatchback",
                body_group %in% c("Pick Up", "SUV", "Compact SUV", "Double Cabin",
"Off-Road Vehicles", "MPV", "Crossover", "Single Cabin") ~ "Utility",
                body_group %in% c("Van", "Truck", "High Roof") ~ "Commercial",
                body_group %in% c("Convertible", "Coupe") ~ "Sports",
                body_group %in% c("Micro Van", "Mini Van", "Mini Vehicles") ~
"Micro/Mini",
                body_group == "" ~ ""))

car_processed %>% count(body_group) %>% arrange(desc(n))

# Proportional imputation
set.seed(1) # for reproducibility
prob <- c(30827, 25193, 9586, 1853, 847, 132) / sum(c(30827, 25193, 9586, 1853, 847,
132))
body <- c('Sedan', 'Hatchback', 'Utility', 'Micro/Mini', 'Commercial', 'Sports')
```

```
# For each empty value, randomly choose a body type based on the known proportions
imputed_body <- sample(body, size = 8857, replace = TRUE, prob = prob)
car_processed$body_group[car_processed$body_group == ""] <- imputed_body

car_processed %>% count(body_group) %>% arrange(desc(n))

# change "body" character column, to factor column
car_processed$body_group = factor(car_processed$body_group, levels = c("Sports",
"Commercial", "Micro/Mini", "Utility", "Hatchback", "Sedan"))
```

```{r assembly_column}
# check unique values of "assembly"
unique(car_processed$assembly)
# in "assembly" variable, change empty values to "local"
car_processed$assembly <- ifelse(car_processed$assembly == "", "Local", "Imported")
# change "assembly" character column, to factor column
car_processed[, "assembly"] <- as.factor(car_processed[, "assembly"])
# check unique variable of "assembly", updated
unique(car_processed$assembly)
# check dimension of remaining dataframe (unchanged)
dim(car_processed)
```

```{r year_column}
# since data is from 2023, we would change name "year" to "age", and calculate the
respective ages
names(car_processed)[4] <- "age"
car_processed$age <- 2023 - car_processed$age

# drop all observations without age (4623 rows, less than 10% of total rows)
# and by common knowledge, (age of car) would be very important in predicting the
price of car
# 77295 -> 72672 rows and 10 columns
library(tidyr)
car_processed <- car_processed %>% drop_na(age) %>% filter(age != "")
dim(car_processed)
# check the count of missing or empty observations for all columns
colSums(is.na(car_processed) | car_processed == "")
```

```{r color_column}
# color would be important in predicting the price of car (paint color)
# change name of color to color_group
names(car_processed)[8] <- "color_group"

# drop all observations without color (1197 rows, less than 10% of total rows)
```

```
# and color can be missing or empty due to a variety of factors
# 72672 rows and 10 columns
library(tidyr)
car_processed <- car_processed %>% drop_na(color_group) %>% filter(color_group != "")
# check the count of each color and rank from top to bottom
car_processed["color_group"] %>% count(color_group) %>% arrange(desc(n))

# in color, if the color contains "White" then its "white and off-white", and so on.

library(stringr)
car_processed = car_processed %>%
  mutate(color_group = case_when(str_detect(color_group, regex("White", ignore_case =
T))~ "White",
                            str_detect(color_group, regex("Black", ignore_case = T)) ~
"Black",
                            str_detect(color_group, regex("Grey", ignore_case = T)) ~
"Gray",
                            str_detect(color_group, regex("Silver", ignore_case = T))
~ "Silver",
                            TRUE ~ "Others"))

car_processed %>% count(color_group) %>% arrange(desc(n))

# check the count of missing or empty observations for all columns
colSums(is.na(car_processed) | car_processed == "")

# change "color" character column, to factor column
car_processed$color_group = factor(car_processed$color_group, levels = c("Gray",
"Black", "Silver", "Others", "White"))
```

```
```{r engine_column}
# only include engine volume 600 to 6600 ml, as they include the majority,
# and by some search, many cars outside this engine volumn range is input incorrectly

library(tidyr)
car_processed <- car_processed %>% drop_na(engine) %>% filter(engine != "")

# create a logical vector that indicates which rows have engine size between 600 and
6600
engine_filter <- car_processed$engine >= 600 & car_processed$engine <= 6600

# subset the dataframe using the logical vector
car_processed <- car_processed[engine_filter, ]

# 71287 -> 71113 rows and 10 columns
dim(car_processed)
# check the count of missing or empty observations for all columns
```
```

```
colSums(is.na(car_processed) | car_processed == "")
```

```{r fuel_column}
# check the count of each fuel and rank from top to bottom
car_processed["fuel"] %>% count(fuel) %>% arrange(desc(n))
# drop all observations without fuel type (613 rows, less than 10% of total rows)
# and by common knowledge, fuel type would be important in predicting the price of car
# 71113 -> 70500 rows and 10 columns
library(tidyr)
car_processed <- car_processed %>% drop_na(fuel) %>% filter(fuel != "")
dim(car_processed)

# check the count of missing or empty observations for all columns
colSums(is.na(car_processed) | car_processed == "")

# change "fuel" character column, to factor column
car_processed$fuel = factor(car_processed$fuel)
```

```{r city_column}
# check the count of each city (city where car is transacted) and rank from top to bottom
car_processed["city"] %>% count(city) %>% arrange(desc(n))

# change the city to 3 groups (popular, moderate, unpopular) based on car transaction count
# First 3 - popular (Lahore, Karachi, Islamabad)
# Rank 4 to 9 - moderate (Rawalpindi, Peshawar, Faisalabad, Multan, Gujranwala, Sialkot)
# Rank 10 and above - unpopular (the rest)
# column name change: city -> city_popularity
names(car_processed)[1] <- "city_popularity"
car_processed$city_popularity <-
  ifelse(car_processed$city_popularity %in% c("Lahore", "Karachi", "Islamabad"), "high",
         ifelse(car_processed$city_popularity %in%
         c("Rawalpindi", "Peshawar", "Faisalabad", "Multan", "Gujranwala", "Sialkot"),
         "medium", "low"))

# change "city_popularity" character column, to factor column
car_processed$city_popularity =
  factor(car_processed$city_popularity, levels = c("low", "medium", "high"))

# check dimension of remaining dataframe (unchanged)
dim(car_processed)
# show the first 10 rows of remaining dataframe
```

```
head(car_processed, 10)
```

```{r transmission column}
# check the count of each transmission and rank from top to bottom
car_processed["transmission"] %>% count(transmission) %>% arrange(desc(n))

# change "transmission" character column, to factor column
car_processed$transmission = factor(car_processed$transmission, levels = c("Manual",
"Automatic"))
```

```{r mileage column}
# no changes needed for mileage
```

Appendix B: R Code Chunks for Exploratory Data Analysis

```
```{r summary_df}
# check the summary statistics of the car_processed dataset
summary(car_processed)
```
```

```
```{r bar_charts}
layout(matrix(1:6, nrow = 2))
# bar plots of categoricals
for (i in c(1,2,6,7)) {
  # Compute the frequency table
  freq_table <- table(car_processed[,i])
  # Order the table by decreasing frequency
  ordered_table <- freq_table[order(-freq_table)]
barplots <- barplot(ordered_table, col = "lightblue",
        main = c("Barplot of",colnames(car_processed[i])), las = 1, cex.names = 1.8,
        ylim = c(0, max(ordered_table) * 1.1))
text(x = barplots, y = ordered_table + 3, labels = ordered_table,
     pos = 3, cex = 1.6, col = "darkgreen")
}
for (i in c(3,8)) {
  # Compute the frequency table
  freq_table <- table(car_processed[,i])
  # Order the table by decreasing frequency
  ordered_table <- freq_table[order(-freq_table)]
barplots <- barplot(ordered_table, col = "lightblue",
 main = c("Barplot of",colnames(car_processed[i])), las = 2,
    cex.names = 1.3, ylim = c(0, max(ordered_table) * 1.1))
text(x = barplots, y = ordered_table + 3, labels = ordered_table,
     pos = 3, cex = 1.3, col = "darkgreen")
}
```
```

```
```{r histograms, fig.width = 10, fig.height = 4}
par(mfrow=c(1,4))
# histograms of numericals
n_breaks = 30
for (i in c(4:5,9:10)) {
hist(car_processed[[i]],main = c("Histogram of",colnames(car_processed)[i]),
     xlab = colnames(car_processed)[i], breaks = n_breaks, cex.main = 1.6, cex.lab =
1.6)
}
```
```

```
```{r approximate_boxcox}
```
```

```
library(MASS)
layout(matrix(1:4, nrow = 2))
boxcox(price~age, data = car_processed)
boxcox(price~engine, data = car_processed)
boxcox(price~mileage, data = car_processed)
```

```{r histograms_after_transformation, fig.width = 10, fig.height = 4}
# add "log_" to column names of all 4 numerical variables
colnames(car_processed)[c(4:5,9:10)] <- paste0("log_",
colnames(car_processed)[c(4:5,9:10)])

# log transform all 4 numerical variables
car_processed[,c(4:5,9:10)] <- log(car_processed[,c(4:5,9:10)])

par(mfrow=c(1,4))
# histograms of numericals
n_breaks = 30
for (i in c(4:5,9:10)) {
hist(log(car_processed[[i]]),main = c("Histogram of",colnames(car_processed)[i]),
     xlab = colnames(car_processed)[i], breaks = n_breaks, cex.main = 1.6, cex.lab =
1.6)
}
```

```{r correlation_of_numericals, fig.width = 80%}
library(corrplot)
# Plot the Correlation
corrplot <- corrplot(cor(car_processed[,c(4:5,9:10)]),
          type = 'lower', tl.col = "black", tl.cex = 1.8, cl.cex = 1.8)
```

```{r scatterplot_of_numericals,fig.width = 8, fig.height = 6}
plot(car_processed[,c(4:5,9:10)], upper.panel = NULL)
```

Appendix C: R Code Chunks for Data Splitting

```r
# separate training and testing, and setting seed for reproducible results
set.seed(1)
train_id <- sample(1:nrow(car_processed),floor(0.8 * nrow(car_processed)))
train <- car_processed[train_id,]
test <- car_processed[-train_id,]
```

Appendix D: R Code Chunks for Linear Model

```{r linear_model_null}
options(scipen = 999)
library(MASS)

# start with all main effects with all 2-way interactions
anova_summary <- anova(lm(log_price~.^2, data = train))
```

```{r linear_model_reduce_process}
# it looks like all main effects should be included
# all 2-way interactions and main effects are too computationally complex
# We choose 2-way interactions effects with 5 highest F value (more than 1000)
# which means 5 lowest p-value, and 5 most significant
# as our final model of linear regression

interaction_terms <- grep(":", rownames(anova_summary), value = TRUE)
anova_summary_main_effects <- anova_summary[!(rownames(anova_summary) %in%
interaction_terms),]
anova_summary_interactions <- anova_summary[rownames(anova_summary) %in%
interaction_terms,]

sorted_anova_summary_top_10 <- head(anova_summary_interactions %>%
  arrange(desc(`F value`)),10)

# View the summary main effects
print(anova_summary_main_effects)
# View the sorted summary interactions show top 10 (take top 5)
print(sorted_anova_summary_top_10)
```

```{r linear_model_reduced}
options(width = 100)
linear_fit_interactions = lm(log_price ~ . + log_age:transmission + log_age:assembly
+
                  assembly:transmission + log_age:log_engine + log_age:log_mileage,
data = train)
anova(linear_fit_interactions)
summary(linear_fit_interactions)
```

```{r linear_model_assumptions}
library(lmtest)
# dw test for autocorrelation
dwtest(linear_fit_interactions)
```

```
# bp test for non-constant variance
bptest(linear_fit_interactions)
# ks test for normality
ks.test(residuals(linear_fit_interactions), "pnorm",
        mean = mean(residuals(linear_fit_interactions)),
        sd = sd(residuals(linear_fit_interactions)))
```

```{r prediction_for_linear_model}
pred_1 <- predict(linear_fit_interactions, test)
# RMSE
RMSE = sqrt(mean((test$log_price - pred_1) ^ 2))
RMSE

#R^2
ss_total <- sum((test$log_price - mean(test$log_price))^2)
ss_res <- sum((test$log_price - pred_1)^2)
r_squared <- 1 - (ss_res / ss_total)
r_squared
```

Appendix E: Interpreting Coefficients for Interaction Terms in Linear Model

- For log_age:log_engine and log_age:log_mileage:
  - Keeping all other variables constant, for every 10% increase in engine or every 10% increase in mileage, given that 10% increase in age happen at the same time, a car on average is respectively 0.59% and 0.28% cheaper, calculated by ln(1+0.1) times the respective coefficients (-0.0621, -0.0289).
- For log_age:assemblyLocal and log_age:transmissionAutomatic:
  - Keeping all other variables constant, a local car (compared to an imported car) or an automatic car (compared to a manual car), given that a 10% increase in age happen at the same time, a car on average is respectively 1.50% and 1.60% more expensive, calculated by ln(1+0.1) times the respective coefficients (0.1571, 0.1682).
- For assemblyLocal:transmissionAutomatic:
  - Keeping all other variables constant, a local and automatic transmission car, compared to a imported and manual transmission car, is on average $(e^{-0.3254}-1) \times 100\% = -27.78\%$ more expensive, or 27.78% cheaper.

Appendix F: R Code Chunks for Elastic Net Model

```
```{r elastic_net_model_setup}
# response column of train and test (converted to matrix)
train_y_matrix <- train[10]
train_y_matrix <- as.matrix(train_y_matrix)
test_y_matrix <- test[10]
test_y_matrix <- as.matrix(test_y_matrix)
# design matrix (train to matrix) and design matrix (test to matrix)
train_design_matrix <- model.matrix(log_price~.^2, data =
data.frame(data.matrix(train)))
test_design_matrix <- model.matrix(log_price~.^2, data =
data.frame(data.matrix(test)))
```
```

```
```{r elastic_net_model}
library(glmnet)
## Elastic Net Model
# Firstly, we would tune the alpha value for the elastic net and choose the alpha
value that minimizes the cross-validation error.
# We create a sequence of alpha values and an empty list to store cv
alpha = seq(0,1,0.01)
cv.list.en = rep(NA, length(alpha))
set.seed(1)

# Then, we create a for loop to iterate through all alpha values
for (i in 1:length(alpha)) {
  en.fit = cv.glmnet(train_design_matrix, train_y_matrix, nfold = 10, alpha =
alpha[i])
  cv.list.en[i] = en.fit$cvm[en.fit$lambda == en.fit$lambda.min]
}

# which alpha value corresponds to the lowest cv error
alpha[which.min(cv.list.en)]

# final model for elastic net
elastic_net_fit = cv.glmnet(train_design_matrix, train_y_matrix, nfold = 10, alpha =
alpha[i])

# The lambda value that minimizes cv error & and minimum mean cv
min(elastic_net_fit$cvm)
elastic_net_fit$lambda.min

# coefficients of elastic net
coef(elastic_net_fit, s = 'lambda.min')
```
```

```{r elastic_net_model_plot, fig.height = 80%}
## Elastic Net Plot
par(mfrow = c(1,2))
plot(elastic_net_fit)
plot(elastic_net_fit$glmnet.fit, 'lambda')
```

```{r prediction_for_elastic_net_model}
pred_2 <- predict(elastic_net_fit, newx = test_design_matrix, s = 'lambda.min')
# RMSE
RMSE = sqrt(sum((test_y_matrix - pred_2)^2) / nrow(test_y_matrix))
RMSE
# R-squared
ss_total_2 <- sum((test_y_matrix - mean(test_y_matrix))^2)
ss_res_2 <- sum((test_y_matrix - pred_2)^2)
r_squared_2 <- 1 - (ss_res_2 / ss_total_2)
r_squared_2
```

Appendix G: R Code Chunks for Random Forest Model

```
```{r random_forest_model_setup}
train_design_y_matrix <- cbind(train_design_matrix,train_y_matrix)
test_design_y_matrix <- cbind(test_design_matrix,test_y_matrix)
library(caret)
library(ranger)
# Set up 10-fold cross-validation
train_control <- trainControl(method = "cv", number = 10, search = "grid")
```
```

```
```{r random_forest_model}
# Define the tuning grid
tune_grid <- expand.grid(mtry = c(10,15,20), min.node.size = c(10,15,20), splitrule =
"variance")

# Train the model
set.seed(1)  # For reproducibility
rf_model <- train(
  log_price ~ .,
  data = train_design_y_matrix,
  method = "ranger",  # Specifies the Random Forest model
  trControl = train_control,
  tuneGrid = tune_grid,
  num.tree = 100,
  metric = "RMSE",  # Choose performance metric, e.g., RMSE for regression tasks
  importance = "impurity"
)

# Print the results
print(rf_model)
```
```

```
```{r random_forest_plot, fig.height = 10, fig.width = 10}
#check variables important plot to find which predictors affect the log_price
seriously,
#the top ones are most important ones
plot(varImp(rf_model))
```
```

```
```{r prediction_rf_model}
#predictions
pred_3 <- predict(rf_model, test_design_y_matrix)
rmse_3 <- sqrt(mean((pred_3 - test_y_matrix)^2))
rmse_3
```
```

```
# R-squared
ss_total_3 <- sum((test_y_matrix - mean(test_y_matrix))^2)
ss_res_3 <- sum((test_y_matrix - pred_3)^2)
r_squared_3 <- 1 - (ss_res_3 / ss_total_3)
r_squared_3
```

Appendix H: R Code Chunks for XGBoost Model

```
```{r xgboost_model_setup}
library(xgboost)

param_grid <- expand.grid(
  eta = c(0.05, 0.1, 0.15),
  max_depth = c(5, 10, 15),
  subsample = c(0.5, 0.7, 0.9),
  colsample_bytree = 0.7,
  objective = "reg:squarederror",
  booster = "gbtree"
)

# Initialize variables to store the best results
best_params <- list()
min_cv_rmse <- Inf
```
```

```
```{r xgboost_model}
# Convert the datasets to XGBoost's DMatrix format
dtrain <- xgb.DMatrix(data = train_design_y_matrix[, -47], label =
train_design_y_matrix[, 47])
dtest <- xgb.DMatrix(data = test_design_y_matrix[, -47], label =
test_design_y_matrix[, 47])

# Loop over the grid
for(i in 1:nrow(param_grid)) {
  set.seed(1)
  # Extract parameters for this iteration
  params <- as.list(param_grid[i,])

  # Perform 10-fold cross-validation
  cv_results <- xgb.cv(
    params = params,
    data = dtrain,
    nrounds = 100,
    nfold = 10,
    showsd = TRUE,
    stratified = FALSE,
    print_every_n = 10,
    early_stopping_rounds = 10,
    maximize = FALSE
  )

  # Find the best round for the current parameter set
```
```

```
  best_rmse <- min(cv_results$evaluation_log$test_rmse_mean)

  # Update best parameters if current model is better
  if (best_rmse < min_cv_rmse) {
    best_params <- params
    min_cv_rmse <- best_rmse
  }

  cat("Finished grid iteration:", i, "/", nrow(param_grid),
      "with RMSE:", best_rmse, "\nBest RMSE so far:", min_cv_rmse, "\n\n")
}

# Print the best parameters
print(best_params)

# Train the final model using the best parameters
best_model <- xgboost(
  set.seed(1),
  params = best_params[1:5],
  data = dtrain,
  nrounds = 100,
  maximize = FALSE
)

print(best_model)
```

```{r xgboost_importance, fig.height = 10}
# Get importance matrix
importance_matrix <- xgb.importance(feature_names = colnames(train_design_matrix),
model = best_model)
print(importance_matrix)
xgb.plot.importance(importance_matrix,cex = 0.8)
```

```{r prediction_for_xgboost_model}
# prediction
pred_4 <- predict(object = best_model, newdata = dtest)
rmse_4 <- sqrt(mean((pred_4 - test_y_matrix)^2))
rmse_4

# R-squared
ss_total_4 <- sum((test_y_matrix - mean(test_y_matrix))^2)
ss_res_4 <- sum((test_y_matrix - pred_4)^2)
r_squared_4 <- 1 - (ss_res_4 / ss_total_4)
r_squared_4
```

References

Ahmad, T. B. (2023, September 29). *Pakistan used car prices 2023*. Kaggle. <https://www.kaggle.com/datasets/talhabarkaatahmad/pakistan-used-car-prices-2023?resource=download>

C. Jin, "Price Prediction of Used Cars Using Machine Learning," 2021 IEEE International Conference on Emergency Science and Information Technology (ICESIT), Chongqing, China, pp. 223-230, 2021.

Laghari, A. (2018a, May 31). *Trends of buying used cars in Pakistan*. PakWheels Blog. <https://www.pakwheels.com/blog/trends-pakistan-used-car/>

S. Peerun, N. H. Chummun and S. Pudaruth, "Predicting the Price of Second-hand Cars using Artificial Neural Networks", The Second International Conference on Data Mining Internet Computing and Big Data, pp. 17-21, 2015.

The World Bank Group. (n.d.). GDP per capita (current US$) - Pakistan. World Bank Open Data. <https://data.worldbank.org/indicator/NY.GDP.PCAP.CD?end=2022&locations=PK&start=1960&view=chart>