

ADVANCED ANALYSIS ON USED BIKE PRICES IN INDIA



GROUP 04

s14853 – Pramudi Rajamanthri
s15030 – Vidura Chathuranga
s15091 – Chalani Wijamunige

ST 3082
Final Project

Abstract

This report aims to outline the findings of the exploratory data analysis which was carried out based on the Used bike prices in India data set obtained from the open-source data site, Kaggle. We have done several comparisons and univariate/bivariate analysis by comparing variables such as; brand, age, ownership, engine power, number of kilometers driven and state where we were able to understand how each of these play a role on the valuation of used bikes. Furthermore, an advanced analysis was carried out on the used bike prices dataset sourced from Kaggle with an objective of analyzing how the price which is our target variable is affected by many different variables on this dataset and a model with high prediction accuracy was selected in predicting the prices of used bikes.

Contents

Abstract.....	1
List of figures.....	1
List of Tables.....	1
1. Introduction.....	1
2. Description of the Question.....	2
3. Description of the data set.....	2
4. Pre-processing.....	2
4.1 Feature Engineering.....	3
5. Important Results of the Descriptive Analysis.....	3
6. Important Results of the Advanced Analysis.....	5
7. Issues Encountered and Proposed Solutions.....	9
8. Discussion and Conclusions.....	10
9. Appendix of the code.....	11

List of Figures

Figure 5.1 : Distribution of Price(In Indian Rupees).....	3
Figure 5.2 : Barplot of bike count Vs brand by ownership	3
Figure 5.3 : Bar plot of bike count Vs state	3
Figure 5.4 : Scatterplot of bike price Vs power.....	4
Figure 5.5 : Scatterplot of bike price Vs age.....	4
Figure 5.6 : Scatterplot of bike price Vs kms_driven.....	4
Figure 5.7 : VIP scores plot.....	4
Figure 5.8 : Silhouette plot.....	4
Figure 5.9 : Cluster plot.....	4
Figure 5.10 : Correlation matrix.....	5
Figure 5.11 : Scatterplot matrix.....	5
Figure 6.1 : Variable importance plot on Random Forest(original response).....	6
Figure 6.2 : Variable importance plot on XGBoost(original response).....	6
Figure 6.3 : Distribution of log transformed price.....	6
Figure 6.4 : Variable importance plot on Random Forest(Log transformed response).....	7
Figure 6.5 : Variable importance plot on XGBoost(Log transformed response).....	7
Figure 6.6 : Distribution of square root transformed price.....	8
Figure 6.7 : Variable importance plot on Random Forest(Square root transformed response).....	9
Figure 6.8 : Variable importance plot on XGBoost(Square root transformed response).....	9
Figure 8.1 : Coefficients plot of the lasso regression model(Square root transformed response).....	10
Figure 8.2 : Scatterplot of predicted Vs observed values on the test data using the selected lasso model	10

List of Tables

Table 3.1 : Variable Summary.....	2
Table 6.1 : Model performance evaluation on the original data.....	5
Table 6.2 : Results of Random Forest and XGBoost models.....	5
Table 6.3 : Model performance evaluation on the log transformed response.....	6
Table 6.4 : Results of Random Forest and XGBoost models on the log transformed response	7
Table 6.5 : Model performance evaluation on the square root transformed response.....	8
Table 6.6 : Results of Random Forest and XGBoost models on the square root transformed response	8
Table 6.7 : Reduced ridge and lasso model performance evaluation on the square root transformed response.....	9

1. Introduction

India is one of the largest markets for two-wheeler vehicles in the world, with over 21 million units sold in 2020 alone (source: Society of Indian Automobile Manufacturers). The motorbike segment accounts for a significant portion of this market, with over 15 million motorbikes sold in 2020. Two-wheelers account for over 80% of the total vehicles sold in India and are a popular mode of transportation for both urban and rural populations.

The Indian motorbike market is highly competitive, with several domestic and international players vying for market share. Also, it is characterized by a wide range of models with varying features, price points, and performance levels, catering to different segments of the market. From entry-level commuter bikes to premium sports bikes, there is a two-wheeler for every budget and need.

The market for used bikes in India has been growing steadily in recent years, as more and more consumers are looking for affordable and reliable transportation options. Used bikes provide a cost-effective alternative to buying a new bike, with prices typically ranging from a few thousand rupees to several lakhs depending on the make, model, and condition of the bike. So, for a vast market like Indian motorbike market, it's essential to have accurate and reliable methods for estimating the value of used motorbikes.

The goal of this project is to develop a predictive model for estimating the resale value of used motorbikes in India. Using data from buying and selling and various attributes related to these motorbikes, we aim to predict their prices accurately. By analyzing this dataset and building a predictive model, we can help individuals and businesses to make informed decisions when buying or selling used motorbikes in India which will be useful to a wide range of stakeholders, including buyers, sellers, insurers, and financiers of used motorbikes.

In the following report, we will describe the process of data cleaning, exploratory data analysis, and model development, along with the results and conclusions of our analysis.

2. Description of the Question

The market for used motorbikes in India is growing, and accurate valuation of these vehicles is crucial for buyers, sellers, and other stakeholders. However, estimating the resale value of a used motorbike can be challenging due to factors such as the bike's age, condition, make, and model. This project aims to address this problem by developing a predictive model that can accurately estimate the resale value of used motorbikes in India, based on various attributes related to these vehicles.

Objectives of the analysis: -

1. Identify the most significant factors that impact the resale value of used motorbikes in India.
2. Develop a predictive model for estimating the resale value of used motorbikes in India.

3. Description of the data set

This dataset contains information about 32648 used motorbikes sold on <https://droom.in/>, an Indian online platform used in buying and selling vehicles. The dataset includes 8 variables which are given below.

Variable	Description	Type
price	Resale price of the bike in Indian Rupees. This is the target variable of the dataset.	Quantitative
bike_name	Full name of the bike. This contains 471 unique categories.	Qualitative
city	City where the bike was listed. This contains 443 unique categories	Qualitative
kms_driven	Total kilometers driven by the bike	Quantitative
owner	Number of owners of the bike 1 – First owner, 2 – Second owner, 3 – Third owner, 4 – Fourth owner	Qualitative
age	Age of the bike in years	Quantitative
power	Power of the bike in CC	Quantitative
brand	Brand of the bike. This contains 23 unique categories.	Qualitative

Table 3.1

4. Data Pre-processing

Used Bikes Prices in India dataset contains data under 8 variables where 4 are categorical and 4 are numerical. In the dataset, out of 32648 records 25324 duplicated entries were discovered and were removed. There were no missing data records in the data set. Finally, cleaned dataset consists of 7324 records.

4.1 Feature Engineering

Initially there were 23 distinct categories in the *brand* variable. After identifying the categories which had the lowest number of cases (Ideal, Indian, LML, Rajdoot, Yezdi, MV and Jawa) they were combined into one category (Others), and a new variable was created named “*brand_New*” with 17 distinct categories.

A variable called “*state*” was created by categorizing the *city* variable which has 443 unique categories, according to the state which they belong out of the 28 states in India.

Additionally, to simplify the analysis and avoid creating too many categories, three states namely, “Arunachal Pradesh”, “Meghalaya”, “Sikkim” with the lowest number of counts were grouped together under the category “Others”. Then, the newly created “*state*” variable contained 25 distinct categories.

Finally, the dataset was randomly split into training and test datasets which contains 80% and 20% of the entries from original dataset respectively.

5. Important Results of the Descriptive Analysis

In the detailed descriptive analysis carried out previously, the behavior of the price was analyzed with several associative factors. Below are some important results obtained.

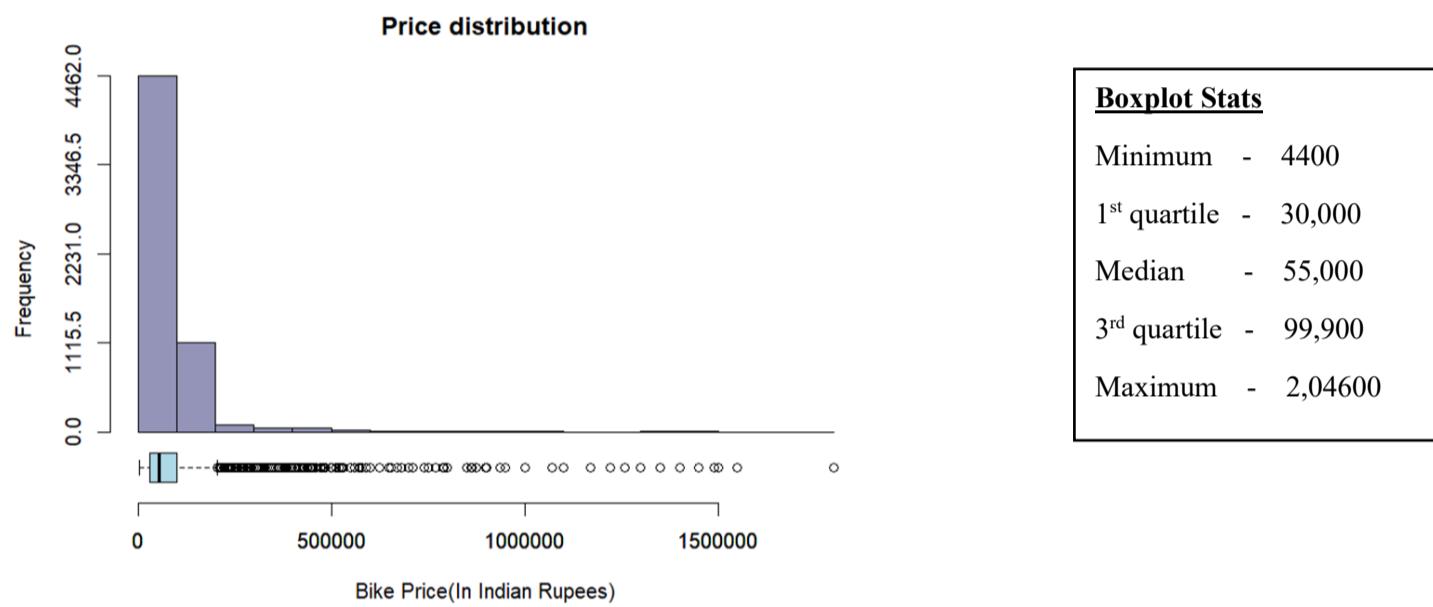


Figure 5.1

- When the price distribution was analyzed, a positively skewed distribution was observed along with 274 outliers. Hence, considering a transformation on the response would help to overcome the issues arising from the above.
- According to the bar plot in Figure 5.2, the order of preference of the used bikes were identified where brands namely, Bajaj, Royal Enfield, Hero and Honda occupy the front positions respectively. Moreover, they are identified as the brands which are well-established all-over India according to the reports of the “globalnewswire” (<https://www.globenewswire.com/en/news-release/2022/06/21/2465834/28124/en/India-Used-Two-Wheeler-Market-Report-2022-2026-A-Highly-Fragmented-Market-with-the-Presence-of-Large-Number-of-Local-Dealers-Small-Mechanics-and-Individual-Sellers.html>) considering their approachable prices regarding the mostly identified requirements for used two-wheeler market among Indians.

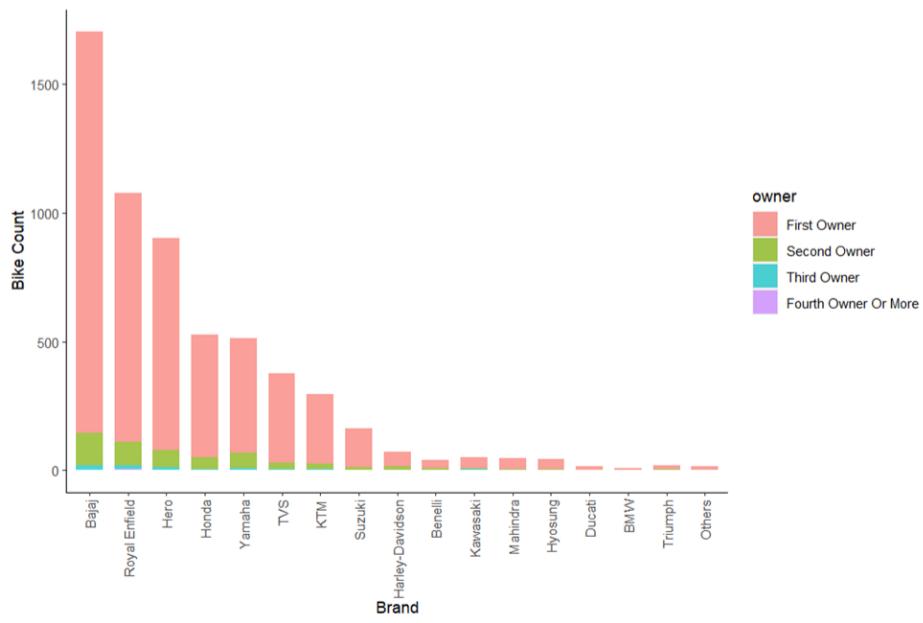


Figure 5.2

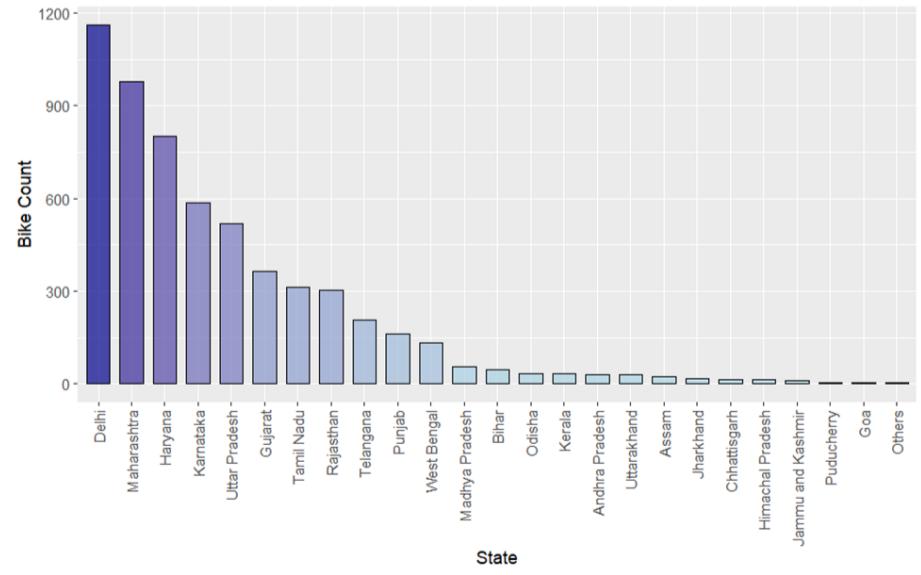


Figure 5.3

- Highly populated states namely, Delhi, Maharashtra seem to have a well-established used bikes market according to the bar plot in Figure 5.3, where “timesofindia” reports (<https://timesofindia.indiatimes.com/auto/bikes/india-is-now-worlds-biggest-2-wheeler-market/articleshow/58555735.cms>) that India has overtaken China in the two-wheeler

market while New Delhi, the Capital of India is identified among the prominent cities where the market is well established.

- According to scatterplot in Figure 5.4, ***power*** of the engine (in CC) depicts an overall positive linear relationship with the bike prices. Even though the fact that the bikes with the less number of kilometers driven with the small years of ownership(***age***) have higher prices, ***kms_driven*** and ***age*** variables do not depict highlighted linear relationships with bike price according to scatterplots in Figure 5.5 and Figure 5.6.

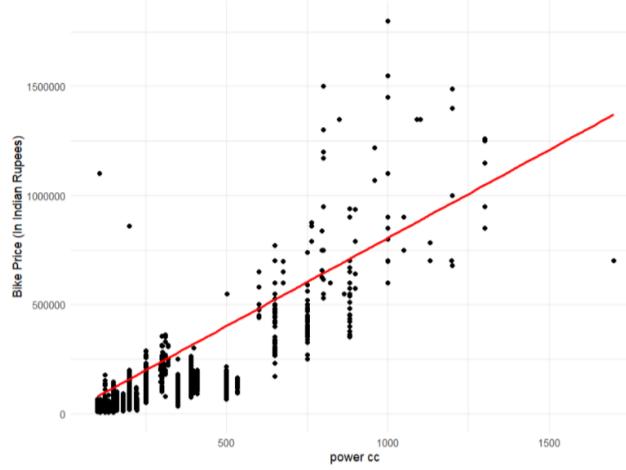


Figure 5.4

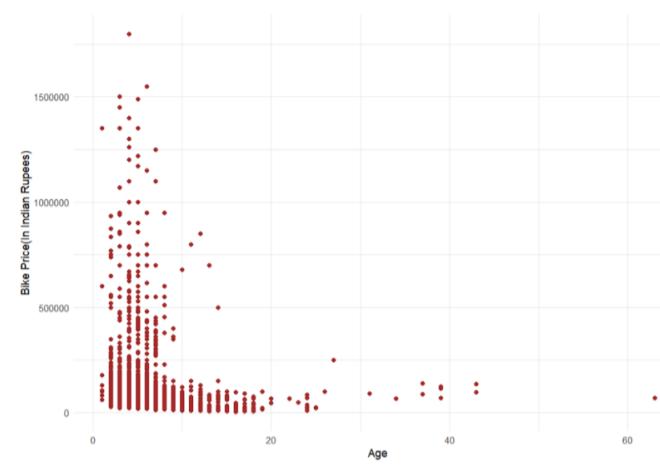


Figure 5.5

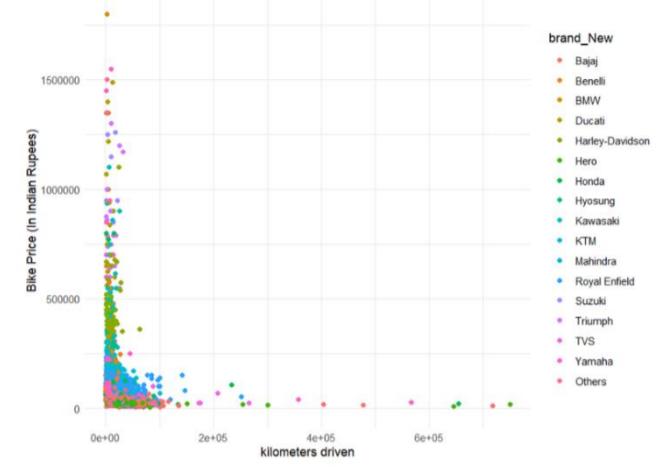


Figure 5.6

- Partial least squares regression (PLSR), resulted dominant VIP scores for ***power***, ***kms_driven*** and ***age***. Moreover, 39 outliers were identified via PLSR.

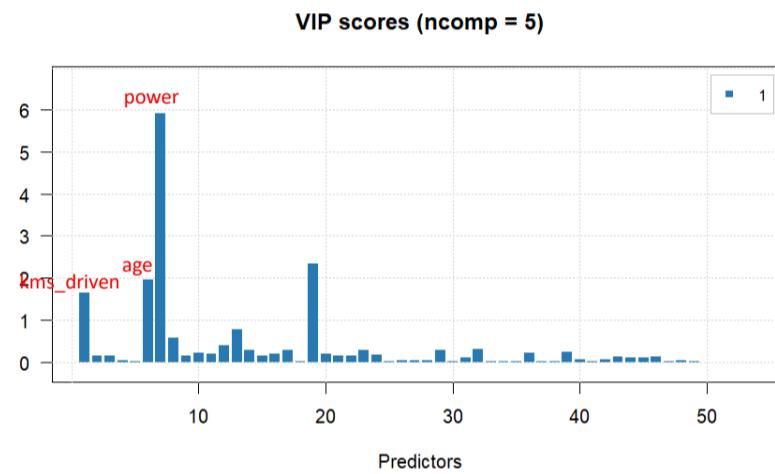


Figure 5.7

- According to Figure 5.8, cluster analysis confirmed the non-existence of clearly distinct clusters where it resulted average silhouette distances less than 0.5 for the considered k values in the range of 2:8 where the highest value was 0.2205 under k=4. Furthermore, Figure 5.9 depicts the unsystematic scattering of the data points with the evidence for the non-existence of clearly distinct clusters.

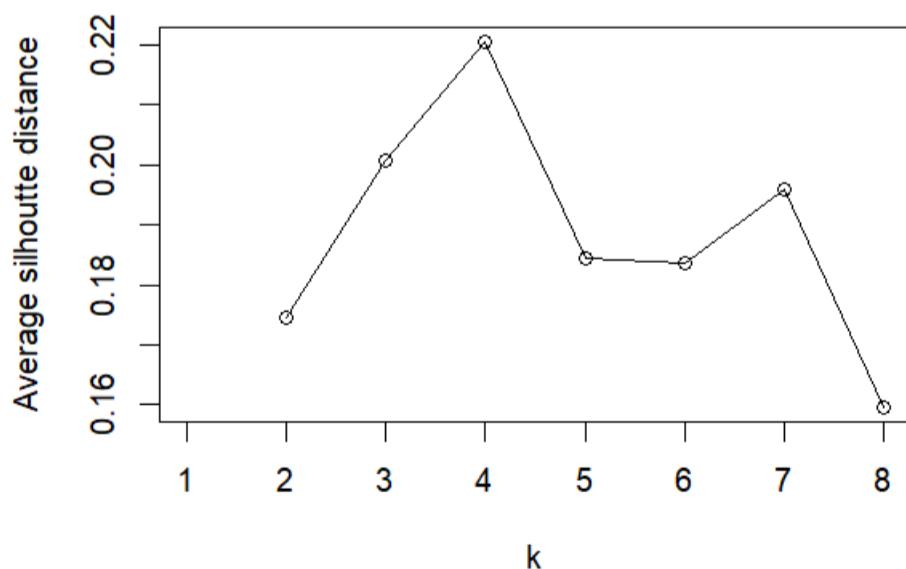


Figure 5.8

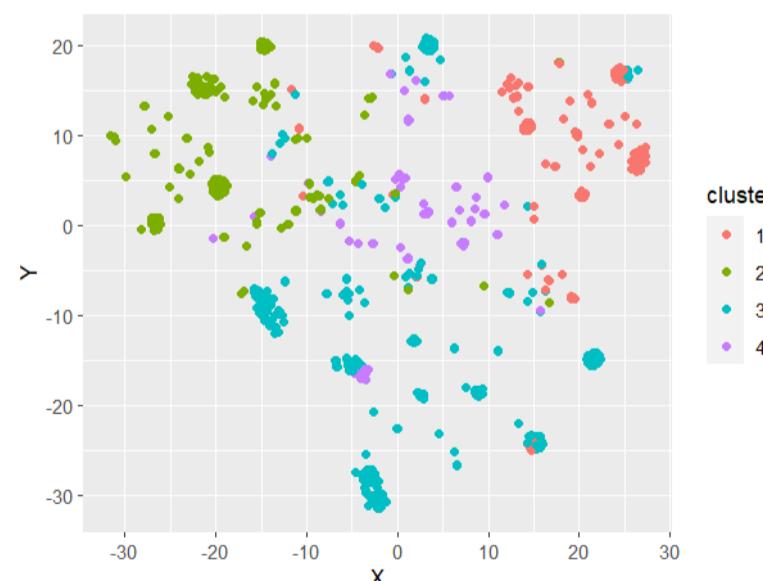


Figure 5.9

- From the correlation matrix in Figure 5.10, a fairly positive correlation was identified among the variables ***kms_driven*** and ***age***, but other correlation values don't show any significance which is more clarified from the scatterplot matrix in Figure 5.11. Hence, we decided not to deny the fact of existence of multicollinearity. Moreover, the kruskal wallis tests resulted significantly small p – values when considering the effect on the median of the numerical variables by the categories of the categorical variables.

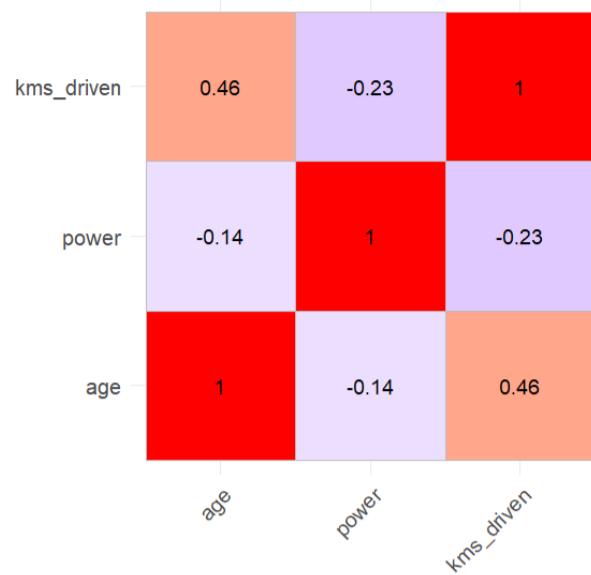


Figure 5.10

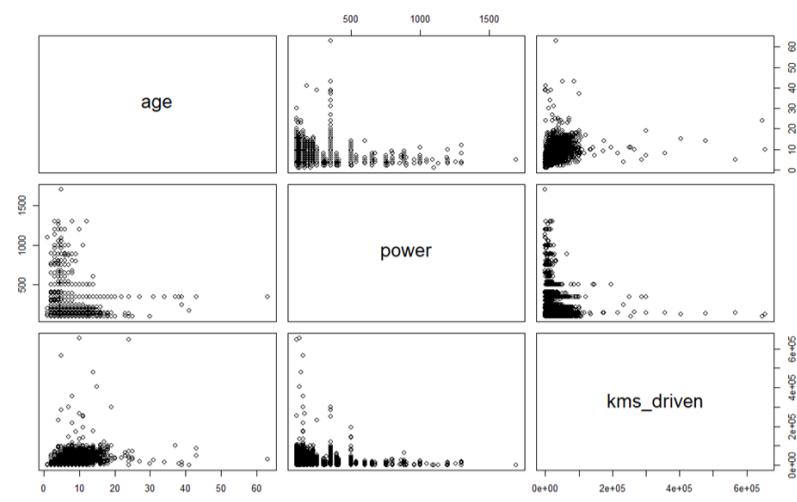


Figure 5.11

6. Important Results of the Advanced Analysis

The predictive models analyzed throughout this analysis are derived with the assumption that the *Price* variable is of continuous data. Accordingly, with regard to the final results of the descriptive analysis mentioned above, fitting a multiple linear regression model is not appropriate due to the presence of associations identified among the predictor variables. Hence, regularization techniques were used to reduce the variance of the coefficient estimates. Table 6.1 below is a model performance evaluation done on the predicted prices and that of actual counts using the Root Mean Squared Error (RMSE) metric with 10- fold cross validation.

Model	Training RMSE	Test RMSE	Training MAPE	Test MAPE
Ridge	29960.91	34176.13	23.31%	25.19%
Lasso	29962.57	34178.73	23.57%	25.43%
Elastic Net	29964.94	38746.6	24.16%	27.81%

Table 6.1

All of the above regularized models indicate approximately similar RMSEs for training data, also it is the same case when it comes to the test RMSEs of ridge and lasso models, while elastic net test RMSE deviates a bit. However, the difference between training and test RMSES of each model suggest overfitting. Obtaining test MAPEs of 25.19%, 25.43% and 27.81% for the three models respectively, indicates that the predictions made by the three models are not fair enough. When considering the training MAPE values, the lower prediction accuracy is much clarified as the model performances on the training data are also unsatisfactory. Hence, selecting a model by considering the above three models would not be effective.

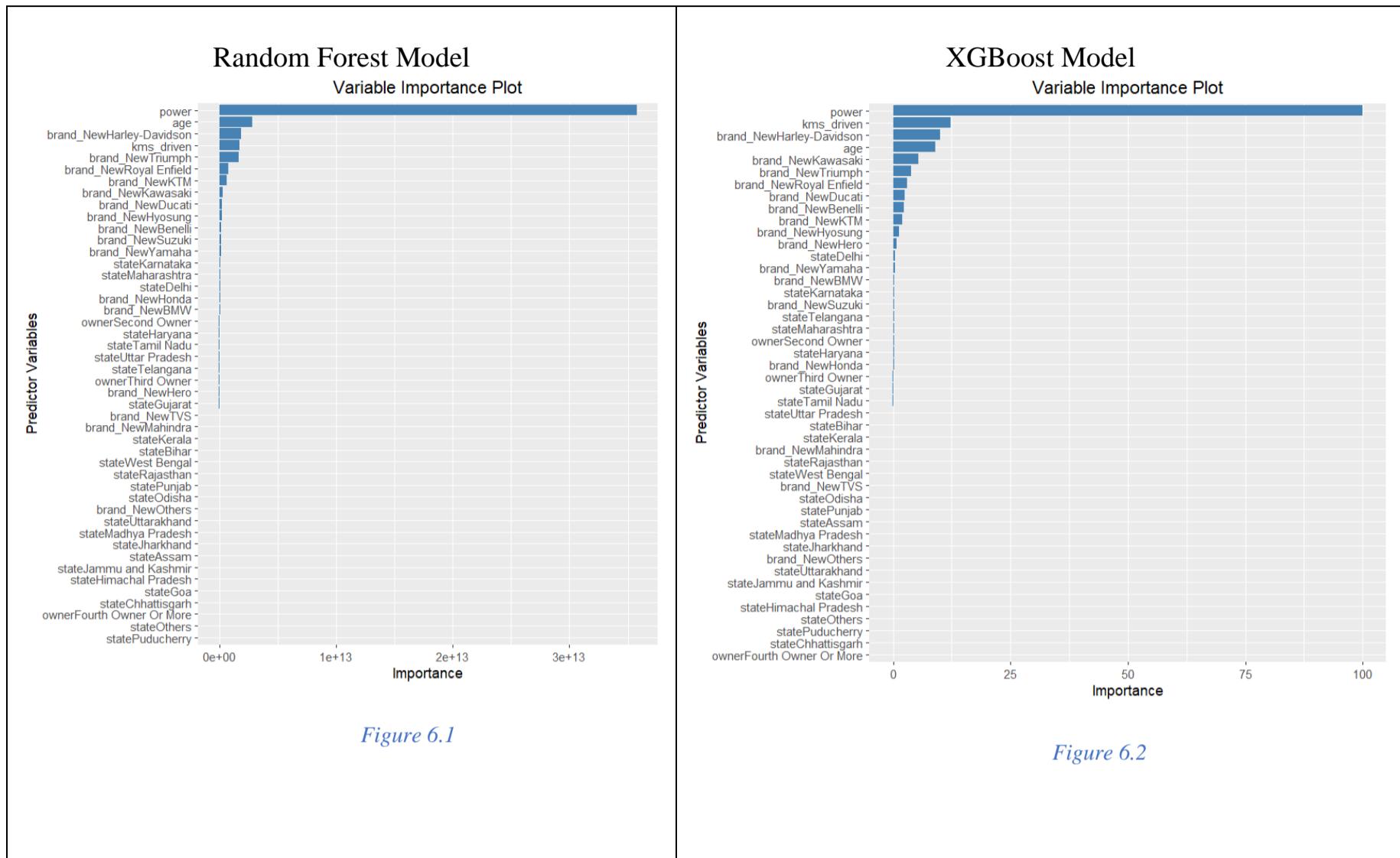
In order to improve the model accuracy further, more sophisticated but that can handle multiple issues simultaneously were applied on the data. Table 6.2 below are the results obtained using random forest and XGBoosting using hyper parameter tuning.

Model	With all the features					With the important features				
	Best Parameters	Training RMSE	Test RMSE	Training MAPE	Test MAPE	Training RMSE	Test RMSE	Training MAPE	Test MAPE	
Random Forest	mtry=30 splitrule=variance min.node.size=5	9753.169	24785.97	8.53%	15.66%	9236.477	25242.09	8.5%	16.13%	
XGBoost	Nrounds=500 max_depth=4 eta=0.1 gamma=0.1 colsample_bytree=0.6 min_child_weight=1 subsample=0.6	11506.04	25009.58	9.06%	15.97%	15463.6	26667.75	9.56%	16.68%	

Table 6.2

Note : The number of important features were selected in random forest and xgboost models was 20 in both cases.

As depicted on Table 6.2, the results obtained when all the features were considered, both the techniques suggest that prediction accuracy is low due to overfitting in terms of both RMSEs and MAPE values. Therefore, after extracting the most important features, the techniques were applied again. But, it didn't remedy the overfitting issue . Figure 6.1 and Figure 6.2 depict the variable importance plots in this scenario regarding the random forest and xgboost models.



The response variable's discreteness and the possession of a positively skewed distribution indicates that using a transformation on the response might produce more accurate predictions. Therefore, **log transformation** is imposed on the response, **Price** .



	Response – Price	Response – ln(Price)
Skewness	4.769269	0.3629236
Kurtosis	35.56085	3.152006

Table 6.3 below is a model performance evaluation done on the predicted delays after transforming them back to the original scale and that of actual counts using the Root Mean Squared Error (RMSE) metric with 10- fold cross validation.

Model	Training RMSE	Test RMSE	Training MAPE	Test MAPE
Ridge	68326.92	69562.18	32.24%	38.39%
Lasso	65687.94	69903.1	32.16%	38.43%
Elastic Net	51363.52	52736.99	31.91%	33.25%

Table 6.3

Above results show that the log transformation has not been successful in fitting linear relationships between the response and predictors with higher RMSEs suggesting the lower prediction accuracy.

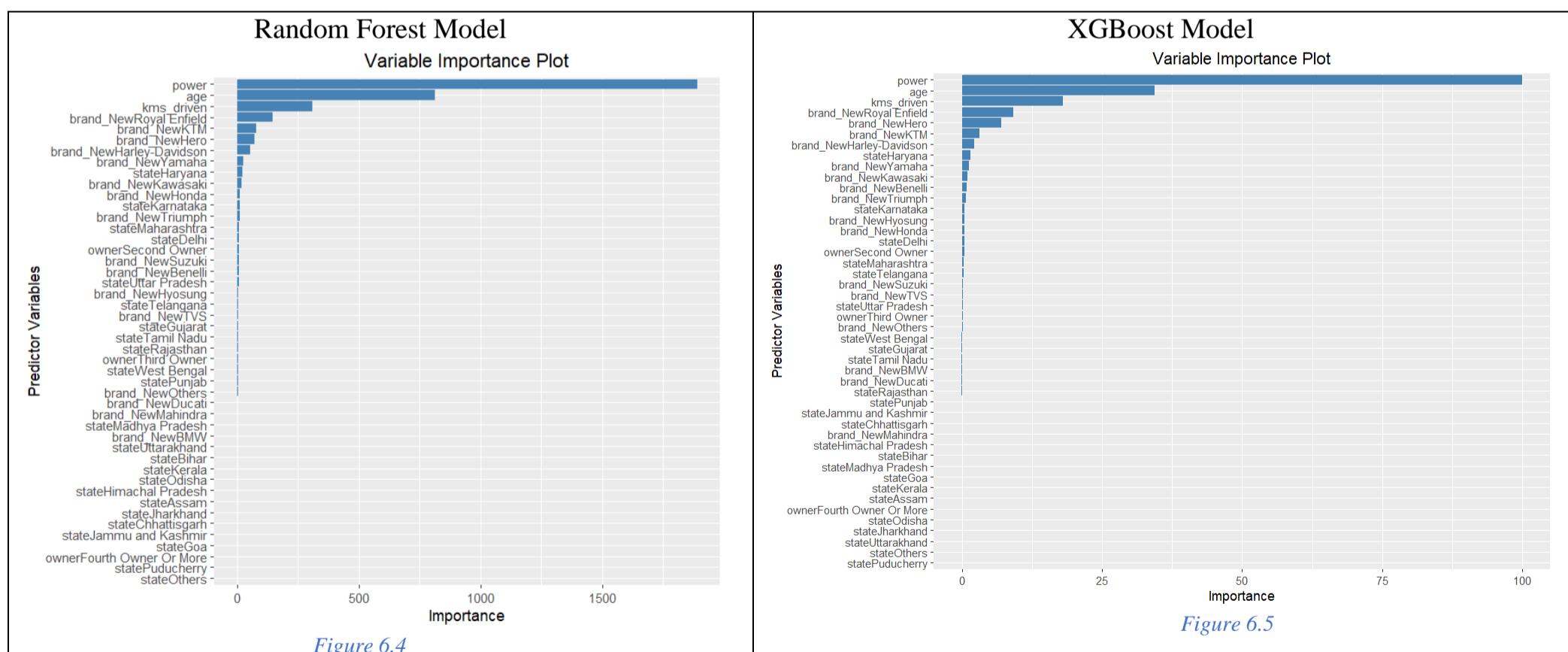
Also, we were attentive on how the transformation affects on the relationships between response variable and predictors, where we found that *power* which shared a fairly positive linear relationship has become a non-linear association with the log price. On that note, we decided to proceed with, Random forest regression and XGBoost. Table 6.4 below are the results obtained using random forest and XGBoosting after transforming them back to the original scale.

Model	With all the features					With the important features			
	Best Parameters	Training RMSE	Test RMSE	Training MAPE	Test MAPE	Training RMSE	Test RMSE	Training MAPE	Test MAPE
Random Forest	mtry=20 splitrule =variance min.node.size =15	16989.51	26014.15	11.43%	16.44%	15731.24	24042.76	9.63%	15.43%
XGBoost	nrounds=500 max_depth=4 eta=0.1 gamma=0.1 colsample_bytree=0.5 min_child_weight=1 subsample=0.6	16928.32	29126.05	10.28%	17.37%	18316.61	32476.92	12.11%	22.2%

Table 6.4

Note : The number of important features were selected in random forest and xgboost models were 11 and 10 respectively.

As depicted on Table 6.4, the results obtained when all the features were considered, both the techniques suggest that prediction accuracy is low due to overfitting. Moreover, extracting the important features also has not been successful in overcoming the overfitting issue. Figure 6.4 and Figure 6.5 depict the variable importance plots in this scenario regarding the random forest and xgboost models.



With the unsatisfactory results after the log transformation, we decided to implement **square root transformation** on the response variable as the techniques we use in model fitting is less affected by the distribution of the response. Here we found that the associations which the numerical predictors shared with the original response is less affected by the transformation, particularly the linear relationships has remained almost the same.

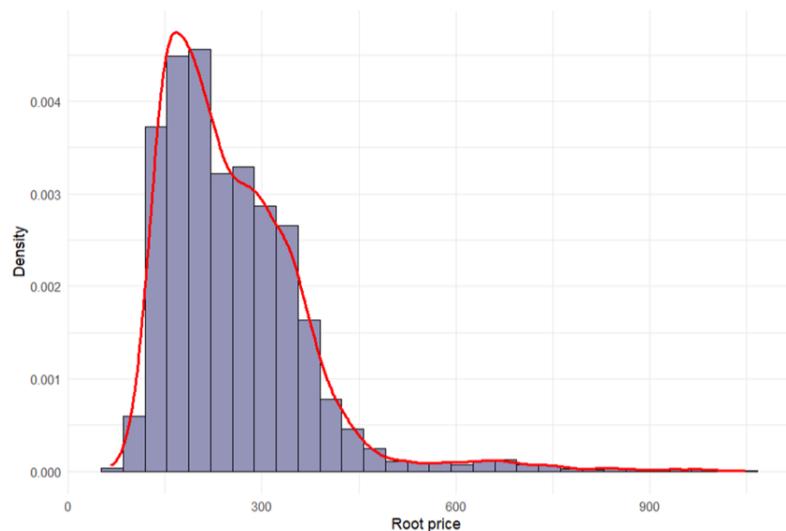


Figure 6.6

Table 6.5 below are the results obtained using regularization techniques after transforming them back to the original scale.

Model	Training RMSE	Test RMSE	Training MAPE	Test MAPE
Ridge	26659.66	27567.9	16.37%	17.103%
Lasso	26646.46	27512.26	16.32%	16.97%
Elastic Net	26684.16	34654.12	16.93%	25.48%

Table 6.5

With reference to the RMSE and MAPE values, the ridge and lasso regression models produce a higher accuracy with less overfitting since they don't show a significant difference between the training RMSE and test RMSE so as the training and test MAPEs. Expecting more accuracy, random forest and xgboost models were fitted on the data with the transformed response. Table 6.6 below are the results obtained using random forest and XGBoosting after transforming them back to the original scale.

Model	Best Parameters	Training RMSE	Test RMSE	Training MAPE	Test MAPE	Training RMSE	Test RMSE	Training MAPE	Test MAPE
Random Forest	mtry=25 splitrule =variance min.node.size =10	12937.58	23464.42	9.25%	14.76%	12340.9	23941.41	9.2%	15.2%
XGBoost	nrounds=500 max_depth=4 eta=0.1 gamma=0.1 colsample_bytree=0.5 min_child_weight=1 subsample =0.6	12014.08	25501.58	9.11%	16.39%	16195.55	28084.67	9.82%	17.21%

Table 6.6

Note : The number of important features were selected in random forest and xgboost models were 20 and 11 respectively.

As depicted on Table 6.6, the results obtained when all the features were considered, both the techniques suggest that prediction accuracy is low due to overfitting in terms of both RMSEs and MAPEs. Therefore, after extracting the most important features, the techniques were applied again. But the results show that satisfactory accuracy has not obtained due to overfitting. Figure 6.7 and Figure 6.8 depict the variable importance plots in this scenario regarding the random forest and xgboost models.

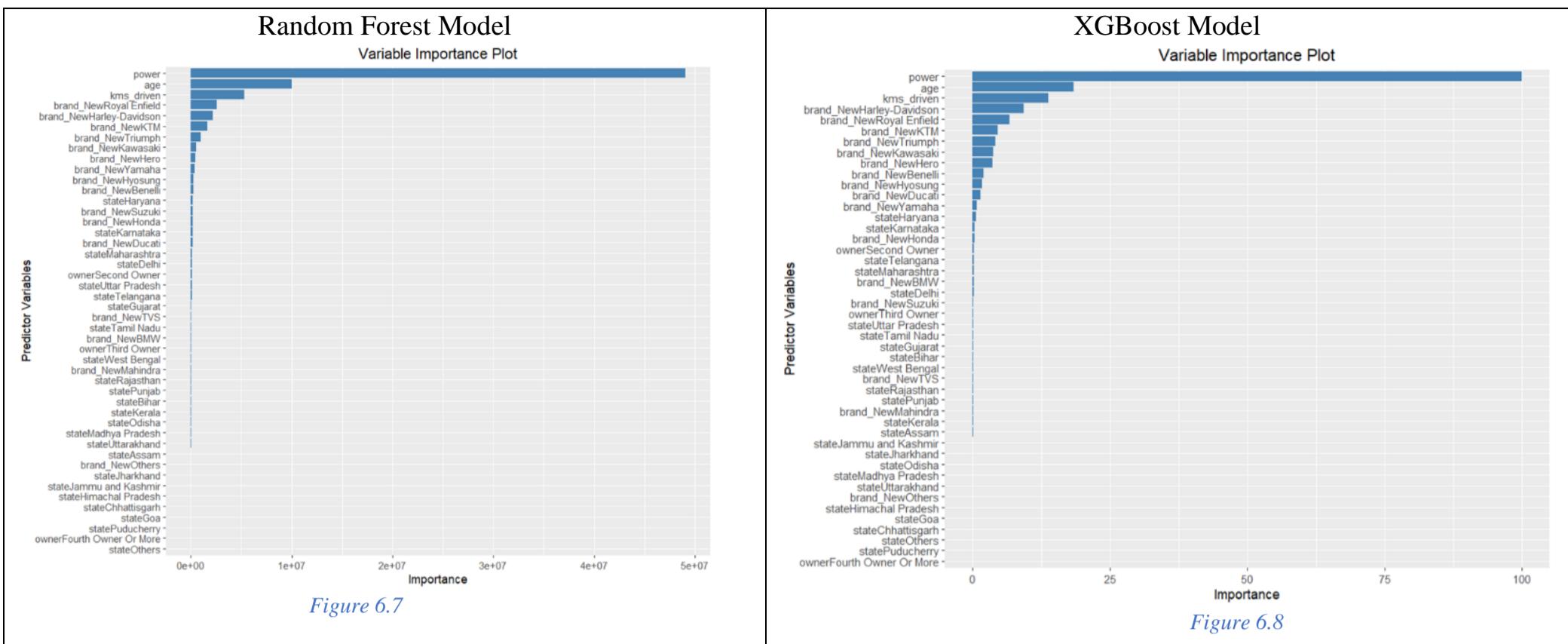


Figure 6.7

Figure 6.8

The ridge and lasso models with the square root transformed response which were identified to have high prediction accuracy with minimum overfitting were refitted with the 20 most important variables which were identified considering both random forest and xgboost in search of more accuracy. Table 6.7 shows the model evaluation results of the refitted models after transforming them back to original scale.

Model	Training RMSE	Test RMSE	Training MAPE	Test MAPE
Ridge	27823.66	29556.44	17.37%	19.62%
Lasso	27342.46	29573.56	17.32%	19.98%

Table 6.7

Even though the results obtained makes no vast difference with the models which were fitted with all the features, the accuracy has reduced in both the models in this case, in terms of both RMSEs and MAPEs.

7. Issues Encountered and Proposed Solutions

- The models fitted on original data implied overfitting and higher value of Test MAPEs. As a solution, transformations were applied on the response but couldn't achieve significant accuracies.
- Presence of a large number of factor levels in a categorical variable would make the model very complicated therefore before fitting the models some factor levels with very low representation were combined together with the intention of reducing the number of factor levels in a categorical variable.
- Random forest and XGBoosting models didn't perform well even after selecting the most important features because it showed significant overfitting. A proper factor analysis could improve these two models.
- The study was only carried out considering the brands of bikes leaving the bike name variable due to the complexity of the models affected by large number of categories Moreover, according to the reports of "kenresearch" website priority of the Indians in buying used bikes is the type of the two wheeler(motor cycle or scooter) more than the brand of the bike which suits their requirements. Otherwise, more accurate values would have been obtained, as bike prices vary with in a brand among various models in wider ranges.
- The dataset which was used for the analysis comprised of categorical variables with many factor levels. In order to fit a model, these variables had to be replaced with dummy variables. As a result, the different factors were separated into many dummy variables, it caused difficulties in interpreting the model since each factor level for the same categorical variable had different levels of importance where some factors had extremely high feature importance while the factors had very low feature importance.

8. Discussion and Conclusions

Since our initial distribution of the response variable was highly skewed to the right, the models which were fitted regarding the regularization techniques Ridge, lasso and Elastic Net resulted considerably higher RMSEs and Test MAPE values. Even though random forest and xgboost models resulted lower RMSE values in both scenarios of considering all the features and considering only the important features, the overfitting issue was clearly existing, which suggested low prediction accuracy which was further proved by the significant differences between their respective training and test MAPE values.

After log transformation of the response, the results obtained for the regularization techniques were still in unacceptable range suggesting a considerable underfitting with larger RMSE values. The random forest and xgboost techniques resulted lower RMSE and test MAPE values with a comparatively less overfitting than when considering the models relevant to the original response.

Thereafter the advanced analysis was carried out on the square root transformed response variable using regularization techniques ridge, lasso and elastic-net. Both ridge and lasso models resulted RMSEs and MAPE values in a justifiable range with comparatively less overfitting among all the regression models applied.

In order to look for further improvement, the attempts of fitting the random forest and xgboost models went in vain due to overfitting issue similar to the previous cases. Despite of the transformation on the original response, all the random forest and xgboost models resulted a higher variable importance for *power*, *kms_driven* and *age*.

When observing all the variable importance plots, it is evident that there are few dominant variables in predicting the prices . Hence they resulted not much difference in accuracy even after considering the important features comparatively with the model results with all the features considered, as the effect of other variables on accuracy is very low.

When all the models compared, the Lasso Regression model applied on the square root transformed response was identified as the best model which produces the most accurate predictions with training and test RMSEs of 26646.46 and 27512.26 respectively, where the training and test MAPEs were 16.32%, 16.97% respectively which suggests a higher prediction accuracy with less overfitting in this scenario. Even though the results of the ridge model doesn't make much difference with the lasso model in this case, we selected the lasso model as the best model based on interpretability since extracting the most important features in predicting the prices of these used bikes is among our main objectives.

Figure 6.7 and Figure 6.8 depict the coefficients plot related to the selected lasso model and the predicted values Vs observed values of the particular model on the test data respectively.

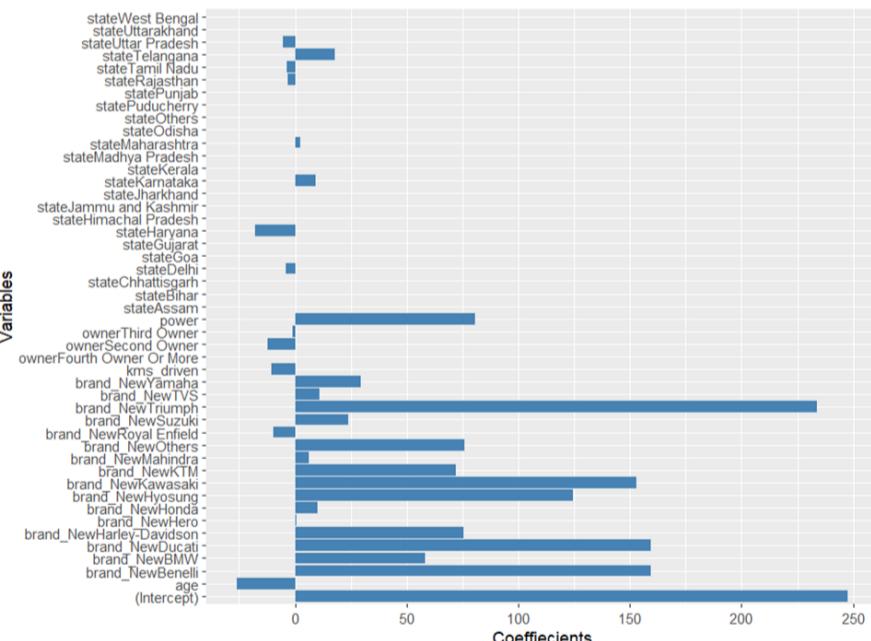


Figure 6.7

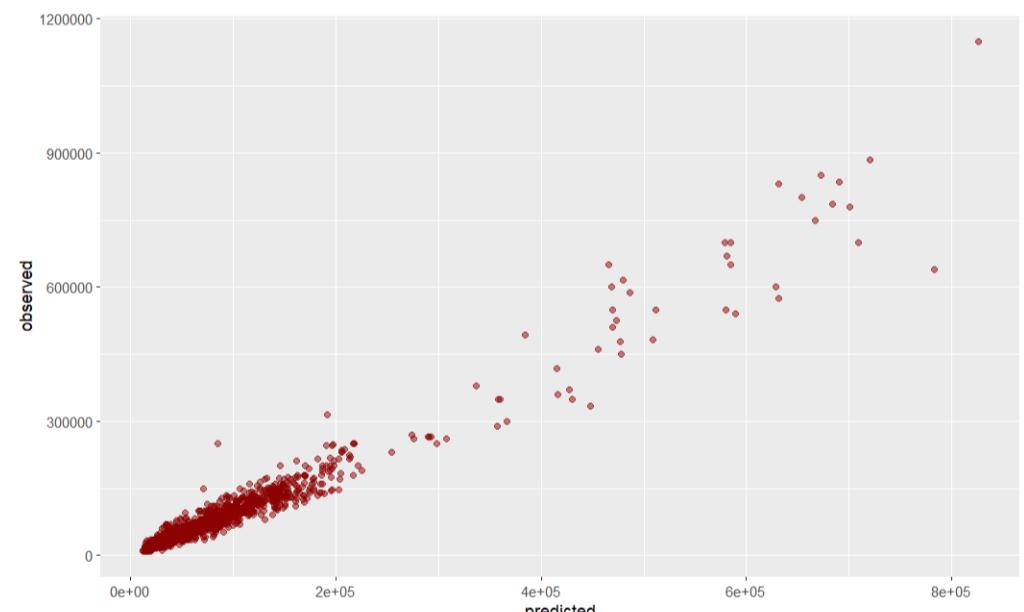


Figure 6.8

9. Appendix of the code

```

1 data=read.csv("D:/3RD YEAR/SEMESTER II/ST3082/Project III Materials//Used_Bikes.csv")
2 View(data)
3 head(data)
4 names(data)
5 summary(data)
6 str(data)
7 library(plyr)
8 library(dplyr)
9 library(stringr)
10 library(mgsub)
11 library(ggplot2)
12 library(ggfortify)
13 library(psych)
14 library(packNG)
15 library(moments)
16 library(tidyverse)
17 library(Metrics)
18
19 #####
20 ##### Pre- processing and Feature Engineering #####
21 #####
22 #Removing duplicates
23 sum(duplicated(data))
24 data_distinct(data)
25 nrow(data)
26 ncol(data) ## with NA
27 data = replace(data, data=="", NA)
28 colSums(is.na(data))
29 #Removing records with price=0
30 data=data[data$price != 0, ]
31 data=data[data$kms_driven != 0, ]
32 data=data[data$age != 0, ]
33 data=data[data$power != 0, ]
34 nrow(data)
35
36 #####brand_New column
37 brand_New<-c()
38 table(data$brand)
39 brand_New<-c()
40 for(i in 1:length(data$brand)){
41   if((data$brand[i]=="Ideal")|(data$brand[i]=="Indian")){
42     (data$brand[i]=="ML")|(data$brand[i]=="Rajdoot"))
43   } else{
44     (data$brand[i]=="Yezdi")|(data$brand[i]=="MV")|(data$brand[i]=="Java"))
45   }
46   brand_New[i]=data$brand[i]
47 }
48
49
50 }
51 table(brand_New)
52 data=cbind(data,brand_New)
53
54
55
56 #Create the state column
57 # read in the second file with unique cities and their corresponding state
58 df_<-read.csv("D:/3RD YEAR/SEMESTER II/ST3082/Project III Materials//cities.csv")
59 sum(duplicated(df))
60 df_distinct(df)
61 table(df$State)
62 nrow(df)
63
64
65 state<-c()
66 for(i in 1:nrow(df)){
67   for(j in 1:nrow(data)){
68     if(data$city[j]==df$city[i]){
69       state[j]=df$State[i]
70     } else{
71       next
72     }
73   }
74 }
75
76 for(i in 1:length(state)){
77   if((state[i]=="Arunachal Pradesh")|(state[i]=="Meghalaya")|
78     (state[i]=="Sikkim")){
79     state[i]="Others"
80   } else{
81     state[i]=state[i]
82   }
83 }
84 table(state)
85
86 #####
87
88 #Factoring
89 data$owner<- factor(data$owner,level=c("First Owner","Second Owner","Third Owner",
90 "Fourth Owner Or More"))
91 data$brand_New = factor(data$brand_New,level=c("Bajaj","Benelli","BMW","Ducati","Harley-Davidson","Hero","Honda",
92 "Hyosung","Kawasaki","KTM","Mahindra","Royal Enfield",
93 "Suzuki","Triumph","TVS","Yamaha","Others"))
94
95 data$state = factor(data$state,level=c("Andhra Pradesh","Assam","Bihar","Chhattisgarh",
96 "Delhi","Goa","Gujarat","Haryana",
97 "Himachal Pradesh","Jammu and Kashmir","Jharkhand","Karnataka",
98 "Kerala","Madhya Pradesh","Maharashtra","Odisha",
99 "Puducherry",
100 "Tamil Nadu","Telangana","Rajasthan",
101 "West Bengal","Others"))
102
103 str(data)
104 #####
105 #Splitting the data in to training and testing sets
106 set.seed(100)
107 indexes=sample(1:nrow(data),0.2*nrow(data))
108 testset=data[indexes,]
109 trainset=data[-indexes,]
110 View(trainset)
111 View(testset)
112 head(testset)
113 head(testset)
114 nrow(trainset)
115 nrow(testset)
116 table(trainset$brand_New)
117 table(trainset$owner)
118 table(trainset$state)
119 table(testset$brand_New)
120 table(testset$owner)
121 table(testset$state)
122 options(repr.plot.width=12,repr.plot.height=7)
123 boxplot(boxplot(trainset$price,main="Price distribution",col="#9494b8",xlab="Bike Price(In Indian Rupees"))
124 boxplot.stats(trainset$price)$stats
125 x=which(trainset$price)%%in% boxplot.stats(trainset$price)$out
126 length(x)
127
128 options(repr.plot.width=12,repr.plot.height=7)
129 hist_boxplot(trainset$price,main="Price distribution",col="#9494b8",xlab="Bike Price(In Indian Rupees")
130 boxplot.stats(trainset$price)$stats
131 x=which(trainset$price)%%in% x
132 length(x)
133 pr_ded=50%% filter(row_number() %%in% x)
134 View(pr_ded)
135 table(df1$brand_New)
136 table(df1$owner)
137 table(df1$state)
138 trainset=trainset %% filter((row_number() %%in% x)
139 table(trainset$brand_New)
140 table(trainset$owner)
141 table(trainset$state)
142 options(repr.plot.width=12,repr.plot.height=7)
143 hist_boxplot(log(trainset$price),main="log(Price) distribution",col="#9494b8",xlab="log Bike Price(In Indian Rupees")
144 boxplot.stats(log(trainset$price))$stats
145 x=which(log(trainset$price) %%in% boxplot.stats(log(trainset$price))$out)
146 length(x)
147
148 #Mean of delay time
149 mean(trainset$price)
150
151 #Skewness and Kurtosis
152 skewness(trainset$price)##1 positively skewed
153 kurtosis(trainset$price)
154 skewness(log(trainset$price))##1 positively skewed
155 kurtosis(log(trainset$price))
156
157 tg2 = dplyr(trainset, "brand_New", summarise, price = median(price))
158
159 ggplot(tg2, aes = brand_New, y = price, group=1) +
160   geom_point(color = "red") +
161   geom_line(color = "black") +
162   scale_x_discrete(guide = guide_axis(angle = 90)) +
163   labs(x = "Brand", y = "Median Bike Price (In Indian Rupees)") +
164   theme_minimal()
165
166 tg3 = trainset %%%
167   group_by(brand_New, owner) %%%
168   summarise(count = n()) %%%
169   arrange(desc(count))
170
171 ggplot(tg3, aes = x = reorder(brand_New, -count), y = count, fill = owner, label = count) +
172   geom_bar(stat = "identity", width = 0.7, alpha = 0.7) +
173   scale_x_discrete(guide = guide_axis(angle = 90)) +
174   labs(x = "Brand", y = "Bike Count") +
175   theme_minimal()
176
177 panel.grid.major = element_blank(),
178 panel.grid.minor = element_blank(),
179
180 #####
181 ##### Cluster Check #####
182
183 library(cluster)
184 library(Rtsne)
185
186 #Computing power distance
187
188 df_trainset_new
189 dummy_coding<-dummyVars(~ ~ , data = df)
190 df_encoded<-predict(dummy_coding, newdata = df)
191 df_encoded<-as.data.frame(df_encoded)
192 View(df_encoded)
193 df_encoded[,c(1,2,7,8)]<-scale(df_encoded[,c(1,2,7,8)], center = TRUE, scale = TRUE)
194 df_encoded[,c(1,2,7,8)]<-lapply(df_encoded[,c(1,2,7,8)], factor)
195 set.seed(100)
196 gml<-kmeans(x=1:nrow(df),size=1000,replace=FALSE)
197 gdl<-df_encoded[gml$cluster==1,]
198 gdl=daisy(gdl,metric="gower")
199 gml<-as.matrix(gdl)
200
201 dfl1<-df1[min(gml$cluster==1,na.rm=TRUE),]
202 dfl2<-df1[max(gml$cluster==1,na.rm=TRUE),]
203 dfl3<-df1[min(gml$cluster==2,na.rm=TRUE),]
204 dfl4<-df1[max(gml$cluster==2,na.rm=TRUE),]
205 dfl5<-df1[min(gml$cluster==3,na.rm=TRUE),]
206 dfl6<-df1[max(gml$cluster==3,na.rm=TRUE),]
207 dfl7<-df1[min(gml$cluster==4,na.rm=TRUE),]
208 dfl8<-df1[max(gml$cluster==4,na.rm=TRUE),]
209
210 #Graph to identify the number of clusters
211 sil_dist<-c()
212 for(i in 2:50){
213   pam_fit1<-pam(gdl,diss=TRUE,k=i)
214   sil_dist[i]<-pam_fit1$silinfo$avg.width
215 }
216
217 dev.off()
218 plot(1:8,sil_dist,
219   xlab = "k",
220   ylab="Average silhoutte distance",
221   lines(1:8,sil_dist))
222
223 # Partitioning around medoids with 3 clusters
224 pam_fit = pam(gdl, diss = TRUE, k = 4)
225
226 # Visualizing the clusters
227 library(Rtsne)
228 tsne_obj<-Rtsne(gdl, is_distance = TRUE)
229
230 tsne_data <- tsne_obj$Y %%%
231 data.frame() %%%
232 setnames(c("X", "Y")) %%%
233 mutate(cluster = factor(pam_fit$clustering))
234
235 ggplot(aes(x = X, y = Y), data = tsne_data) +
236   geom_point(aes(color = cluster))
237
238 ##### Suggestions for advanced analysis
239 quantitative<-trainset_new %%%
240 select(c(age,power,kms_driven))%%%
241 as.data.frame()
242 corr=round(corr(quantitative[,]),2)
243 library(ggcopplot)
244 ggcopplot(corr,hc.order=F,lab=TRUE)
245 data5<-trainset_new[-c(100:4000),]
246
247 #Suggestions for advanced analysis
248 #Correlation matrix
249 quantitative<-trainset_new %%%
250 select(c(power,kms_driven))%%%
251 as.data.frame()
252 corr=round(corr(quantitative[,]),2)
253 library(ggcopplot)
254 ggcopplot(corr,hc.order=F,lab=TRUE)
255 cor_test_result = cor.test(trainset_new$kms_driven, trainset_new$age)
256 cor_test_result2 = cor.test(trainset_new$kms_driven, trainset_new$power)
257 cor_test_result3 = cor.test(trainset_new$power, trainset_new$age)
258 # create a scatterplot matrix
259 pmat<-cor.test(cbind(kms_driven,age, power,age, power, kms_driven))
260 library(ggpubr)
261 ggcopplot(quantitative[,],lab=TRUE)
262
263 library(rstatistix)
264 ggcopplot(trainset, "age", facet_by = "brand_New")
265 ggcopplot(trainset, "age", facet_by = "owner")
266 ggcopplot(trainset, "age", facet_by = "state")
267 ggcopplot(trainset, "power", facet_by = "brand_New")
268 ggcopplot(trainset, "power", facet_by = "owner")
269 ggcopplot(trainset, "power", facet_by = "state")
270 ggcopplot(trainset, "kms_driven", facet_by = "brand_New")
271 ggcopplot(trainset, "kms_driven", facet_by = "owner")
272 ggcopplot(trainset, "price", facet_by = "state")
273
274 #Since ANOVA normality assumptions are violated we are using Kruskal wallis test
275 # Performing Kruskal-wallis test
276 result = kruskal.test(price ~ state,
277                       data = trainset)
278 print(result)
279
280 #####

```

```

393 #####PLS#####
394 trainset_new<-subset(trainset,select=-c(bike_name,city,brand))
395 testset_new<-subset(testset,select=-c(bike_name,city,brand))
396
397 library(mdatools)
398 library(caret)
399 dummy_coding<-dummyVars(~ . , data = trainset_new)
400 trainset_encoded<-predict(dummy_coding, newdata =trainset_new)
401
402 View(trainset_encoded)
403
404 x<-trainset_encoded[,1]
405 x[,c(1,6,7)]<-scale(x[,c(1,6,7)], center = TRUE, scale = TRUE)
406 y<-as.matrix(trainset_encoded[,1])
407 set.seed(100)
408 plotIPScorers(x,y, cv=5, info = "Bike Price prediction")
409 #View summary of model fitting
410 summary(ModelPLS)
411
412 #visualize CV plots
413 dev.off()
414 plot(ModelPLS)
415 plotXScore(ModelPLS,show_label = FALSE)
416 plotXResiduals(ModelPLS,show_label = FALSE)
417 plotIPScores(ModelPLS,ncp=10, type = "h",show_label = TRUE)
418 summary(ModelPLS$coeffs)
419 plot(ModelPLS$coeffs,show_label = TRUE)
420 Model0<-ModelPLS
421 #Checking Outliers
422 Model0$setdimnames(limits=(ModelPLS$lim,type = ModelPLS$lim,type,alpha=0.05)
423 plotXResiduals(Model0,show_labels=FALSE,Tables="indices")
424
425 # get row indices for outliers in calibration set
426 outliers <- which(categorize(Model0, ModelPLS$rescal) == "outlier")
427 length(outliers)
428 View(df1)
429 table(df1$brand_New)
430 table(df1$owner)
431 table(df1$state)
432 trainset_new<-trainset_new %>% filter(!row_number() %in% outliers)
433
485 #model_cv = gllmnet(x = model_matrix, y= y1, alpha = 0, Lambda = best_lambda)
486 model_cv2 = gllmnet(x = model_matrix, y=y1, alpha = 0, Lambda = best_lambda2,standardize = FALSE)
488
489 y_hat_ridge2 = predict(model_cv2, model_matrix)
490
491 library(Metrics)
492 actual<-y1
493 pred2<-y_hat_ridge2
494
495 mape_m = mape(y1, y_hat_ridge2)
496 rmse_m = rmse(y1, y_hat_ridge2)
497 mape_m
498 rmse_m
499 rmse_m
500
501
502
503 c2<-coef(model_cv2 ,>best_lambda2)
504 #For the test set
505 data2<-testset[,-1]
506 data2[,c(2,4,5)]<-scale(data2[,c(2,4,5)], center = TRUE, scale = TRUE)
507 data2[,4]<-as.numeric(data2[,4])
508 data2[,5]<-as.numeric(data2[,5])
509 data2[,6]<-as.numeric(data2[,6])
510
511 View(data2)
512 nrow(data2)
513 model_matrix1 = model.matrix(price~., data = data2[, -1])
514
515 # Separate the response variable from the predictors
516 y1 = data2$Min_Delay
517 y2 = data2$price
518 model_cv2 = gllmnet(x = model_matrix1, y=y2, alpha = 0, lambda = best_lambda2,standardize = FALSE)
519
520 y_hat_ridge2 = predict(model_cv2, model_matrix1)
521
522
523 actual<-y2
524 pred2<-y_hat_ridge2
525 mape_m = mape(y2, pred2)
526 rmse_m = rmse(y2, pred2)
527
528 dev.off()
529 plot(lasso_cv)
530
531 # Best cross-validated lambda
532 Lambda_cv2 = lasso_cv$lambda.1se
533
534 # Fit final model, get its sum of squared residuals and multiple R-squared
535 #model_cv = gllmnet(x = model_matrix, y= y1, alpha = 1, lambda = lambda_cv)
536 model_cv2 = gllmnet(x = model_matrix, y= y1, alpha = 1, lambda = lambda_cv2,standardize = FALSE)
537
538 y_hat_lasso2 = predict(model_cv2, model_matrix)
539
540 actual<-y1
541 pred2<-y_hat_lasso2
542
543 mape_m = mape(y1, pred2)
544 rmse_m = rmse(y1, pred2)
545 mape_m
546 rmse_m
547
548 #R2_cv = cor(y, y_hat_cv)^2
549 coef(model_cv1, s=lambda_cv1)
550 coef(model_cv2, s=lambda_cv2)
551
552 table(trainset_new$brand_New)
553 # See how increasing lambda shrinks the coefficients -----
554
555 res = gllmnet(x = model_matrix, y= y1, alpha = 1, lambda = lambda_seq,standardize = FALSE)
556 plot(res, xvar = "Lambda")
557 legend("bottomright", lwd = 1, col = 1:6, legend = colnames(model_matrix), cex = .7)
558
559 #For the test set
560
561 model_cv2 = gllmnet(x = model_matrix1, y=y2, alpha = 1, lambda = lambda_cv2,standardize = FALSE)
562
563 y_hat_lasso2 = predict(model_cv2, model_matrix1)
564
565 actual<-y2
566
567 # Check multiple R-squared
568 y_hat_enet = predict(elastic_net_model, data2[, -1])
569
570 actual<-y2
571 pred2<-y_hat_enet
572 rmse_m = rmse(y2,pred)
573 rmse_m
574 mape_m = mape(y2, pred)
575 mape_m
576
577 #Random Forest
578
579 # Define the tuning grid
580 tune_grid = expand.grid(
581   mtry=c(15,20,25,30),
582   splitrule = "variance",
583   min.node.size = c(1, 5, 10,15)
584 )
585
586 # Tune the hyperparameters using the tune() function
587 library(ranger)
588
589 # Perform 5-fold cross-validation
590 set.seed(100)
591 rf_model = train(
592   trainControlMethod = "boot", number = ifelse(grepl("cv", method), 10,25), repeats = ifelse(grepl("cv", method), 10,25), search = "grid", initialWindow = NULL, horizon = 1, fixedWindow =
593   data = data,
594   method = "ranger",
595   trcontrol = trainControl(method = "cv", number = 5, verboseItter = TRUE),
596   tuneGrid=tune_grid,
597   importance="impuity"
598 )
599 # Get best hyperparameters
600 best_params = rf_model$bestTune
601 plot(rf_model)
602
603 # For the training set
604 y_hat_enet = predict(rf_model, data1[, -1])
605
606 actual<-y1
607 pred2<-y_hat_enet
608
609 # Make predictions
610
611 #For the training set
612 y_hat_enet = predict(rf_model, data1[, -1])
613 pred2<-y_hat_enet
614 rmse_m = rmse(y1,pred)
615 rmse_m
616 mape_m = mape(y1, pred)
617 mape_m
618
619 #For the test set
620 y_hat_enet = predict(rf_model, data2[, -1])
621
622 actual<-y2
623 pred2<-y_hat_enet
624 rmse_m = rmse(y2,pred)
625 rmse_m
626 mape_m = mape(y2, pred)
627 mape_m
628
629 # get variable importance
630 # Print the variable importance measures
631 # print(rf_model$finalModel$variable.importance)
632 # print the variable importance
633 print(var_imp)
634
635 # load ggplot2 library
636 library(ggplot2)
637
638 var_imp = data.frame(variables = names(rf_model$finalModel$variable.importance),
639   Importance = rf_model$finalModel$variable.importance,
640   row.names = NULL)
641
642 ggplot(data = var_imp, aes(x = reorder(variables, Importance), y = Importance)) +
643   geom_bar(stat = "identity", fill = "#steelblue")
644 labs(title = "Variable Importance Plot", x = "Predictor Variables", y = "Importance") +
645   theme(plot.title = element_text(hjust = 0.5))+
646   coord_flip()
647
648 min.node.size = c(1, 5, 10,15)
649
650 set.seed(100)
651 rf_model = train(
652   price~.,
653   data = data,
654   method = "ranger",
655   trcontrol = trainControl(method = "cv", number = 5, verboseItter = TRUE),
656   tuneGrid=tune_grid,
657   importance="impuity"
658 )
659
660 # Make predictions
661
662 #For the training set
663 y_hat_enet = predict(rf_model, data1[, -1])
664 pred2<-y_hat_enet
665 rmse_m = rmse(y1,pred)
666 rmse_m
667 mape_m = mape(y1, pred)
668 mape_m
669
670 #For the test set
671 y_hat_enet = predict(rf_model, data2[, -1])
672
673 actual<-y2
674 pred2<-y_hat_enet
675 rmse_m = rmse(y2,pred)
676 rmse_m
677 mape_m = mape(y2, pred)
678 mape_m
679
680 # Remove least important variables from data
681 dummy_data1<-dummyVars(~ . , data = data1,sep = "")
682 data1_encoded<-predict(dummy_data1, newdata = data1)
683
684 View(data1_encoded)
685 data1_im<-data1[, -(colnames(data1_encoded) %in% vars_to_remove)]
686
687 View(data1_im)
688
689 base_var<-c("brand_NewBajaj","ownerFirst Owner","stateAndhra Pradesh")
690 data1_im<-data1_im[, -(colnames(data1_im) %in% base_var)]
691 View(data1_im)
692 str(data1_im)
693
694 data_im<-as.data.frame(data1_im)
695 as.data.frame(data_im)
696 a<-as.data.frame(data_im)
697 a$commodity<-data_im$commodity
698 a$price<-data_im$price
699 a$km_driven<-data_im$km_driven
700 a$age<-data_im$age
701 a$power<-data_im$power
702
703 # Convert non-numeric columns to factors
704 factor_cols<-setdiff(names(data1_im), num_cols)
705 data1_im[factor_cols] <- apply(data1_im[factor_cols], 2, function(x) as.factor(x))
706
707 # Check the data types
708 str(data1_im)
709
710 # Define the tuning grid for random forest hyperparameters
711 tune_grid = expand.grid(
712   mtry = 32,
713   splitrule = "variance",
714   min.node.size = c(1, 5, 10,15)
715 )
716
717 df_var_imp<-var_im$Importance
718 View(df)
719
720 nrow(df)
721
722 # Sort variable importances
723 var_im <- var_im[order(var_im$Importance), ]
724
725 # Calculate number of variables to be removed
726
727 vars_to_remove = df$Variables[33:46]
728
729 # Remove least important variables from data
730 dummy_data2<-dummyVars(~ . , data = data2,sep = "")
731 data2_encoded<-predict(dummy_data2, newdata = data2)
732
733 View(data2_encoded)
734 data2_im<-data2[, -(colnames(data2_encoded) %in% vars_to_remove)]
735
736 View(data2_im)
737 str(data2_im)
738
739 base_var<-c("brand_NewBajaj","ownerFirst Owner","stateAndhra Pradesh")
740 data2_im<-data2_im[, -(colnames(data2_im) %in% base_var)]
741 View(data2_im)
742
743 base_var<-c("brand_NewBajaj","ownerFirst Owner","stateAndhra Pradesh")
744 data2_im<-data2_im[, -(colnames(data2_im) %in% base_var)]
745 View(data2_im)
746
747 tune_grid = expand.grid(
748   mtry = 32,
749   splitrule = "variance",
750   min.node.size = c(1, 5, 10,15)
751 )
752
753 # Check the data types
754 str(data2_im)
755
756 # View summary of model fitting
757 summary(ModelPLS)
758
759 #View summary of model fitting
760
761 # Make predictions
762
763 #For the training set
764 y_hat_enet = predict(rf_model, data1[, -1])
765 pred2<-y_hat_enet
766 rmse_m = rmse(y1,pred)
767 rmse_m
768 mape_m = mape(y1, pred)
769 mape_m
770
771 #For the test set
772 y_hat_enet = predict(rf_model, data2[, -1])
773 pred2<-y_hat_enet
774 rmse_m = rmse(y2,pred)
775 rmse_m
776 mape_m = mape(y2, pred)
777 mape_m
778
779 # Remove least important variables from data
780 dummy_data2<-dummyVars(~ . , data = data2,sep = "")
781 data2_encoded<-predict(dummy_data2, newdata = data2)
782
783 View(data2_encoded)
784 data2_im<-data2[, -(colnames(data2_encoded) %in% vars_to_remove)]
785
786 View(data2_im)
787 str(data2_im)
788
789 base_var<-c("brand_NewBajaj","ownerFirst Owner","stateAndhra Pradesh")
790 data2_im<-data2_im[, -(colnames(data2_im) %in% base_var)]
791 View(data2_im)
792
793 tune_grid = expand.grid(
794   mtry = 32,
795   splitrule = "variance",
796   min.node.size = c(1, 5, 10,15)
797 )
798
799 # Make predictions
800
801 #For the training set
802 y_hat_enet = predict(rf_model, data1[, -1])
803 pred2<-y_hat_enet
804 rmse_m = rmse(y1,pred)
805 rmse_m
806 mape_m = mape(y1, pred)
807 mape_m
808
809 #For the test set
810 y_hat_enet = predict(rf_model, data2[, -1])
811 pred2<-y_hat_enet
812 rmse_m = rmse(y2,pred)
813 rmse_m
814 mape_m = mape(y2, pred)
815 mape_m
816
817 # Make predictions
818
819 #For the training set
820 y_hat_enet = predict(rf_model, data1[, -1])
821 pred2<-y_hat_enet
822 rmse_m = rmse(y1,pred)
823 rmse_m
824 mape_m = mape(y1, pred)
825 mape_m
826
827 #For the test set
828 y_hat_enet = predict(rf_model, data2[, -1])
829 pred2<-y_hat_enet
830 rmse_m = rmse(y2,pred)
831 rmse_m
832 mape_m = mape(y2, pred)
833 mape_m
834
835 # Make predictions
836
837 #For the training set
838 y_hat_enet = predict(rf_model, data1[, -1])
839 pred2<-y_hat_enet
840 rmse_m = rmse(y1,pred)
841 rmse_m
842 mape_m = mape(y1, pred)
843 mape_m
844
845 #For the test set
846 y_hat_enet = predict(rf_model, data2[, -1])
847 pred2<-y_hat_enet
848 rmse_m = rmse(y2,pred)
849 rmse_m
850 mape_m = mape(y2, pred)
851 mape_m
852
853 # Make predictions
854
855 #For the training set
856 y_hat_enet = predict(rf_model, data1[, -1])
857 pred2<-y_hat_enet
858 rmse_m = rmse(y1,pred)
859 rmse_m
860 mape_m = mape(y1, pred)
861 mape_m
862
863 #For the test set
864 y_hat_enet = predict(rf_model, data2[, -1])
865 pred2<-y_hat_enet
866 rmse_m = rmse(y2,pred)
867 rmse_m
868 mape_m = mape(y2, pred)
869 mape_m
870
871 # Make predictions
872
873 #For the training set
874 y_hat_enet = predict(rf_model, data1[, -1])
875 pred2<-y_hat_enet
876 rmse_m = rmse(y1,pred)
877 rmse_m
878 mape_m = mape(y1, pred)
879 mape_m
880
881 #For the test set
882 y_hat_enet = predict(rf_model, data2[, -1])
883 pred2<-y_hat_enet
884 rmse_m = rmse(y2,pred)
885 rmse_m
886 mape_m = mape(y2, pred)
887 mape_m
888
889 # Make predictions
890
891 #For the training set
892 y_hat_enet = predict(rf_model, data1[, -1])
893 pred2<-y_hat_enet
894 rmse_m = rmse(y1,pred)
895 rmse_m
896 mape_m = mape(y1, pred)
897 mape_m
898
899 #For the test set
900 y_hat_enet = predict(rf_model, data2[, -1])
901 pred2<-y_hat_enet
902 rmse_m = rmse(y2,pred)
903 rmse_m
904 mape_m = mape(y2, pred)
905 mape_m
906
907 # Make predictions
908
909 #For the training set
910 y_hat_enet = predict(rf_model, data1[, -1])
911 pred2<-y_hat_enet
912 rmse_m = rmse(y1,pred)
913 rmse_m
914 mape_m = mape(y1, pred)
915 mape_m
916
917 #For the test set
918 y_hat_enet = predict(rf_model, data2[, -1])
919 pred2<-y_hat_enet
920 rmse_m = rmse(y2,pred)
921 rmse_m
922 mape_m = mape(y2, pred)
923 mape_m
924
925 # Make predictions
926
927 #For the training set
928 y_hat_enet = predict(rf_model, data1[, -1])
929 pred2<-y_hat_enet
930 rmse_m = rmse(y1,pred)
931 rmse_m
932 mape_m = mape(y1, pred)
933 mape_m
934
935 #For the test set
936 y_hat_enet = predict(rf_model, data2[, -1])
937 pred2<-y_hat_enet
938 rmse_m = rmse(y2,pred)
939 rmse_m
940 mape_m = mape(y2, pred)
941 mape_m
942
943 # Make predictions
944
945 #For the training set
946 y_hat_enet = predict(rf_model, data1[, -1])
947 pred2<-y_hat_enet
948 rmse_m = rmse(y1,pred)
949 rmse_m
950 mape_m = mape(y1, pred)
951 mape_m
952
953 #For the test set
954 y_hat_enet = predict(rf_model, data2[, -1])
955 pred2<-y_hat_enet
956 rmse_m = rmse(y2,pred)
957 rmse_m
958 mape_m = mape(y2, pred)
959 mape_m
960
961 # Make predictions
962
963 #For the training set
964 y_hat_enet = predict(rf_model, data1[, -1])
965 pred2<-y_hat_enet
966 rmse_m = rmse(y1,pred)
967 rmse_m
968 mape_m = mape(y1, pred)
969 mape_m
970
971 #For the test set
972 y_hat_enet = predict(rf_model, data2[, -1])
973 pred2<-y_hat_enet
974 rmse_m = rmse(y2,pred)
975 rmse_m
976 mape_m = mape(y2, pred)
977 mape_m
978
979 # Make predictions
980
981 #For the training set
982 y_hat_enet = predict(rf_model, data1[, -1])
983 pred2<-y_hat_enet
984 rmse_m = rmse(y1,pred)
985 rmse_m
986 mape_m = mape(y1, pred)
987 mape_m
988
989 #For the test set
990 y_hat_enet = predict(rf_model, data2[, -1])
991 pred2<-y_hat_enet
992 rmse_m = rmse(y2,pred)
993 rmse_m
994 mape_m = mape(y2, pred)
995 mape_m
996
997 # Make predictions
998
999 #For the training set
1000 y_hat_enet = predict(rf_model, data1[, -1])
1001 pred2<-y_hat_enet
1002 rmse_m = rmse(y1,pred)
1003 rmse_m
1004 mape_m = mape(y1, pred)
1005 mape_m
1006
1007 #For the test set
1008 y_hat_enet = predict(rf_model, data2[, -1])
1009 pred2<-y_hat_enet
1010 rmse_m = rmse(y2,pred)
1011 rmse_m
1012 mape_m = mape(y2, pred)
1013 mape_m
1014
1015 # Make predictions
1016
1017 #For the training set
1018 y_hat_enet = predict(rf_model, data1[, -1])
1019 pred2<-y_hat_enet
1020 rmse_m = rmse(y1,pred)
1021 rmse_m
1022 mape_m = mape(y1, pred)
1023 mape_m
1024
1025 #For the test set
1026 y_hat_enet = predict(rf_model, data2[, -1])
1027 pred2<-y_hat_enet
1028 rmse_m = rmse(y2,pred)
1029 rmse_m
1030 mape_m = mape(y2, pred)
1031 mape_m
1032
1033 # Make predictions
1034
1035 #For the training set
1036 y_hat_enet = predict(rf_model, data1[, -1])
1037 pred2<-y_hat_enet
1038 rmse_m = rmse(y1,pred)
1039 rmse_m
1040 mape_m = mape(y1, pred)
1041 mape_m
1042
1043 #For the test set
1044 y_hat_enet = predict(rf_model, data2[, -1])
1045 pred2<-y_hat_enet
1046 rmse_m = rmse(y2,pred)
1047 rmse_m
1048 mape_m = mape(y2, pred)
1049 mape_m
1050
1051 # Make predictions
1052
1053 #For the training set
1054 y_hat_enet = predict(rf_model, data1[, -1])
1055 pred2<-y_hat_enet
1056 rmse_m = rmse(y1,pred)
1057 rmse_m
1058 mape_m = mape(y1, pred)
1059 mape_m
1060
1061 #For the test set
1062 y_hat_enet = predict(rf_model, data2[, -1])
1063 pred2<-y_hat_enet
1064 rmse_m = rmse(y2,pred)
1065 rmse_m
1066 mape_m = mape(y2, pred)
1067 mape_m
1068
1069 # Make predictions
1070
1071 #For the training set
1072 y_hat_enet = predict(rf_model, data1[, -1])
1073 pred2<-y_hat_enet
1074 rmse_m = rmse(y1,pred)
1075 rmse_m
1076 mape_m = mape(y1, pred)
1077 mape_m
1078
1079 #For the test set
1080 y_hat_enet = predict(rf_model, data2[, -1])
1081 pred2<-y_hat_enet
1082 rmse_m = rmse(y2,pred)
1083 rmse_m
1084 mape_m = mape(y2, pred)
1085 mape_m
1086
1087 # Make predictions
1088
1089 #For the training set
1090 y_hat_enet = predict(rf_model, data1[, -1])
1091 pred2<-y_hat_enet
1092 rmse_m = rmse(y1,pred)
1093 rmse_m
1094 mape_m = mape(y1, pred)
1095 mape_m
1096
1097 #For the test set
1098 y_hat_enet = predict(rf_model, data2[, -1])
1099 pred2<-y_hat_enet
1100 rmse_m = rmse(y2,pred)
1101 rmse_m
1102 mape_m = mape(y2, pred)
1103 mape_m
1104
1105 # Make predictions
1106
1107 #For the training set
1108 y_hat_enet = predict(rf_model, data1[, -1])
1109 pred2<-y_hat_enet
1110 rmse_m = rmse(y1,pred)
1111 rmse_m
1112 mape_m = mape(y1, pred)
1113 mape_m
1114
1115 #For the test set
1116 y_hat_enet = predict(rf_model, data2[, -1])
1117 pred2<-y_hat_enet
1118 rmse_m = rmse(y2,pred)
1119 rmse_m
1120 mape_m = mape(y2, pred)
1121 mape_m
1122
1123 # Make predictions
1124
1125 #For the training set
1126 y_hat_enet = predict(rf_model, data1[, -1])
1127 pred2<-y_hat_enet
1128 rmse_m = rmse(y1,pred)
1129 rmse_m
1130 mape_m = mape(y1, pred)
1131 mape_m
1132
1133 #For the test set
1134 y_hat_enet = predict(rf_model, data2[, -1])
1135 pred2<-y_hat_enet
1136 rmse_m = rmse(y2,pred)
1137 rmse_m
1138 mape_m = mape(y2, pred)
1139 mape_m
1140
1141 # Make predictions
1142
1143 #For the training set
1144 y_hat_enet = predict(rf_model, data1[, -1])
1145 pred2<-y_hat_enet
1146 rmse_m = rmse(y1,pred)
1147 rmse_m
1148 mape_m = mape(y1, pred)
1149 mape_m
1150
1151 #For the test set
1152 y_hat_enet = predict(rf_model, data2[, -1])
1153 pred2<-y_hat_enet
1154 rmse_m = rmse(y2,pred)
1155 rmse_m
1156 mape_m = mape(y2, pred)
1157 mape_m
1158
1159 # Make predictions
1160
1161 #For the training set
1162 y_hat_enet = predict(rf_model, data1[, -1])
1163 pred2<-y_hat_enet
1164 rmse_m = rmse(y1,pred)
1165 rmse_m
1166 mape_m = mape(y1, pred)
1167 mape_m
1168
1169 #For the test set
1170 y_hat_enet = predict(rf_model, data2[, -1])
1171 pred2<-y_hat_enet
1172 rmse_m = rmse(y2,pred)
1173 rmse_m
1174 mape_m = mape(y2, pred)
1175 mape_m
1176
1177 # Make predictions
1178
1179 #For the training set
1180 y_hat_enet = predict(rf_model, data1[, -1])
1181 pred2<-y_hat_enet
1182 rmse_m = rmse(y1,pred)
1183 rmse_m
1184 mape_m = mape(y1, pred)
1185 mape_m
1186
1187 #For the test set
1188 y_hat_enet = predict(rf_model, data2[, -1])
1189 pred2<-y_hat_enet
1190 rmse_m = rmse(y2,pred)
1191 rmse_m
1192 mape_m = mape(y2, pred)
1193 mape_m
1194
1195 # Make predictions
1196
1197 #For the training set
1198 y_hat_enet = predict(rf_model, data1[, -1])
1199 pred2<-y_hat_enet
1200 rmse_m = rmse(y1,pred)
1201 rmse_m
1202 mape_m = mape(y1, pred)
1203 mape_m
1204
1205 #For the test set
1206 y_hat_enet = predict(rf_model, data2[, -1])
1207 pred2<-y_hat_enet
1208 rmse_m = rmse(y2,pred)
1209 rmse_m
1210 mape_m = mape(y2, pred)
1211 mape_m
1212
1213 # Make predictions
1214
1215 #For the training set
1216 y_hat_enet = predict(rf_model, data1[, -1])
1217 pred2<-y_hat_enet
1218 rmse_m = rmse(y1,pred)
1219 rmse_m
1220 mape_m = mape(y1, pred)
1221 mape_m
1222
1223 #For the test set
1224 y_hat_enet = predict(rf_model, data2[, -1])
1225 pred2<-y_hat_enet
1226 rmse_m = rmse(y2,pred)
1227 rmse_m
1228 mape_m = mape(y2, pred)
1229 mape_m
1230
1231 # Make predictions
1232
1233 #For the training set
1234 y_hat_enet = predict(rf_model, data1[, -1])
1235 pred2<-y_hat_enet
1236 rmse_m = rmse(y1,pred)
1237 rmse_m
1238 mape_m = mape(y1, pred)
1239 mape_m
1240
1241 #For the test set
1242 y_hat_enet = predict(rf_model, data2[, -1])
1243 pred2<-y_hat_enet
1244 rmse_m = rmse(y2,pred)
1245 rmse_m
1246 mape_m = mape(y2, pred)
1247 mape_m
1248
1249 # Make predictions
1250
1251 #For the training set
1252 y_hat_enet = predict(rf_model, data1[, -1])
1253 pred2<-y_hat_enet
1254 rmse_m = rmse(y1,pred)
1255 rmse_m
1256 mape_m = mape(y1, pred)
1257 mape_m
1258
1259 #For the test set
1260 y_hat_enet = predict(rf_model, data2[, -1])
1261 pred2<-y_hat_enet
1262 rmse_m = rmse(y2,pred)
1263 rmse_m
1264 mape_m = mape(y2, pred)
1265 mape_m
1266
1267 # Make predictions
1268
1269 #For the training set
1270 y_hat_enet = predict(rf_model, data1[, -1])
1271 pred2<-y_hat_enet
1272 rmse_m = rmse(y1,pred)
1273 rmse_m
1274 mape_m = mape(y1, pred)
1275 mape_m
1276
1277 #For the test set
1278 y_hat_enet = predict(rf_model, data2[, -1])
1279 pred2<-y_hat_enet
1280 rmse_m = rmse(y2,pred)
1281 rmse_m
1282 mape_m = mape(y2, pred)
1283 mape_m
1284
1285 # Make predictions
1286
1287 #For the training set
1288 y_hat_enet = predict(rf_model, data1[, -1])
1289 pred2<-y_hat_enet
1290 rmse_m = rmse(y1,pred)
1291 rmse_m
1292 mape_m = mape(y1, pred)
1293 mape_m
1294
1295 #For the test set
1296 y_hat_enet = predict(rf_model, data2[, -1])
1297 pred2<-y_hat_enet
1298 rmse_m = rmse(y2,pred)
1299 rmse_m
1300 mape_m = mape(y2, pred)
1301 mape_m
1302
1303 # Make predictions
1304
1305 #For the training set
1306 y_hat_enet = predict(rf_model, data1[, -1])
1307 pred2<-y_hat_enet
1308 rmse_m = rmse(y1,pred)
1309 rmse_m
1310 mape_m = mape(y1, pred)
1311 mape_m
1312
1313 #For the test set
1314 y_hat_enet = predict(rf_model, data2[, -1])
1315 pred2<-y_hat_enet
1316 rmse_m = rmse(y2,pred)
1317 rmse_m
1318 mape_m = mape(y2
```

```

795 data2_imp<-data.frame(data2_imp)
796 num_cols = c("price", "kms_driven", "age", "power")
797 # Convert non-numeric columns to factors
798 factor_cols = apply(data2_imp[, -num_cols], 2, function(x) as.factor(x))
799
800 y_hat_enet = predict(rf_model, data2_imp[, -1])
801 actual_y2
802 pred<-c(y_hat_enet )
803 rmse_m = rmse(y2,pred)
804 mape_m = mape(y2, pred)
805 mape_m
806
807
808 #####xgbTree#####
809 library(xgboost)
810 set.seed(100)
811 Xtrain = xgb.DMatrix(data = as.matrix(data1[,-1]),label=y1)
812 y_train = data1$price
813 Xtest = xgb.DMatrix(data = as.matrix(data2[,-1]),label=y2)
814 y_test = data2$price
815 xgb_model = train(xtrain,
816   method = "xgbTree",
817   objective = "reg:squarederror",
818   trControl = trainControl(method = "cv",
819     number = 5,
820     verboseIter = TRUE),
821
822 tuneGrid = expand.grid(nrounds = c(500,1000),
823   eta=0.1,
824   max_depth = c(2,4,6),
825   colsample_bytree = c(0.5,0.6),
826   subsample = c(0.5,0.6),
827   gamma=0.1,
828   min_child_weight = c(1,3,5)
829 )
830
831 plot(xgb_model)
832 y_hat_enet = predict(xgb_model, xtrain)
833 actual_y2
834 pred<-c(y_hat_enet )
835 rmse_m = rmse(y2,pred)
836 mape_m = mape(y2, pred)
837 mape_m
838
839 #for the test set
840
841 y_hat_enet = predict(xgb_model, xtest)
842 actual_y2
843 pred<-c(y_hat_enet )
844 rmse_m = rmse(y2,pred)
845 mape_m = mape(y2, pred)
846 mape_m
847
848
849
850
851
852
853
854
855
856
857
858
859
860
861
862
863
864
865
866
867
868
869
870
871
872
873
874
875
876
877
878
879
880
881
882
883
884
885
886
887
888
889
890
891
892
893
894
895
896
897
898
899
900
901
902
903
904
905
906
907
908
909
910
911
912
913
914
915
916
917
918
919
920
921
922
923
924
925
926
927
928
929
930
931
932
933
934
935
936
937
938
939
940
941
942
943
944
945
946
947
948
949
950
951
952
953
954
955
956
957
958
959
960
961
962
963
964
965
966
967
968
969
970
971
972
973
974
975
976
977
978
979
980
981
982
983
984
985
986
987
988
989
990
991
992
993
994
995
996
997
998
999
1000
1001
1002
1003
1004
1005
1006
1007
1008
1009
1010
1011
1012
1013
1014
1015
1016
1017
1018
1019
1020
1021
1022
1023
1024
1025
1026
1027
1028
1029
1030
1031
1032
1033
1034
1035
1036
1037
1038
1039
1040
1041
1042
1043
1044
1045
1046
1047
1048
1049
1050
1051
1052
1053
1054
1055
1056
1057
1058
1059
1060
1061
1062
1063
1064
1065
1066
1067
1068
1069
1070
1071
1072
1073
1074
1075
1076
1077
1078
1079
1080
1081
1082
1083
1084
1085
1086
1087
1088
1089
1090
1091
1092
1093
1094
1095
1096
1097
1098
1099
1100
1101
1102
1103
1104
1105
1106
1107
1108
1109
1110
1111
1112
1113
1114
1115
1116
1117
1118
1119
1120
1121
1122
1123
1124
1125
1126
1127
1128
1129
1130
1131
1132
1133
1134
1135
1136
1137
1138
1139
1140
1141
1142
1143
1144
1145
1146
1147
1148
1149
1150
1151
1152
1153
1154
1155
1156
1157
1158
1159
1160
1161
1162
1163
1164
1165
1166
1167
1168
1169
1170
1171
1172
1173
1174
1175
1176
1177
1178
1179
1180
1181
1182
1183
1184
1185
1186
1187
1188
1189
1190
1191
1192
1193
1194
1195
1196
1197
1198
1199
1200
1201
1202
1203
1204
1205
1206
1207
1208
1209
1210
1211
1212
1213
1214
1215
1216
1217
1218
1219
1220
1221
1222
1223
1224
1225
1226
1227
1228
1229
1230
1231
1232
1233
1234
1235
1236
1237
1238
1239
1240
1241
1242
1243
1244
1245
1246
1247
1248
1249
1250
1251
1252
1253
1254
1255
1256
1257
1258
1259
1260
1261
1262
1263
1264
1265
1266
1267
1268
1269
1270
1271
1272
1273
1274
1275
1276
1277
1278
1279
1280
1281
1282
1283
1284
1285
1286
1287
1288
1289
1290
1291
1292
1293
1294
1295
1296
1297
1298
1299
1300
1301
1302
1303
1304
1305
1306
1307
1308
1309
1310
1311
1312
1313
1314
1315
1316
1317
1318
1319
1320
1321
1322
1323
1324
1325
1326
1327
1328
1329
1330
1331
1332
1333
1334
1335
1336
1337
1338
1339
1340
1341
1342
1343
1344
1345
1346
1347
1348
1349
1350
1351
1352
1353
1354
1355
1356
1357
1358
1359
1360
1361
1362
1363
1364
1365
1366
1367
1368
1369
1370
1371
1372
1373
1374
1375
1376
1377
1378
1379
1380
1381
1382
1383
1384
1385
1386
1387
1388
1389
1390
1391
1392
1393
1394
1395
1396
1397
1398
1399
1400
1401
1402
1403
1404
1405
1406
1407
1408
1409
1410
1411
1412
1413
1414
1415
1416
1417
1418
1419
1420
1421
1422
1423
1424
1425
1426
1427
1428
1429
1430
1431
1432
1433
1434
1435
1436
1437
1438
1439
1440
1441
1442
1443
1444
1445
1446
1447
1448
1449
1450
1451
1452
1453
1454
1455
1456
1457
1458
1459
1460
1461
1462
1463
1464
1465
1466
1467
1468
1469
1470
1471
1472
1473
1474
1475
1476
1477
1478
1479
1480
1481
1482
1483
1484
1485
1486
1487
1488
1489
1490
1491
1492
1493
1494
1495
1496
1497
1498
1499
1500
1501
1502
1503
1504
1505
1506
1507
1508
1509
1510
1511
1512
1513
1514
1515
1516
1517
1518
1519
1520
1521
1522
1523
1524
1525
1526
1527
1528
1529
1530
1531
1532
1533
1534
1535
1536
1537
1538
1539
1540
1541
1542
1543
1544
1545
1546
1547
1548
1549
1550
1551
1552
1553
1554
1555
1556
1557
1558
1559
1560
1561
1562
1563
1564
1565
1566
1567
1568
1569
1570
1571
1572
1573
1574
1575
1576
1577
1578
1579
1580
1581
1582
1583
1584
1585
1586
1587
1588
1589
1590
1591
1592
1593
1594
1595
1596
1597
1598
1599
1600
1601
1602
1603
1604
1605
1606
1607
1608
1609
1610
1611
1612
1613
1614
1615
1616
1617
1618
1619
1620
1621
1622
1623
1624
1625
1626
1627
1628
1629
1630
1631
1632
1633
1634
1635
1636
1637
1638
1639
1640
1641
1642
1643
1644
1645
1646
1647
1648
1649
1650
1651
1652
1653
1654
1655
1656
1657
1658
1659
1660
1661
1662
1663
1664
1665
1666
1667
1668
1669
1670
1671
1672
1673
1674
1675
1676
1677
1678
1679
1680
1681
1682
1683
1684
1685
1686
1687
1688
1689
1690
1691
1692
1693
1694
1695
1696
1697
1698
1699
1700
1701
1702
1703
1704
1705
1706
1707
1708
1709
1710
1711
1712
1713
1714
1715
1716
1717
1718
1719
1720
1721
1722
1723
1724
1725
1726
1727
1728
1729
1730
1731
1732
1733
1734
1735
1736
1737
1738
1739
1740
1741
1742
1743
1744
1745
1746
1747
1748
1749
1750
1751
1752
1753
1754
1755
1756
1757
1758
1759
1760
1761
1762
1763
1764
1765
1766
1767
1768
1769
1770
1771
1772
1773
1774
1775
1776
1777
1778
1779
1780
1781
1782
1783
1784
1785
1786
1787
1788
1789
1790
1791
1792
1793
1794
1795
1796
1797
1798
1799
1800
1801
1802
1803
1804
1805
1806
1807
1808
1809
1810
1811
1812
1813
1814
1815
1816
1817
1818
1819
1820
1821
1822
1823
1824
1825
1826
1827
1828
1829
1830
1831
1832
1833
1834
1835
1836
1837
1838
1839
1840
1841
1842
1843
1844
1845
1846
1847
1848
1849
1850
1851
1852
1853
1854
1855
1856
1857
1858
1859
1860
1861
1862
1863
1864
1865
1866
1867
1868
1869
1870
1871
1872
1873
1874
1875
1876
1877
1878
1879
1880
1881
1882
1883
1884
1885
1886
1887
1888
1889
1890
1891
1892
1893
1894
1895
1896
1897
1898
1899
1900
1901
1902
1903
1904
1905
1906
1907
1908
1909
1910
1911
1912
1913
1914
1915
1916
1917
1918
1919
1920
1921
1922
1923
1924
1925
1926
1927
1928
1929
1930
1931
1932
1933
1934
1935
1936
1937
1938
1939
1940
1941
1942
1943
1944
1945
1946
1947
1948
1949
1950
1951
1952
1953
1954
1955
1956
1957
1958
1959
1960
1961
1962
1963
1964
1965
1966
1967
1968
1969
1970
1971
1972
1973
1974
1975
1976
1977
1978
1979
1980
1981
1982
1983
1984
1985
1986
1987
1988
1989
1990
1991
1992
1993
1994
1995
1996
1997
1998
1999
2000
2001
2002
2003
2004
2005
2006
2007
2008
2009
2010
2011
2012
2013
2014
2015
2016
2017
2018
2019
2020
2021
2022
2023
2024
2025
2026
2027
2028
2029
2030
2031
2032
2033
2034
2035
2036
2037
2038
2039
2040
2041
2042
2043
2044
2045
2046
2047
2048
2049
2050
2051
2052
2053
2054
2055
2056
2057
2058
2059
2060
2061
2062
2063
2064
2065
2066
2067
2068
2069
2070
2071
2072
2073
2074
2075
2076
2077
2078
2079
2080
2081
2082
2083
2084
2085
2086
2087
2088
2089
2090
2091
2092
2093
2094
2095
2096
2097
2098
2099
2100
2101
2102
2103
2104
2105
2106
2107
2108
2109
2110
2111
2112
2113
2114
2115
2116
2117
2118
2119
2120
2121
2122
2123
2124
2125
2126
2127
2128
2129
2130
2131
2132
2133
2134
2135
2136
2137
2138
2139
2140
2141
2142
2143
2144
2145
2146
2147
2148
2149
2150
2151
2152
2153
2154
2155
2156
2157
2158
2159
2160
2161
2162
2163
2164
2165
2166
2167
2168
2169
2170
2171
2172
2173
2174
2175
2176
2177
2178
2179
2180
2181
2182
2183
2184
2185
2186
2187
2188
2189
2190
2191
2192
2193
2194
2195
2196
2197
2198
2199
2200
2201
2202
2203
2204
2205
2206
2207
2208
2209
2210
2211
2212
2213
2214
2215
2216
2217
2218
2219
2220
2221
2222
2223
2224
2225
2226
2227
2228
2229
2230
2231
2232
2233
2234
2235
2236
2237
2238
2239
2240
2241
2242
2243
2244
2245
2246
2247
2248
2249
2250
2251
2252
2253
2254
2255
2256
2257
2258
2259
2260
2261
2262
2263
2264
2265
2266
2267
2268
2269
2270
2271
2272
2273
2274
2275
2276
2277
2278
2279
2280
2281
2282
2283
2284
2285
2286
2287
2288
2289
2290
2291
2292
2293
2294
2295
2296
2297
2298
2299
2300
2301
2302
2303
2304
2305
2306
2307
2308
2
```

```

1218 mape_m = mape(d_actual, d_pred)
1219 mape_m
1220 #For the test set
1221 #elastic_net_model = train(log_delay ~ ., data = data2,method = "glmnet",trControl = train_control)
1222
1223 # Check multiple R-squared
1224 y_hat_enet = predict(elastic_net_model, data2[,-1])
1225
1226 actual_y2
1227 pred<-c(y_hat_enet )
1228 d_actual=testset_new$price
1229 d_pred<-as.integer(exp(pred))
1230 rmse_m = rmse(d_actual, d_pred)
1231 rmse_m
1232 mape_m = mape(d_actual, d_pred)
1233 mape_m
1234
1235 #Random Forest
1236
1237 # Define the tuning grid
1238 tune_grid = expand.grid(
1239   mtry = c(15,20,25,30),
1240   splitrule = "variance",
1241   min.node.size = c(1, 5, 10,15)
1242 )
1243
1244 # Tune the hyperparameters using the tune() function
1245 library(ranger)
1246
1247 # Perform 5-fold cross-validation
1248 set.seed(100)
1249 rf_model = train(
1250   log_price ~.,
1251   data = data1,
1252   method = "ranger",
1253   trControl = trainControl(method = "cv", number = 5, verboseIter = TRUE),
1254   tuneGrid=tune_grid,
1255   importance="impurity"
1256 )
1257 # Get best hyperparameters
1258 best_params = rf_model$bestTune
1259
1260 plot(rf_model)
1261 #for the training set
1262 y_hat_enet = predict(rf_model, data1[,-1])
1263 actual_y1
1264 pred<-c(y_hat_enet)
1265 d_actual=traintest_new$price
1266 d_pred<-as.integer(exp(pred))
1267 mape_m = mape(d_actual, d_pred)
1268 rmse_m = rmse(d_actual, d_pred)
1269 mape_m
1270 rmse_m
1271
1272 #for the test set
1273 y_hat_enet = predict(rf_model, data2[,-1])
1274
1275 actual_y2
1276 pred<-c(y_hat_enet )
1277 d_actual=testset_new$price
1278 d_pred<-as.integer(exp(pred))
1279 mape_m = mape(d_actual, d_pred)
1280 rmse_m = rmse(d_actual, d_pred)
1281 mape_m
1282 rmse_m
1283
1284
1285 # get variable importance
1286 # Print the variable importance measures
1287 var_imp_rf_model$finalModel$variable.importance
1288
1289 # print the variable importance
1290 print(var_imp)
1291
1292 # load ggplot2 library
1293 library(ggplot2)
1294
1295
1296 var_imp = data.frame(variables = names(rf_model$finalModel$variable.importance),
1297   importance = rf_model$finalModel$variable.importance,
1298   row.names = NULL)
1299
1300 ggplot(data = var_imp, aes(x = reorder(variables, Importance), y = Importance)) +
1301   geom_bar(stat = "identity", fill = "#steelblue") +
1302   labs(title = "Variable Importance Plot", x = "Predictor Variables", y = "Importance") +
1303   theme(plot.title = element_text(hjust = 0.5))+
1304   coord_flip()
1305
1306
1307 df$var_imp[order(-var_imp$Importance),]
1308 View(df)
1309 df[c(1:11),]
1310
1311 nrow(df)
1312
1313 # Calculate number of variables to be removed
1314
1315 vars_to_remove = df$Variables[12:46]
1316
1317 # Remove least important variables from data1
1318 dummy_data1=dummyVars(~ . , data = data1,sep = "")
1319 data1_encoded=predict(dummy_data1, newdata = data1)
1320
1321 View(data1_encoded)
1322 data1 = data1_encoded[, !(colnames(data1_encoded) %in% vars_to_remove)]
1323 View(data1)
1324
1325 data1$imp = as.data.frame(data1$imp)
1326
1327 # Convert dummy variables to factors
1328 num_col = c("log_price", "kms_driven", "age", "power")
1329 num_col = setdiff(names(data1$imp), num_col)
1330
1331 # Convert non-numeric columns to factors
1332 data1$imp$num_col = as.factor(data1$imp$num_col)
1333 factor_cols = setdiff(names(data1$imp), num_col)
1334 data1$imp[factor_cols] = apply(data1$imp[factor_cols], 2, function(x) as.factor(x))
1335
1336 # Check the data types
1337 str(data1)
1338
1339 # Define the tuning grid for random forest hyperparameters
1340 tune_grid = expand.grid(
1341   mtry = c(15,20,25,30),
1342   splitrule = "variance",
1343   min.node.size = c(1, 5, 10,15)
1344 )
1345
1346
1347 data2$imp = as.data.frame(data2$imp)
1348
1349 data2$imp$base_var = c("brand_NewBajaj","ownerFirst Owner","stateAndhra Pradesh")
1350 data2$imp = data2$imp[, !(colnames(data2$imp) %in% base_var)]
1351 View(data2$imp)
1352
1353 data2$imp = as.data.frame(data2$imp)
1354
1355 # Convert dummy variables to Factors
1356 num_col = c("log_price", "kms_driven", "age", "power")
1357 num_col = setdiff(names(data2$imp), num_col)
1358
1359 factor_cols = setdiff(names(data2$imp), num_col)
1360 data2$imp[factor_cols] = apply(data2$imp[factor_cols], 2, function(x) as.factor(x))
1361
1362 # Check the data types
1363 str(data2)
1364
1365 # Remove least important variables from data2
1366 dummy_data2=dummyVars(~ . , data = data2,sep = "")
1367 data2_encoded=predict(dummy_data2, newdata = data2)
1368 View(data2_encoded)
1369 data2 = data2_encoded[, !(colnames(data2_encoded) %in% vars_to_remove)]
1370 View(data2)
1371
1372 data2$imp = as.data.frame(data2$imp)
1373
1374 # Convert dummy variables to Factors
1375 num_col = c("log_price", "kms_driven", "age", "power")
1376 num_col = setdiff(names(data2$imp), num_col)
1377
1378 factor_cols = setdiff(names(data2$imp), num_col)
1379 data2$imp[factor_cols] = apply(data2$imp[factor_cols], 2, function(x) as.factor(x))
1380
1381 # Check the data types
1382 str(data2)
1383
1384 # Define the tuning grid for random forest hyperparameters
1385 tune_grid = expand.grid(
1386   mtry = c(15,20,25,30),
1387   splitrule = "variance",
1388   min.node.size = c(1, 5, 10,15)
1389 )
1390
1391
1392 data2$imp$base_var = c("brand_NewBajaj","ownerFirst Owner","stateAndhra Pradesh")
1393 data2$imp = data2$imp[, !(colnames(data2$imp) %in% base_var)]
1394 View(data2$imp)
1395
1396 data2$imp = as.data.frame(data2$imp)
1397
1398 data2$imp$base_var = c("brand_NewBajaj","ownerFirst Owner","stateAndhra Pradesh")
1399 data2$imp = data2$imp[, !(colnames(data2$imp) %in% base_var)]
1400 View(data2$imp)
1401
1402 data2$imp$base_var = c("brand_NewBajaj","ownerFirst Owner","stateAndhra Pradesh")
1403 data2$imp = data2$imp[, !(colnames(data2$imp) %in% base_var)]
1404 View(data2$imp)
1405
1406 data2$imp$base_var = c("brand_NewBajaj","ownerFirst Owner","stateAndhra Pradesh")
1407 data2$imp = data2$imp[, !(colnames(data2$imp) %in% base_var)]
1408 View(data2$imp)
1409
1410 data2$imp$base_var = c("brand_NewBajaj","ownerFirst Owner","stateAndhra Pradesh")
1411 data2$imp = data2$imp[, !(colnames(data2$imp) %in% base_var)]
1412 View(data2$imp)
1413
1414
1415
1416
1417
1418
1419
1420
1421
1422
1423
1424
1425
1426
1427
1428
1429
1430
1431
1432
1433
1434
1435
1436
1437
1438
1439
1440
1441
1442
1443
1444
1445
1446
1447
1448
1449
1450
1451
1452
1453
1454
1455
1456
1457
1458
1459
1460
1461
1462
1463
1464
1465
1466
1467
1468
1469
1470
1471
1472
1473
1474
1475
1476
1477
1478
1479
1480
1481
1482
1483
1484
1485
1486
1487
1488
1489
1490
1491
1492
1493
1494
1495
1496
1497
1498
1499
1500
1501
1502
1503
1504
1505
1506
1507
1508
1509
1510
1511
1512
1513
1514
1515
1516
1517
1518
1519
1520
1521
1522
1523
1524
1525
1526
1527
1528
1529
1530
1531
1532
1533
1534
1535
1536
1537
1538
1539
1540
1541
1542
1543
1544
1545
1546
1547
1548
1549
1550
1551
1552
1553
1554
1555
1556
1557
1558
1559
1560
1561
1562
1563
1564
1565
1566
1567
1568
1569
1570
1571
1572
1573
1574
1575
1576
1577
1578
1579
1580
1581
1582
1583
1584
1585
1586
1587
1588
1589
1590
1591
1592
1593
1594
1595
1596
1597
1598
1599
1600
1601
1602
1603
1604
1605
1606
1607
1608
1609
1610
1611
1612
1613
1614
1615
1616
1617
1618
1619
1620
1621
1622
1623
1624
1625
1626
1627
1628
1629
1630
1631
1632
1633
1634
1635
1636
1637
1638
1639
1640
1641
1642
1643
1644
1645
1646
1647
1648
1649
1650
1651
1652
1653
1654
1655
1656
1657
1658
1659
1660
1661
1662
1663
1664
1665
1666
1667
1668
1669
1670
1671
1672
1673
1674
1675
1676
1677
1678
1679
1680
1681
1682
1683
1684
1685
1686
1687
1688
1689
1690
1691
1692
1693
1694
1695
1696
1697
1698
1699
1700
1701
1702
1703
1704
1705
1706
1707
1708
1709
1710
1711
1712
1713
1714
1715
1716
1717
1718
1719
1720
1721
1722
1723
1724
1725
1726
1727
1728
1729
1730
1731
1732
1733
1734
1735
1736
1737
1738
1739
1740
1741
1742
1743
1744
1745
1746
1747
1748
1749
1750
1751
1752
1753
1754
1755
1756
1757
1758
1759
1760
1761
1762
1763
1764
1765
1766
1767
1768
1769
1770
1771
1772
1773
1774
1775
1776
1777
1778
1779
1780
1781
1782
1783
1784
1785
1786
1787
1788
1789
1790
1791
1792
1793
1794
1795
1796
1797
1798
1799
1800
1801
1802
1803
1804
1805
1806
1807
1808
1809
1810
1811
1812
1813
1814
1815
1816
1817
1818
1819
1820
1821
1822
1823
1824
1825
1826
1827
1828
1829
1830
1831
1832
1833
1834
1835
1836
1837
1838
1839
1840
1841
1842
1843
1844
1845
1846
1847
1848
1849
1850
1851
1852
1853
1854
1855
1856
1857
1858
1859
1860
1861
1862
1863
1864
1865
1866
1867
1868
1869
1870
1871
1872
1873
1874
1875
1876
1877
1878
1879
1880
1881
1882
1883
1884
1885
1886
1887
1888
1889
1890
1891
1892
1893
1894
1895
1896
1897
1898
1899
1900
1901
1902
1903
1904
1905
1906
1907
1908
1909
1910
1911
1912
1913
1914
1915
1916
1917
1918
1919
1920
1921
1922
1923
1924
1925
1926
1927
1928
1929
1930
1931
1932
1933
1934
1935
1936
1937
1938
1939
1940
1941
1942
1943
1944
1945
1946
1947
1948
1949
1950
1951
1952
1953
1954
1955
1956
1957
1958
1959
1960
1961
1962
1963
1964
1965
1966
1967
1968
1969
1970
1971
1972
1973
1974
1975
1976
1977
1978
1979
1980
1981
1982
1983
1984
1985
1986
1987
1988
1989
1990
1991
1992
1993
1994
1995
1996
1997
1998
1999
2000
2001
2002
2003
2004
2005
2006
2007
2008
2009
2010
2011
2012
2013
2014
2015
2016
2017
2018
2019
2020
2021
2022
2023
2024
2025
2026
2027
2028
2029
2030
2031
2032
2033
2034
2035
2036
2037
2038
2039
2040
2041
2042
2043
2044
2045
2046
2047
2048
2049
2050
2051
2052
2053
2054
2055
2056
2057
2058
2059
2060
2061
2062
2063
2064
2065
2066
2067
2068
2069
2070
2071
2072
2073
2074
2075
2076
2077
2078
2079
2080
2081
2082
2083
2084
2085
2086
2087
2088
2089
2090
2091
2092
2093
2094
2095
2096
2097
2098
2099
2100
2101
2102
2103
2104
2105
2106
2107
2108
2109
2110
2111
2112
2113
2114
2115
2116
2117
2118
2119
2120
2121
2122
2123
2124
2125
2126
2127
2128
2129
2130
2131
2132
2133
2134
2135
2136
2137
2138
2139
2140
2141
2142
2143
2144
2145
2146
2147
2148
2149
2150
2151
2152
2153
2154
2155
2156
2157
2158
2159
2160
2161
2162
2163
2164
2165
2166
2167
2168
2169
2170
2171
2172
2173
2174
2175
2176
2177
2178
2179
2180
2181
2182
2183
2184
2185
2186
2187
2188
2189
2190
2191
2192
2193
2194
2195
2196
2197
2198
2199
2200
2201
2202
2203
2204
2205
2206
2207
2208
2209
2210
2211
2212
2213
2214
2215
2216
2217
2218
2219
2220
2221
2222
2223
2224
2225
2226
2227
2228
2229
2230
2231
2232
2233
2234
2235
2236
2237
2238
2239
2240
2241
2242
2243
2244
2245
2246
2247
2248
2249
2250
2251
2252
2253
2254
2255
2256
2257
2258
2259
2260
2261
2262
2263
2264
2265
2266
2267
2268
2269
2270
2271
2272
2273
2274
2275
2276
2277
2278
2279
2280
2281
2282
2283
2284
2285
2286
2287
2288
2289
2290
2291
2292
2293
2294
2295
2296
2297
2298
2299
2300
2301
2302
2303
2304
2305
2306
2307
2308
2309
2310
2311
2312
2313
2314
2315
2316
2317
2318
2319
2320
2321
2322
2323
2324
2325
2326
2327
2328
2329
2330
2331
2332
2333
2334
2335
2336
2337
2338
2339
2340
2341
2342
2343
2344
2345
2346
2347
2348
2349
2350
2351
2352
2353
2354
2355
2356
2357
2358
2359
2360
2361
2362
2363
2364
2365
2366
2367
2368
2369
2370
2371
2372
2373
2374
2375
2376
2377
2378
2379
2380
2381
2382
2383
2384
2385
2386
2387
2388
2389
2390
2391
2392
2393
2394
2395
2396
2397
2398
2399
2400
2401
2402
2403
2404
2405
2406
2407
2408
2409
2410
2411
2412
2413
2414
2415
2416
2417
2418
2419
2420
2421
2422
2423
2424
2425
2426
2427
2428
2429
2430
2431
2432
2433
2434
2435
2436
2437
2438
2439
2440
2441
2442
2443
2444
2445
2446
2447
2448
2449
2450
2451
2452
2453
2454
2455
2456
2457
2458
2459
2460
2461
2462
2463
2464
2465
2466
2467
2
```

```

1650 ######Square root transformation#####
1651 #Creating the model matrix
1652 View(trainset_new)
1653 View(testset_new)
1654 View(data1)
1655 data1[,c(2,4,5)]<-scale(data1[,c(2,4,5)], center = TRUE, scale = TRUE)
1656 data1[,2]<-as.numeric(data1[,2])
1657 data1[,4]<-as.numeric(data1[,4])
1658 data1[,5]<-as.numeric(data1[,5])
1659 data1[,5]<-numeric(c(data1[,5]))
1660
1661 View(data1)
1662
1663 # Create model matrix with categorical variables as dummies and numerical variable as is
1664 #model_matrix = model.matrix(~model_matrix[, -1], data = data1)
1665 #model_matrix2 = model.matrix(log_delay~, data = data1)[-1]
1666 #model_matrix = model.matrix(log_delay~, data = data1)[-1]
1667 #model_matrix = model.matrix(log_delay~, data = data1)[-1]
1668
1669 #model_matrix = model.matrix(Min.Delay~, data = df)
1670
1671 # Separate the response variable from the predictors
1672 y1 = data1$so_price
1673 data1 = data1[-1]
1674 is.matrix(model_matrix)
1675 dim(model_matrix)
1676
1677 # Create a sequence of lambda values to test
1678 #lambda_seq<-10^seq(-3, 10, length.out = 100)
1679 library(glmnet)
1680 #use cv.glmnet to perform cross-validation and select the optimal lambda value
1681 #cv_fit = cv.glmnet(x = model_matrix, y= y1 , alpha = 0, lambda = lambda_seq)
1682 set.seed(100)
1683 cv_fit = cv.glmnet(x = model_matrix, y= y1 , alpha = 0, lambda = lambda_seq,nfold=10,standardize=FALSE)
1684
1685 # View the optimal lambda value
1686 best_lambda=cv_fit$lambda.1se
1687
1688 # Plot the cross-validation results
1689 dev.off()
1690 plot(cv_fit)
1691
1692 # Fit final model, get its sum of squared residuals and multiple R-squared
1693 #model_cv = glmnet(x = model_matrix, y= y1, alpha = 0, lambda = best_lambda)
1694
1695 #For the test set
1696
1697 data2$testset_new[,2]
1698 data2[,c(2,4,5)]<-scale(data2[,c(2,4,5)], center = TRUE, scale = TRUE)
1699 data2[,2]<-as.numeric(data2[,2])
1700 data2[,4]<-as.numeric(data2[,4])
1701 data2[,5]<-as.numeric(data2[,5])
1702
1703 View(data2)
1704 View(data2)
1705 #model_matrix1 = model.matrix(log_delay~, data = data2)
1706 #model_matrix1 = model.matrix(so_price~, data = data2)[-1]
1707 # Separate the response variable from the predictors
1708 y1 = data2$so_price
1709 data2 = data2[-1]
1710 #model_cv2 = glmnet(x = model_matrix1, y=y2, alpha = 0, lambda = best_lambda2,standardize = FALSE)
1711
1712 # Fit final model, get its sum of squared residuals and multiple R-squared
1713 #model_cv2 = glmnet(x = model_matrix, y= y1, alpha = 0, lambda = best_lambda)
1714
1715 #For the test set,newprice
1716
1717 data2$testset_new[,2]
1718 data2[,c(2,4,5)]<-scale(data2[,c(2,4,5)], center = TRUE, scale = TRUE)
1719 data2[,2]<-as.numeric(data2[,2])
1720 data2[,4]<-as.numeric(data2[,4])
1721 data2[,5]<-as.numeric(data2[,5])
1722
1723 View(data2)
1724 View(data2)
1725 #model_matrix1 = model.matrix(log_delay~, data = data2)
1726 #model_matrix1 = model.matrix(so_price~, data = data2)[-1]
1727 # Separate the response variable from the predictors
1728 y1 = data2$so_price
1729 data2 = data2[-1]
1730 #model_cv2 = glmnet(x = model_matrix1, y=y2, alpha = 0, lambda = best_lambda2,standardize = FALSE)
1731
1732 # Fit final model, get its sum of squared residuals and multiple R-squared
1733 #model_cv2 = glmnet(x = model_matrix, y= y1, alpha = 0, lambda = best_lambda)
1734
1735 #actual-y2
1736 pred1<-y1_hat_ridge1
1737 pred2<-y1_hat_ridge2
1738
1739 d_actual<-testset_newprice
1740 d_pred1<-as.integer(pred1)
1741 d_pred2<-as.integer(exp(pred2))
1742 d_actual<-testset_newprice
1743 d_pred1<-as.integer(pred1)
1744 d_pred2<-as.integer(pred2)
1745
1746 rmse_m = rmse(d.actual, d.pred1)
1747 rmse_m = rmse(d.actual, d.pred2)
1748 rmse_m = mape(d.actual, d.pred1)
1749 rmse_m = mape(d.actual, d.pred2)
1750 mape_m = mape(d.actual, d.pred1)
1751 mape_m = mape(d.actual, d.pred2)
1752 mape_m = mape(d.actual, d.pred1)
1753 mape_m = mape(d.actual, d.pred2)
1754 mape_m = mape(d.actual, d.pred1)
1755
1756
1757
1758
1759
1760
1761
1762
1763
1764
1765
1766
1767
1768
1769
1770
1771
1772
1773
1774
1775
1776
1777
1778
1779
1780
1781
1782
1783
1784
1785
1786
1787
1788
1789
1790
1791
1792
1793
1794
1795
1796
1797
1798
1799
1800
1801
1802
1803
1804
1805
1806
1807
1808
1809
1810
1811
1812
1813
1814
1815
1816
1817
1818
1819
1820
1821
1822
1823
1824
1825
1826
1827
1828
1829
1830
1831
1832
1833
1834
1835
1836
1837
1838
1839
1840
1841
1842
1843
1844
1845
1846
1847
1848
1849
1850
1851
1852
1853
1854
1855
1856
1857
1858
1859
1860
1861
1862
1863
1864
1865
1866
1867
1868
1869
1870
1871
1872
1873
1874
1875
1876
1877
1878
1879
1880
1881
1882
1883
1884
1885
1886
1887
1888
1889
1890
1891
1892
1893
1894
1895
1896
1897
1898
1899
1900
1901
1902
1903
1904
1905
1906
1907
1908
1909
1910
1911
1912
1913
1914
1915
1916
1917
1918
1919
1920
1921
1922
1923
1924
1925
1926
1927
1928
1929
1930
1931
1932
1933
1934
1935
1936
1937
1938
1939
1940
1941
1942
1943
1944
1945
1946
1947
1948
1949
1950
1951
1952
1953
1954
1955
1956
1957
1958
1959
1960
1961
1962
1963
1964
1965
1966
1967
1968
1969
1970
1971
1972
1973
1974
1975
1976
1977
1978
1979
1980
1981
1982
1983
1984
1985
1986
1987
1988
1989
1990
1991
1992
1993
1994
1995
1996
1997
1998
1999
2000
2001
2002
2003
2004
2005
2006
2007
2008
2009
2010
2011
2012
2013
2014
2015
2016
2017
2018
2019
2020
2021
2022
2023
2024
2025
2026
2027
2028
2029
2030
2031
2032
2033
2034
2035
2036
2037
2038
2039
2040
2041
2042
2043
2044
2045
2046
2047
2048
2049
2050
2051
2052
2053
2054
2055
2056
2057
2058
2059
2060
2061
2062
2063
2064
2065
2066
2067
2068
2069
2070
2071
2072
2073
2074
2075
2076
2077
2078
2079
2080
2081
2082
2083
2084
2085
2086
2087
2088
2089
2090
2091
2092
2093
2094
2095
2096
2097
2098
2099
2100
2101
2102
2103
2104
2105
2106
2107
2108
2109
2110
2111
2112
2113
2114
2115
2116
2117
2118
2119
2120
2121
2122
2123
2124
2125
2126
2127
2128
2129
2130
2131
2132
2133
2134
2135
2136
2137
2138
2139
2140
2141
2142
2143
2144
2145
2146
2147
2148
2149
2150
2151
2152
2153
2154
2155
2156
2157
2158
2159
2160
2161
2162
2163
2164
2165
2166
2167
2168
2169
2170
2171
2172
2173
2174
2175
2176
2177
2178
2179
2180
2181
2182
2183
2184
2185
2186
2187
2188
2189
2190
2191
2192
2193
2194
2195
2196
2197
2198
2199
2200
2201
2202
2203
2204
2205
2206
2207
2208
2209
2210
2211
2212
2213
2214
2215
2216
2217
2218
2219
2220
2221
2222
2223
2224
2225
2226
2227
2228
2229
2230
2231
2232
2233
2234
2235
2236
2237
2238
2239
2240
2241
2242
2243
2244
2245
2246
2247
2248
2249
2250
2251
2252
2253
2254
2255
2256
2257
2258
2259
2260
2261
2262
2263
2264
2265
2266
2267
2268
2269
2270
2271
2272
2273
2274
2275
2276
2277
2278
2279
2280
2281
2282
2283
2284
2285
2286
2287
2288
2289
2290
2291
2292
2293
2294
2295
2296
2297
2298
2299
2300
2301
2302
2303
2304
2305
2306
2307
2308
2309
2310
2311
2312
2313
2314
2315
2316
2317
2318
2319
2320
2321
2322
2323
2324
2325
2326
2327
2328
2329
2330
2331
2332
2333
2334
2335
2336
2337
2338
2339
2340
2341
2342
2343
2344
2345
2346
2347
2348
2349
2350
2351
2352
2353
2354
2355
2356
2357
2358
2359
2360
2361
2362
2363
2364
2365
2366
2367
2368
2369
2370
2371
2372
2373
2374
2375
2376
2377
2378
2379
2380
2381
2382
2383
2384
2385
2386
2387
2388
2389
2390
2391
2392
2393
2394
2395
2396
2397
2398
2399
2400
2401
2402
2403
2404
2405
2406
2407
2408
2409
2410
2411
2412
2413
2414
2415
2416
2417
2418
2419
2420
2421
2422
2423
2424
2425
2426
2427
2428
2429
2430
2431
2432
2433
2434
2435
2436
2437
2438
2439
2440
2441
2442
2443
2444
2445
2446
2447
2448
2449
2450
2451
2452
2453
2454
2455
2456
2457
2458
2459
2460
2461
2462
2463
2464
2465
2466
2467
2468
2469
2470
2471
2472
2473
2474
2475
2476
2477
2478
2479
2480
2481
2482
2483
2484
2485
2486
2487
2488
2489
2490
2491
2492
2493
2494
2495
2496
2497
2498
2499
2500
2501
2502
2503
2504
2505
2506
2507
2508
2509
2510
2511
2512
2513
2514
2515
2516
2517
2518
2519
2520
2521
2522
2523
2524
2525
2526
2527
2528
2529
2530
2531
2532
2533
2534
2535
2536
2537
2538
2539
2540
2541
2542
2543
2544
2545
2546
2547
2548
2549
2550
2551
2552
2553
2554
2555
2556
2557
2558
2559
2560
2561
2562
2563
2564
2565
2566
2567
2568
2569
2570
2571
2572
2573
2574
2575
2576
2577
2578
2579
2580
2581
2582
2583
2584
2585
2586
2587
2588
2589
2590
2591
2592
2593
2594
2595
2596
2597
2598
2599
2600
2601
2602
2603
2604
2605
2606
2607
2608
2609
2610
2611
2612
2613
2614
2615
2616
2617
2618
2619
2620
2621
2622
2623
2624
2625
2626
2627
2628
2629
2630
2631
2632
2633
2634
2635
2636
2637
2638
2639
2640
2641
2642
2643
2644
2645
2646
2647
2648
2649
2650
2651
2652
2653
2654
2655
2656
2657
2658
2659
2660
2661
2662
2663
2664
2665
2666
2667
2668
2669
2670
2671
2672
2673
2674
2675
2676
2677
2678
2679
2680
2681
2682
2683
2684
2685
2686
2687
2688
2689
2690
2691
2692
2693
2694
2695
2696
2697
2698
2699
2700
2701
2702
2703
2704
2705
2706
2707
2708
2709
2710
2711
2712
2713
2714
2715
2716
2717
2718
2719
2720
2721
2722
2723
2724
2725
2726
2727
2728
2729
2730
2731
2732
2733
2734
2735
2736
2737
2738
2739
2740
2741
2742
2743
2744
2745
2746
2747
2748
2749
2750
2751
2752
2753
2754
2755
2756
2757
2758
2759
2760
2761
2762
2763
2764
2765
2766
2767
2768
2769
2770
2771
2772
2773
2774
2775
2776
2777
2778
2779
2780
2781
2782
2783
2784
2785
2786
2787
2788
2789
2790
2791
2792
2793
2794
2795
2796
2797
2798
2799
2800
2801
2802
2803
2804
2805
2806
2807
2808
2809
2810
2811
2812
2813
2814
2815
2816
2817
2818
2819
2820
2821
2822
2823
2824
2825
2826
2827
2828
2829
2830
2831
2832
2833
2834
2835
2836
2837
2838
2839
2840
2841
2842
2843
2844
2845
2846
2847
2848
2849
2850
2851
2852
2853
2854
2855
2856
2857
2858
2859
2860
2861
2862
2863
2864
2865
2866
2867
2868
2869
2870
2871
2872
2873
2874
2875
2876
2877
2878
2879
2880
2881
2882
2883
2884
2885
2886
2887
2888
2889
2890
2891
2892
2893
2894
2895
2896
2897
2898
2899
2900
2901
2902
2903
2904
2905
2906
2907
2908
2909
2910
2911
2912
2913
2914
2915
2916
2917
2918
2919
2920
2921
2922
2923
2924
2925
2926
2927
2928
2929
2930
2931
2932
2933
2934
2935
2936
2937
2938
2939
2940
2941
2942
2943
2944
2945
2946
2947
2948
2949
2950
2951
2952
2953
2954
2955
2956
2957
2958
2959
2960
2961
2962
2963
2964
2965
2966
2967
2968
2969
2970
2971
2972
2973
2974
2975
2976
2977
2978
2979
2980
2981
2982
2983
2984
2985
2986
2987
2988
2989
2990
2991
299
```

```

2133 data2_imp[factor_cols] = apply(data2_imp[factor_cols], 2, function(x) as.factor(x))
2134
2135 y_hat_enet = predict(rf_model, data2_imp[,-1])
2136
2137 actual_y2
2138 d_actual=actual_setnewprice
2139 d_pred-as.integer(pred2)
2140 mape_m = mape(d_actual, d_pred)
2141 rmse_m = rmse(d_actual, d_pred)
2142
2143 mape_m
2144 rmse_m
2145 rmse_m = rmse(y2,pred)
2146
2147 mape_m = mape(y2, pred)
2148 mape_m
2149
2150
2151 #xgboss3
2152 set.seed(100)
2153 Xtrain = xgb.DMatrix(data = as.matrix(data1[,-1]),label=y1)
2154 y_train = datalimp$price
2155 xgb_model = train(Xtrain,
2156   y_train,
2157   objective = "xgbTree",
2158   max_depth = c(2,4,6),
2159   colsample_bytree = c(0.5,0.6),
2160   subsample = c(0.5,0.6),
2161   gamma=0.1,
2162   min_child_weight = c(1,3,5)
2163 )
2164
2165 plot(xgb_model)
2166 y_hat_enet = predict(xgb_model, Xtrain)
2167 actual_y1
2168 pred=c(y_hat_enet)
2169 d_actual=actual_setnewprice
2170 d_pred-as.integer(pred2)
2171 mape_m = mape(d_actual, d_pred)
2172 rmse_m = rmse(d_actual, d_pred)
2173
2174 mape_m
2175 rmse_m
2176
2177
2178
2179
2180
2181 tuneGrid = expand.grid(nrounds = c(500,1000),
2182   eta=0.1,
2183   max_depth = c(2,4,6),
2184   colsample_bytree = c(0.5,0.6),
2185   subsample = c(0.5,0.6),
2186   gamma=0.1,
2187   min_child_weight = c(1,3,5)
2188 )
2189
2190
2191 plot(xgb_model)
2192 y_hat_enet = predict(xgb_model, Xtrain)
2193 actual_y1
2194 pred=c(y_hat_enet)
2195 d_actual=actual_setnewprice
2196 d_pred-as.integer(pred2)
2197 mape_m = mape(d_actual, d_pred)
2198 rmse_m = rmse(d_actual, d_pred)
2199 mape_m
2200
2201 # get variable importance
2202 var_imp=varImp(xgb_model)
2203
2204 # print the variable importance
2205 print(var_imp)
2206
2207 # load ggplot2 library
2208 library(ggplot2)
2209
2210 # plot the variable importance
2211 ggplot(data = var_imp, aes(x = reorder(rnames(var_imp), Overall), y = Overall)) +
2212   geom_bar(stat = "identity". fill = "steelblue") +
2213
2214
2215
2216
2217
2218
2219
2220
2221
2222
2223
2224
2225
2226
2227
2228
2229
2230
2231
2232
2233
2234
2235
2236
2237
2238
2239
2240
2241
2242
2243
2244
2245
2246
2247
2248
2249
2250
2251
2252
2253
2254
2255
2256
2257
2258
2259
2260
2261
2262
2263
2264
2265
2266
2267
2268
2269
2270
2271
2272
2273
2274
2275
2276
2277
2278
2279
2280
2281
2282
2283
2284
2285
2286
2287
2288
2289
2290
2291
2292
2293
2294
2295
2296
2297
2298
2299
2299
2300
2301
2302
2303
2304
2305
2306
2307
2308
2309
2310
2311
2312
2313
2314
2315
2316
2317
2318
2319
2320
2321
2322
2323
2324
2325
2326
2327
2328
2329
2330
2331
2332
2333
2334
2335
2336
2337
2338
2339
2340
2341
2342
2343
2344
2345
2346
2347
2348
2349
2350
2351
2352
2353
2354
2355
2356
2357
2358
2359
2360
2361
2362
2363
2364
2365
2366
2367
2368
2369
2370
2371
2372
2373
2374
2375
2376
2377
2378
2379
2380
2381
2382
2383
2384
2385
2386
2387
2388
2389
2390
2391
2392
2393
2394
2395
2396
2397
2398
2399
2400
2401
2402
2403
2404
2405
2406
2407
2408
2409
2410
2411
2412
2413
2414
2415
2416
2417
2418
2419
2420
2421
2422
2423
2424
2425
2426
2427
2428
2429
2430
2431
2432
2433
2434
2435
2436
2437
2438
2439
2440
2441
2442
2443
2444
2445
2446
2447
2448
2449
2450
2451
2452
2453
2454
2455
2456
2457
2458
2459
2460
2461
2462
2463
2464
2465
2466
2467
2468
2469
2470
2471
2472
2473
2474
2475
2476
2477
2478
2479
2480
2481
2482
2483
2484
2485
2486
2487
2488
2489
2490
2491
2492
2493
2494
2495
2496
2497
2498
2499
2500
2501
2502
2503
2504
2505
2506
2507
2508
2509
2510
2511
2512
2513
2514
2515
2516
2517
2518
2519
2520
2521
2522
2523
2524
2525
2526
2527
2528
2529
2530
2531
2532
2533
2534
2535
2536
2537
2538
2539
2540
2541
2542
2543
2544
2545
2546
2547
2548
2549
2550
2551
2552
2553
2554
2555
2556
2557
2558
2559
2560
2561
2562
2563
2564
2565
2566
2567
2568
2569
2570
2571
2572
2573
2574
2575
2576
2577
2578
2579
2580
2581
2582
2583
2584
2585
2586
2587
2588
2589
2590
2591
2592
2593
2594
2595
2596
2597
2598
2599
2600
2601
2602
2603
2604
2605
2606
2607
2608
2609
2610
2611
2612
2613
2614
2615
2616
2617
2618
2619
2620
2621
2622
2623
2624
2625
2626
2627
2628
2629
2630
2631
2632
2633
2634
2635
2636
2637
2638
2639
2640
2641
2642
2643
2644
2645
2646
2647
2648
2649
2650
2651
2652
2653
2654
2655
2656
2657
2658
2659
2660
2661
2662
2663
2664
2665
2666
2667
2668
2669
2670
2671
2672
2673
2674
2675
2676
2677
2678
2679
2680
2681
2682
2683
2684
2685
2686
2687
2688
2689
2690
2691
2692
2693
2694
2695
2696
2697
2698
2699
2700
2701
2702
2703
2704
2705
2706
2707
2708
2709
2710
2711
2712
2713
2714
2715
2716
2717
2718
2719
2720
2721
2722
2723
2724
2725
2726
2727
2728
2729
2730
2731
2732
2733
2734
2735
2736
2737
2738
2739
2740
2741
2742
2743
2744
2745
2746
2747
2748
2749
2750
2751
2752
2753
2754
2755
2756
2757
2758
2759
2760
2761
2762
2763
2764
2765
2766
2767
2768
2769
2770
2771
2772
2773
2774
2775
2776
2777
2778
2779
2780
2781
2782
2783
2784
2785
2786
2787
2788
2789
2790
2791
2792
2793
2794
2795
2796
2797
2798
2799
2800
2801
2802
2803
2804
2805
2806
2807
2808
2809
2810
2811
2812
2813
2814
2815
2816
2817
2818
2819
2820
2821
2822
2823
2824
2825
2826
2827
2828
2829
2830
2831
2832
2833
2834
2835
2836
2837
2838
2839
2840
2841
2842
2843
2844
2845
2846
2847
2848
2849
2850
2851
2852
2853
2854
2855
2856
2857
2858
2859
2860
2861
2862
2863
2864
2865
2866
2867
2868
2869
2870
2871
2872
2873
2874
2875
2876
2877
2878
2879
2880
2881
2882
2883
2884
2885
2886
2887
2888
2889
2889
2890
2891
2892
2893
2894
2895
2896
2897
2898
2899
2899
2900
2901
2902
2903
2904
2905
2906
2907
2908
2909
2910
2911
2912
2913
2914
2915
2916
2917
2918
2919
2920
2921
2922
2923
2924
2925
2926
2927
2928
2929
2930
2931
2932
2933
2934
2935
2936
2937
2938
2939
2940
2941
2942
2943
2944
2945
2946
2947
2948
2949
2950
2951
2952
2953
2954
2955
2956
2957
2958
2959
2960
2961
2962
2963
2964
2965
2966
2967
2968
2969
2970
2971
2972
2973
2974
2975
2976
2977
2978
2979
2980
2981
2982
2983
2984
2985
2986
2987
2988
2989
2990
2991
2992
2993
2994
2995
2996
2997
2998
2999
2999
3000
3001
3002
3003
3004
3005
3006
3007
3008
3009
3010
3011
3012
3013
3014
3015
3016
3017
3018
3019
3020
3021
3022
3023
3024
3025
3026
3027
3028
3029
3030
3031
3032
3033
3034
3035
3036
3037
3038
3039
3040
3041
3042
3043
3044
3045
3046
3047
3048
3049
3050
3051
3052
3053
3054
3055
3056
3057
3058
3059
3060
3061
3062
3063
3064
3065
3066
3067
3068
3069
3070
3071
3072
3073
3074
3075
3076
3077
3078
3079
3080
3081
3082
3083
3084
3085
3086
3087
3088
3089
3090
3091
3092
3093
3094
3095
3096
3097
3098
3099
3099
3100
3101
3102
3103
3104
3105
3106
3107
3108
3109
3110
3111
3112
3113
3114
3115
3116
3117
3118
3119
3120
3121
3122
3123
3124
3125
3126
3127
3128
3129
3130
3131
3132
3133
3134
3135
3136
3137
3138
3139
3140
3141
3142
3143
3144
3145
3146
3147
3148
3149
3150
3151
3152
3153
3154
3155
3156
3157
3158
3159
3160
3161
3162
3163
3164
3165
3166
3167
3168
3169
3170
3171
3172
3173
3174
3175
3176
3177
3178
3179
3180
3181
3182
3183
3184
3185
3186
3187
3188
3189
3190
3191
3192
3193
3194
3195
3196
3197
3198
3199
3199
3200
3201
3202
3203
3204
3205
3206
3207
3208
3209
3210
3211
3212
3213
3214
3215
3216
3217
3218
3219
3220
3221
3222
3223
3224
3225
3226
3227
3228
3229
3230
3231
3232
3233
3234
3235
3236
3237
3238
3239
3239
3240
3241
3242
3243
3244
3245
3246
3247
3248
3249
3250
3251
3252
3253
3254
3255
3256
3257
3258
3259
3259
3260
3261
3262
3263
3264
3265
3266
3267
3268
3269
3270
3271
3272
3273
3274
3275
3276
3277
3278
3279
3279
3280
3281
3282
3283
3284
3285
3286
3287
3288
3289
3289
3290
3291
3292
3293
3294
3295
3296
3297
3298
3299
3299
3300
3301
3302
3303
3304
3305
3306
3307
3308
3309
3309
3310
3311
3312
3313
3314
3315
3316
3317
3318
3319
3319
3320
3321
3322
3323
3324
3325
3326
3327
3328
3329
3329
3330
3331
3332
3333
3334
3335
3336
3337
3338
3339
3339
3340
3341
3342
3343
3344
3345
3346
3347
3348
3349
3349
3350
3351
3352
3353
3354
3355
3356
3357
3358
3359
3359
3360
3361
3362
3363
3364
3365
3366
3367
3368
3369
3369
3370
3371
3372
3373
3374
3375
3376
3377
3378
3379
3379
3380
3381
3382
3383
3384
3385
3386
3387
3388
3389
3389
3390
3391
3392
3393
3394
3395
3396
3397
3398
3399
3399
3400
3401
3402
3403
3404
3405
3406
3407
3408
3409
3409
3410
3411
3412
3413
3414
3415
3416
3417
3418
3419
3419
3420
3421
3422
3423
3424
3425
3426
3427
3428
3429
3429
3430
3431
3432
3433
3434
3435
3436
3437
3438
3439
3439
3440
3441
3442
3443
3444
3445
3446
3447
3448
3449
3449
3450
3451
3452
3453
3454
3455
3456
3457
3458
3459
3459
3460
3461
3462
3463
3464
3465
3466
3467
3468
3469
3469
3470
3471
3472
3473
3474
3475
3476
3477
3478
3479
3479
3480
3481
3482
3483
3484
3485
3486
3487
3488
3489
3489
3490
3491
3492
3493
3494
3495
3496
3497
3498
3499
3499
3500
3501
3502
3503
3504
3505
3506
3507
3508
3509
3509
3510
3511
3512
3513
3514
3515
3516
3517
3518
3519
3519
3520
3521
3522
3523
3524
3525
3526
3527
3528
3529
3529
3530
3531
3532
3533
3534
3535
3536
3537
3538
3539
3539
3540
3541
3542
3543
3544
3545
3546
3547
3548
354
```