ALY6015 – INTERMEDIATE ANALYTICS

FINAL PROJECT PROPOSAL – CAR PRICE PREDICTION

Team Members

Name	NUID
AESHWANTH VENKIDESH	**002934780**
SAI SACHIN CHANDRASEKAR	**002934358**
SRIKRISHNA VENKATESAN	**002778279**

FEBURARY 18, 2022

Introduction:

The car price prediction dataset is a collection of data that contains information about various car attributes such as make, model, year, engine size, mileage, and price. The objective of this analysis is to predict the price of a car based on its attributes. The steps involved in this analysis include Exploratory Data Analysis (EDA), selecting appropriate statistical methods, and building predictive models.

The price of the car is a dependent variable, and a correlation matrix is shown to identify the independent variables for the dependent variables. Following this, the data is tested for overfitting using the two popular methods like Lasso and Ridge methods. Overfitting occurs when a model is too complex and fits the training data too well, resulting in poor performance on unseen data.

After the model is found that there is no overfitting, the categorical variables are also analysed by using the logistic regression method. The logistic regression model is based on the logistic function, which is used to model the relationship between the independent variables and the categorical variable. The logistic function is a sigmoid curve that maps any real-valued number to a value between 0 and 1.

Question:

1. What is the impact of the year the car was manufactured on the sales price of old cars in the dataset, and how does this relationship compare to the impact of other independent variables such as the kms driven, engine cc and number of seats in the car?
2. Is there any evidence of overfitting in the Lasso and Ridge regression models when predicting the prices of old cars in the old car price prediction dataset, and how does each model address overfitting?
3. Is there a relationship between the transmission type of a car and the sale price, and can this relationship be used to make predictions about the sale price of a car based on its transmission type?

Methods:

- To model the relationship between the year the car was manufactured and other independent variables such as the make and model, and the number of seats in the car, can use a multiple linear regression model. This type of model allows you to examine the impact of multiple independent variables on the dependent variable, which in this case is the sales price of the old car.
- In order to determine the presence of overfitting in the old car price prediction model, a training and testing dataset will be created. The Lasso and Ridge regression models will then be applied to these datasets to evaluate their performance. The root mean square values will be calculated to determine the degree of overfitting, if any, in the model. This will help us identify and eliminate overfitting in the data, ensuring that the model provides accurate and reliable predictions.
- In logistic regression, a mathematical equation is created to model the relationship between the dependent and independent variables. The equation takes into account the coefficients of the independent variables, which can be used to make predictions about the sale price of old cars based on their transmission type. Logistic regression provides a powerful tool for analysing the relationship between the transmission type of old cars and their sale price and can be used to make predictions and inform business decisions in the old car market.

Exploratory Data Analysis (EDA):

The car price prediction dataset is structured as a table with 5512 rows (observations or records) and 9 columns (attributes or variables). Each row represents an individual car and its attributes, and each column represents a specific attribute of the car. The 9 attributes in the dataset are:

Car Name: The name of the car model

Driven KM: The total distance driven by the car

Fuel Type: The type of fuel used by the car (e.g., petrol, diesel, etc.)

Transmission: The type of transmission system used by the car (e.g., manual, automatic, etc.)

Year of manufacture: The year in which the car was manufactured

Engine CC: The engine capacity of the car in cubic centimetres

No. of seats: The number of seats in the car

Price: The price of the car (the target or response variable)

Owner Type: The type of owner of the car (e.g., first owner, second owner, etc.)

Some of these attributes, such as Car Name, Fuel Type, Transmission, and Owner Type, are categorical variables and are represented by character values. Other attributes, such as Driven KM, Year of manufacture, Engine CC, No. of seats, and Price, are numerical variables and are represented by numeric values. This structure of the dataset allows for a variety of statistical methods to be used in the analysis to predict the price of an old car based on its attributes. Below the variables are mentioned in a tabular representation for easy identification of the categorical and numerical variables.

Sr.No	Variables	Type	Meaning
1	Car name	Character	Car names
2	Car price in rupees	Character	Price of the car in rupees
3	Kms driven	Character	Km driven by the car before sale
4	Fuel type	Character	Fuel type depends on the car
5	Transmission	Character	Transmission type
6	Ownership	Character	Ownership type
7	Manufacture	Integer	Manufacture year
8	Engine	Character	Engine power
9	Seats	Character	Number of seats

Data Cleaning:

As can be observed, the dataset contains the majority of character characteristics, but it will not be useful for logistic and linear regression because those methods require numerical and continuous variables, which are not present in this dataset. Data cleaning must therefore be done.

The str() method, which provides extensive information about the variables, is used to describe the dataset's structure. As can be seen in the figure below, every variable has some form of character data that needs to be eliminated. For instance, the car price can be converted to numbers from lakhs and crore. The engine values can be expressed in cubic centimetres and the Kilometer Driven variable can

simply have the value of the Kilometer, but they are not required to be included in every record.

```
> str(dataset)
'data.frame':   5512 obs. of  9 variables:
 $ car_name          : chr  "Jeep Compass 2.0 Longitude Option BSIV" "Renault Duster RXZ Turbo CVT" "Toyota Camry 2.5 G" "Honda Jazz VX CV
T" ...
 $ car_prices_in_rupee: chr  "10.03 Lakh" "12.83 Lakh" "16.40 Lakh" "7.77 Lakh" ...
 $ kms_driven        : chr  "86,226 kms" "13,248 kms" "60,343 kms" "26,696 kms" ...
 $ fuel_type         : chr  "Diesel" "Petrol" "Petrol" "Petrol" ...
 $ transmission      : chr  "Manual" "Automatic" "Automatic" "Automatic" ...
 $ ownership         : chr  "1st Owner" "1st Owner" "1st Owner" "1st Owner" ...
 $ manufacture       : int  2017 2021 2016 2018 2016 2017 2017 2021 2015 2017 ...
 $ engine            : chr  "1956 cc" "1330 cc" "2494 cc" "1199 cc" ...
 $ Seats             : chr  "5 Seats" "5 Seats" "5 Seats" "5 Seats" ...
>
```

The below image represents the structure of the dataset after the cleaning has been performed and converted to numeric data values. The Kilometers driven, ownership, seats, and engine columns were all cleaned to ensure accuracy and avoid any missing values.

```
> summary(dataset)
   car_name         car_prices_in_rupee   kms_driven      fuel_type         transmission
 Length:5512        Min.   :   35000    Min.   :   250   Length:5512        Length:5512
 Class :character   1st Qu.:  315000    1st Qu.: 33152   Class :character   Class :character
 Mode  :character   Median :  550000    Median : 59000   Mode  :character   Mode  :character
                    Mean   : 1162510    Mean   : 63212
                    3rd Qu.: 1025750    3rd Qu.: 84265
                    Max.   :19200000    Max.   :560000
   ownership      manufacture      engine          Seats
 Min.   :0.000   Min.   :1995   Min.   :   0   Min.   :2.000
 1st Qu.:1.000   1st Qu.:2013   1st Qu.:1197   1st Qu.:5.000
 Median :1.000   Median :2016   Median :1396   Median :5.000
 Mean   :1.422   Mean   :2015   Mean   :1532   Mean   :5.251
 3rd Qu.:2.000   3rd Qu.:2018   3rd Qu.:1950   3rd Qu.:5.000
 Max.   :5.000   Max.   :2022   Max.   :5950   Max.   :8.000
>
```

Below image depicts the relationship between the year of manufacture for a car and its price in Rupees. The color of each data point represents the type of transmission for the car, either automatic or manual.

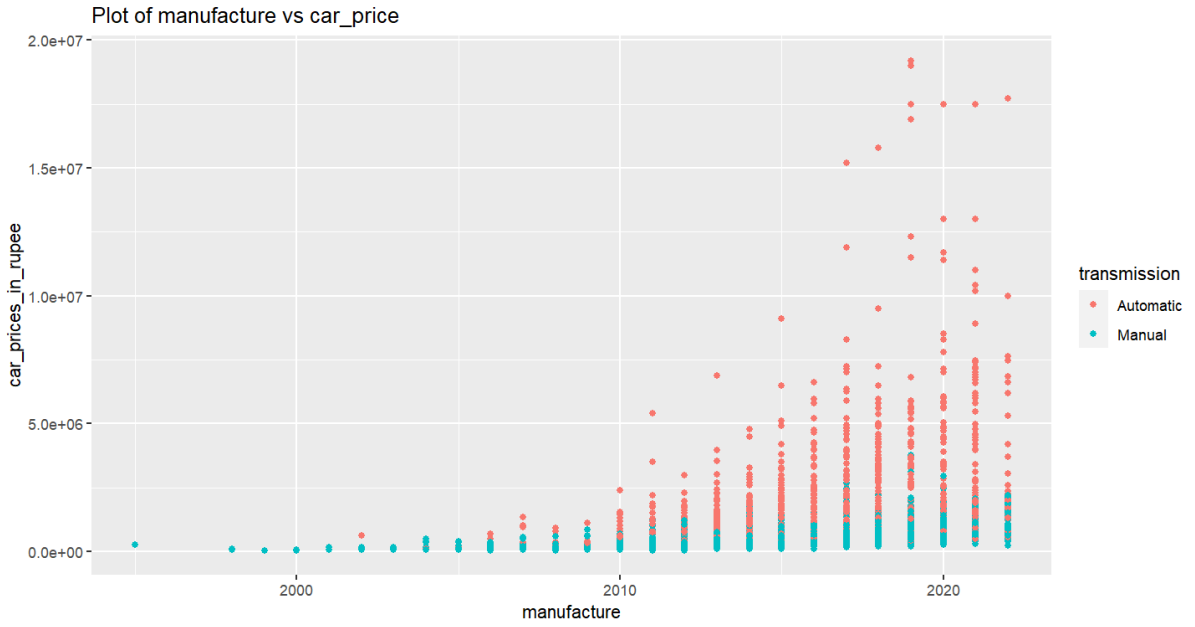

Figure 1.1: Plot representing the manufacture year and the car price in rupees

The plot indicates a steady rise in car prices as the years go by. Additionally, there is a higher number of car prices under automatic transmission compared to those with manual transmission. The number

of cars which the price is higher is during a period of decade from 2010 to 2020, with some exceptional case. The price of the car is calculated according to the manufacture year of the car."

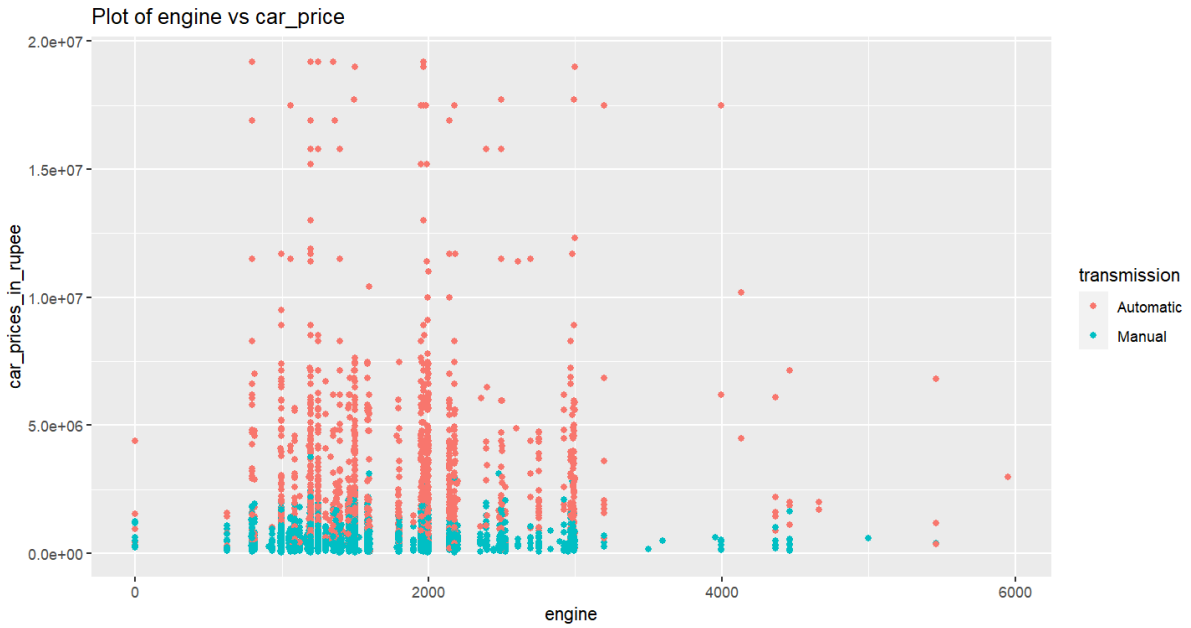

Figure 1.2: Plot representing the engine and the car price in rupees based on transmission

Above plot indicates that the engine of a car has a significant impact on its sales price. When the engine's cc is between 1000 and 3000, there is a higher number of sales with some scattered values (higher in price). Additionally, manual cars are not as popular as automatic cars, resulting in decrease in car price.

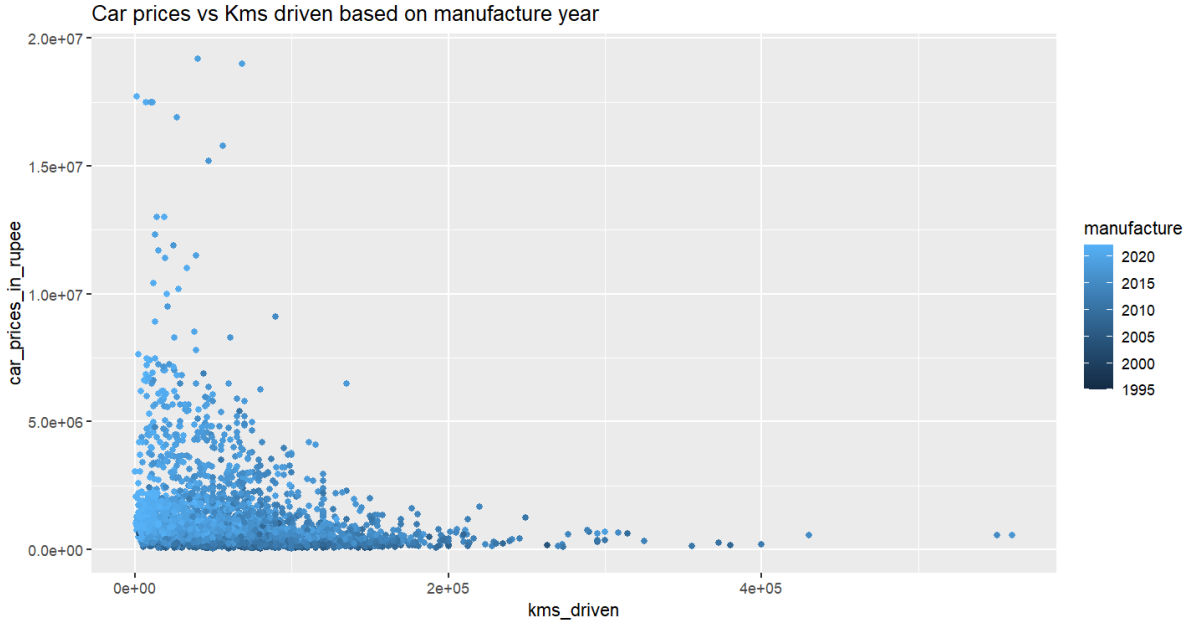

Figure 1.3: Plot representing Car price vs KMS driven based on manufacture year

The image illustrates the correlation between car prices and the number of kilometers driven based on the manufacturing year. It is clear from the image that newer cars have higher prices. Furthermore, the

cars with the lowest mileage have the highest prices. This trend suggests that newer cars with fewer kilometers driven are more highly valued in the market, likely due to their perceived higher quality and longer lifespan.

Correlation Plot:

A correlation plot is a graphical representation of the relationship between two or more variables. It is used to visualize the linear relationship between the variables and to determine if there is a positive, negative, or neutral relationship between them. The plot typically consists of dots or scatter points, each representing one data point, and the points are positioned based on their values for the two variables. The direction and strength of the relationship are indicated by the slope and distribution of the points on the plot.

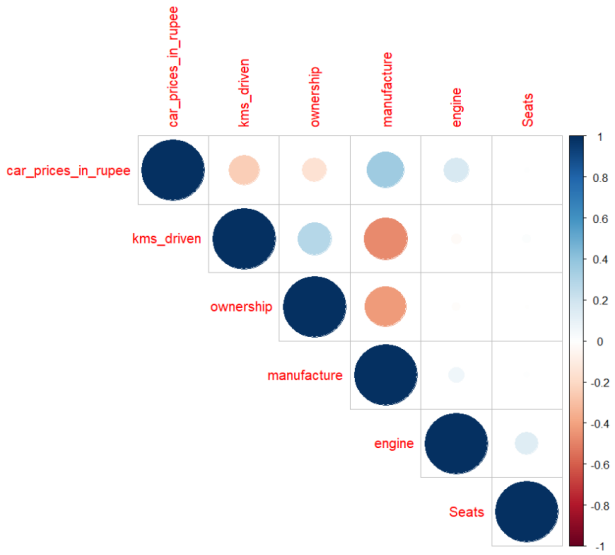

Figure 1.4 Correlation Plot of the car dataset

Observations:

- The variable car price in rupees is the variable of interest and with that it is seen that the variable manufacture year is positively correlated and the variable Engine in cubic centimeters is also correlated.
- The other variables like Ownership , the Kms driven which is in Cubic centimeter variable are negatively correlated. This tells that for every increase in the value of car price in rupees the variables like ownership and kms driven are gonna decrease so there is no use in comparing these variables.
- The variable Seats is seen to have a neutral relationship with that of the car price in rupees variable.

Analysis:

Linear regression is a statistical method used to model the relationship between a dependent variable and one or more independent variables. The goal of the initial analysis using linear regression is to fit a linear equation to the data, which can then be used to make predictions about the dependent variable based on the values of the independent variables. Given below is the summary of the linear regression model and the car price is considered as the independent variable and the manufacture, engine, seats, kms driven and the ownership are all considered the dependant variables.

```
> fit <- lm(car_prices_in_rupee~kms_driven+fuel_type+transmission+ownership+manufacture+engine+Seats,data=dataset)
> summary(fit)

Call:
lm(formula = car_prices_in_rupee ~ kms_driven + fuel_type + transmission +
    ownership + manufacture + engine + Seats, data = dataset)

Residuals:
     Min       1Q   Median       3Q      Max
-3390852  -478324   -84104   264046 16376579

Coefficients:
              Estimate Std. Error t value Pr(>|t|)
(Intercept)  -2.066e+08  1.540e+07 -13.421  < 2e-16 ***
kms_driven   -3.235e+00  6.901e-01  -4.688 2.83e-06 ***
fuel_type     2.915e+05  4.220e+04   6.908 5.55e-12 ***
transmission  1.989e+06  5.518e+04  36.048  < 2e-16 ***
ownership    -6.575e+04  3.779e+04  -1.740   0.0819 .
manufacture   1.026e+05  7.626e+03  13.454  < 2e-16 ***
engine        2.441e+02  4.356e+01   5.604 2.21e-08 ***
Seats        -1.958e+04  3.278e+04  -0.598   0.5502
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 1632000 on 4734 degrees of freedom
Multiple R-squared:  0.3482,    Adjusted R-squared:  0.3472
F-statistic: 361.3 on 7 and 4734 DF,  p-value: < 2.2e-16
```

Observations:

- From the data, it is seen that the manufacture, fuel_type,transmission,engine, kms driven are all seeming to be significant since the p-value is less than 0.05.
- The Seats and the ownership are clearly not significant as the p value of all these points are greater than 0.05 and this means that those coefficients shouldn't be considered for analysis.
- It is seen that the r-squared values are both below 40% and it is not a perfect model.

```
> fit_lm <- lm(car_prices_in_rupee~fuel_type+transmission+manufacture+engine+kms_driven,data=numeric)
> summary(fit_lm)

Call:
lm(formula = car_prices_in_rupee ~ fuel_type + transmission +
    manufacture + engine + kms_driven, data = numeric)

Residuals:
     Min       1Q   Median       3Q      Max
-1345011  -173244   -28557   134197  1429681

Coefficients:
              Estimate Std. Error t value Pr(>|t|)
(Intercept)  -1.156e+08  3.264e+06 -35.430  < 2e-16 ***
fuel_type     1.532e+05  8.889e+03  17.239  < 2e-16 ***
transmission  5.308e+05  1.272e+04  41.716  < 2e-16 ***
manufacture   5.750e+04  1.618e+03  35.546  < 2e-16 ***
engine        5.623e+01  1.081e+01   5.203 2.06e-07 ***
kms_driven   -5.326e-01  1.838e-01  -2.898  0.00378 **
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 315700 on 3957 degrees of freedom
Multiple R-squared:  0.5321,    Adjusted R-squared:  0.5315
F-statistic: 900.1 on 5 and 3957 DF,  p-value: < 2.2e-16
```

Observations:

- The above model is very significant since all the p-values of all the features are less than the 95% confidence interval range of 0.05.
- The Adjusted R squared value of the model is 0.15 which represents that the model will be able to accurately make prediction about 15% of the data.

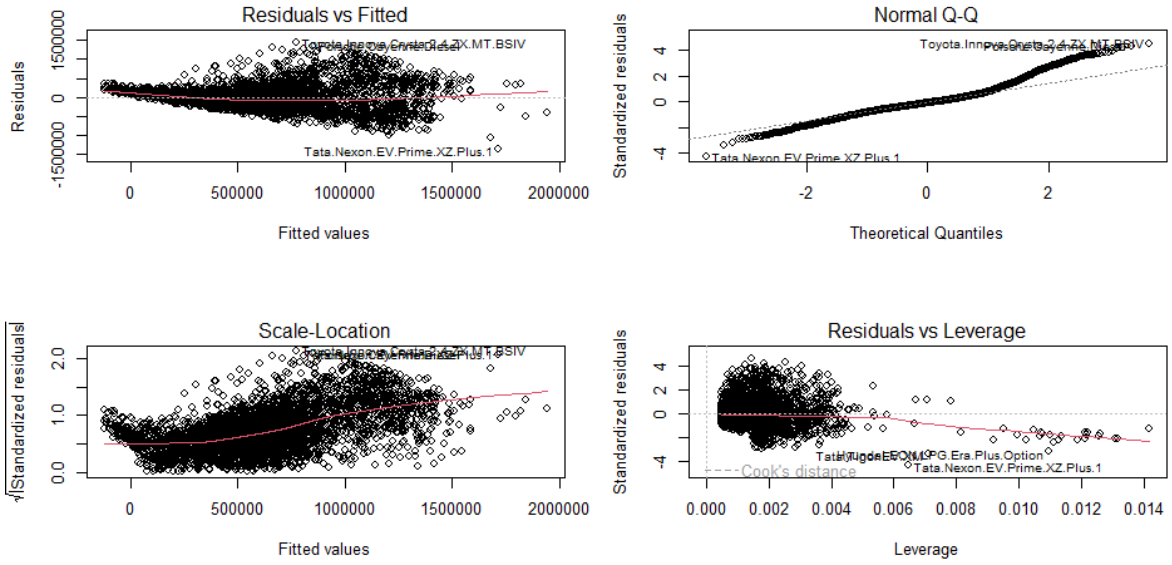

Figure:1.1: Plot to represent the regression model

Residuals vs fitted, Normal Q-Q, Scale location, and residuals vs leverage are the four graphs created. The residuals vs. fitted plot is used to find outliers and nonlinearity. It demonstrates how many outliers are there and how linearity is disrupted. We can plainly observe the presence of the outliers that need to be dealt with in the abovementioned graphs.

Checking for multicollinearity:

When two or more independent variables in a regression model have a high correlation with one another, multicollinearity arises. This implies that in a regression model, one independent variable can be predicted from another independent variable.

```
> vif(fit_lm)
   fuel_type transmission  manufacture       engine   kms_driven
    1.132186     1.025820     1.390604     1.011663     1.531457
> outlierTest(model = fit_lm)
                                    rstudent unadjusted p-value Bonferroni p
Toyota.Innova.Crysta.2.4.ZX.MT.BSIV 4.543567          5.696e-06     0.022573
```

From the above image we can see that there is no variables which are causes of concern as the multicollinear test indicates the values are less than 5 for all the 3 variables and hence the dataset are not having any multicollinearity. The next test values indicates the presence of the outliers within the dataset that can be observed with the help of outlierTest(). From the above image it is observed that there are literally less or no outliers which are already taken care off.

Splitting the car price dataset into train and test values:

A machine learning model is trained on a dataset called the training set, and then evaluated on a separate dataset called the test set. The purpose of using a separate test set is to evaluate the performance of the trained model on unseen data, so that we can get an estimate of how well the model will perform on new, real-world data.

The training set is used to train the model, which means that the model is fed the input-output pairs from the training set and the model's parameters are adjusted to minimize the prediction error. The test set is used to evaluate the performance of the trained model. The model is given input from the test set and the model's predictions are compared to the actual output to calculate the performance metrics such as accuracy, precision, recall, F1-score etc.

A common practice is to split the original dataset into two parts, training set and test set, with a typical split being **80% for training and 20% for testing**. Below image displays the way how the dataset is slitted into two models and used for the analysis with 60 and 40 for better model accuracy.

```
> #1. Split the data
> set.seed(123)
> trainIndex <- sample(x= nrow(df), size=nrow(df)*0.6)
> train <- df[trainIndex,]
> test <- df[-trainIndex,]
> train_x <- model.matrix(car_prices_in_rupee~.,train)[,-1]
> test_x <- model.matrix(car_prices_in_rupee~., test)[,-1]
> train_y <- train$car_prices_in_rupee
> test_y <- test$car_prices_in_rupee
```

Ridge Regression:

Ridge Regression is a type of regularization method used in **linear regression** to address the issue of **multicollinearity among the predictors**. In ridge regression, a penalty term, **the L2 regularization term,** is added to **the ordinary least squares (OLS) objective** function. This penalty term shrinks the **regression coefficients towards zero,** which helps to **reduce the variance** of the model and improve its generalization performance.

```
> #lambda min (minimizes out of sample box)
> lambda.min <- cv.ridge$lambda.min
> lambda.min
[1] 24214.63
> log(lambda.min)
[1] 10.09471
>
> #lambda.1se (largest value of lambda within 1 standard error of lambda min)
> lambda.1se <- cv.ridge$lambda.1se
> lambda.1se
[1] 107285.9
> log(lambda.1se)
[1] 11.58325
```

- Lambda.min and lambda.1se are two values used in the context of regularization.
- Lambda.min is the value of the regularization parameter that results in the smallest cross-validation error. This means that for this value of the parameter, the model has the best performance on the validation data.
- lambda.1se is the value of the regularization parameter that results in one standard error above the minimum cross-validation error. This value is considered as a good default choice for the regularization parameter, as it balances model complexity and model fit to the data.
- Comparing the values of lambda.min 10.08818 and lambda.1se 11.5762, it can be seen that lambda.1se is much larger than lambda.min.
- This means that the model will be less complex (i.e., have fewer coefficients) when lambda.1se is used as the regularization parameter. This is because a larger value of the regularization parameter reduces the coefficients more, effectively shrinking them towards zero.

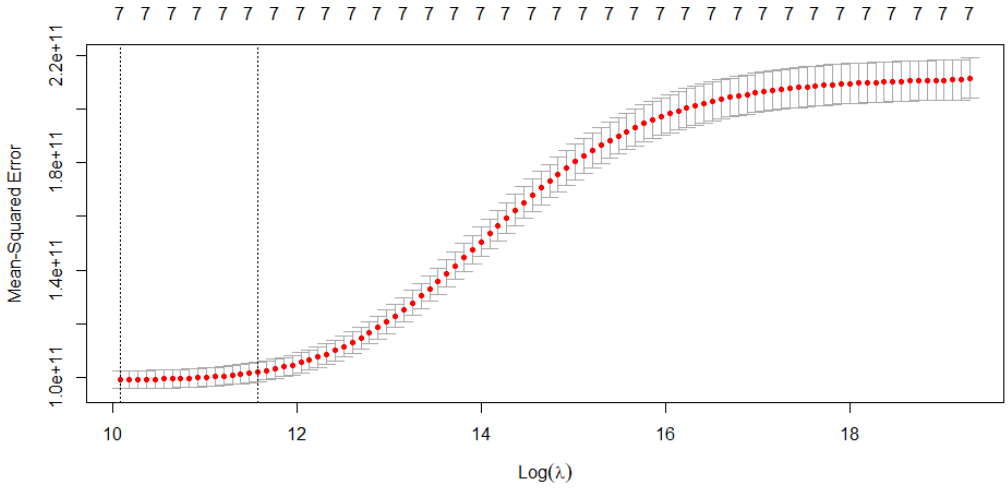

Figure 1.2: Plot of Cv.Ridge

Observations:

- There **isn't significant difference** in the coefficients during the lambda.min and the lambda.1se.
- The lambda.min value is seen at the value around 10 and the lambda.1se value is seen at around 11.
- The values of 7 in the top represents **the number of variables which remains constant** and never change with the lambda.min and the lambda.min.

Fitting a regression model:

A Ridge min model is obtained by **putting the value of alpha as zero** and the minimum value of lambda is considered. Taking the function of glmnet() the results tell the lambda value is 69850 for the degree of freedom 5. On taking the coefficient of the model, it is observed that the value of the coefficients are almost shrinked to zero but **none of the coefficients are eliminated** from the model. Most of the coefficient values are approaching zero and some values are negative.

```
> #Fit Ridge regression model against the training set
> model.ridge.min <- glmnet(train_x, train_y, alpha = 0, lambda =cv.ridge$lambda.min) #lambda min
> model.ridge.min

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

  Df  %Dev Lambda
1  7 53.75  24210
> coef(model.ridge.min)
8 x 1 sparse Matrix of class "dgCMatrix"
                      s0
(Intercept)  -1.083530e+08
kms_driven   -4.136311e-01
fuel_type     1.633585e+05
transmission  5.220932e+05
ownership    -2.929830e+04
manufacture   5.391098e+04
engine        3.515702e+01
Seats         2.862438e+03
>
> #Regularization
> model.ridge.1se <- glmnet(train_x, train_y, alpha = 0, lambda =cv.ridge$lambda.1se) #lambda 1se
> model.ridge.1se

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

  Df  %Dev Lambda
1  7 52.25 107300
> coef(model.ridge.1se)
8 x 1 sparse Matrix of class "dgCMatrix"
                      s0
(Intercept)  -9.045286e+07
kms_driven   -7.088064e-01
fuel_type     1.428116e+05
transmission  4.513766e+05
ownership    -3.496047e+04
manufacture   4.506095e+04
engine        3.821423e+01
Seats         3.049735e+03
```

A significant difference is observed in terms of the s0 values of the coefficients in the lambda.1se function as well but as stated above none of the coefficients are shrinked to zero.

Calculation of Root mean square Error (RMSE):

RMSE (Root Mean Squared Error) is a measure of **the error or deviation between the predicted and actual values** in a regression problem. In Ridge Regression, the RMSE is used to evaluate the **performance of the model**. The smaller the RMSE, the better the model fits the data. The RMSE is calculated by taking the **square root of the mean of the squared differences between the predicted and actual values**. The choice of lambda affects the magnitude of the coefficients, and a **smaller lambda value means larger coefficients** and a **more complex model, while a larger lambda results in smaller coefficients** and a simpler model.

```
> #Train set prediction of RIDGE model by calculating RMSE
> #lamda min - RIDGE
> pred.ridge.min <- predict(model.ridge.min, newx = train_x)
> train.ridge.rmse.min <- rmse(train_y, pred.ridge.min)
> train.ridge.rmse.min
[1] 314979.2
>
> #lambda 1se - RIDGE
> pred.ridge.1se <- predict(model.ridge.1se, newx = train_x)
> train.ridge.rmse.1se <- rmse(train_y, pred.ridge.1se)
> train.ridge.rmse.1se
[1] 320030.5
```

It is seen that the RMSE value of the train dataset with the lambda.min is 314979.2 and with the lambda.1se value is 320030. Now these values need to be compared with the best model to come to a conclusion.

Comparing with test dataset:

To determine if a model is **overfitting, you need to compare the RMSE** (or any other performance metric) on the training set and the test set. If the **RMSE on the test set is significantly higher than the RMSE on the training set, it may indicate overfitting.**

```
> #Test set prediction of ridge model using RMSE
> #lambda min - RMSE RIDGE
> pred.ridge.min.test <- predict(model.ridge.min, newx = test_x)
> test.ridge.rmse.min <- rmse(test_y, pred.ridge.min.test)
> test.ridge.rmse.min
[1] 316502.1
>
> #lambda 1se  - RMSE RIDGE
> pred.ridge.1se.test <- predict(model.ridge.1se, newx = test_x)
> test.ridge.rmse.1se <- rmse(test_y, pred.ridge.1se.test)
> test.ridge.rmse.1se
[1] 318572.8
```

Observations:

- From the above values it is seen that the **RMSE of the test set is lower in both the case of minimum and also in the case of 1se**.
- Thus, we can conclude that there is **no overfitting of the data** in both the cases and this is concluded with the help of Root mean square error as the values in both the train and the test dataset values are not much different.
- The RMSE of the lambda of minimum value is less than that of the lambda value of the test value of the 1se value and thus the ridge model for minimum value is more significant.

LASSO Regression:

Lasso Regression is a type of **linear regression** algorithm that addresses the problem of multicollinearity and overfitting in the input data. It adds a penalty term **(L1 regularization)** to the cost function that penalizes the **absolute magnitude of the coefficients of the predictors**, which helps to reduce overfitting and improve the generalization ability of the model. The model tries to balance between fitting the data well (**minimizing the residual sum of squares**) and having sparse coefficients (penalizing large coefficients). Unlike Ridge Regression, Lasso Regression can effectively reduce the number of predictors to a subset of the most important variables, making it a useful feature selection method.

```
> # LASSO Regression
> #Find best values of lambda
> cv.lasso <- cv.glmnet(train_x, train_y, nfolds=10)
> cv.lasso

Call:  cv.glmnet(x = train_x, y = train_y, nfolds = 10)

Measure: Mean-Squared Error

    Lambda Index   Measure        SE Nonzero
min   3056    48 9.975e+10 4.428e+09       6
1se  37670    21 1.041e+11 4.686e+09       3
>
> #lambda min
> cv.lasso$lambda.min
[1] 3055.54
> #lambda.1se
> cv.lasso$lambda.1se
[1] 37670.13
```

Observations:

- The value of lambda.1se is much higher than that of the lambda.min value which is because a larger value of the regularization parameter reduces the coefficients more, effectively shrinking them towards zero.
- On checking the values in both the ridge function and that of the lasso we can see that the values of lambda is much less in the case of lasso. In **Lasso Regression**, the **penalty term (lambda) is applied as an absolute value of the coefficients,** whereas in Ridge Regression, the penalty term is applied as the square of the coefficients.
- As a result, the magnitude of the penalty term in **Lasso Regression is smaller compared to Ridge Regression,** which makes Lasso Regression more likely to shrink coefficients to zero and result in sparse models with only a few features.

Plot to represent the lasso regression:

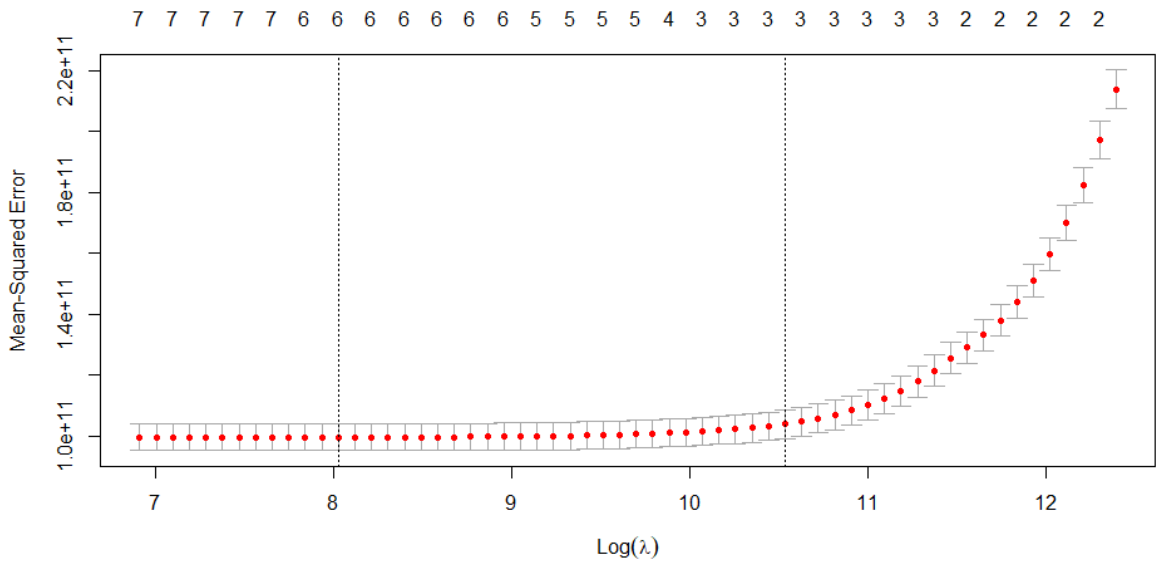

Figure 1.3: Plot of Cv.Lasso

Observations:

- The **y-axis is the mean squared error** and the **x-axis is the log of lambda** and across the graph on the top there is the non-zero coefficients.
- The **red dots** which is marked in the graph is the **error estimates** and the line up and below the red dots represents the confidence interval of the error matrix. The red dots are computed through the **cross validation of lasso method.**
- The two vertical lines represents the value of the lambda.min and the lambda.1se which is represented from left to right.

- It is represented from the plot that the value of lambda.min is somewhere on 8 and that of lambda.1se is somewhere close to 10.5.
- The lambda.min retains only the 6 predictor values of the model and the lambda.1se only **retains 3 coefficient values** and the others are discarded.

Lasso Regression model:

```
> #fit lasso regression model against the training set
> model.lasso.min <- glmnet(train_x, train_y, alpha=1, lambda =cv.lasso$lambda.min)#lambda min
> model.lasso.min

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

  Df  %Dev Lambda
1  6 53.84   3056
> coef(model.lasso.min)
8 x 1 sparse Matrix of class "dgCMatrix"
                      s0
(Intercept)  -1.153987e+08
kms_driven   -1.629083e-01
fuel_type     1.629417e+05
transmission  5.409346e+05
ownership    -2.329245e+04
manufacture   5.740597e+04
engine        2.876046e+01
Seats         .
>
> model.lasso.1se <- glmnet(train_x, train_y, alpha = 1, lambda =cv.lasso$lambda.1se)#lambda 1se
> model.lasso.1se

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

  Df  %Dev Lambda
1  3 51.68  37670
> coef(model.lasso.1se)
8 x 1 sparse Matrix of class "dgCMatrix"
                      s0
(Intercept)  -102580588.33
kms_driven            .
fuel_type         99632.42
transmission     465450.39
ownership             .
manufacture       51099.93
engine                .
Seats                 .
```

Observations:

- From the image above it is seen that the values of **kms_driven, ownership, engine and seats is not significant** and is eliminated in the minimum value curve of lambda.
- When compared to the ridge regression values and the lasso regression, for the lasso regression, the values is **mostly significant as out of the 7 coefficients**, 4 of the coefficient values are reduced to zero.
- Further analysis is needed to find out if the variables are not significant or it can be removed to simplify the model and avoid overfitting.

```
> #Train set prediction of LASSO model by calculating RMSE
> #lambda min
> pred.lasso.min <- predict(model.lasso.min, newx = train_x)
> train.lasso.rmse <- rmse(train_y, pred.lasso.min)
> train.lasso.rmse
[1] 314655.2
>
> #lambda 1se
> pred.lasso.1se <- predict(model.lasso.1se, newx = train_x)
> train.lasso.rmse.1se <- rmse(train_y, pred.lasso.1se)
> train.lasso.rmse.1se
[1] 321930.7
```

- Comparing the RMSE values of the train and test dataset of the lasso regression, it is seen that the **test set values are less compared to the train set values**.

```
> #Test set prediction of lasso model using RMSE
> #lambda min
> pred.lasso.min.test <- predict(model.lasso.min, newx = test_x)
> test.lasso.rmse.min <- rmse(test_y, pred.lasso.min.test)
> test.lasso.rmse.min
[1] 317286.3
>
> #lambda 1se
> pre.lasso.1se.test <- predict(model.lasso.1se, newx = test_x)
> test.lasso.rmse.1se <- rmse(test_y, pre.lasso.1se.test)
> test.lasso.rmse.1se
[1] 320619.6
```

- Hence there is **no overfitting of the values** in the model in both the cases and this is concluded with the help of Root mean square error as the values in both the train and the test dataset values are not much different.

Logistic Regression:

- Logistic regression is applied **to predict the categorical dependent variable**. In other words, it's used when the prediction is categorical, for example, yes or no, true or false, 0 or 1
- The logistic regression model is a statistic model which is used to find the relationship between the dependant and one or more independent variables. In this section, a logistic model is fitted by using glm() function in R.
- Two models are created using the function which gives the ability to analysis which model is more appropriate.
- The first model included all independent variables from the original dataset and the second model included only two independent variables that are significant which is decided based on the p-values.

```
> #Logistic regression model
> model1 <- glm(transmission~., data=train, family = binomial(link = "logit"))
> summary(model1)

Call:
glm(formula = transmission ~ ., family = binomial(link = "logit"),
    data = train)

Deviance Residuals:
    Min       1Q   Median       3Q      Max
-2.3808  -0.5220  -0.4185  -0.3031   2.7435

Coefficients:
                    Estimate Std. Error z value Pr(>|z|)
(Intercept)         3.106e+02  4.593e+01   6.762 1.36e-11 ***
car_prices_in_rupee 3.274e-06  1.568e-07  20.878  < 2e-16 ***
kms_driven         -6.295e-06  2.081e-06  -3.025  0.00249 **
ownership           2.360e-01  9.235e-02   2.556  0.01060 *
manufacture        -1.555e-01  2.277e-02  -6.830 8.47e-12 ***
engine              1.188e-04  1.264e-04   0.939  0.34753
Seats              -2.075e-01  8.475e-02  -2.448  0.01435 *
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 2768.9  on 2774  degrees of freedom
Residual deviance: 2032.3  on 2768  degrees of freedom
AIC: 2046.3

Number of Fisher Scoring iterations: 5
```

Observations:

- From the above image it is observed that the value of intercept which is the transmission value is significant since the value is less than 95% confidence interval of 0.05
- The variables like car_price_in_rupees, manufacture, engine, seats and ownership are significant since their values are less compared to the 0.05 value.
- The AIC value of the model is 2046.3 and this can be compared to the another model to estimate a better model.

Model 2:

```
> model2 <- glm(transmission~car_prices_in_rupee+Seats+manufacture+ownership+kms_driven, data=train, family = binomial
(link = "logit"))
> #view(model2)
> summary(model2)

Call:
glm(formula = transmission ~ car_prices_in_rupee + Seats + manufacture +
    ownership + kms_driven, family = binomial(link = "logit"),
    data = train)

Deviance Residuals:
    Min       1Q   Median       3Q      Max
-2.3703  -0.5223  -0.4191  -0.3042   2.7294

Coefficients:
                   Estimate Std. Error z value Pr(>|z|)
(Intercept)        3.114e+02  4.593e+01   6.780 1.20e-11 ***
car_prices_in_rupee 3.292e-06  1.559e-07  21.108  < 2e-16 ***
Seats              -1.935e-01  8.331e-02  -2.322  0.02023 *
manufacture        -1.559e-01  2.277e-02  -6.846 7.61e-12 ***
ownership           2.384e-01  9.225e-02   2.585  0.00974 **
kms_driven         -6.228e-06  2.081e-06  -2.993  0.00276 **
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 2768.9  on 2774  degrees of freedom
Residual deviance: 2033.2  on 2769  degrees of freedom
AIC: 2045.2

Number of Fisher Scoring iterations: 5
```

Observation:

- From the above image it is observed that the value of intercept which is the transmission value is significant since the value is less than 95% confidence interval of 0.05
- This model is a preffered more since all the variables are significant having p values less than 0.05 which is the confidence interval of 95%.
- The AIC value of the model is 2045.2 and when compared to the previous model, this model 2 is considered a better model.

Confusion Matrix:

A confusion matrix is a table used to evaluate the performance of a machine learning model for classification tasks. It compares the actual values of the target variable with the predicted values of the model. The matrix is usually a square matrix with rows representing the actual class and columns representing the predicted class. The confusion matrix helps to calculate various performance metrics like accuracy, precision, recall, and F1 score. These metrics are useful for assessing the effectiveness of a model and identifying areas that need improvement.

```
Confusion Matrix and Statistics

          Reference
Prediction    0    1
         0 2144  320
         1   79  232

               Accuracy : 0.8562
                 95% CI : (0.8426, 0.8691)
    No Information Rate : 0.8011
    P-Value [Acc > NIR] : 2.403e-14

                  Kappa : 0.4603

 Mcnemar's Test P-Value : < 2.2e-16

            Sensitivity : 0.9645
            Specificity : 0.4203
         Pos Pred Value : 0.8701
         Neg Pred Value : 0.7460
             Prevalence : 0.8011
         Detection Rate : 0.7726
   Detection Prevalence : 0.8879
      Balanced Accuracy : 0.6924

       'Positive' Class : 0
```

The confusion matrix summarizes the performance of a machine learning model for predicting the price of old cars based on variables such as transmission, model, fuel type, seats, kms driven, and ownership. The model achieved an accuracy of 85%, indicating that it correctly predicted 85% of the instances. The sensitivity of 96% suggests that the model is good at identifying positive instances, while the specificity of 42% indicates that it is not as effective in identifying negative instances. The prevalence of 80% suggests that the majority of the instances belong to the positive class.

ROC curve:

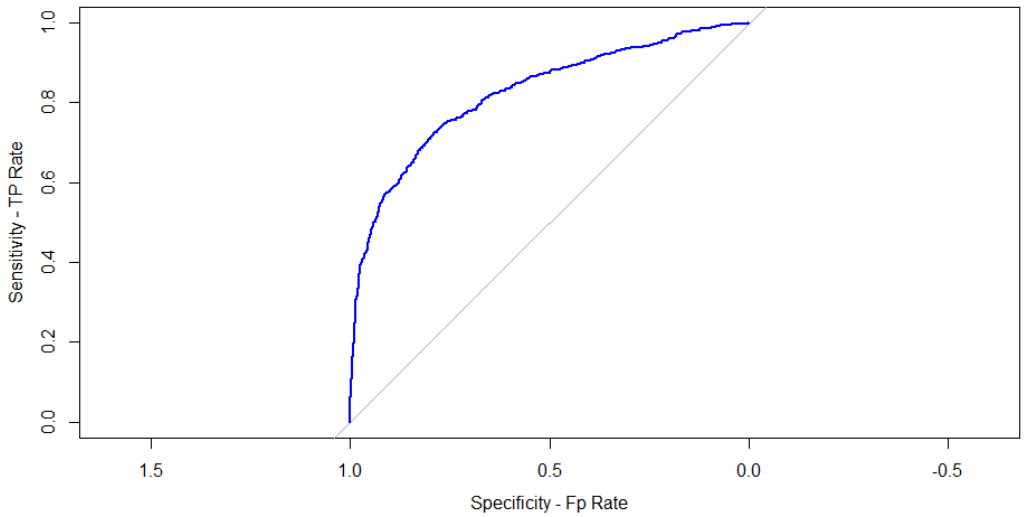

Figure 1.4: Plot of ROC curve

- The Receiver Operating Characteristic (ROC) curve is a graphical representation of the performance of a classification model, showing the trade-off between the true positive rate (TPR) and the false positive rate (FPR) as the decision threshold is varied.
- To generate the ROC curve for the dataset of old car price prediction, we need to have a set of predicted probabilities for each instance.
- The ROC curve is plotted by varying the decision threshold from 0 to 1, calculating the TPR and FPR for each threshold, and plotting the resulting points on a graph with TPR on the y-axis and FPR on the x-axis.
- The Area Under the Curve (AUC) is a metric used to evaluate the performance of a classification model based on the Receiver Operating Characteristic (ROC) curve.
- The AUC of the above model is 82% signifying that it is a good model.

Conclusion:

In conclusion, the study analyzed the car dataset and built two models using linear regression to estimate the car sale price. The results showed that the car sale price was positively correlated with the variables such as the year of manufacture, engine cc, and kms driven. The dataset was found to have no multicollinearity, and the Lasso and Ridge regression models were constructed to evaluate the overfitting of the data. The Ridge model retained all the variables, while the Lasso model reduced the variables to one. The study also performed Logistic regression and observed a relationship between the transmission and car price in Rupees. These findings provide valuable insights into the factors that influence the car sale price, and can help buyers and sellers make informed decisions. Thus with the help of logistic regression an overall significance of 82% is achieved which represents a good model.

Reference:

- Goyal, C. (2021, May 15). The Game of Increasing R-squared in a Regression Model. Analytics Vidhya. https://www.analyticsvidhya.com/blog/2021/05/the-game-of-increasing-r-squared-in-a-regression-model/
- Kumar, A. (2017, September 3). “confusionMatrix” function in R – The data contain levels not found in the data. DATA ANALYTICS BLOG SERIES. https://arulvelkumar.wordpress.com/2017/07/06/confusionmatrix-function-in-r-the-data-contain-levels-not-found-in-the-data/
- N. (2022, August 26). Reorder Columns of DataFrame in R. Spark by {Examples}. https://sparkbyexamples.com/r-programming/reorder-columns-in-r/
- Loft, S. R. |. (2020, June 16). Understanding Lasso and Ridge Regression. R-bloggers. https://www.r-bloggers.com/2020/06/understanding-lasso-and-ridge-regression/
- https://www.kaggle.com/code/minghsieh0302/old-car-price-predcitions-ming/data

Appendix:

```
print("Final Project")

packages <- install.packages(c("corrplot","RColorBrewer","car",
                               "dplyr","ggplot2","MASS","tidyverse",
                               "psych","mice","Hmisc","car","leaps"))

install.packages("Metrics")

install.packages("outliers")

library(outliers) #for outliers

library(ggplot2)   #for ggplot

library(dplyr)    # for mutate, select, filter, summarize, arrange

library(mice)

library(corrplot)

library(ISLR)

library(psych)

library(pROC)

library(gridExtra)

library(MASS)

library(tidyverse)

library(glmnet)

library(caret)

library(Metrics)

library(RColorBrewer)  # for color
```

```
library(stats) # for correlation
library(car) # for vif
library(leaps) # for regsubsets
lapply(packages,library, character.only = TRUE)

#1. Loading the dataset
dataset <- read.csv("C:\\Users\\krish\\Desktop\\NEU\\Course\\ALY6015\\module6\\car_price.csv",row.names= NULL)

#2. Exploratory statistics
View(dataset)
names(dataset)
dim(dataset)
summary(dataset)
str(dataset)
#describe(dataset)
library(stringr)  #for string operations
sapply(dataset,class)

#identify missing values
sapply(dataset, function(x) sum(is.na(x)))

attach(dataset)

#cleaning the dataset
#removing X column
dataset <- dataset[,-1]
names(dataset)

dataset$car_prices_in_rupee <- str_replace_all(car_prices_in_rupee,",","")
dataset$kms_driven<-str_replace_all(kms_driven,",","")
x <- dataset$kms_driven
```

```
x <- str_replace_all(x," kms","")
dataset$kms_driven<-x
dataset$kms_driven

Value_unity <- ifelse(str_detect(dataset$car_prices_in_rupee, ' Lakh'), 1e5, ifelse(str_detect(dataset$car_prices_in_rupee, ' Crore'), 1e7, 1))
Value_new <- Value_unity * as.numeric(str_remove(dataset$car_prices_in_rupee, ' Lakh| Crore'))
Value_new
dataset$car_prices_in_rupee<-Value_new

dataset$Seats <- str_remove_all(Seats," Seats")
dataset$engine <- str_remove_all(engine," cc")

dataset$ownership<-str_remove_all(ownership,"[^(-?(\\d*\\.)?\\d+)]")

#converting dataset to numeric values

dataset$engine <- as.numeric(dataset$engine)
dataset$ownership <- as.numeric(dataset$ownership)
dataset$manufacture <- as.numeric(dataset$manufacture)
dataset$Seats <- as.numeric(dataset$Seats)
dataset$kms_driven <-as.numeric(dataset$kms_driven)

#remove duplicated values
dataset[!duplicated(dataset$car_name), ]
dataset<- subset(dataset, engine>1000)

dataset <- within(dataset, {
  fuel_type[fuel_type == "Petrol"] <- 1
  fuel_type[fuel_type == "Diesel"] <- 2
  fuel_type[fuel_type == "Cng"] <- 3
  fuel_type[fuel_type== "Electric"]<-4
  fuel_type[fuel_type=="Lpg"]<-5
```

```
})
dataset$transmission <- ifelse(dataset$transmission == "Automatic", 1, 0)

dataset$transmission <-as.numeric(dataset$transmission)
dataset$fuel_type <-as.numeric(dataset$fuel_type)

#install.packages("tibble")
library(tibble)

rownames(dataset) <- make.names(dataset[,1], unique = TRUE)
dataset <- dataset[,-1]

View(dataset)

#pair <- subset(final_numeric_clean,select=c(""))
#corrplot(cor(pair),method="circle", col=brewer.pal(n=20,name="RdYlBu"))

class(car_prices_in_rupee)
#finding outliers
# Q <- quantile(dataset$car_prices_in_rupee, probs=c(.25, .75), na.rm = FALSE)
# Q[1]
# iqr <- IQR(dataset$car_prices_in_rupee)
# iqr
# up <-  Q[2]+1.5*iqr # Upper Range
# up
# low<- Q[1]-1.5*iqr # Lower Range
# low
# eliminated<- subset(dataset, dataset$car_prices_in_rupee > 325000 )
# View(eliminated)
# boxplot(car_prices_in_rupee)

boxplot(dataset$car_prices_in_rupee, plot=FALSE)$out
```

```
outliers_car_price <- boxplot(dataset$car_prices_in_rupee, plot=FALSE)$out
x<-dataset
x<- x[-which(x$car_prices_in_rupee %in% outliers_car_price),]

outliers_kms_driven<-boxplot(dataset$kms_driven,plot=FALSE)$out
x<-x[-which(x$kms_driven %in% outliers_kms_driven),]

outliers_engine<-boxplot(dataset$engine,plot=FALSE)$out
x<-x[-which(x$engine %in% outliers_engine),]

outliers_manufacture<-boxplot(dataset$manufacture,plot=FALSE)$out
x<-x[-which(x$manufacture %in% outliers_manufacture),]
boxplot(dataset$manufacture)

#filtering the required numerical values for correlation plot
numeric <- x %>% dplyr::select(where(is.numeric))
View(numeric)
names(numeric)

#plotting the correlation matrix
cor(numeric,use="everything")
corrplot(cor(numeric),method="circle",type="upper")

#checking for linear regression
fit <-
lm(car_prices_in_rupee~kms_driven+fuel_type+transmission+ownership+manufacture+engine+Seats
,data=dataset)
summary(fit)
#avoiding this fit as its not significant, removing the seats, ownership as both are above 0.05 scope

fit2<-
regsubsets(car_prices_in_rupee~fuel_type+transmission+manufacture+engine+kms_driven,data=data
set)
summary.out<- summary(fit2)
```

```
summary.out


fit_lm <- lm(car_prices_in_rupee~fuel_type+transmission+manufacture+engine+kms_driven,data=numeric)
summary(fit_lm)


# model2 is more significant as due to less aic value, not much but can be considered


#Plot the regression model
par(mfrow=c(2,2))
plot(fit_lm)
dev.off()


#checking for multicollinearity
vif(fit_lm)
outlierTest(model = fit_lm)


plot(fit_lm,scale="adjr2",main="Plot of RegressSubsets of variables")


stepAIC(fit,direction="forward")
stepAIC(fit_lm,direction="forward")


ggplot(dataset, aes( x=manufacture  ,y=car_prices_in_rupee,color=transmission)) +
  geom_point(fill = "#0c4c8a")


qplot(data=dataset,x=kms_driven,y=car_prices_in_rupee,color=manufacture,main = "Car prices vs Kms driven based on manufacture year ")


df <- numeric


#1. Split the data
set.seed(123)
trainIndex <- sample(x= nrow(df), size=nrow(df)*0.6)
```

```
train <- df[trainIndex,]
test <- df[-trainIndex,]
train_x <- model.matrix(car_prices_in_rupee~.,train)[,-1]
test_x <- model.matrix(car_prices_in_rupee~., test)[,-1]
train_y <- train$car_prices_in_rupee
test_y <- test$car_prices_in_rupee
head(train_x,5)
head(test_x,5)
# using car price in rupees as response variable in which removes the variables in Grd rate and
#adds the categorical variable in training dataset

# RIDGE Regression
#Find the best lambda using cross-validation
set.seed(123)
cv.ridge <- cv.glmnet(train_x, train_y,alpha=0, nfolds=10)
cv.ridge

#Plot the results
dev.off()
plot(cv.ridge)

#lambda min (minimizes out of sample box)
lambda.min <- cv.ridge$lambda.min
lambda.min
log(lambda.min)

#lambda.1se (largest value of lambda within 1 standard error of lambda min)
lambda.1se <- cv.ridge$lambda.1se
lambda.1se
log(lambda.1se)

#Fit Ridge regression model against the training set
```

```
model.ridge.min <- glmnet(train_x, train_y, alpha = 0, lambda =cv.ridge$lambda.min) #lambda min
model.ridge.min
coef(model.ridge.min)

#Regularization
model.ridge.1se <- glmnet(train_x, train_y, alpha = 0, lambda =cv.ridge$lambda.1se) #lambda 1se
model.ridge.1se
coef(model.ridge.1se)

#Train set prediction of RIDGE model by calculating RMSE
#lamda min - RIDGE
pred.ridge.min <- predict(model.ridge.min, newx = train_x)
train.ridge.rmse.min <- rmse(train_y, pred.ridge.min)
train.ridge.rmse.min

#lambda 1se - RIDGE
pred.ridge.1se <- predict(model.ridge.1se, newx = train_x)
train.ridge.rmse.1se <- rmse(train_y, pred.ridge.1se)
train.ridge.rmse.1se

#Test set prediction of ridge model using RMSE
#lambda min - RMSE RIDGE
pred.ridge.min.test <- predict(model.ridge.min, newx = test_x)
test.ridge.rmse.min <- rmse(test_y, pred.ridge.min.test)
test.ridge.rmse.min

#lambda 1se  - RMSE RIDGE
pred.ridge.1se.test <- predict(model.ridge.1se, newx = test_x)
test.ridge.rmse.1se <- rmse(test_y, pred.ridge.1se.test)
test.ridge.rmse.1se

#display coefficient of ols model with no regularization
```

```
ols <- lm(car_prices_in_rupee ~., data = train)
coef(ols)

#view RMSE of full model
preds.ols <- predict(ols, new = test)
rmse(test$car_prices_in_rupee, preds.ols)

# LASSO Regression
#Find best values of lambda
cv.lasso <- cv.glmnet(train_x, train_y, nfolds=10)
cv.lasso
View(numeric)
#lambda min
cv.lasso$lambda.min
#lambda.1se
cv.lasso$lambda.1se

#Plot the results
plot(cv.lasso)

#fit lasso regression model against the training set
model.lasso.min <- glmnet(train_x, train_y, alpha=1, lambda =cv.lasso$lambda.min)#lambda min
model.lasso.min
coef(model.lasso.min)

model.lasso.1se <- glmnet(train_x, train_y, alpha = 1, lambda =cv.lasso$lambda.1se)#lambda 1se
model.lasso.1se
coef(model.lasso.1se)

#Train set prediction of LASSO model by calculating RMSE
#lambda min
pred.lasso.min <- predict(model.lasso.min, newx = train_x)
```

```
train.lasso.rmse <- rmse(train_y, pred.lasso.min)
train.lasso.rmse

#lambda 1se
pred.lasso.1se <- predict(model.lasso.1se, newx = train_x)
train.lasso.rmse.1se <- rmse(train_y, pred.lasso.1se)
train.lasso.rmse.1se

#Test set prediction of lasso model using RMSE
#lambda min
pred.lasso.min.test <- predict(model.lasso.min, newx = test_x)
test.lasso.rmse.min <- rmse(test_y, pred.lasso.min.test)
test.lasso.rmse.min

#lambda 1se
pre.lasso.1se.test <- predict(model.lasso.1se, newx = test_x)
test.lasso.rmse.1se <- rmse(test_y, pre.lasso.1se.test)
test.lasso.rmse.1se

#Stepwise selection and regularization
#forward selection
step(lm(car_prices_in_rupee ~ 1, data = train), direction = 'forward', scope = ~kms_driven+ownership+manufacture+Seats+engine)
#backward selection
step(lm(car_prices_in_rupee ~ ., data = train), direction = 'backward')
#comparring regression models
fit1 <- lm(formula = car_prices_in_rupee ~ kms_driven+ownership+manufacture+Seats+engine, data = train)
fit2 <- lm(formula = car_prices_in_rupee ~ kms_driven+ownership+manufacture+Seats+engine, data = train)
fit3 <- lm(formula = car_prices_in_rupee ~ kms_driven+ownership+manufacture+Seats+engine, data = train)
AIC(fit1, fit2, fit3)
BIC(fit1,fit2, fit3)
```

```
#ROC curve

#plotting

qplot(x=fuel_type, y=car_prices_in_rupee, color=transmission, geom = "point")
+scale_shape(solid=FALSE)+theme(text = element_text(size=10))

qplot(x=engine, y=manufacture, color=transmission, geom = "point")
+scale_shape(solid=FALSE)+theme(text = element_text(size=10))


#Split data into train and test set

logdata <- numeric

logdata<- logdata[,-3]

names(logdata)

View(logdata)

attach(logdata)

logdata$transmission<- as.factor(logdata$transmission)


set.seed(123)

trainIndex <- createDataPartition(logdata$transmission, p=0.7, list = FALSE)

train <- logdata[trainIndex,]

test <- logdata[-trainIndex,]

head(train)

#Logistic regression model

model1 <- glm(transmission~., data=train, family = binomial(link = "logit"))

summary(model1)

#2185.1

model2 <- glm(transmission~car_prices_in_rupee+Seats+manufacture+ownership+kms_driven,
data=train, family = binomial(link = "logit"))

#View(model2)

summary(model2)


probabilities.train <- predict(model2, newdata = train, type = "response")

predicted.classes.min <- as.factor(ifelse(probabilities.train >= 0.5,

                                  "1", "0"))
```

```
#Confusion Matrix / Model Accuracy
confusionMatrix(predicted.classes.min,train$transmission)

ROC1 <- roc(train$transmission, probabilities.train)
plot(ROC1, col="blue", ylab="Sensitivity - TP Rate", xlab= "Specificity - Fp Rate")

#AUC
auc <-auc(ROC1)
auc
```