Bright Kyeremeh
DS501
Introduction To Data Science
April 20, 2023

Motor Trend Car Road Tests

Predicting and Deploying The Miles Per Gallon of Cars Using Relevant Predicted Variables Via R Shiny

Introduction:

In this case study, I will be using the **mtcars** dataset to build a predictive model that will based on many variables such as car's Gross horsepower, Number of cylinders, Displacement, Rear axle ratio, Weight, 1/4 mile time, Engine, Transmission, Number of forward gears to predict the Miles/gallon of the respective car.

Data Collection:

The first step of this case study is to load the dataset into rstudio. The **mtcars** dataset is a built-in dataset in R that contains measurements on 11 different attributes for 32 different cars. Since this is an inbuilt dataset, I simply load into rstudio using data(mtcars). We can take a look at the first six rows of the dataset by using the **head()** function:

```
> data(mtcars)
> head(mtcars)
                   mpg cyl disp  hp drat   wt qsec vs am gear carb
Mazda RX4         21.0   6  160 110 3.90 2.62 16.5  0  1    4    4
Mazda RX4 Wag     21.0   6  160 110 3.90 2.88 17.0  0  1    4    4
Datsun 710        22.8   4  108  93 3.85 2.32 18.6  1  1    4    1
Hornet 4 Drive    21.4   6  258 110 3.08 3.21 19.4  1  0    3    1
Hornet Sportabout 18.7   8  360 175 3.15 3.44 17.0  0  0    3    2
Valiant           18.1   6  225 105 2.76 3.46 20.2  1  0    3    1
>
```

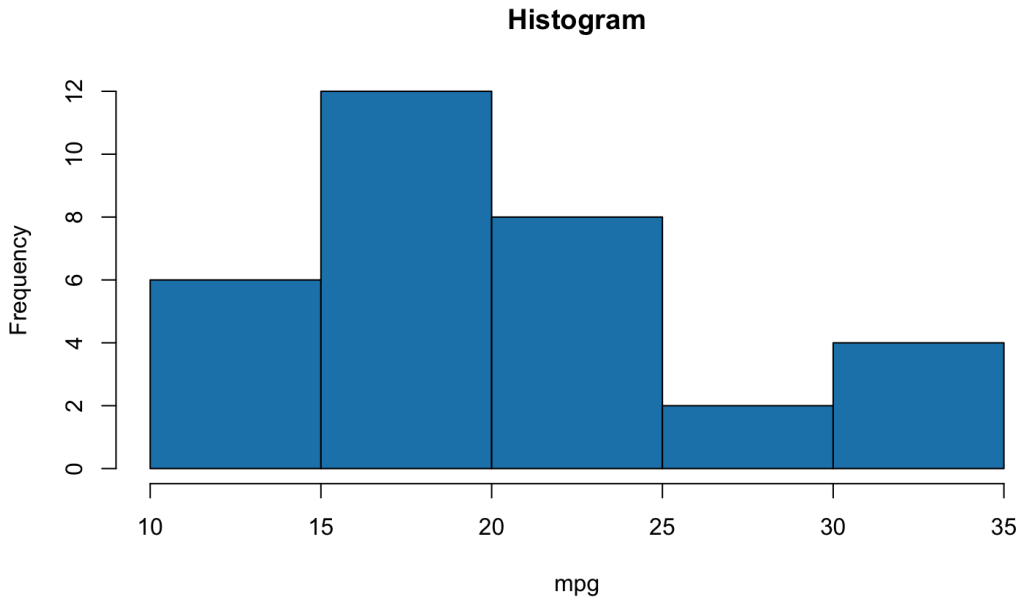
Data Preprocessing :

Summarize the **mtcars** Dataset

The second step was to preprocess the collected data to clean and prepare it for analysis. This involved techniques such as removing duplicates, handling missing values, and performing feature engineering. I also explore the data further using the summary() function to see the summary statistics of the dataset :

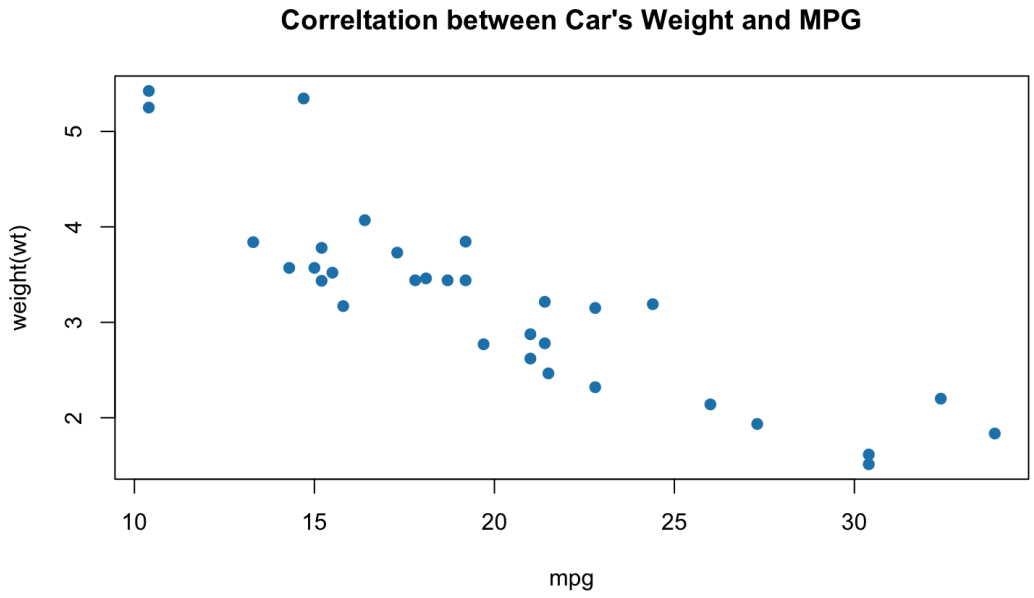
```
> summary(mtcars)
      mpg             cyl             disp             hp             drat
 Min.   :10.4   Min.   :4.00   Min.   : 71   Min.   : 52   Min.   :2.76
 1st Qu.:15.4   1st Qu.:4.00   1st Qu.:121   1st Qu.: 96   1st Qu.:3.08
 Median :19.2   Median :6.00   Median :196   Median :123   Median :3.69
 Mean   :20.1   Mean   :6.19   Mean   :231   Mean   :147   Mean   :3.60
 3rd Qu.:22.8   3rd Qu.:8.00   3rd Qu.:326   3rd Qu.:180   3rd Qu.:3.92
 Max.   :33.9   Max.   :8.00   Max.   :472   Max.   :335   Max.   :4.93
       wt             qsec             vs               am              gear
 Min.   :1.51   Min.   :14.5   Min.   :0.000   Min.   :0.000   Min.   :3.00
 1st Qu.:2.58   1st Qu.:16.9   1st Qu.:0.000   1st Qu.:0.000   1st Qu.:3.00
 Median :3.33   Median :17.7   Median :0.000   Median :0.000   Median :4.00
 Mean   :3.22   Mean   :17.9   Mean   :0.438   Mean   :0.406   Mean   :3.69
 3rd Qu.:3.61   3rd Qu.:18.9   3rd Qu.:1.000   3rd Qu.:1.000   3rd Qu.:4.00
 Max.   :5.42   Max.   :22.9   Max.   :1.000   Max.   :1.000   Max.   :5.00
      carb
 Min.   :1.00
 1st Qu.:2.00
 Median :2.00
 Mean   :2.81
 3rd Qu.:4.00
 Max.   :8.00
>
```

I again checked the distribution of the miles per gallon since it is our variable of interest:



We can see that most of the cars have around 15-20 mpg.

Next, I checked the correlation that exist between the target/predictor variable and the other variables. For example I will want to know what is the correlation between cars weight and its miles per gallon:



Model Building:

For this used case, I build Linear regression model based on user selection of X, Y variables and train/test splits parameter.

The maths behind Linear Regression:

In linear regression, we obtain an estimate of the unknown variable (denoted by y; the output of our model) by computing a weighted sum of our known variables (denoted by x_i; the inputs) to which we add a bias term.

$$y = b + \sum_{i=1}^{n} x_i \cdot w_i$$

Where **n** is the number of data points we have and **b** is bias. Adding a bias is the same thing as imagining we have an extra input variable that's always 1 and using only the weights. We will consider this case to make the math notation a little easier.

$$y = \sum_{i=0}^{n} x_i \cdot w_i$$

Where $\mathbf{x_0}$ is always 1, and $\mathbf{w_0}$ is our previous b. To make the notation a little easier, we will transition from the above sum notation to matrix notation. The weighted sum in the equation above is equivalent to the multiplication of a row-vector of all the input variables with a column-vector of all the weights. That is:

$$y = \begin{bmatrix} x_0 & x_1 & \dots & x_n \end{bmatrix} \begin{bmatrix} w_0 \\ w_1 \\ \dots \\ w_n \end{bmatrix}$$

The equation above is for just one data point. If we want to compute the outputs of more data points at once, we can concatenate the input rows into one matrix which we will denote by **X**. The weights vector will remain the same for all those different input rows and we will denote it by **w**. Now **y** will be used to denote a column-vector with all the outputs instead of just a single value. This new equation, the matrix form, is given below:

$$y = Xw \qquad (1)$$

Given an input matrix **X** and a weights vector **w**, we can easily get a value for **y** using the formula above. The input values are assumed to be known, or at least to be easy to obtain. But the problem is: How do we obtain the weights vector. To learn the weights, we need a dataset in which we know both x and y values, and based on those we will find the weights vector.If our data points are the minimum required to define our regression line (one more than the number of inputs), then we can simply solve equation (1) for **w**:

$$w = X^{-1}y$$

We call this thing a regression line, but actually, it is a line only for 1 input. For 2 inputs it will be a plane, for 3 inputs it will be some kind of "3D plane", and so on. If equation (1) doesn't have a solution, this means that **y** doesn't belong to the column space of **X**. So, instead of **y**, we will use the projection of **y** onto the column space of **X**. This is the closest vector to **y** that also belongs to the column space of **X**. If we multiply (on the left) both sides of eq. (1) by the transpose of **X**, we will get an equation in which this projection is considered.

$$\begin{aligned} y &= Xw \\ X^T \cdot \mid \quad y &= Xw \\ X^T y &= X^T Xw \\ (X^T X)^{-1} \cdot \mid \; X^T y &= X^T Xw \\ (X^T X)^{-1} X^T y &= w \\ \boxed{w &= (X^T X)^{-1} X^T y} \end{aligned}$$

```
y = b0 + b1x1 + b2x2 + ... + bnxn
```

where:

- y is the dependent variable
- b0 is the intercept
- b1, b2, ..., bn are the coefficients for the independent variables
- x1, x2, ..., xn are the independent variables

The goal of linear regression is to find the values of the coefficients `b0`, `b1`, `b2`, ..., `bn` that minimize the sum of squared errors. The sum of squared errors is a measure of how far the predicted values are from the actual values.

Building R Shiny App

User interface (UI):

This is where we define our layout : place holders which will be populated at the runtime from processed data/plot from the server.

Server:

This is where you write most of your logic, data wrangling, plotting, etc. Most heavy lifting is done here. I start by adding the two dropdown fields one for independent variables and the other for selecting target. create multiple tabs, each performing having specific functionality as detailed below:

Data – To view the raw data in the tabular form,

Data Summary – View the basic stats for our dataset.

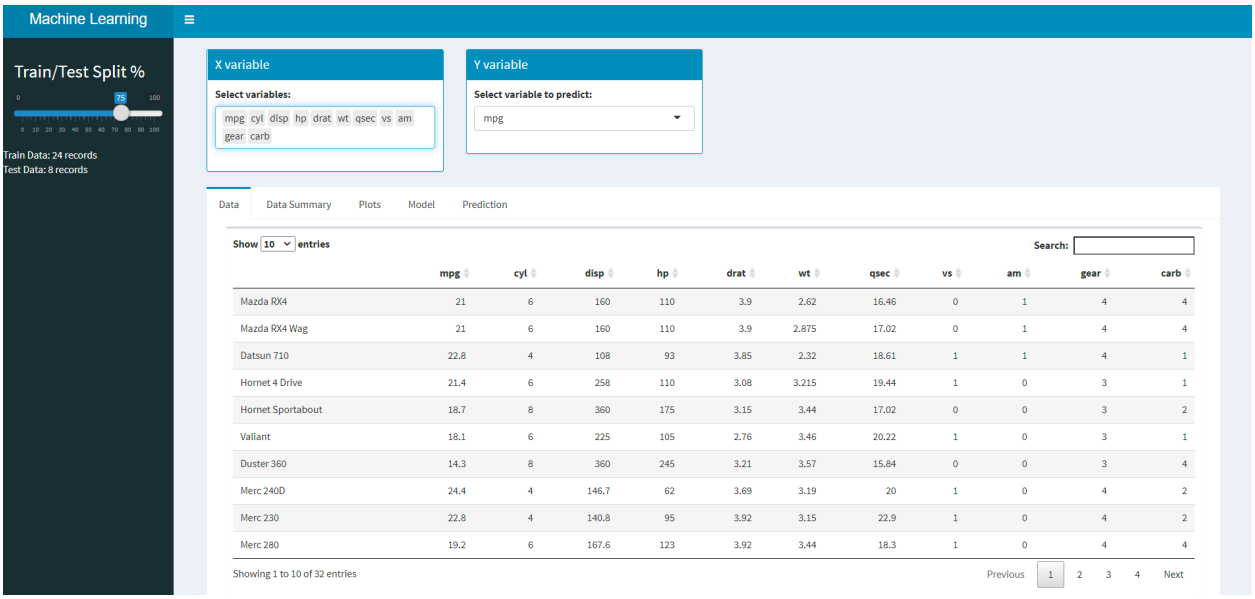
Plots – In this case, we will create only a correlation plot but more relevant plots can be added if required.

Model – Build Linear regression model based on user selection of X, Y variables and train/test splits

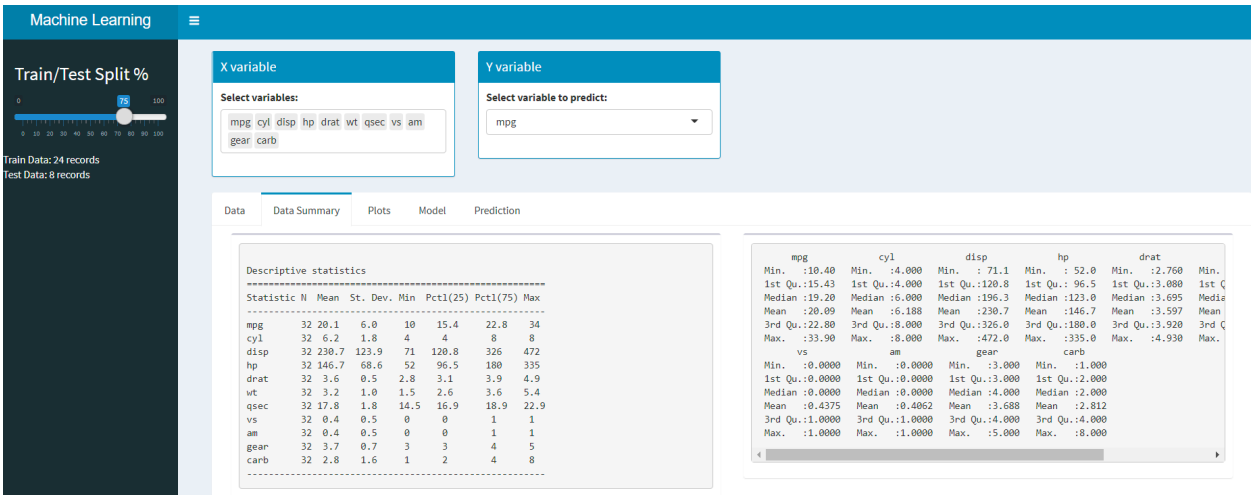
Prediction – Predict on the test set.

Overview of Dashboard:

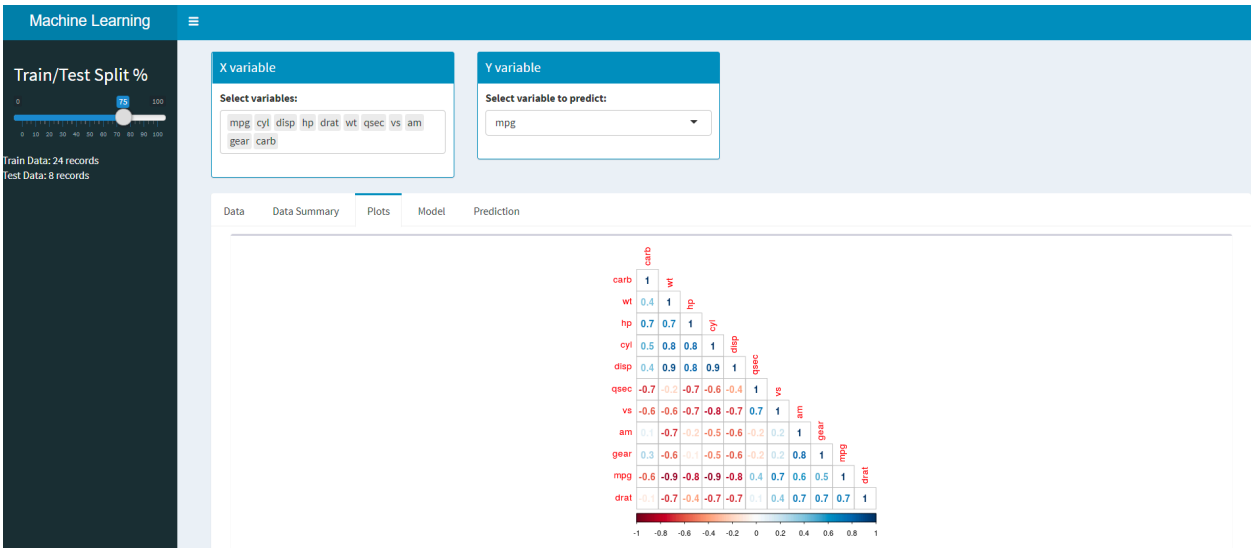
Data Tab:



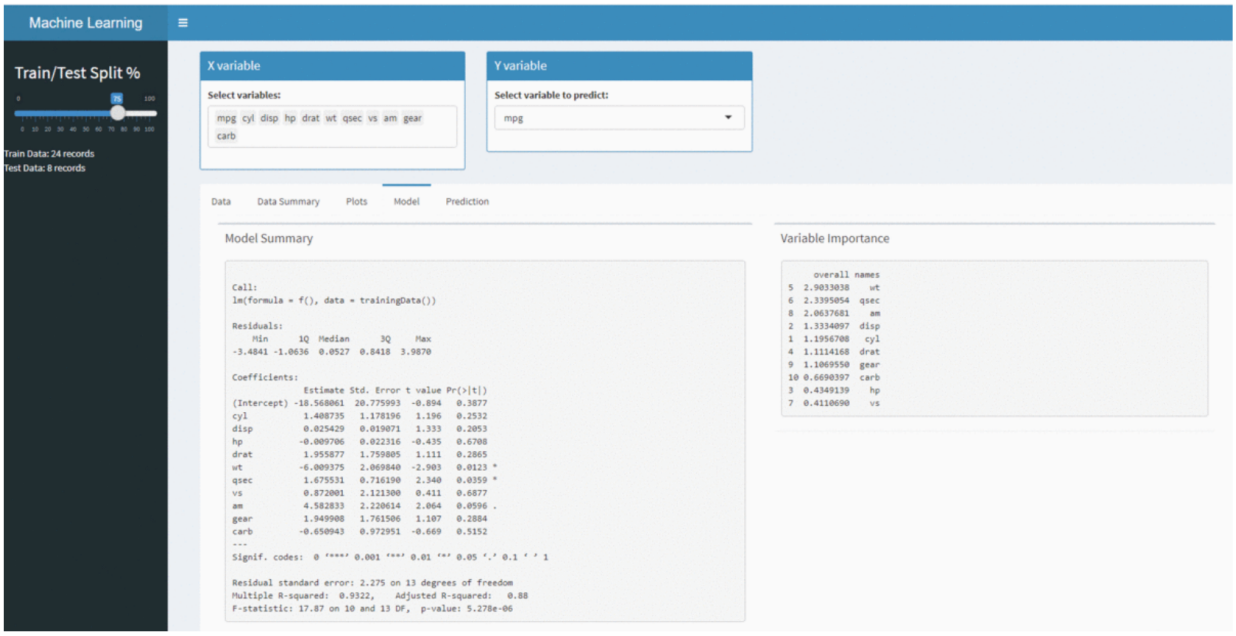
Data Summary Tab:



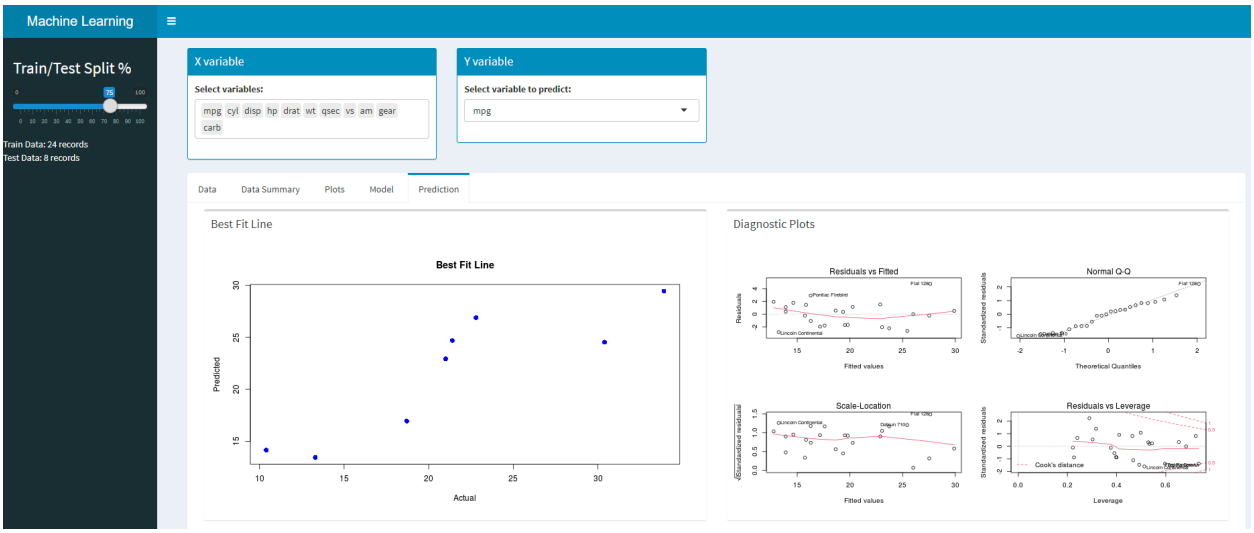
Plot Tab:



Model Tab:



Prediction Tab:



We can see that we have a low positive best fit line suggesting that the car's mpg increases as their other predictor variables such as Gross horsepower, Number of cylinders, Displacement, Rear axle ratio, etc. increases.

Deployment and Integration:

Finally, we deployed the model via https://www.shinyapps.io

Link to deployment : https://brightkyeremeh.shinyapps.io/Interactive-Modelling-with-Shiny-main/

Conclusion:

In conclusion, building an R shiny predictor application involves collecting data, preprocessing it, building a machine learning model, evaluating its performance, and deploying and integrating the model with other systems(in this case through shiny app.io). This will help car owners and yet to be car owners to be in the known of the other variables that affect their ownership of a car and thereby influencing their buying preferences and making better decisions.