COE 379L Project 1: Fuel Efficiency Linear Regression Model

Antonio Jimenez; aoj268

Exploring the Data

Before preparing the data for linear regression it is important to know information about the data that you are working with. From analyzing the shape of the database using .shape, I found the data set to be 398 rows and 9 columns. I followed by using the .head() function and .info() function to evaluate what the data looked like, the data type, and the number of non-null values the database contained. From the .info() output we see that horsepower columns is of type object. Upon further investigation we find that the horsepower column is missing data which is labeled by a "?". Lastly, I used .describe() to evaluate the database for outliers. Specific columns like weight, displacement and acceleration had large differences between the min and max values. Weight also differed in the fact that its standard deviation was 846.

Preparing the Data

Having found missing data in the horsepower, I found there to be 6 rows with missing horsepower data. To resolve these values, I began by converting the "?" to numpy NaN values. I then replaced the NaN values with the mean horsepower of cars with the same name as they differ slightly. I was able to perform this change for 2/6 of the rows. The remaining 4 rows did not have data with the same car name, so I was unable to get a mean to replace its NaN value. I then decided to delete the four rows as our dataset is large relative to the number of rows we eliminated. With no more NaN values in the column, I proceeded to change the type of the horsepower column from object to int. I proceeded to eliminate the car_name column as we would not be able to make it numeric or a bool which is required by our linear regression model. Furthermore, the car_name does not impact the miles per gallon (mpg). While it is possible for a cars_name to indicate a possible range for horsepower I found it to be unnecessary and could cause the model to not be as general.

Similarly to the horsepower column, I needed to convert the data type of origin to Boolean as there were only 3 options. Through one hot encoding we were able to transform the origin column into Booleans for when origin=2 and origin=3.

Analyze the Remaining Data

Having prepared the data, I proceeded to create univariate and multivariate plots to gain a deeper understanding of the data, as well as see if there were any outliers we might have missed.

In the univariate plots (Cells: 24,25,26) we are able to see the distribution of model years, weight, and displacement. I found most important the range of weight and the small number of vehicles that weight around 5000 lbs.

More importantly, through a correlation heatmap (Cell: 27) I was able to visually see the correlation between the different columns. In the top-left section of the heat map we see a blue box which indicates that there is a positive correlation between cylinders, displacement, horsepower, and weight. If we look slightly above the blue box, we can observe that these columns are negatively correlated with mpg. Additionally, we see that acceleration and model year are positively correlated with mpg. This was interesting as I would have imagined acceleration to be negatively correlated with mgp, and it also allows us to see that newer cars are more fuel efficient. Lastly, I found interesting that cars that originated from Japan (3) were more fuel efficient.

Overall, through these plots I was able to analyze the data to find patterns that our linear regression model will be able to use to estimate mpg. Additional plots were created to show the negative correlation between weight v mpg, horsepower v mpg, and cylinders v mpg (Cells: 28,29,30).

Training the Linear Regression Model

To train the linear regression model I began by separating my dependent and independent variables. For this model, we made the independent variables (x) all of the columns except for mpg. We then set the mpg column (y) to be our dependent variable. I then proceeded to split the data using the sklearn train_test_split() function using 30% of the dataset for testing. The other 70 percent is used for training the model. It is important to create this split as we do not want to test our model using data that it has already seen.

I then created a LinearRegression object from sklearn.linear_model which executes a least squares approximation algorithm to find a linear model that minimizes the error function. We then used the fit(x_train,y_train) function that belongs to the LinearRegression object to train our model using the training data.

Analyzing Linear Regression Model Performance

To analyze the performance of our model I began by feeding it the x_test data, which is our independent variables which were saved for testing. The model then returned predicted mpg values. I then calculated the error our of predictions by taking the absolute value of actual mpg minus predicted mpg. The mean of our error was around 2.49, which means that on average our model is off by 2.49 mpg. Using the scorel() function I found the R2 error on the test data to be 0.834 and 0.816 on the training data. Additionally, I found that the mean average error to be large than the mean squared error for the test data, which means that the mean average error is a product of small errors and no large outliers. All of these metrics help gain an understanding of our models behavior. However, performance on test data gives us a clearer picture how it will perform against new data which is more important for our task.

Lastly, I am confident in the linear regression model as the mean average error is relatively small and our error does not result from extremely large errors. If we simply wish to have a close estimate of the fuel efficiency, I believe the model does a relatively good job. Performance could be futher improved by acquiring more data. By analyzing the training data I found the split to have resulted in the training data having no vehicles with 3 and 5 cylinders. Acquiring more data could help resolve this problem.