Project 01

Thursday, Feb.22, 2024
COE 379L

Osvaldo Salinas
Joe Stubbs

What did you do to prepare the data?

The data was in a DATA filetype, so I read it in as a pandas data frame using the pd.read_csv() function, the name of the data file, and the comma delimiters as the function arguments. Then I checked to see what the columns in the data were, what non-null values each column has, and their data type using the info() function. I got rid of the car name and origin columns because they didn't really have any relevant information in them for linear regression. There weren't any null values at first, and I knew there weren't any because there were 398 entries and each column had 398 non-null values. However, I noticed that the horsepower column had object type data when it should be numeric. To see why that is, I used the unique() function on the horsepower column to see what all the unique values in the column are. I noticed that one of the unique values was a question mark, indicating that there are missing values there. I used pd.to_numeric() function to convert the horsepower column to numeric values, and when a cell in that column was not convertible to numeric values they were converted to NaN values. The arguments for the pd.to_numeric() function were the column autos['horsepower'] and errors='coerce' to convert the question marks to NaN. I couldn't leave the NaN cells in the column, so I replaced them with the mean of that column with the fillna() function so that the column was still usable for the linear regression model. The arguments I used in the function were the mean calculation autos['horsepower'].mean() and inplace=True to replace the null values. There was no need to perform one-hot encoding because most of the data is represented by number except car name. There are too many unique values in the car_name column for one-hot coding to be necessary.

After having a full data frame without null values, I made use of univariate and bivariate plots to display the data in the data frame and the describe() function to derive statistical information. I created histograms and box plots for each variable, a heatmap plot to find correlations between columns, and pair plots to visualize how each variable relates to each other. I created the plots using a combination of seaborn and matplotlib.

What insights did you get from your data preparation?

Going through the data preparation process unveiled the need for examining datasets thoroughly. At first glance there were no missing values in the dataset, but by looking more closely at the data types for each column, there were question marks in place of question marks. Additionally, through the describe function and univariate and multivariate plots for each variable, we can better see how they relate to each other.
The .describe() method summarizes column statistics through information like the range of values (min/max), standard deviation, and percentiles (25th and 75th) for data distribution insights. Each of these characteristics can help figure out where to look for outliers first. For example, looking at the minimum and maximum values that fall far from the 25th or 75th percentiles may be considered outliers. Weight and displacement had the highest standard deviations with 846.8 and 104.3 respectively. The high standard deviations indicate a wide dispersion of values from the mean, suggesting a diverse range of car weights and engine displacements within the dataset. The cylinders variable had a small range that went from 3 to 8, and all the percentiles are integers. This makes sense because cars can only have a set number of cylinders and there isn't great variability. The miles per gallon (mpg) column had a maximum value of 46.6 while the 75th percentile was 29.0, suggesting that there are outliers for that variable. The box plot demonstrates that there is indeed an outlier, and the histogram and box plot show how most of the values are between 15 and 30 mpg. I can verify this by looking at the derived statistics and seeing that the 25th 50th and 70th percentiles are 17.5, 23.0, and 29.0 respectively. All these values are between the 15 and 30mpg range of values in the box plot and histogram. According to the box plots for the other variables, acceleration and horsepower had the most outliers, and this might skew the data and make an analysis less accurate.

Multivariate plots, like the heatmap and pair plots, illustrate column relationships. Each square in the heatmap indicates the strength of the connection between two columns, with red for positive and blue for negative. Darker shades signify stronger connections. The variables that correlate the most with mpg are cylinders, displacement, horsepower, and weight. There is a negative correlation for these four variables. The variables that correlate the least are acceleration and model year. The pair plots plot every combination of variables against each other. For example, there's a plot for mpg vs cylinders as well as a plot for cylinders vs mpg. There is also a trend line for each of these plots. The pair plots are useful to look at alongside the heatmap because I can go to a cell in the heat map and see how high correlation data looks like. I realized that the plots with variables with distinct ranges looked more like a series of undefined functions than the scatter plots I was expecting. Both the cylinders vs weight and weight vs cylinders plots have points that create vertical lines instead of clusters around the trendline. Despite this, there seems to be a high positive correlation between these two variables on the heat map. This suggests that while the scatter plots may not display the typical linear relationship expected due to the distinct ranges, the correlation coefficient still captures the overall association between the variables. This highlights the importance of considering both visualizations like scatter plots and quantitative measures like correlation coefficients to fully understand the relationships between variables.

What procedure did you use to train the model?

The model training procedure begins with the preparation of the dataset where the mpg column is treated as the dependent variable and the remaining columns serve as independent variables. The dataset is then split into training and test sets using the train_test_split() function, with a test size of 0.5 and a random state of 1 to ensure reproducibility. Then, a linear regression model is done using the linear_model.LinearRegression() function from the sklearn library. This model is then trained on the training data using the fit method, adjusting its parameters to minimize the disparity between the actual target variable y_train and the predicted values based on X_train. Following training, the model is utilized to make predictions on the test set X_test via the predict method, specifically predicting the target variable for the first 199 instances.

How does the model perform to predict the fuel efficiency?

The code uses score from scikit-learn's linear regression model to compute R2 for both training and test sets. R2 is coefficient of determination, and it measures how well the model explains the variability in the dependent variable (mpg) using the independent variables. R2 ranges from 0 to 1, with 1 indicating a perfect fit. Using score() for the for the test set as such lr.score(X_test, y_test) calculates the R2 value to assess the model's performance on unseen data. This was also done for lr.score(X_train, y_train), but this test is less relevant. The result of lr.score(X_test, y_test) is about 0.82, meaning that it is closer to 1 than 0.

How confident are you in the model?

With a value of about 0.82, this means that the model explains about 82% of the variability in the dependent variable mpg using the independent variables. The model is pretty good, but it's definitely not perfect, so the predictions made by the model are usable but should be taken with a grain of salt.