Python Project: Multiple Linear Regression/
Model Evaluation Using Visualization by Kate Rogatina

In this project, we will develop several models that will predict the price of the car using the variables or features. This is just an estimate but should give us an objective idea of how much the car should cost.

Some questions we want to ask in this module

- Do I know if the dealer is offering fair value for my trade-in?
- Do I know if I put a fair value on my car?

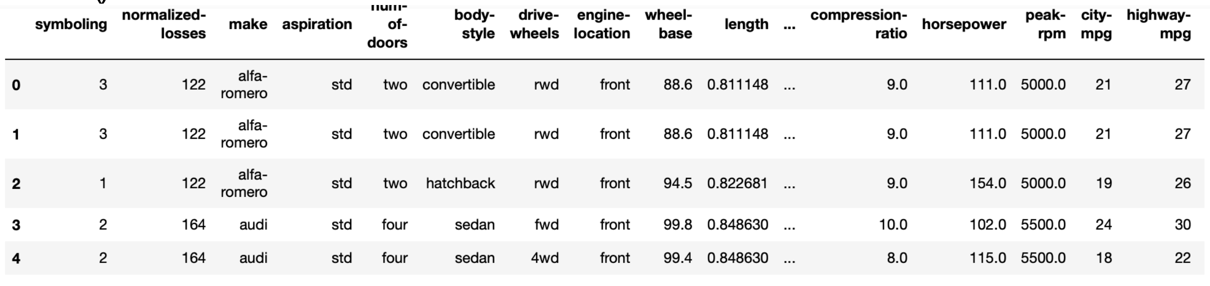
In data analytics, we often use **Model Development** to help us predict future observations from the data we have.
A model will help us understand the exact relationship between different variables and how these variables are used to predict the result.

Import libraries:

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
```

Load the data and store it in dataframe df:

```
df = pd.read_csv('https://cf-courses-data.s3.us.cloud-object-storage.appdomain.cloud/IBMDeveloperSkillsNetwork-DA0101EN-SkillsNetwork/labs/Data%20files/automobileEDA.csv')
df.head()
```



	symboling	normalized-losses	make	aspiration	num-of-doors	body-style	drive-wheels	engine-location	wheel-base	length	...	compression-ratio	horsepower	peak-rpm	city-mpg	highway-mpg
0	3	122	alfa-romero	std	two	convertible	rwd	front	88.6	0.811148	...	9.0	111.0	5000.0	21	27
1	3	122	alfa-romero	std	two	convertible	rwd	front	88.6	0.811148	...	9.0	111.0	5000.0	21	27
2	1	122	alfa-romero	std	two	hatchback	rwd	front	94.5	0.822681	...	9.0	154.0	5000.0	19	26
3	2	164	audi	std	four	sedan	fwd	front	99.8	0.848630	...	10.0	102.0	5500.0	24	30
4	2	164	audi	std	four	sedan	4wd	front	99.4	0.848630	...	8.0	115.0	5500.0	18	22

5 rows × 29 columns

Multiple Linear Regression

What if we want to predict car price using more than one variable?
If we want to use more variables in our model to predict car price, we can use **Multiple Linear Regression**. Multiple Linear Regression is very similar to

Simple Linear Regression, but this method is used to explain the relationship between one continuous response (dependent) variable and **two or more** predictor (independent) variables. Most of the real-world regression models involve multiple predictors. We will illustrate the structure by using four predictor variables, but these results can generalize to any integer:

The equation is given by:
Yhat=a+b1X1+b2X2+b3X3+b4X4

From the previous section we know that other good predictors of price could be:

- Horsepower
- Curb-weight
- Engine-size
- Highway-mpg

Let's develop a model using these variables as the predictor variables.

```
Z = df[['horsepower', 'curb-weight', 'engine-size', 'highway-mpg']]
```

Fit the linear model using the four above-mentioned variables.

```
lm.fit(Z, df['price'])
```

What is the value of the intercept(a)?

```
lm.intercept_
-15806.624626329227
```

What are the values of the coefficients (b1, b2, b3, b4)?

```
lm.coef_
array([53.49574423,  4.70770099, 81.53026382, 36.05748882])
```

As we saw above, we should get a final linear function with the structure:

Yhat = a + b_1 X_1 + b_2 X_2 + b_3 X_3 + b_4 X_4

Price = -15678.742628061467 + 52.65851272 x **horsepower** + 4.69878948 x **curb-weight** + 81.95906216 x **engine-size** + 33.58258185 x **highway-mpg**

Create and train a Multiple Linear Regression model "lm2" where the response variable is "price", and the predictor variable is "normalized-losses" and "highway-mpg".

lm2 = LinearRegression()

lm2

lm2.fit(df[['normalized-losses' , 'highway-mpg']],df['price'])

lm2.coef_

```
array([   1.49789586, -820.45434016])
```

After the development of some models, we evaluate our models and choose the best one. One way to do this is by using a visualization.

Import the visualization package, seaborn:

```
# import the visualization package: seaborn
import seaborn as sns
%matplotlib inline
```

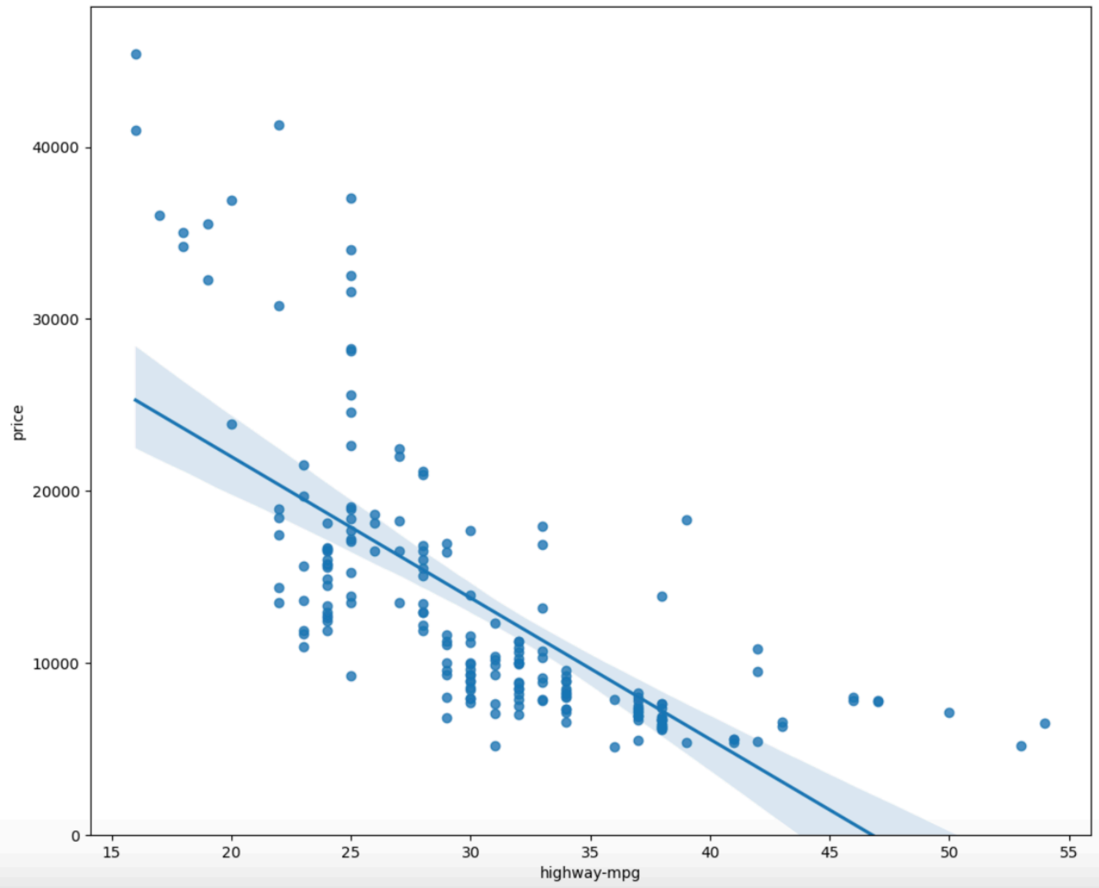
Regression Plot

When it comes to simple linear regression, an excellent way to visualize the fit of our model is by using **regression plots**.
This plot will show a combination of a scattered data points (a **scatterplot**), as well as the fitted **linear regression** line going through the data. This will give us a reasonable estimate of the relationship between the two variables, the strength of the correlation, as well as the direction (positive or negative correlation).

Let's visualize **highway-mpg** as potential predictor variable of price:

```
width = 12
height = 10
plt.figure(figsize=(width, height))
sns.regplot(x="highway-mpg", y="price", data=df)
plt.ylim(0,)
```

```
Out[39]: (0.0, 48188.48081967767)
```

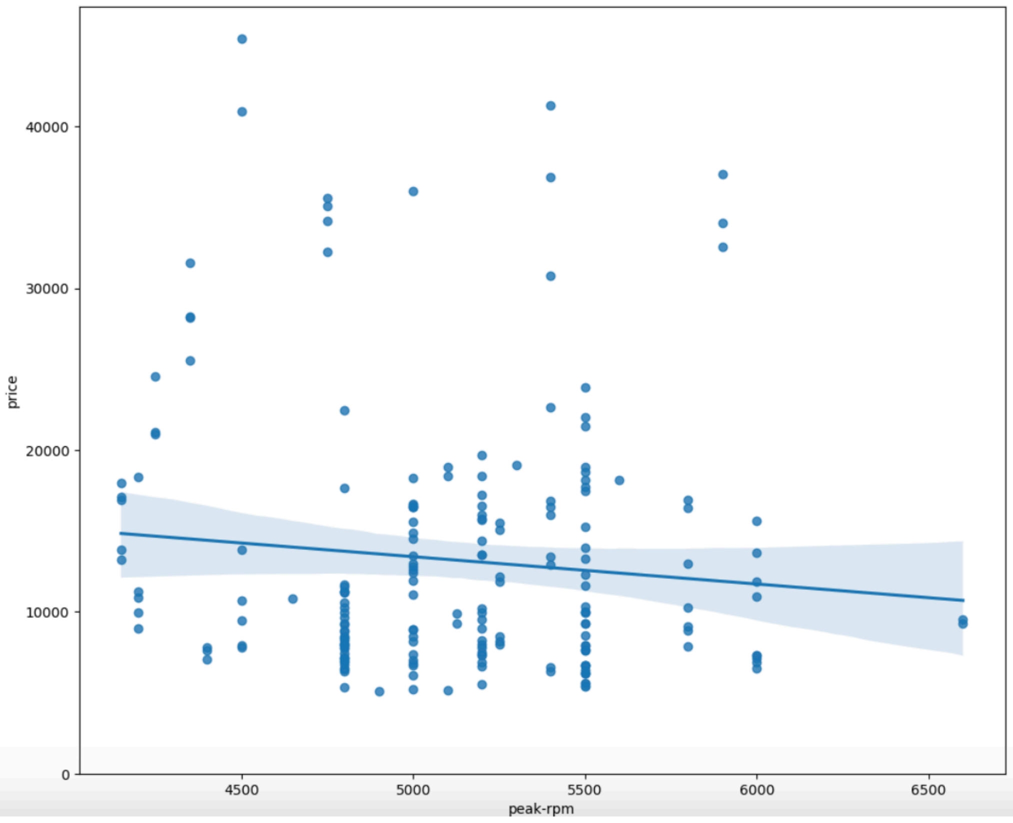


We can see from this plot that price is negatively correlated to highway-mpg since the regression slope is negative.
One thing to keep in mind when looking at a regression plot is to pay attention to how scattered the data points are around the regression line. This will give you a good indication of the variance of the data and whether a linear model would be the best fit or not. If the data is too far off from the line, this linear model might not be the best model for this data.

Let's compare this plot to the regression plot of "peak-rpm".

```
plt.figure(figsize=(width, height))
sns.regplot(x="peak-rpm", y="price", data=df)
plt.ylim(0,)
```

```
Out[40]: (0.0, 47414.1)
```



Comparing the regression plot of "peak-rpm" and "highway-mpg", we see that the points for "highway-mpg" are much closer to the generated line and, on average, decrease. The points for "peak-rpm" have more spread around the predicted line and it is much harder to determine if the points are decreasing or increasing as the "peak-rpm" increases.

For this project I used Jupyter Notebook.