Understanding Multiple Linear Regression

October 22, 2019

0.1 Multiple linear regression

What is Linear Regression?

Linear regression is a predictive analysis that predicts the value on a dependent variable based on an independent variable or multiple independent variables. The predictive analysis where the dependent variable is predicted from one independent variable is Simple linear regression, and when multiple independent variable are used the it is called 'Multiple linear regression'.

Multiple linear regression

This analysis predicts the value of dependent variable (Y) by using the line of best fit, which can be defined by the equation :

i. E(y) = $\beta_0 + \beta_1 X_1 + \beta_2 X_2 + \beta_3 X_3 +$ ..... $+ \beta_n X_n$

Where E(y) is the predicted value of y.
X_1 , X_2 , X_3,. . . .$X_n $ are the values of independent variables.
β_0 is the intercept and
β_1 , β_2 , β_3 are the regression cofficients of X_1, X_2, X_3, X_n.

Looking at an example of Multiple linear regression model

[1]:
```
# import the required libraries

import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import numpy as np
import pylab as pl
import missingno as msno
import stat
from scipy.stats.stats import pearsonr
import scipy.stats as stats

from sklearn.metrics import mean_squared_error, r2_score


# import train test split for splitting the data in train and test
from sklearn.model_selection import train_test_split
```

```python
from sklearn.linear_model import LinearRegression
```

[2]:
```python
# importing the data
data = "https://raw.githubusercontent.com/Blackman9t/Machine_Learning/master/
↪Original_2000_2014_Fuel_Consumption_Ratings.csv"
df = pd.read_csv(data)
```

[3]:
```python
#look at the shape of the data
df.shape
```

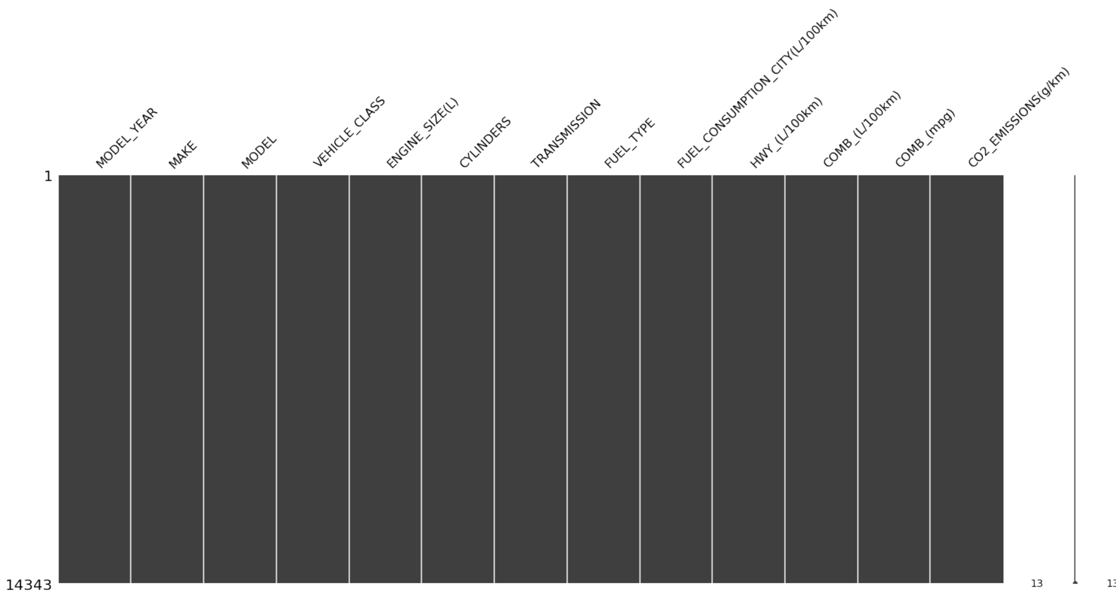
[3]:
```
(14343, 13)
```

[6]:
```python
# see if any data is missing
df.isna().any()
```

[6]:
```
MODEL_YEAR                          False
MAKE                                False
MODEL                               False
VEHICLE_CLASS                       False
ENGINE_SIZE(L)                      False
CYLINDERS                           False
TRANSMISSION                        False
FUEL_TYPE                           False
FUEL_CONSUMPTION_CITY(L/100km)      False
HWY_(L/100km)                       False
COMB_(L/100km)                      False
COMB_(mpg)                          False
CO2_EMISSIONS(g/km)                 False
dtype: bool
```

[12]:
```python
# Visualize missing data using missingno
msno.matrix(df)
```

[12]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x1a1985d0b8>
```



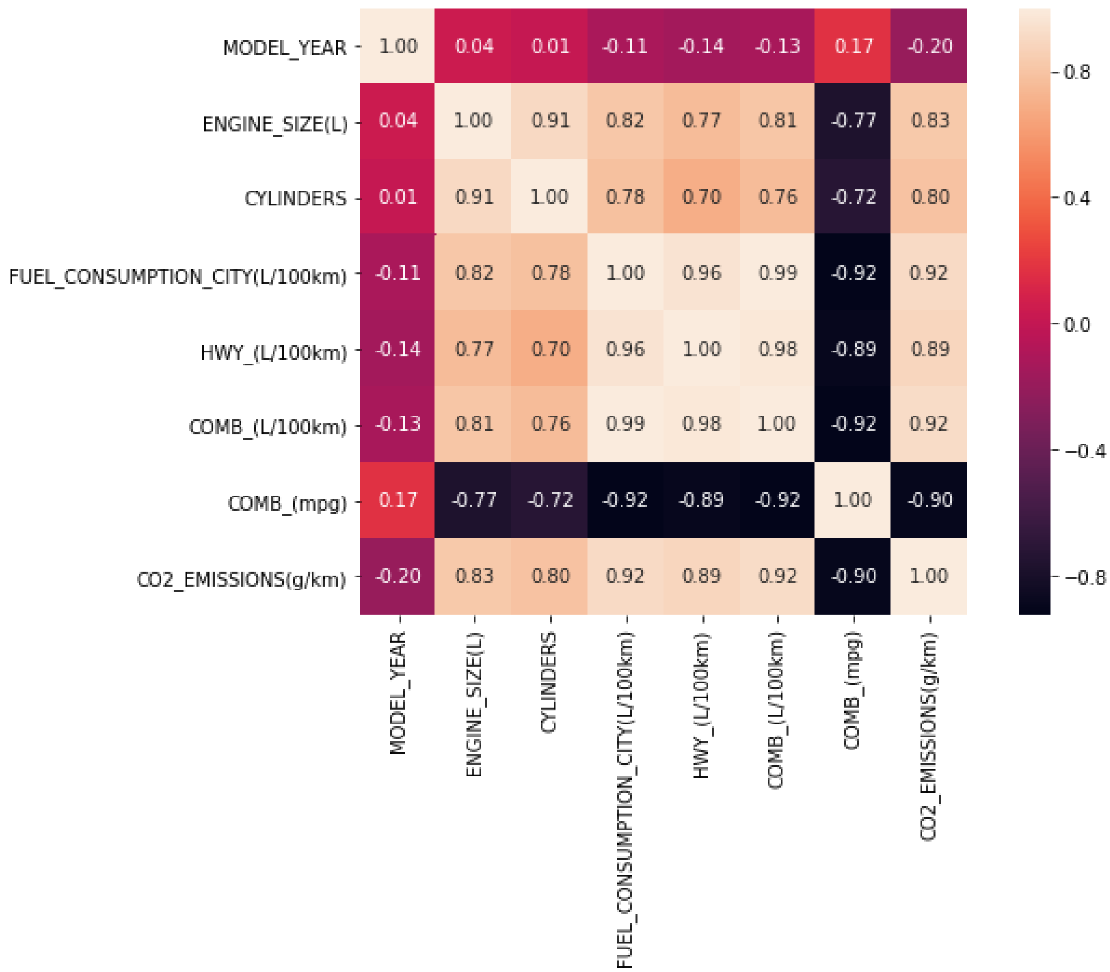
We can see therefore by visualization that no data points are missing.

We want to identify the rate of change emission with respect with other independent variables. In this case CO2 Emission(g/km) is our dependent variable.

Let us look at the correlation between the dependent and the independent variables.

[13]:
```
#outputs a correlation pandas dataframe
corrmat = df.corr()
plt.figure(figsize=(10,6))
# sns.heatmap(corrmat,vmax=1, square=True
sns.heatmap(corrmat, cbar=True, annot=True, square=True, fmt='.2f',␣
 ↪annot_kws={'size': 10})
```

[13]: <matplotlib.axes._subplots.AxesSubplot at 0x1a197a1e80>



From the correlation heatmap we can see the Engine_Size(L), Cylinders, Fuel_consumption in city, highway(HWY_) are positively correlated with CO2 Emission. We can also see the the COMB_(MPG) or combined miles per gallon is negatively correlated to the CO2 emission.

For this project we will take our Engine_size, cylinders and COMB_(mpg) as our independent variables.

Why?

We can see that the Fuel consumption in city, HWY_(L100) and comb_(L/100) have a strong correlation between themselves. This is called multi-colinearity. Moderate multicolinearity might not pose a problem, however, like in our case where a colinearlity of 0.99 is seen this might cause problems, such as:

- Estimates for regression cofficients to be unreliable
- Multicolinearity saps the statistical power of the analysis can cause the coefficients to switch signs, distorting the correct model.

One can test multi-colinearity with Variance Inflation Factors (VIF)however, we won't delve. into that in this paper. I will delve into VIF in another paper in more detail.

Therefore we get a subtle suggestion that the larger the engine size and cylinders the greater CO2 emission, and greater the milege per gallon for a vehicle the lesser the CO2 emission.

[24]:
```
# taking a subset of our data. (Only using independent variables )
X = df[['ENGINE_SIZE(L)','CYLINDERS','COMB_(mpg)']]
# CO2 emission
y = df[['CO2_EMISSIONS(g/km)']]
```

[26]:
```
# Separating the dataset into training and test dataset
# Splitting the data into training and testing dataset
X_train, X_test, y_train, y_test = train_test_split(X,y,test_size = 0.25,␣
↪shuffle = True)
```

[60]:
```
# print(type(df_pred_co2),type(y_test))
df_pred_co2.head(), y_test.head()
```

[60]:
```
(             0
 0  308.451612
 1  294.718224
 2  209.642996
 3  196.662431
 4  250.532403,        CO2_EMISSIONS(g/km)
 9430                   299
 10334                  269
 12099                  209
 9641                   196
 2238                   242)
```

[27]:
```
# Display the shape of the training and test data set
print('X train shape is', X_train.shape)
print('X test shape is', X_test.shape)
print('Y train shape is', y_train.shape)
print('y test shape is', y_test.shape)
```

```
X train shape is (10757, 3)
X test shape is (3586, 3)
Y train shape is (10757, 1)
y test shape is (3586, 1)
```

We can use the least squares estimates(LSE) to find the intercepts. The LSE works very well for computing the coeffients of linear regression, whether simple or multiple linear regression. The only drawback to the LSE is if the independent variable are linearly-dependent.

Mathematically we can get the values of β by

E(y) = 0+11+22+33+. . . ..+

However, we are not going into solving the equation algebraically, and will use the sckit's Linear Regression model instead.

[28]:
```
# instantiate a linear regressin model
model = LinearRegression()
#Next train the model
model.fit(X_train, y_train)
```

[28]: LinearRegression(copy_X=True, fit_intercept=True, n_jobs=None, normalize=False)

[123]:
```
coefficients_X = model.coef_
beta_0 = model.intercept_
print(f"Slope{coefficients_X}, Intercept{beta_0}")
engine_size = coefficients_X[0][0]
cylinders = coefficients_X[0][1]
mpg = coefficients_X[0][2]
intercept = beta_0[0]
```

```
Slope[[ 7.52822873  6.33490332 -4.98463704]], Intercept[320.75938234]
```

[125]:
```
```

[125]: 320.7593823352039

Here we can see the coffiecents of Engine_Size, cylinders and COMB_mpg is 7.52, 6.33 and -4.98 respectievely. Also, the intercept value of β_0 is 320.75

Therefore our equation is:

i. E(y) = 320.75 + 7.52 (Engine_size) + 6.33(Cylinders) + (-4.98)COMB_mpg

[78]:
```
# Using the model to predict values of X_test
predicted_C02_em = model.predict(X_test)
y_test_df = y_test.reset_index()
```

[116]:
```
pred_co2_df = pd.DataFrame(predicted_C02_em.reshape(1,-1)).T
```

[80]:
```
# Combining the actual values and predicted values to
co2_predicted_test = pd.concat([y_test_df, pd.DataFrame(predicted_C02_em)],␣
↪axis = 1)
```

[81]:
```
# We can see the values are close to one another. But how accurate is how model?
co2_predicted_test.head(10)
```

[81]:

	index	CO2_EMISSIONS(g/km)	0
0	9430	299	308.451612
1	10334	269	294.718224
2	12099	209	209.642996
3	9641	196	196.662431
4	2238	242	250.532403
5	6202	285	265.641693
6	7140	242	288.903853
7	1126	315	278.934579
8	4942	315	306.660843
9	7025	193	194.689085

[82]:
```
accuracy = model.score(X_test, y_test)
```

[84]:
```
round(accuracy,2)
```

[84]: 0.86

We can see our model is 86% confident with our predictions.

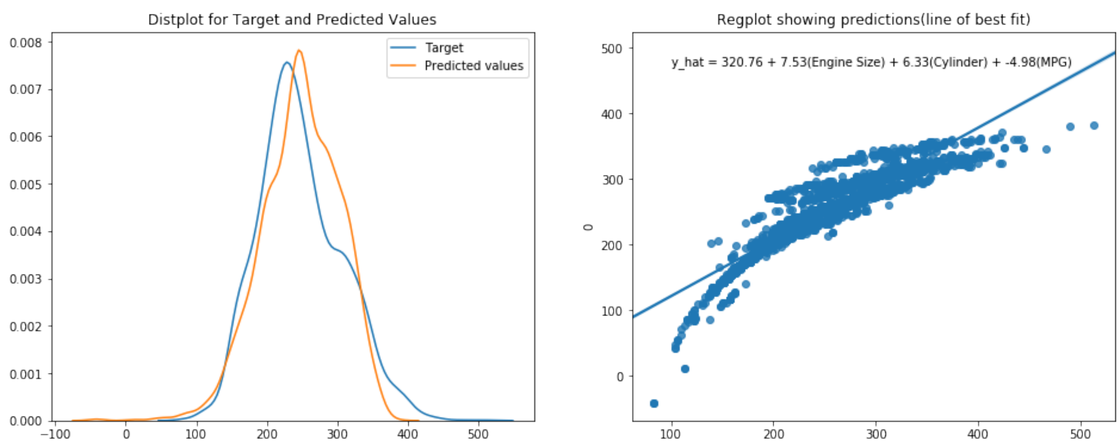
Furthermore, we can graph a distplot to visualize how our predicted values might slighly differ from actual values.

[129]:
```
# Visualization:
# plt.figure(figsize=(10,6))

fig, axs = plt.subplots(1, 2, figsize=(16, 6))
# fig.suptitle ('Plotting the ')

sns.distplot(y_test, hist = False, label = 'Target', ax = axs[0])
axs[0].set_title('Distplot for Target and Predicted Values')
sns.distplot(predicted_C02_em, hist = False, label = 'Predicted values', ax =
 ↪axs[0])

sns.regplot(y_test, pred_co2_df[0], ax = axs[1])
axs[1].set_title('Regplot showing predictions(line of best fit)')
plt.text(100, 475, f"y_hat = {round(intercept,2)} +
 ↪{round(engine_size,2)}(Engine Size) + {round(cylinders,2)}(Cylinder) +
 ↪{round(mpg,2)}(MPG)")
plt.savefig('regression.png')
```



[ ]: