Machine Learning Foundations:

Supervised Machine Learning: Regression course

Part of IBM Machine Learning certificate

How does the vehicle's engine size influence CO_2 emissions?

By Zahra Adahman, MBS.

1. Introduction

Background and objective:

Greenhouse gas emissions are released to the environment due to a variety of human activities. These gases trap heat in the atmosphere leading to climate change. The top four greenhouse gases are Carbon Dioxide (CO_2), Methane (CH_4), Nitrous oxide (N_2O) and Fluorinated Gases (HFCs, PFCs, SF6). However, Carbon Dioxide (CO_2) is the largest greenhouse gas released. According to the US EPA (United States Environmental Protection Agency), transportation accounts for 29 percent of 2019 greenhouse gas emissions) – The transportation sector generates the largest share of greenhouse gas emissions of which is mostly CO_2. CO_2 gas emissions from transportation primarily comes from burning fossil fuels (gas, diesel and so on). About 4.6 metric tons of CO_2 gas is emitted from a typical passenger vehicle every year.

This project aims to develop a regression model to interpret the relationship between engine size and CO_2 emissions. The main (first) objective of this model will help describe the relationship between engine size and CO_2 emissions. In addition, the second objective of this model will be to prescribe what engine size is best to regulate the amount of CO_2 gas emissions. This model can help government and environmental experts in climate change decide on policies for vehicle manufactures to follow when building fossil fuel automobiles to promote less CO_2 gas emissions.

Data acquisition:

The data fuelcomsumption.csv was used for the study were gotten from a previous IBM Data Science course material. The data was imported using pandas.

```
In [7]: CarEmission.columns.values
Out[7]: array(['MODELYEAR', 'MAKE', 'MODEL', 'VEHICLECLASS', 'ENGINESIZE',
               'CYLINDERS', 'TRANSMISSION', 'FUELTYPE', 'FUELCONSUMPTION_CITY',
               'FUELCONSUMPTION_HWY', 'FUELCONSUMPTION_COMB',
               'FUELCONSUMPTION_COMB_MPG', 'CO2EMISSIONS'], dtype=object)

In [8]: CarEmission.describe()
Out[8]:
```

	MODELYEAR	ENGINESIZE	CYLINDERS	FUELCONSUMPTION_CITY	FUELCONSUMPTION_HWY	FUELCONSUMPTION_COMB	FUELCONSUMPTION_COMB_M
count	1067.0	1067.000000	1067.000000	1067.000000	1067.000000	1067.000000	1067.000
mean	2014.0	3.346298	5.794752	13.296532	9.474602	11.580881	26.441
std	0.0	1.415895	1.797447	4.101253	2.794510	3.485595	7.468
min	2014.0	1.000000	3.000000	4.600000	4.900000	4.700000	11.000
25%	2014.0	2.000000	4.000000	10.250000	7.500000	9.000000	21.000
50%	2014.0	3.400000	6.000000	12.600000	8.800000	10.900000	26.000
75%	2014.0	4.300000	8.000000	15.550000	10.850000	13.350000	31.000
max	2014.0	8.400000	12.000000	30.200000	20.500000	25.800000	60.000

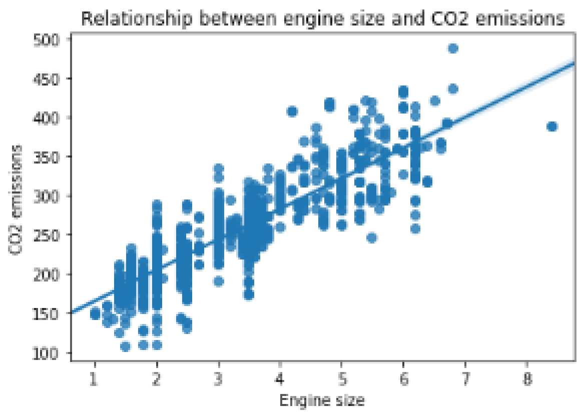
Fig.1 Fuelconsumption.csv data as well as column values.

2. Methodology

Visualization

The final data frame was visualized on using a seaborn (regplot and lmplot) and matplotlib packages, to visualize the distribution of engine size to the CO_2 emissions. Here is the regression plot from the seahorse package.

```
In [10]: import seaborn as sns
         axlin=sns.regplot(x="ENGINESIZE", y="CO2EMISSIONS", data=CarEmission);
         axlin.set(xlabel='Engine size', ylabel='CO2 emissions');
         axlin.set_title('Relationship between engine size and CO2 emissions');
```




```
In [46]: from scipy import stats
         import numpy as np
         slope, intercept, r_value, p_value, std_err = stats.linregress(x,y)
         print ("r-squared:", r_value**2)
         print ("p value:", p_value)
         print ("std err:", std_err)

         r-squared: 0.7641458597854809
         p value: 0.0
         std err: 0.6660631152468043
```

Fig.2. Seaborn regplot showing the relationship between engine size and CO2 emissions

A linear relationship is observed (Fig. 2).

Modeling

Data standardization was done using standard scaler. A linear regression analysis was performed on the standardizes data and the regular data between the independent variable, engine size, and the dependent or target variable, and CO2 emissions. The Scikit-learn package was used to model the training data from a subset (70%) of the final data frame. The coefficient of the standardized and regular data set was different (Fig.3). Cross validation was performed using SciKit Learn with `cross_val_predict` and `GridSearchCV` (Fig.4). Both Lasso and Ridge analysis were performed (Fig.5-7). Regularization was done using the Lasso analysis (Fig.8).

```
In [10]: x = CarEmission.iloc[:, 4].values #ENGINESIZE
         y = CarEmission.iloc[:, 12].values #CO2EMISSIONS
         # Here is the trick
         x = x.reshape(-1,1)

         #X_data= X_data.reshape(-1, 1)
         #Y_data= Y_data.reshape(-1, 1)
```

3. Fit a basic linear regression model on the training data

Data standardization using standardscaler

```
In [11]: from sklearn.preprocessing import StandardScaler

         s = StandardScaler()
         X_ss = s.fit_transform(x)
```

```
In [12]: from sklearn.linear_model import LinearRegression
         lr = LinearRegression()
         lr.fit(x, y)
         print(lr.coef_)

         [39.12519979]
```

a. Fit, and transform using `StandardScaler`

```
In [13]: s = StandardScaler()
         X_ss = s.fit_transform(x)

         lr2 = LinearRegression()
         lr2.fit(X_ss, y)
         print(lr2.coef_) # coefficients  now "on the same scale"

         [55.37121136]
```

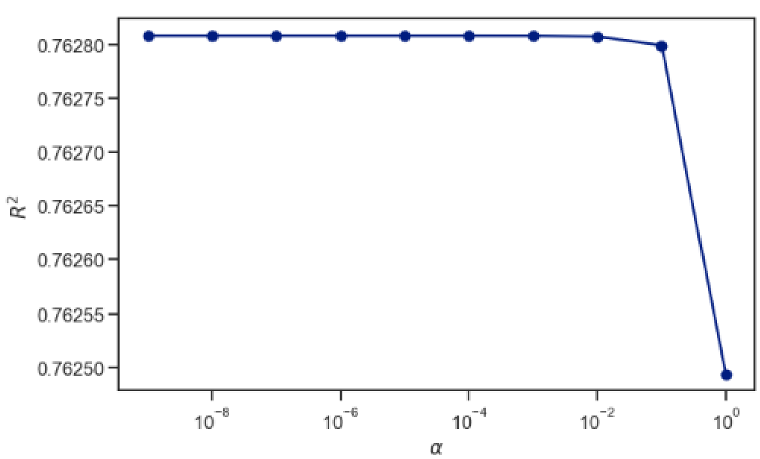
b. Split into train and test sets

```
In [14]: from sklearn.model_selection import train_test_split

         x = CarEmission.iloc[:, 4].values #ENGINESIZE
         y = CarEmission.iloc[:, 12].values #CO2EMISSIONS
         # Here is the trick
         x = x.reshape(-1,1)

         x_train, x_test, y_train, y_test = train_test_split(x, y,
                                                             test_size=0.3, random_state=42)
```

Fig.3 Linear regression analysis of the standardized and regular dataset showing the coefficent of the relationship between engine size, and the dependent or target variable, and CO2 emissions.



When the alpha of $10e^{-1}$, the R^2 is just as good as an alpha of $10e^{-9}$

Fig.4 Cross-validation is just as good at alpha 10^1 as 10^9.

Calculate R^2

```
In [25]: from sklearn.metrics import r2_score
         r2_score(y,las.predict(x_pf_ss))

Out[25]: 0.7638966252533124
```

```
In [26]:
# Decreasing regularization and ensuring convergence
las001 = Lasso(alpha = 0.001, max_iter=100000)

# Transforming training set to get standardized units
x_train_s = s.fit_transform(x_train)

# Fitting model to training set
las001.fit(x_train_s, y_train)

# Transforming test set using the parameters defined from training set
x_test_s = s.transform(x_test)

# Finding prediction on test set
y_pred = las001.predict(x_test_s)

# Calculating r2 score
print("r2 score for alpha = 0.001:", r2_score(y_pred, y_test))


# Part 2

# Using vanilla Linear Regression
lr = LinearRegression()

# Fitting model to training set
lr.fit(x_train_s, y_train)

# predicting on test set
y_pred_lr = lr.predict(x_test_s)

# Calculating r2 score
print("r2 score for Linear Regression:", r2_score(y_pred_lr, y_test))


# Part 3
print('Magnitude of Lasso coefficients:', abs(las001.coef_).sum())
print('Number of coeffients not equal to 0 for Lasso:', (las001.coef_!=0).sum())

print('Magnitude of Linear Regression coefficients:', abs(lr.coef_).sum())
print('Number of coeffients not equal to 0 for Linear Regression:', (lr.coef_!=0).sum)

r2 score for alpha = 0.001: 0.7045550800938665
r2 score for Linear Regression: 0.7045662440527289
Magnitude of Lasso coefficients: 55.103773898000654
Number of coeffients not equal to 0 for Lasso: 1
Magnitude of Linear Regression coefficients: 55.10477389800064
Number of coeffients not equal to 0 for Linear Regression: <built-in method sum of numpy.ndarray object at 0x000001AC
C30BE6C0>
```

Fig.5 The R^2 score was calculated and the Lasso regression was calculated.

```
In [22]: las = Lasso()
         las.fit(x_pf_ss, y)
         las.coef_

Out[22]: array([54.37121136,  0.        ])

In [23]: #alpha of 0.1
         las01 = Lasso(alpha = 0.1)
         las01.fit(x_pf_ss, y)
         print('sum of coefficients:', abs(las01.coef_).sum() )
         print('number of coefficients not equal to 0:', (las01.coef_!=0).sum())

         sum of coefficients: 76.76410074896772
         number of coefficients not equal to 0: 2

In [24]: #alpha of 1
         las1 = Lasso(alpha = 1)
         las1.fit(x_pf_ss, y)
         print('sum of coefficients:', abs(las01.coef_).sum() )
         print('number of coefficients not equal to 0:', (las01.coef_!=0).sum())

         sum of coefficients: 76.76410074896772
         number of coefficients not equal to 0: 2
```

Fig.6 The coefficient of the Lasso regression was calculated.

Ridge Regression

In [27]:
```
from sklearn.linear_model import Ridge

# Decreasing regularization and ensuring convergence
r = Ridge(alpha = 0.001)
x_train_s = s.fit_transform(x_train)
r.fit(x_train_s, y_train)
x_test_s = s.transform(x_test)
y_pred_r = r.predict(x_test_s)

# Calculating r2 score
r.coef_
### END SOLUTION
```

Out[27]:
```
array([55.10470003])
```

In [28]:
```
print(np.sum(np.abs(r.coef_)))
print(np.sum(np.abs(las001.coef_)))

print(np.sum(r.coef_ != 0))
print(np.sum(las001.coef_ != 0))
```

```
55.104700031110525
55.103773898000654
1
1
```

In [29]:
```
y_pred_r = r.predict(x_test_s)
print("r2 score for ridge regression:", r2_score(y_pred_r, y_test))

y_pred_lr = lr.predict(x_test_s)
print("r2 score for Linear Regression:", r2_score(y_pred_lr, y_test))
```

```
r2 score for ridge regression: 0.7045654194494002
r2 score for Linear Regression: 0.7045662440527289
```

Fig. 7 The coefficient of the Ridge regression was calculated.

c. Plot Predictions vs truths

In [20]:
```
import matplotlib.pyplot as plt
import seaborn as sns
%matplotlib inline


sns.set_context('talk')
sns.set_style('ticks')
sns.set_palette('dark')

plt.figure(figsize=(12,8))
ax = plt.axes()
# we are going to use y_test, y_test_pred
ax.scatter(y_test, y_test_pred, alpha=.5)

ax.set(xlabel='Engine Size',
       ylabel='CO2 Emissions',
       title='Car Emissions Predictions vs Truth, using Linear Regression');
```

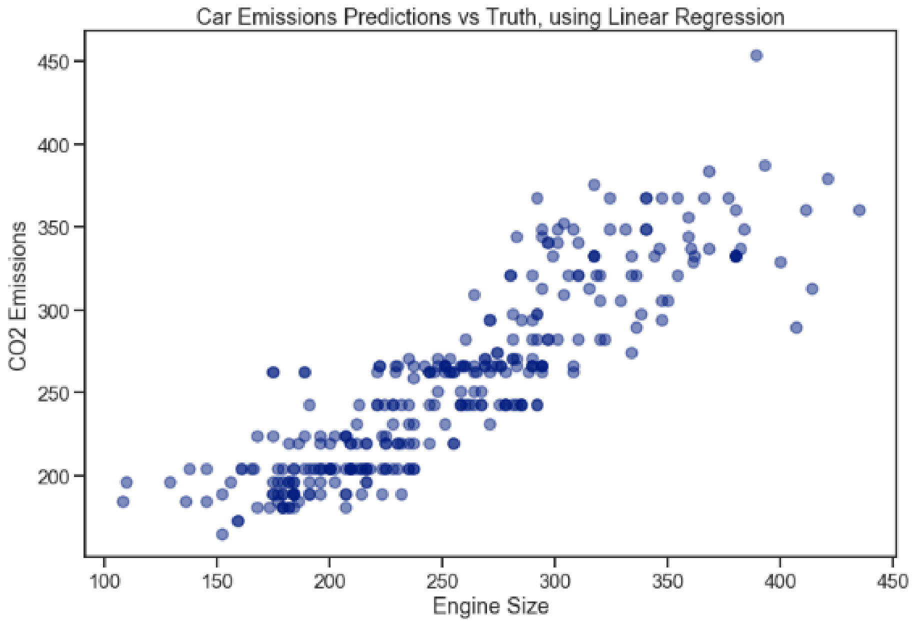


Fig.8 Regularization using Lasso and Ridge regression.

3. Results

These results do show a positive linear relationship between of engine size to the CO_2 emissions. The R^2 score of the linear regression was 0.76 showing that there is a positive correlation between the two variables. The Cross-validation analysis show that an alpha of 10^1 is just as good as 10^9. Lasso and Ridge analysis was done. The R2 score was for Lasso regression was 0.70 and ridge regression 0.70.

4. Discussion and Conclusion

There is no significant difference in the lasso and ridge regression analysis. Either analysis will be good to build the model. The larger engine size correlated with more the CO_2 emissions.

5. Next Steps

The dataset is not very large and the data set was from a previous course. It would be best to test this model on a more diverse dataset.

Reducing CO_2 emissions is not only from controlling fossil-fuels burning vehicles. Hence, tackling the release of CO_2 and other greenhouse gases is not only from using a model for this variables. Other factors are important to consider as it is a multifaceted issue leading to climate change

Appendix

```
In [1]: import numpy as np
        import pandas as pd
        import matplotlib.pyplot as plt
        from sklearn.preprocessing import StandardScaler, PolynomialFeatures
        from sklearn.model_selection import KFold, cross_val_predict
        from sklearn.linear_model import LinearRegression, Lasso, Ridge
        from sklearn.metrics import r2_score
        from sklearn.pipeline import Pipeline
```

Fig.9 Libraries used for regression modeling.

6. References.

a. Overview of Greenhouse Gases, US EPA.

b. Transportation CO2 emissions, US EPA

c. FourSquare API for developers.