(https://www.bigdatauniversity.com)

Multiple Linear Regression

About this Notebook

In this notebook, we learn how to use scikit-learn to implement Multiple linear regression. We download a dataset that is related to fuel consumption and Carbon dioxide emission of cars. Then, we split our data into training and test sets, create a model using training set, Evaluate your model using test set, and finally use model to predict unknown value

Table of contents

Importing Needed packages

```
In [1]: import matplotlib.pyplot as plt
        import pandas as pd
        import pylab as pl
        import numpy as np
        %matplotlib inline
```

Downloading Data

To download the data, we will use !wget to download it from IBM Object Storage.

```
In [2]:  !wget -O FuelConsumption.csv https://s3-api.us-geo.objectstorage.softlayer.net/c
         f-courses-data/CognitiveClass/ML0101ENv3/labs/FuelConsumptionCo2.csv
```

```
--2020-01-28 19:18:04--  https://s3-api.us-geo.objectstorage.softlayer.net/cf-
courses-data/CognitiveClass/ML0101ENv3/labs/FuelConsumptionCo2.csv
Resolving s3-api.us-geo.objectstorage.softlayer.net (s3-api.us-geo.objectstora
ge.softlayer.net)... 67.228.254.196
Connecting to s3-api.us-geo.objectstorage.softlayer.net (s3-api.us-geo.objects
torage.softlayer.net)|67.228.254.196|:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: 72629 (71K) [text/csv]
Saving to: 'FuelConsumption.csv'

FuelConsumption.csv 100%[===================>]  70.93K  --.-KB/s    in 0.04s

2020-01-28 19:18:04 (1.59 MB/s) - 'FuelConsumption.csv' saved [72629/72629]
```

Did you know? When it comes to Machine Learning, you will likely be working with large datasets. As a business, where can you host your data? IBM is offering a unique opportunity for businesses, with 10 Tb of IBM Cloud Object Storage: Sign up now for free (http://cocl.us/ML0101EN-IBM-Offer-CC)

Understanding the Data

`FuelConsumption.csv:`

We have downloaded a fuel consumption dataset, `FuelConsumption.csv` , which contains model-specific fuel consumption ratings and estimated carbon dioxide emissions for new light-duty vehicles for retail sale in Canada. Dataset source (http://open.canada.ca/data/en/dataset/98f1a129-f628-4ce4-b24d-6f16bf24dd64)

- **MODELYEAR** e.g. 2014
- **MAKE** e.g. Acura
- **MODEL** e.g. ILX
- **VEHICLE CLASS** e.g. SUV
- **ENGINE SIZE** e.g. 4.7
- **CYLINDERS** e.g 6
- **TRANSMISSION** e.g. A6
- **FUELTYPE** e.g. z
- **FUEL CONSUMPTION in CITY(L/100 km)** e.g. 9.9
- **FUEL CONSUMPTION in HWY (L/100 km)** e.g. 8.9
- **FUEL CONSUMPTION COMB (L/100 km)** e.g. 9.2
- **CO2 EMISSIONS (g/km)** e.g. 182 --> low --> 0

Reading the data in

In [3]:
```python
df = pd.read_csv("FuelConsumption.csv")

# take a look at the dataset
df.head()
```

Out[3]:

	MODELYEAR	MAKE	MODEL	VEHICLECLASS	ENGINESIZE	CYLINDERS	TRANSMISSION	FUELTYPE
0	2014	ACURA	ILX	COMPACT	2.0	4	AS5	Z
1	2014	ACURA	ILX	COMPACT	2.4	4	M6	Z
2	2014	ACURA	ILX HYBRID	COMPACT	1.5	4	AV7	Z
3	2014	ACURA	MDX 4WD	SUV - SMALL	3.5	6	AS6	Z
4	2014	ACURA	RDX AWD	SUV - SMALL	3.5	6	AS6	Z

Lets select some features that we want to use for regression.

In [4]:
```python
cdf = df[['ENGINESIZE','CYLINDERS','FUELCONSUMPTION_CITY','FUELCONSUMPTION_HWY
','FUELCONSUMPTION_COMB','CO2EMISSIONS']]
cdf.head(9)
```

Out[4]:

	ENGINESIZE	CYLINDERS	FUELCONSUMPTION_CITY	FUELCONSUMPTION_HWY	FUELCONSUMPTION_C
0	2.0	4	9.9	6.7	
1	2.4	4	11.2	7.7	
2	1.5	4	6.0	5.8	
3	3.5	6	12.7	9.1	
4	3.5	6	12.1	8.7	
5	3.5	6	11.9	7.7	
6	3.5	6	11.8	8.1	
7	3.7	6	12.8	9.0	
8	3.7	6	13.4	9.5	

Lets plot Emission values with respect to Engine size:

In [5]:
```python
plt.scatter(cdf.ENGINESIZE, cdf.CO2EMISSIONS,  color='blue')
plt.xlabel("Engine size")
plt.ylabel("Emission")
plt.show()
```
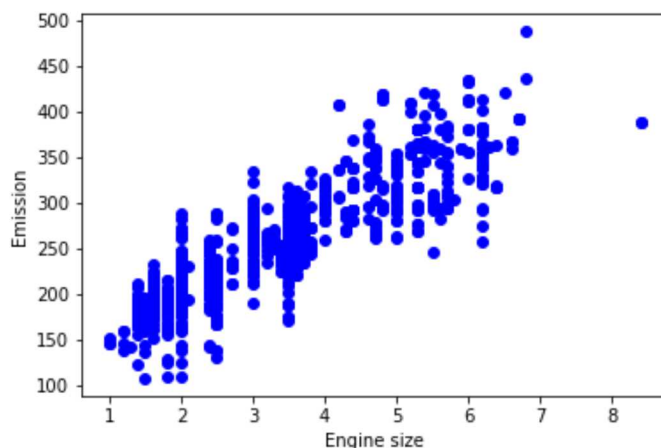
Creating train and test dataset

Train/Test Split involves splitting the dataset into training and testing sets respectively, which are mutually exclusive. After which, you train with the training set and test with the testing set. This will provide a more accurate evaluation on out-of-sample accuracy because the testing dataset is not part of the dataset that have been used to train the data. It is more realistic for real world problems.

This means that we know the outcome of each data point in this dataset, making it great to test with! And since this data has not been used to train the model, the model has no knowledge of the outcome of these data points. So, in essence, it's truly an out-of-sample testing.

```
In [6]: msk = np.random.rand(len(df)) < 0.8
        train = cdf[msk]
        test = cdf[~msk]
        test.head(9)
```

Out[6]:

	ENGINESIZE	CYLINDERS	FUELCONSUMPTION_CITY	FUELCONSUMPTION_HWY	FUELCONSUMPTION_(
2	1.5	4	6.0	5.8	
3	3.5	6	12.7	9.1	
8	3.7	6	13.4	9.5	
23	2.0	4	11.5	8.1	
24	2.0	4	10.8	7.5	
27	3.0	6	9.8	6.4	
29	3.0	6	9.8	6.4	
35	6.3	12	18.2	12.6	
49	4.2	8	21.2	13.4	

Train data distribution

```
In [7]: plt.scatter(train.ENGINESIZE, train.CO2EMISSIONS,  color='blue')
        plt.xlabel("Engine size")
        plt.ylabel("Emission")
        plt.show()
```
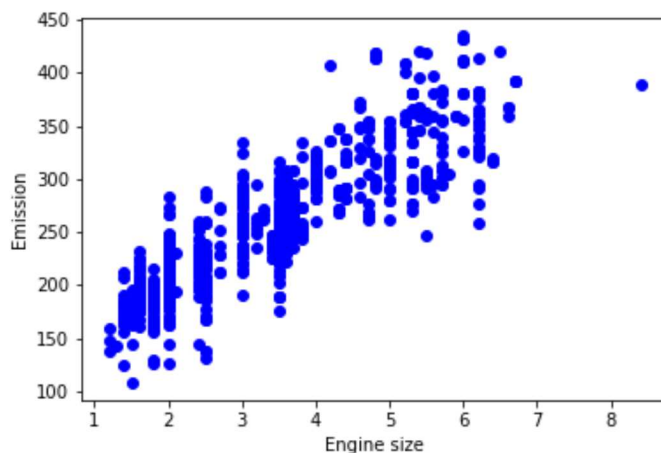

Multiple Regression Model

In reality, there are multiple variables that predict the Co2emission. When more than one independent variable is present, the process is called multiple linear regression. For example, predicting co2emission using FUELCONSUMPTION_COMB, EngineSize and Cylinders of cars. The good thing here is that Multiple linear regression is the extension of simple linear regression model.

```
In [8]: from sklearn import linear_model
        regr = linear_model.LinearRegression()
        x = np.asanyarray(train[['ENGINESIZE','CYLINDERS','FUELCONSUMPTION_COMB']])
        y = np.asanyarray(train[['CO2EMISSIONS']])
        regr.fit (x, y)
        # The coefficients
        print ('Coefficients: ', regr.coef_)
        print ('Intercept: ', regr.intercept_) # grey
```

```
Coefficients:  [[11.12786171  7.5705586   9.34556165]]
Intercept:  [67.11108216]
```

As mentioned before, **Coefficient** and **Intercept** , are the parameters of the fit line. Given that it is a multiple linear regression, with 3 parameters, and knowing that the parameters are the intercept and coefficients of hyperplane, sklearn can estimate them from our data. Scikit-learn uses plain Ordinary Least Squares method to solve this problem.

Ordinary Least Squares (OLS)

OLS is a method for estimating the unknown parameters in a linear regression model. OLS chooses the parameters of a linear function of a set of explanatory variables by minimizing the sum of the squares of the differences between the target dependent variable and those predicted by the linear function. In other words, it tries to minimizes the sum of squared errors (SSE) or mean squared error (MSE) between the target variable (y) and our predicted output ($\hat{y}$) over all samples in the dataset.

OLS can find the best parameters using of the following methods:

```
    - Solving the model parameters analytically using closed-form equations
    - Using an optimization algorithm (Gradient Descent, Stochastic Gradient Descent, N
    ewton's Method, etc.)
```

Prediction

```
In [9]: y_hat= regr.predict(test[['ENGINESIZE','CYLINDERS','FUELCONSUMPTION_COMB']])
        x = np.asanyarray(test[['ENGINESIZE','CYLINDERS','FUELCONSUMPTION_COMB']])
        y = np.asanyarray(test[['CO2EMISSIONS']])
        print("Residual sum of squares: %.2f"
              % np.mean((y_hat - y) ** 2))

        # Explained variance score: 1 is perfect prediction
        print('Variance score: %.2f' % regr.score(x, y))
```

```
Residual sum of squares: 620.93
Variance score: 0.87
```

explained variance regression score:
If $\hat{y}$ is the estimated target output, y the corresponding (correct) target output, and Var is Variance, the square of the standard deviation, then the explained variance is estimated as follow:

$\texttt{explainedVariance}(y, \hat{y}) = 1 - \frac{Var\{ y - \hat{y}\}}{Var\{y\}}$
The best possible score is 1.0, lower values are worse.

Practice

Try to use a multiple linear regression with the same dataset but this time use **FUEL CONSUMPTION in CITY** and **FUEL CONSUMPTION in HWY** instead of FUELCONSUMPTION_COMB. Does it result in better accuracy?

```
In [10]:  # write your code here

          # step 1: train = estimate
          regr2 = linear_model.LinearRegression()
          x2 = np.asanyarray(train[['ENGINESIZE','CYLINDERS','FUELCONSUMPTION_CITY', 'FUEL
          CONSUMPTION_HWY']])
          y2 = np.asanyarray(train[['CO2EMISSIONS']])
          #y2 = y

          regr2.fit (x2, y2)
          print("Coefficients: ", regr2.coef_)
          print("Intercept: ", regr2.intercept_)
```

```
          Coefficients:  [[11.30748332  6.96588582  6.50089782  2.41420334]]
          Intercept:  [68.91332232]
```

```
In [17]:  # step 2: test = predict
          #x2_test = test[['ENGINESIZE','CYLINDERS','FUELCONSUMPTION_CITY', 'FUELCONSUMPTI
          ON_CITY']]
          #y2_hat = regr.predict(x2_test)
          #y_= regr2.predict(test[['ENGINESIZE','CYLINDERS','FUELCONSUMPTION_CITY','FUELCO
          NSUMPTION_HWY']]) # error: regr2, not regr
          y2_hat = regr2.predict(test[['ENGINESIZE','CYLINDERS','FUELCONSUMPTION_CITY','FU
          ELCONSUMPTION_HWY']]) # error: regr2, not regr
          x2_test = np.asanyarray(test[['ENGINESIZE','CYLINDERS','FUELCONSUMPTION_CITY', '
          FUELCONSUMPTION_HWY']]) # error: CITY, HWY; not COMB
          y2_test = np.asanyarray(test[['CO2EMISSIONS']])

          print("RSS: %.2f" % np.mean((y2_test - y2_hat)**2))
          print("Variance Score: %.2f" % regr2.score(x2_test, y2_test)) # error: format %.
          2f; not %2.f
```

```
          RSS: 625.53
          Variance Score: 0.86
```

Double-click **here** for the solution.

Want to learn more?

IBM SPSS Modeler is a comprehensive analytics platform that has many machine learning algorithms. It has been designed to bring predictive intelligence to decisions made by individuals, by groups, by systems – by your enterprise as a whole. A free trial is available through this course, available here: SPSS Modeler (http://cocl.us/ML0101EN-SPSSModeler)

Also, you can use Watson Studio to run these notebooks faster with bigger datasets. Watson Studio is IBM's leading cloud solution for data scientists, built by data scientists. With Jupyter notebooks, RStudio, Apache Spark and popular libraries pre-packaged in the cloud, Watson Studio enables data scientists to collaborate on their projects without having to install anything. Join the fast-growing community of Watson Studio users today with a free account at Watson Studio (https://cocl.us/ML0101EN_DSX)

Thanks for completing this lesson!

Author: Saeed Aghabozorgi (https://ca.linkedin.com/in/saeedaghabozorgi)

Saeed Aghabozorgi (https://ca.linkedin.com/in/saeedaghabozorgi), PhD is a Data Scientist in IBM with a track record of developing enterprise level applications that substantially increases clients' ability to turn data into actionable knowledge. He is a researcher in data mining field and expert in developing advanced analytic methods like machine learning and statistical modelling on large datasets.