**Multiple Linear Regression**[**¶**](https://jupyterlab-4-labs-prod-jupyterlab-us-east-0.labs.cognitiveclass.ai/user/akulasamson/lab/tree/labs/coursera/ML0101EN/ML0101EN-Reg-Mulitple-Linear-Regression-Co2.ipynb#Multiple-Linear-Regression)

Estimated time needed: **15** minutes

**Objectives**

After completing this lab you will be able to:

* Use scikit-learn to implement Multiple Linear Regression
* Create a model, train it, test it and use the model

**Table of contents**

1. [Understanding the Data](https://jupyterlab-4-labs-prod-jupyterlab-us-east-0.labs.cognitiveclass.ai/user/akulasamson/files/labs/coursera/ML0101EN/https%3A/%23understanding-data?_xsrf=2%7C34313e40%7C6238ca29ce5b2e428c5db70dff082614%7C1630087457)
2. [Reading the Data in](https://jupyterlab-4-labs-prod-jupyterlab-us-east-0.labs.cognitiveclass.ai/user/akulasamson/files/labs/coursera/ML0101EN/https%3A/%23reading_data?_xsrf=2%7C34313e40%7C6238ca29ce5b2e428c5db70dff082614%7C1630087457)
3. [Multiple Regression Model](https://jupyterlab-4-labs-prod-jupyterlab-us-east-0.labs.cognitiveclass.ai/user/akulasamson/files/labs/coursera/ML0101EN/https%3A/%23multiple_regression_model?_xsrf=2%7C34313e40%7C6238ca29ce5b2e428c5db70dff082614%7C1630087457)
4. [Prediction](https://jupyterlab-4-labs-prod-jupyterlab-us-east-0.labs.cognitiveclass.ai/user/akulasamson/files/labs/coursera/ML0101EN/https%3A/%23prediction?_xsrf=2%7C34313e40%7C6238ca29ce5b2e428c5db70dff082614%7C1630087457)
5. [Practice](https://jupyterlab-4-labs-prod-jupyterlab-us-east-0.labs.cognitiveclass.ai/user/akulasamson/files/labs/coursera/ML0101EN/https%3A/%23practice?_xsrf=2%7C34313e40%7C6238ca29ce5b2e428c5db70dff082614%7C1630087457)

**Importing Needed packages**

[ ]:



**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.

[ ]:



**!**wget **-**O FuelConsumption.csv https:**//**cf**-**courses**-**data.s3.us.cloud**-**object**-**storage.appdomain.cloud**/**IBMDeveloperSkillsNetwork**-**ML0101EN**-**SkillsNetwork**/**labs**/**Module**%**202**/**data**/**FuelConsumptionCo2.csv

**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?utm_medium=Exinfluencer&utm_source=Exinfluencer&utm_content=000026UJ&utm_term=10006555&utm_id=NA-SkillsNetwork-Channel-SkillsNetworkCoursesIBMDeveloperSkillsNetworkML0101ENSkillsNetwork20718538-2021-01-01)

* **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**

[ ]:



df **=** pd.read\_csv("FuelConsumption.csv")

​

*# take a look at the dataset*

df.head()

Let's select some features that we want to use for regression.

[ ]:



cdf **=** df[['ENGINESIZE','CYLINDERS','FUELCONSUMPTION\_CITY','FUELCONSUMPTION\_HWY','FUELCONSUMPTION\_COMB','CO2EMISSIONS']]

cdf.head(9)

Let's plot Emission values with respect to Engine size:

[ ]:



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 model. Therefore, it gives us a better understanding of how well our model generalizes on new data.

We know the outcome of each data point in the testing dataset, making it great to test with! 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 is truly an out-of-sample testing.

Let's split our dataset into train and test sets. Around 80% of the entire dataset will be used for training and 20% for testing. We create a mask to select random rows using the **np.random.rand()** function:

[ ]:



msk **=** np.random.rand(len(df)) **<** 0.8

train **=** cdf[msk]

test **=** cdf[**~**msk]

**Train data distribution**

**Did you know? IBM Watson Studio lets you build and deploy an AI solution, using the best of open source and IBM software and giving your team a single environment to work in.**[**Learn more here.**](https://cocl.us/ibm_watson_studio_infobox)

[ ]:



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 impact the Co2emission. When more than one independent variable is present, the process is called multiple linear regression. An example of multiple linear regression is predicting co2emission using the features FUELCONSUMPTION\_COMB, EngineSize and Cylinders of cars. The good thing here is that multiple linear regression model is the extension of the simple linear regression model.

[ ]:



**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\_)

As mentioned before, **Coefficient** and **Intercept** are the parameters of the fitted line. Given that it is a multiple linear regression model with 3 parameters and that the parameters are the intercept and coefficients of the 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 (𝑦̂ 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, Newton’s Method, etc.)

**Prediction**

[ ]:



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))

\*\*Explained variance regression score:\*\*  
Let 𝑦̂ y^ be the estimated target output, y the corresponding (correct) target output, and Var be the Variance (the square of the standard deviation). Then the explained variance is estimated as follows:

𝚎𝚡𝚙𝚕𝚊𝚒𝚗𝚎𝚍𝚅𝚊𝚛𝚒𝚊𝚗𝚌(𝑦,𝑦̂ )=1−𝑉𝑎𝑟𝑦−𝑦̂ 𝑉𝑎𝑟𝑦explainedVariance(y,y^)=1−Vary−y^Vary  
The best possible score is 1.0, the 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?

[ ]:



*# write your code here*

​

​

Click here for the solution

regr **=** linear\_model.LinearRegression()

x **=** np.asanyarray(train[['ENGINESIZE','CYLINDERS','FUELCONSUMPTION\_CITY','FUELCONSUMPTION\_HWY']])

y **=** np.asanyarray(train[['CO2EMISSIONS']])

regr.fit (x, y)

print ('Coefficients: ', regr.coef\_)

y\_**=** regr.predict(test[['ENGINESIZE','CYLINDERS','FUELCONSUMPTION\_CITY','FUELCONSUMPTION\_HWY']])

x **=** np.asanyarray(test[['ENGINESIZE','CYLINDERS','FUELCONSUMPTION\_CITY','FUELCONSUMPTION\_HWY']])

y **=** np.asanyarray(test[['CO2EMISSIONS']])

print("Residual sum of squares: %.2f"**%** np.mean((y\_ **-** y) **\*\*** 2))

print('Variance score: %.2f' **%** regr.score(x, y))