

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.

```
[]: msk = np.random.rand(len(df)) < 0.8
train = cdf[msk]
test = cdf[~msk]</pre>
```

Train data distribution

Did you know? IBM Watson Studio lets you build and deploy an Al solution, using the best of open source and IBM software and giving your team a single environment to work in. Learn more here.

```
[]: plt.scatter(train.ENGINESIZE, train.CO2EMISSIONS, color='blue')
plt.xlabel("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.

```
[]: from_sklearn_import_linear_model
regr = linear_model.LinearRegression()
x = np.asanyarray(train[['ENGINESIZE','CYLINGERS','FUELCONSUMPTION_COMB']])
y = np.asanyarray(train[['COZEMISSIONS']])
regr.fit_(x, y)
# The coefficients
print_('Coefficients', regr.coef_)
```

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, Newton's Method, etc.)

Prediction

explained variance regression score:

If \hat{y} is the estimated arget 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:

```
explainedVariance(y, \hat{y}) = 1 - \frac{Vary - \hat{y}}{Vary}
```

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?

```
[ ]: # 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[['COZEMISSIONS']])
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[['COZEMISSIONS']])
print("Residual sum of squares: %.2f" % np.mean((y_ - y) ** 2))
print('Variance score: %.2f' % regr.score(x, y))
```

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

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

Thank you for completing this lab!

Saeed Aghabozorgi

Other Contributors

Joseph Santarcangelo

Change Log

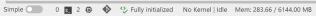
Date (YYYY-MM-DD)	Version	Changed By	Change Description
2020-11-03	2.1	Lakshmi	Made changes in URL
2020-08-27	2.0	Lavanya	Moved lab to course repo in GitLab

 $\ensuremath{\mathbb{C}}$ IBM Corporation 2020. All rights reserved.











Mode: Command 😵 Ln 1, Col 1 English (American) ML0101EN-Reg-Mulitple-Linear-Regression-Co2-py-v1.ipynb