# Before your start:

* Read the README.md file
* Comment as much as you can and use the resources in the README.md file
* Happy learning!

# Import your libraries:  
import pandas as pd  
from sklearn.datasets import load\_diabetes

# Challenge 1 - Explore the Scikit-Learn Datasets

Before starting to work on our own datasets, let's first explore the datasets that are included in this Python library. These datasets have been cleaned and formatted for use in ML algorithms.

First, we will load the diabetes dataset. Do this in the cell below by importing the datasets and then loading the dataset to the diabetes variable using the load\_diabetes() function.

# Your code here:

Let's explore this variable by looking at the different attributes. Do this by looking at the keys() of this variable.

# Your code here:

The next step is to read the description of the dataset. Print the description in the cell below using the DESCR attribute of the diabetes variable

# Your code here:

What are the variables in this dataset according to the description? List them in the markdown cell below

* age age in years
  + sex
  + bmi body mass index
  + bp average blood pressure
  + s1 tc, total serum cholesterol
  + s2 ldl, low-density lipoproteins
  + s3 hdl, high-density lipoproteins
  + s4 tch, total cholesterol / HDL
  + s5 ltg, possibly log of serum triglycerides level
  + s6 glu, blood sugar level

#### Enter your answer here:

age, sex, body mass index, average blood pressure, s1, s2, s3, s4, s5, s6

Now explore the data. Scikit-learn typically takes in 2D numpy arrays as input (though pandas dataframes are also accepted). In the cell below find the shape of the numpy array contained in the data portion of the diabetes variable.

# Your code here:

# Challenge 2 - Perform Supervised Learning on the Dataset

#### The data has already been split to predictor and response variables. The response variable is in the target portion of the variable.

Given this information, let's apply what we have previously learned about linear regression and apply the algorithm to the diabetes dataset. In the cell below, import the linear regression class from sklearn.

# Your code here:

Initialize the model in the variable diabetes\_model

# Your code here:

In the cell below, fit the model and print the intercept and coefficients of the model.

# Your code here:

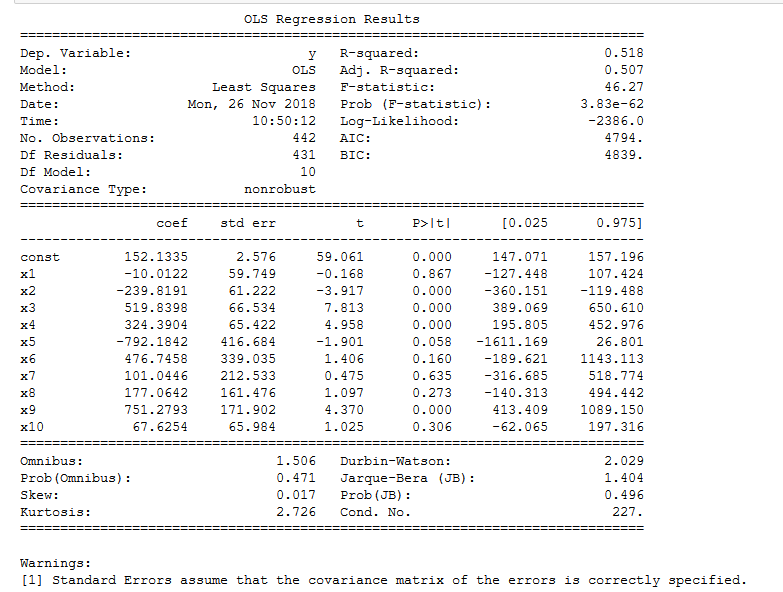
# Bonus Challenge 1 - Conduct a Hypothesis Test on the Model

Once we have generated a linear model, we can test each coefficient using a t-test to see whether the confidence interval for the variable contains zero. We can also perform an overall F test to check whether at least one coefficient is significantly different from zero.

Refer to the resource in this [link](https://onlinecourses.science.psu.edu/stat501/node/297/) for more details and perform the t-tests for the model above. Additionally, interpret the results and list coefficients are significantly different from zero.

Hint: use the statsmodels package.

Result should look like this:



# Your code here:

# Challenge 3 - Peform Supervised Learning on a Pandas Dataframe

Now that we have looked at data that has been formatted for scikit-learn, let's look at data that we will need to format ourselves.

In the next cell, load the auto-mpg.csv file included in this folder and assign it to a variable called auto.

# Your code here:

Look at the first 5 rows using the head() function:

# Your code here:

Evaluate the data to ensure that all numeric columns are correctly detected as such by pandas. If a column is misclassified as object, coerce it to numeric.

# Your code here:

What is the newest model year and the oldest model year?

# Your code here:

Check the dataset for missing values and remove all rows containing at least one missing value.

# Your code here:

auto.dropna(axis=0, how="any", inplace=True)

Find the frequency table for the cylinders column using the value\_counts() function. How many possible values of cylinders are there?

# Your code here:

We would like to generate a linear regression model that will predict mpg. To do this, first drop the car\_name column since it does not contain any quantitative data. Next separate the dataframe to predictor and response variables. Separate those into test and training data with 80% of the data in the training set and the remainder in the test set.

Assign the predictor and response training data to X\_train and y\_train respectively. Similarly, assign the predictor and response test data to X\_test and y\_test.

# Your code here:

Now we will the dataset that we processed and peform linear regression on this data to predict the mpg for each vehicle. Initialize the model in the cell below.

# Your code here:

Next, fit the model in the cell below.

# Your code here:

# Challenge 4 - Evaluate the Model

the r squared score of a model tells us how much variation is explained by the model. In a typical dataset, most observations differ from the mean. When we create a model, we are trying to generate an equation that will tell us by how much each observation will differ from the mean. Obviously, the vast majority of models are not perfect. They can only predict some of the variation from the mean but not all of it. We attribute the rest of the difference between the actual value and the mean to random error. We would like random error to explain the as little as possible of the variation. This is why the r squared score is an important metric.

In the next cell, compute the r squared score of the model. Do this by first computing the predicted values and assign them to y\_pred.

# Your code here:

#### Our next step is to evaluate the model using the test data. We would like to ensure that our model is not overfitting the data. This means that our model will not be able to generalize well outside of the training data.

In the cell below, use the model to generate the predicted values for the training data and assign them to y\_test\_pred. Compute the r squared score for the test data by comparing the oberserved y\_train data and the predicted y\_test\_pred.

# Your code here:

# Challenge 5 - Improve the Model Fit

While the most common way to improve the fit of a model is by using regularization, there are other simpler ways to improve model fit. The first is to create a simpler model. The second is to increase the train sample size.

Let us start with the easier option and increase our train sample size to 90% of the data. Create a new test train split and name the new predictors and response variables X\_train09, X\_test09, y\_train09, y\_test09.

# Your code here:

Initialize a new model. Name this model auto\_model09. Fit the model to the new sample data.

# Your code here:

Compute the predicted values and r squared score for our new model and new sample data.

# Your code here:

Compute the r squared score for the smaller test set. Is there an improvement in the test r squared?

# Your code here:

# Bonus Challenge 2 - Backward Elimination

The main way to produce a simpler linear regression model is to reduce the number of variables used in the model. In scikit-learn, we can do this by using recursive feature elimination. You can read more about RFE [here](https://scikit-learn.org/stable/modules/generated/sklearn.feature_selection.RFE.html).

In the next cell, we will import RFE

from sklearn.feature\_selection import RFE

Follow the documentation and initialize an RFE model using the auto\_model linear regression model. Set n\_features\_to\_select=3

# Your code here:

Fit the model and print the ranking

# Your code here:

Feature importance is ranked from most important (1) to least important (4). Generate a model with the three most important features. The features correspond to variable names. For example, feature 1 is cylinders and feature 2 is displacement.

Perform a test-train split on this reduced column data and call the split data X\_train\_reduced, X\_test\_reduced, y\_test\_reduced, y\_train\_reduced. Use an 80% split.

# Your code here:

Generate a new model called auto\_model\_reduced and fit this model. Then proceed to compute the r squared score for the model. Did this cause an improvement in the r squared score?

# Your code here: