# Using the Anaconda Distribution for Python

## Overview

Python is a powerful tool for scientific computing, data analysis, and machine learning. In this lab, we will learn how simple it is to perform

a machine learning experiment using the Anaconda distribution installed via the Visual Studio Data Science workload. In doing so, we will interact

with some of the fundamental Python tools for VS to set-up our environment and execute the experiment.

## The experiment

Predictive Analysis involves using existing trend data to train models to predict future movement of that trend. In this lab we will use open data

to predict stock prices using three different regression models. Our goal is to find the regression method that leads to most accurate predictions.

Our Python program will break the stock data into two sets: a training set and a test set. The test test will be used to compare predicted values

with actual values.

To understand how Anaconda can help abstract away environment set-up, we will first conduct the experiment with a manually set-up environment, and

once with the Anaconda environment.

## Let's get started!

### Preparing the experiment script

1. Open the Cookiecutter Explorer (Tools -> Python Tools -> Cookiecutter Explorer)

2. Create a simple regression script by searching for a template. Choose the electir/build-regression template.

3. Import requisite packages by adding the following lines of code under the imports section:

Data will be read into a Pandas Data Frame:

from pandas import read\_table

The Pandas Data Frame will be transmuted into NumPy arrays before model calc.NumPy is optimized for matrix calculations.

import numpy as np

Your favorite plotting library

import matplotlib.pyplot as plt

A powerful ML package for Python

from sklearn.svm import SVR

4. In the get\_features\_and\_labels() method, add the following lines of code to create training and test sets

from sklearn.cross\_validation import train\_test\_split

X\_train, \_, y\_train, \_ = train\_test\_split(X, y, test\_size=0.5)

X\_test, y\_test = X, y

5. In the evaluate\_learner() method, add code to train each of the three regression learners

For Radial Basis Function:

svr = SVR(kernel='rbf', gamma=0.1)

svr.fit(X\_train, y\_train)

y\_pred = svr.predict(X\_test)

r\_2 = svr.score(X\_test, y\_test)

yield 'RBF Model ($R^2={:.3f}$)'.format(r\_2), y\_test, y\_pred

For Linear Kernel:

svr = SVR(kernel='linear')

svr.fit(X\_train, y\_train)

y\_pred = svr.predict(X\_test)

r\_2 = svr.score(X\_test, y\_test)

yield 'Linear Model ($R^2={:.3f}$)'.format(r\_2), y\_test, y\_pred

For Polynomial Kernel

svr = SVR(kernel='poly', degree=2)

svr.fit(X\_train, y\_train)

y\_pred = svr.predict(X\_test)

r\_2 = svr.score(X\_test, y\_test)

yield 'Polynomial Model ($R^2={:.3f}$)'.format(r\_2), y\_test, y\_pred

7. At this point, we have all the code to run the experiment. Take some time to look through other methods of the script.

### Set up the environment manually & Run

1. Open the Environments Window (Tools -> Python Tools -> Environment)

2. Create a custom Python 3 environment, make it the default for all programs. If you don't see one, use the tool to create one.

a. The path prefix is: C:\Users\Administrator\AppData\Local\Programs\Python\Python36

3. Navigate to packages and install the NumPy, SciPy, sklearn, pandas, and matplotlib.

You should see a plot of precited values. The R^2 value is a measure of accuracy of the prediction. The closer this number is to 1, the better

the predicted values curve fits with the curve of actual values. Which regression technique best suits this problem?

### Re-run with Anaconda

1. Delete the Python 3 environment you created in the previous step.

2. Select the Anaconda environment as the default for all projects.

3. Run the script again.

As you can see using Anaconda precludes the need to customize the environment - it already comes loaded with many popular Python packages for

scientific computing.