ST0249   
AI & Machine Learning

Practical 3  
Python Machine Learning



What you will learn / do in this lab

1. Explore using Python for Machine Learning
2. Explore Scikit-learn
3. Conduct an experiment with the iris dataset

TAble of  
Contents

1. Overview 1

Scikit-learn 1

Installation 2

2. using Scikit-Learn (SKLEARN) 4

Using jupyter 4

Using python source file 6

Data loading and preparation 7

3. ML Experiment with sKLEARN 10

Data Preparation 10

Modelling 10

Scoring/Predicting 11

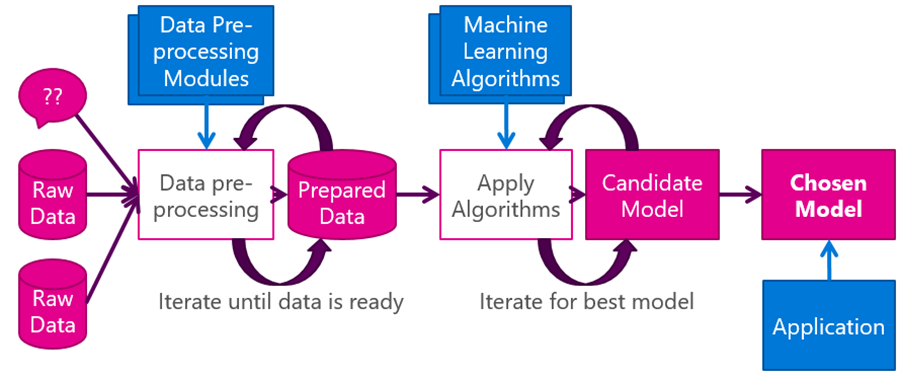
Evaluation 12

# 1. Overview

In this practical we will learn how to conduct the machine learning workflow using the python programming language and the scikit-learn machine learning library.

## Scikit-learn

Scikit-learn is a python library that contains classes and functions that help us with most the most common tasks in a machine learning workflow. The machine learning workflow which we are referring to should be familiar to you by now; it is the same as the one we used with Azure ML Studio. We start with formulating the questions first, then data preparation, etc.



Relationship between Numpy, Scipy, Pandas and Scikit-learn

Numpy and Scipy add arrays/vectors and matrix computations to python. Pandas brings DataFrames to python. DataFrames is a core data structure in data analysis. Building on top of these data structures, Scikit-learn brings implementations of best-of-breed machine-learning algorithms, all under a standardized library.

* PREPROCESSING WITH PANDAS
  + Reading data
  + Selecting columns and rows
  + Filtering
  + Vectorized string operations
  + Missing values
  + Handling time
  + Time series
* NUMPY, SCIPY
  + Arrays
  + Indexing, Slicing, and Iterating
  + Reshaping
  + Shallow vs deep copy
  + Broadcasting
  + Indexing (advanced)
  + Matrices
  + Matrix decompositions
* SCIKIT-LEARN
  + Feature extraction
  + Classification
  + Regression
  + Clustering
  + Dimension reduction
  + Model selection

## Installation

If you are using Anaconda, scikit-learn is already included in the distribution; you don’t need to install scikit-learn separately.

To install scikit-learn with other python distributions, refer to the instructions on the scikit-learn site:

[http://scikit-learn.org/stable/install.html#](http://scikit-learn.org/stable/install.html)

If you already have a working installation of numpy and scipy, the easiest way to install scikit-learn is using pip

pip install -U scikit-learn

Since Anaconda already comes with scikit-learn, you should use conda to update scikit-learn to the latest version (instead of pip)

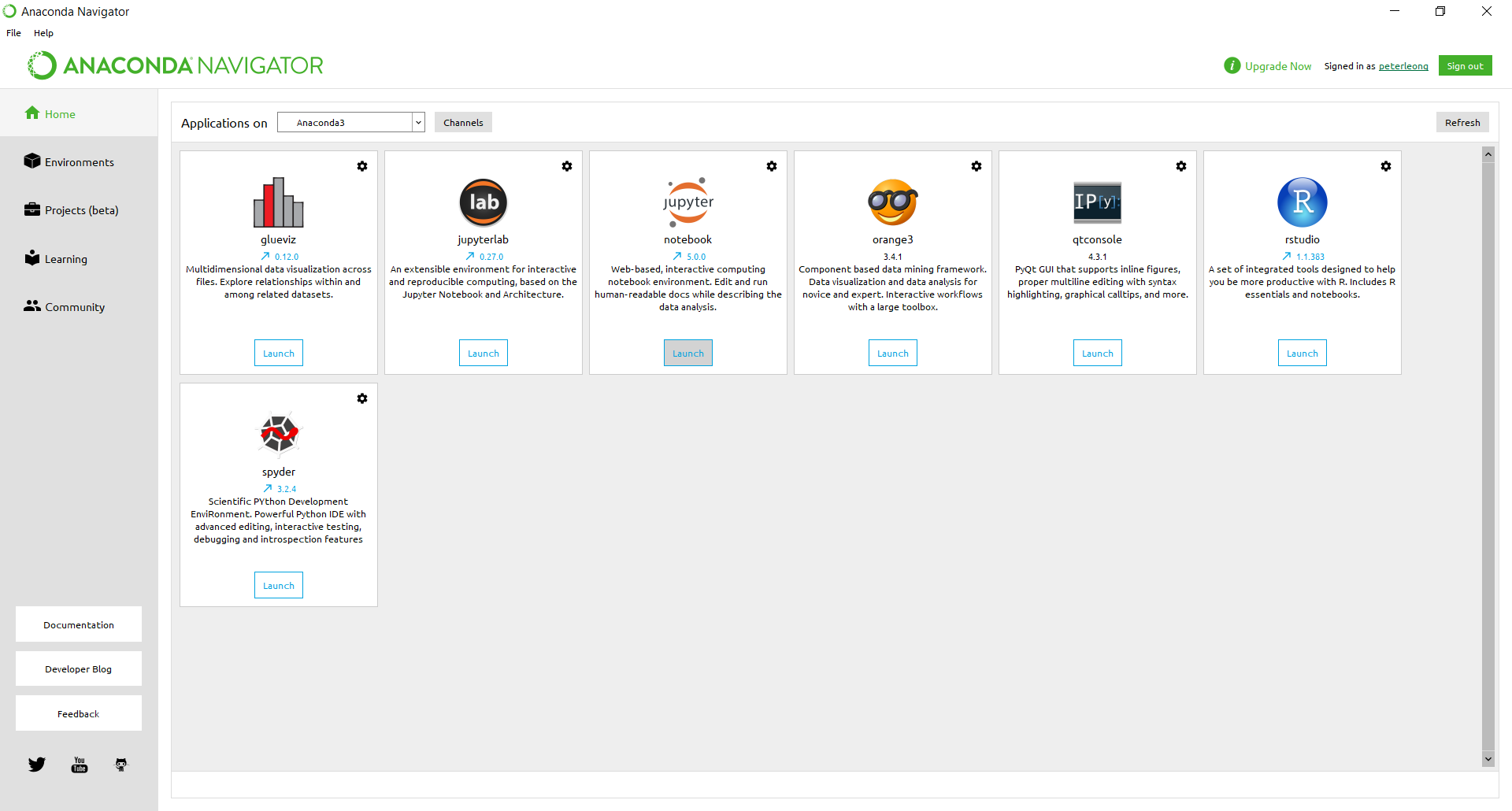
conda update scikit-learn

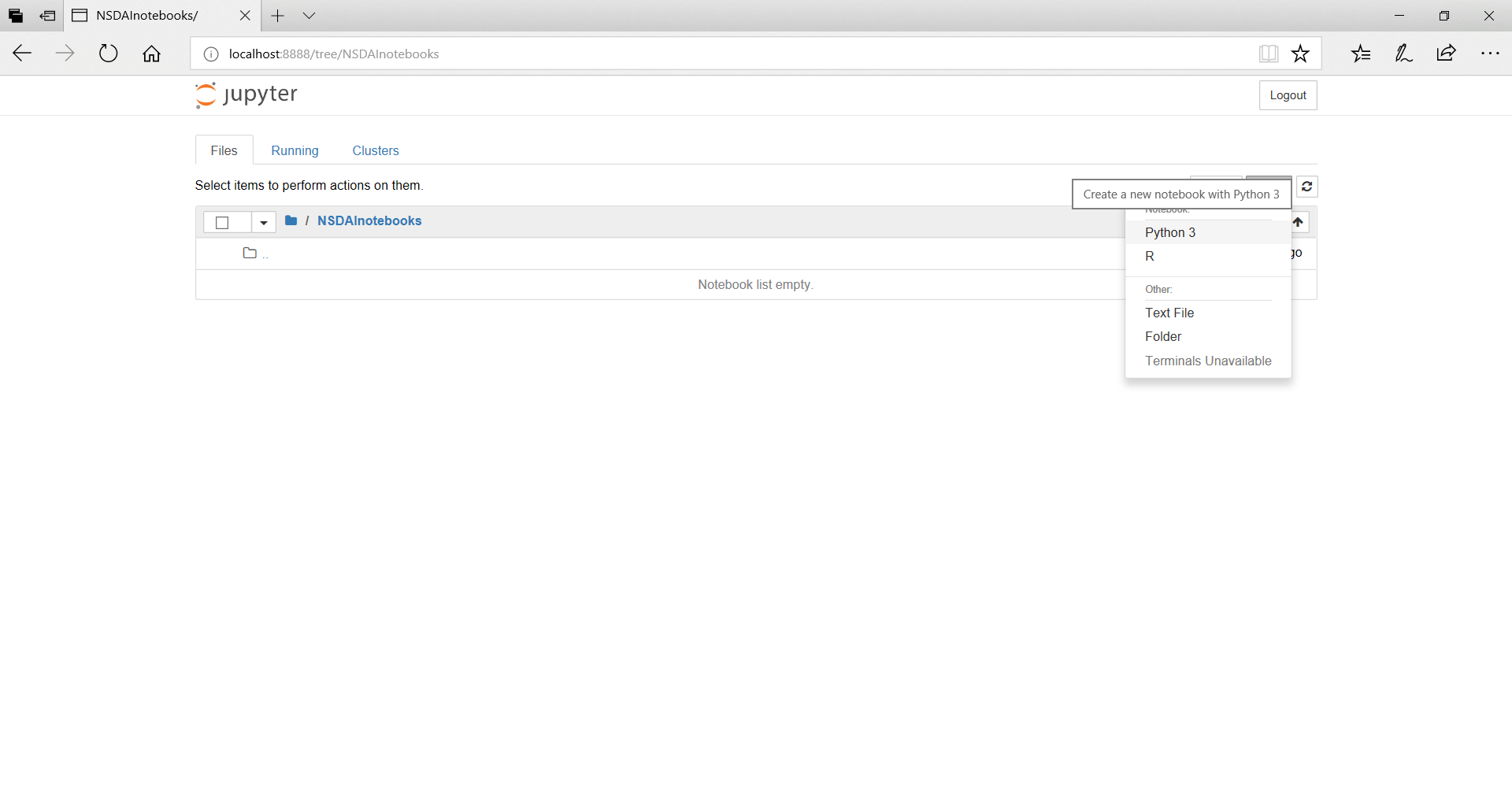
# 2. using Scikit-Learn (SKLEARN)

In this section, we will go through the basics of programming using scikit-learn.

## Using jupyter

You can use scikit-learn within a jupyter notebook. Launch a python notebook from within the Anaconda Naviagator



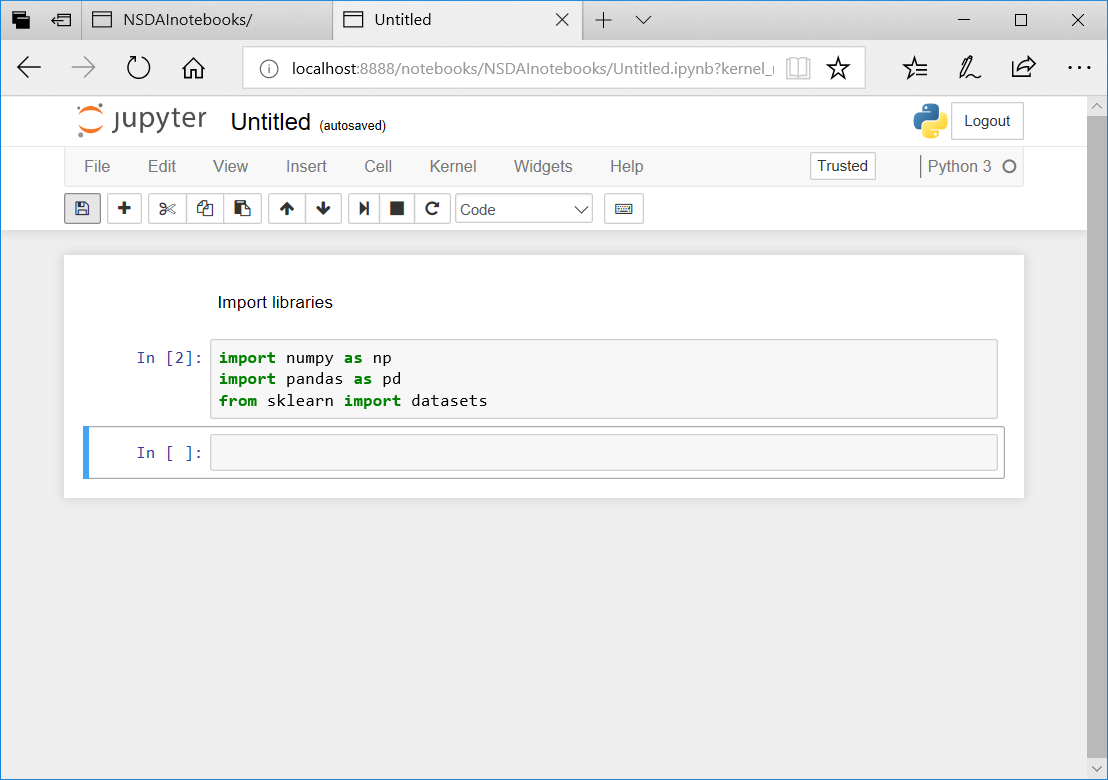


Enter into a cell and run

import numpy as np

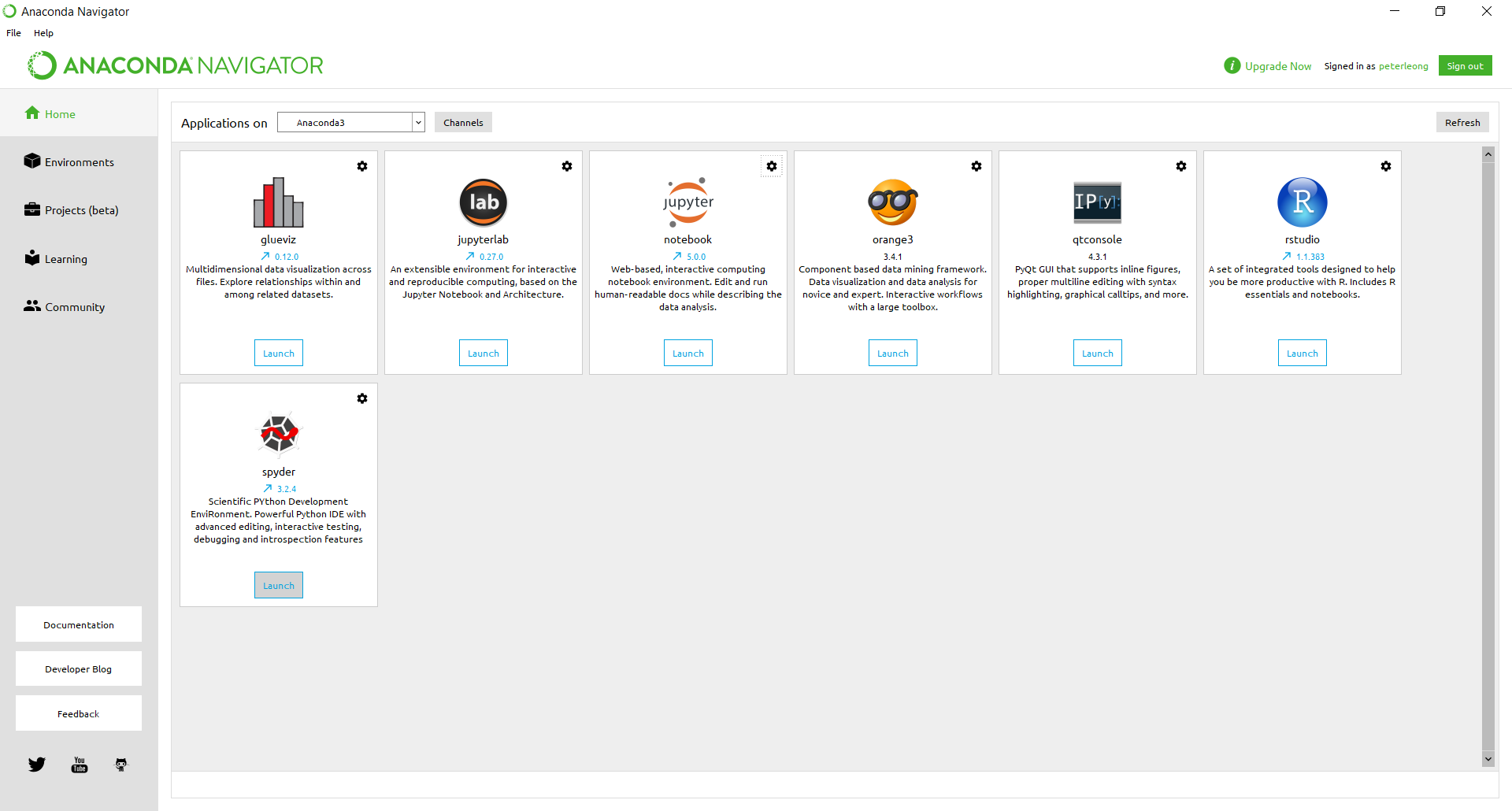
import pandas as pd

from sklearn import datasets



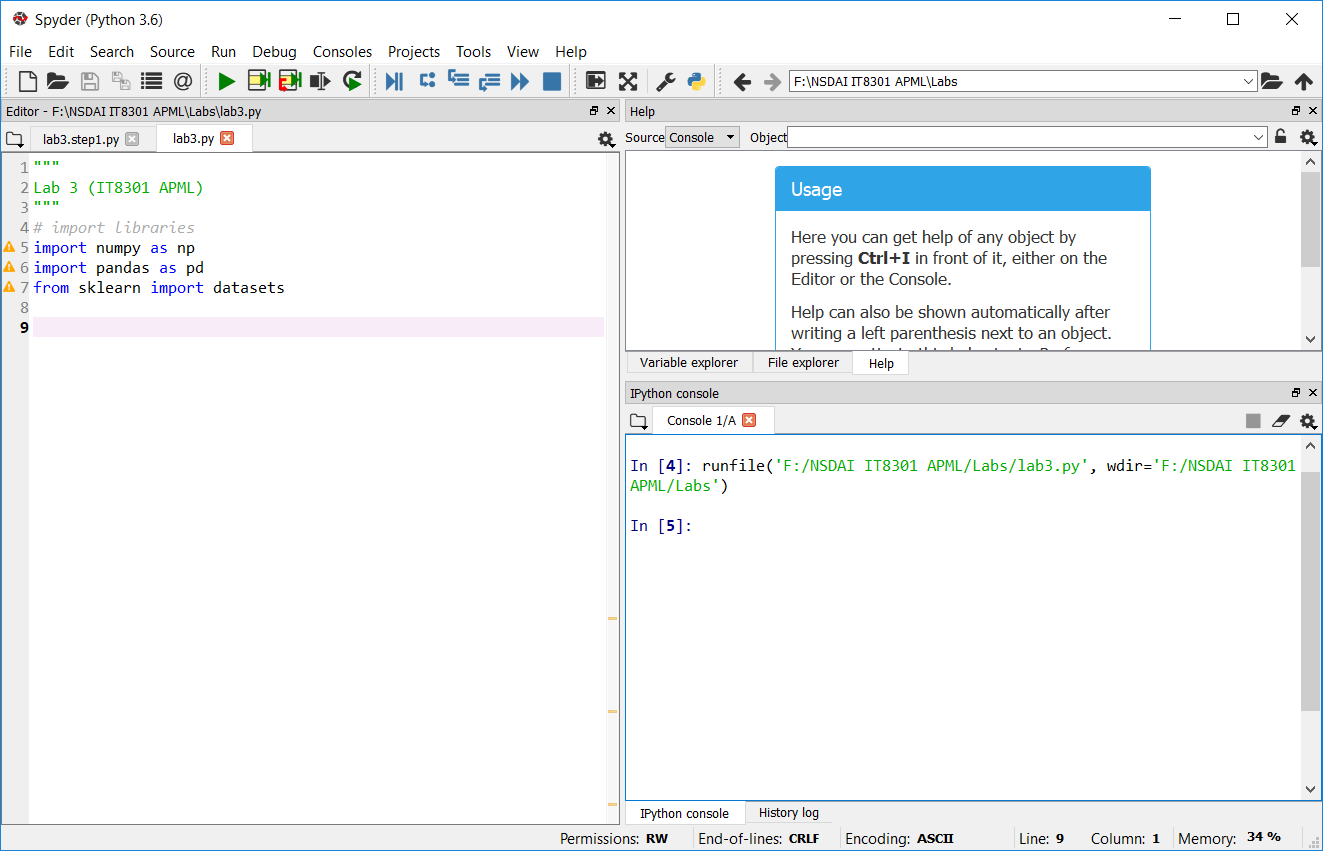
## Using python source file

Launch the Spyder editor from the Anaconda Navigator



Save the file as lab3.py

Enter the same code as before and run



Note: the warning labels at the side are there because we have not actually used the imported modules.

## Data loading and preparation

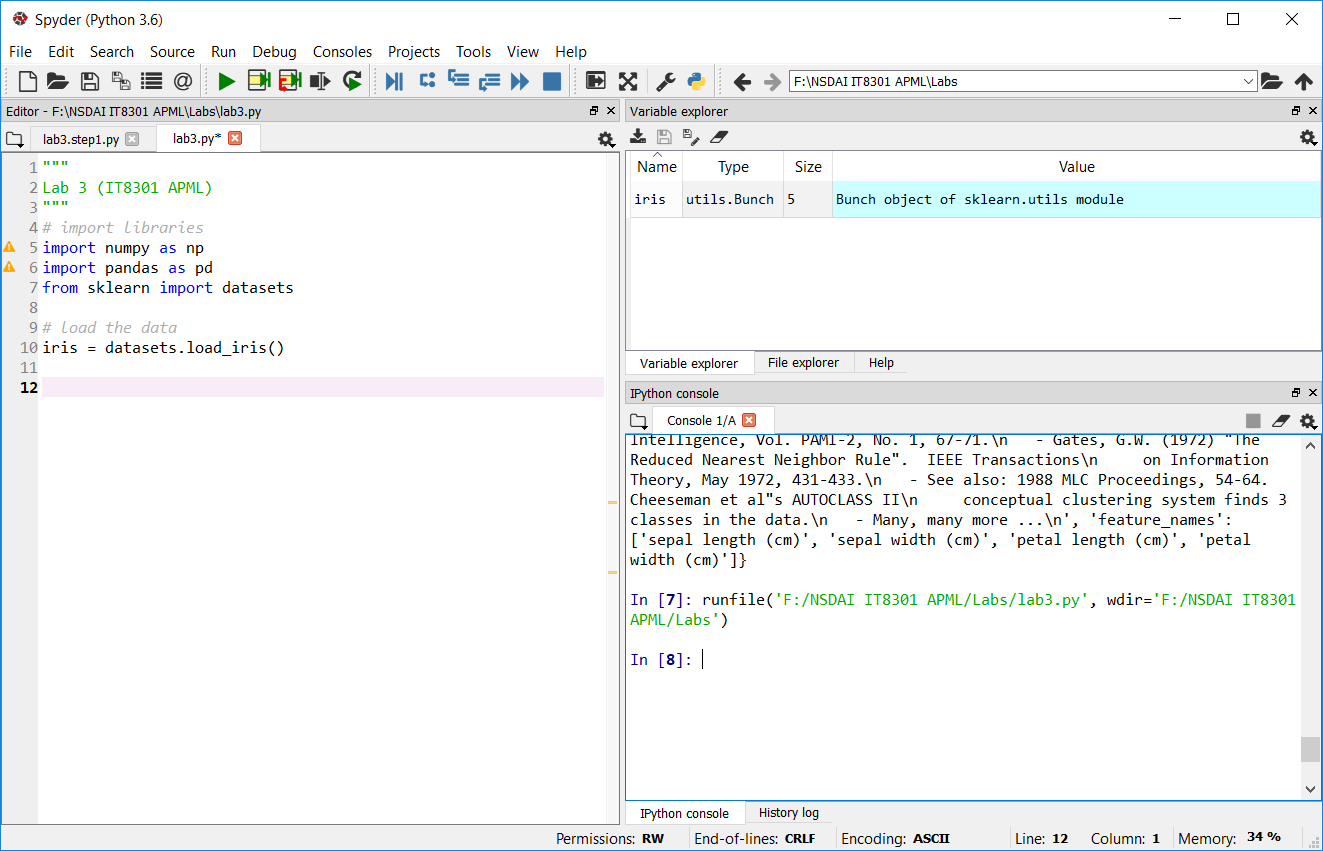
scikit-learn comes with a few standard datasets such as iris dataset.

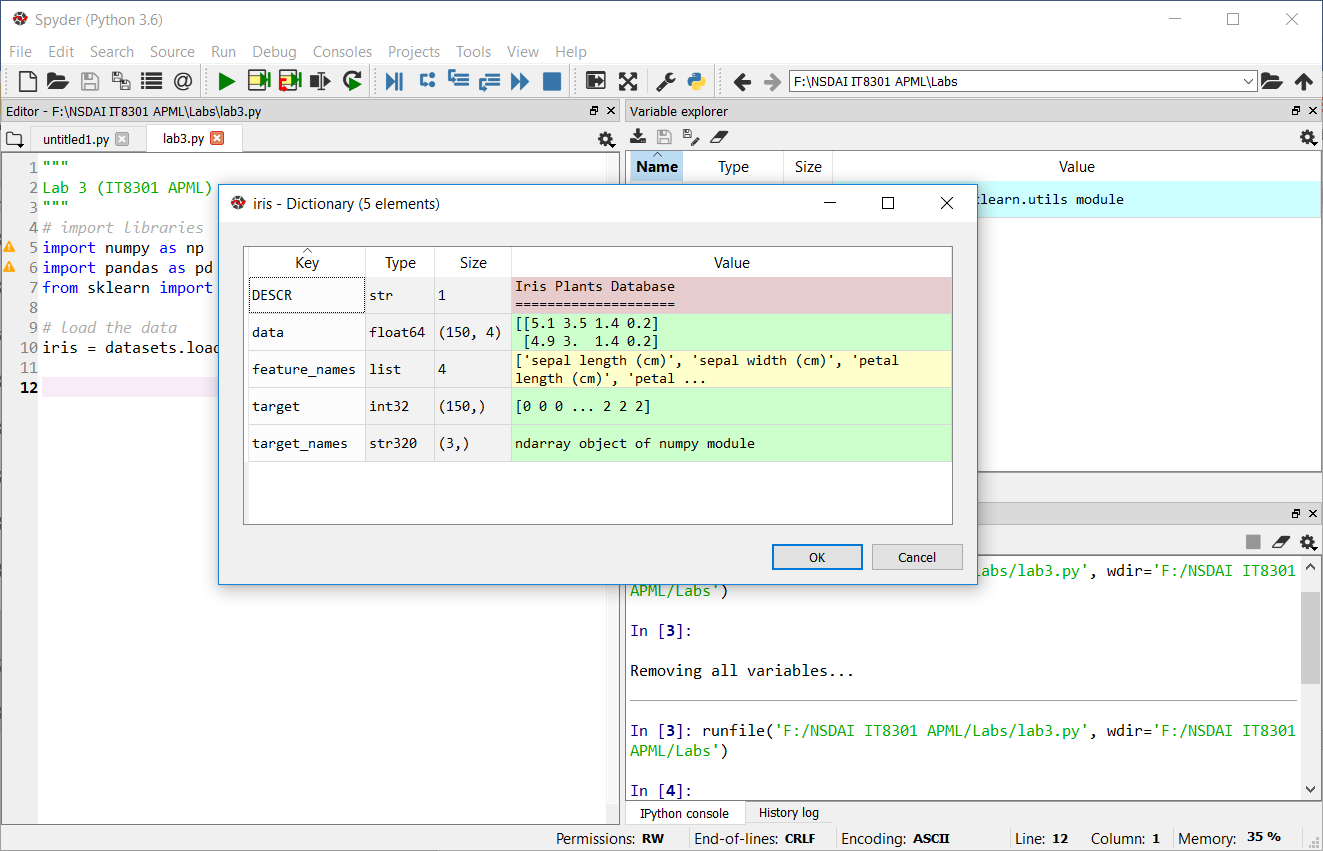
Enter the following code to load the iris dataset.

# load the data

iris = datasets.load\_iris()

Run the code. Use the [Variable explorer] (top-right pane) to view the data. Double-click on the [iris] line in the Variable explorer.





Next we load the data matrix in X and the target vector in y. After that, we split the dataset into a training set and a test set using the train\_test\_split function from sklearn.model\_selection.

# get the data (X) and target (y)

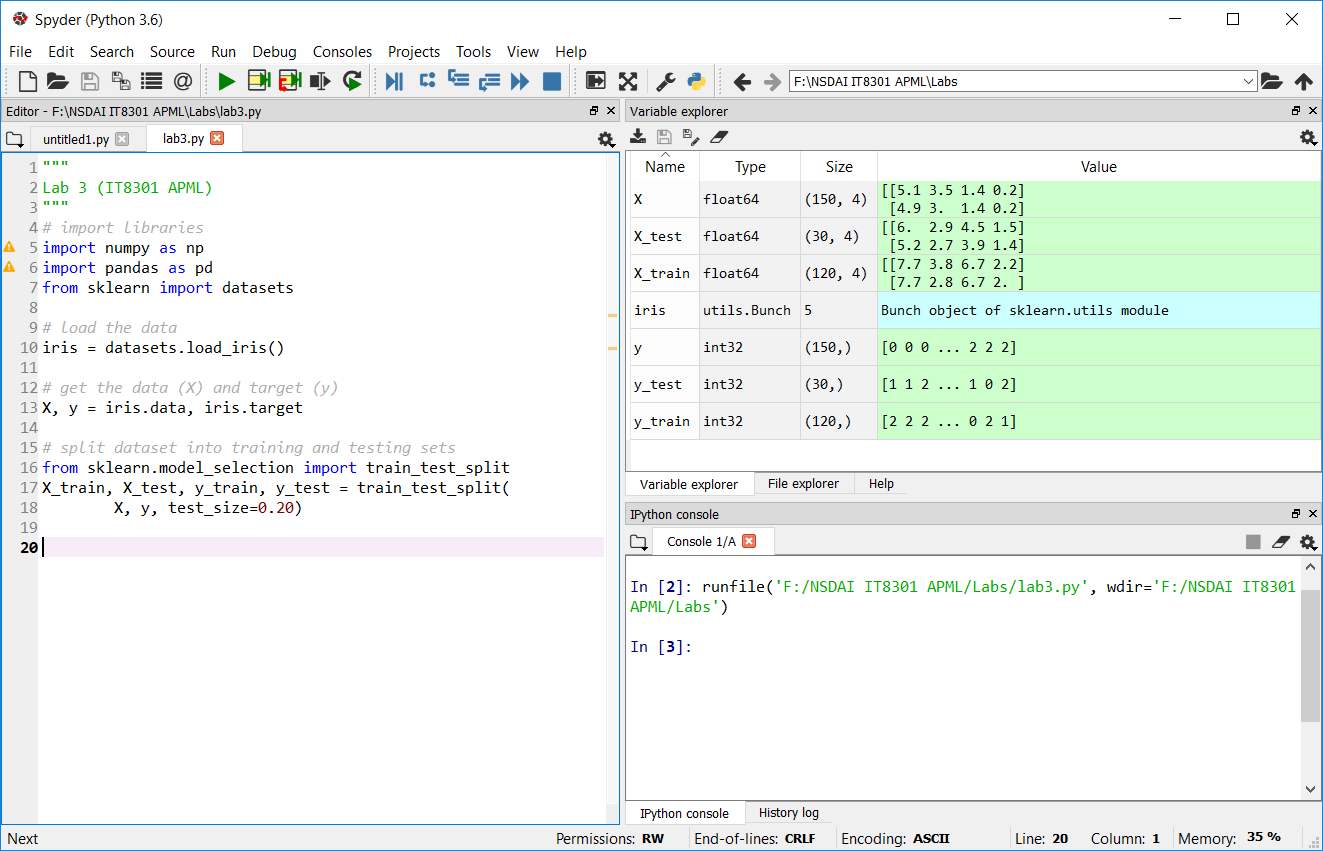
X, y = iris.data, iris.target

# split dataset into training and testing sets

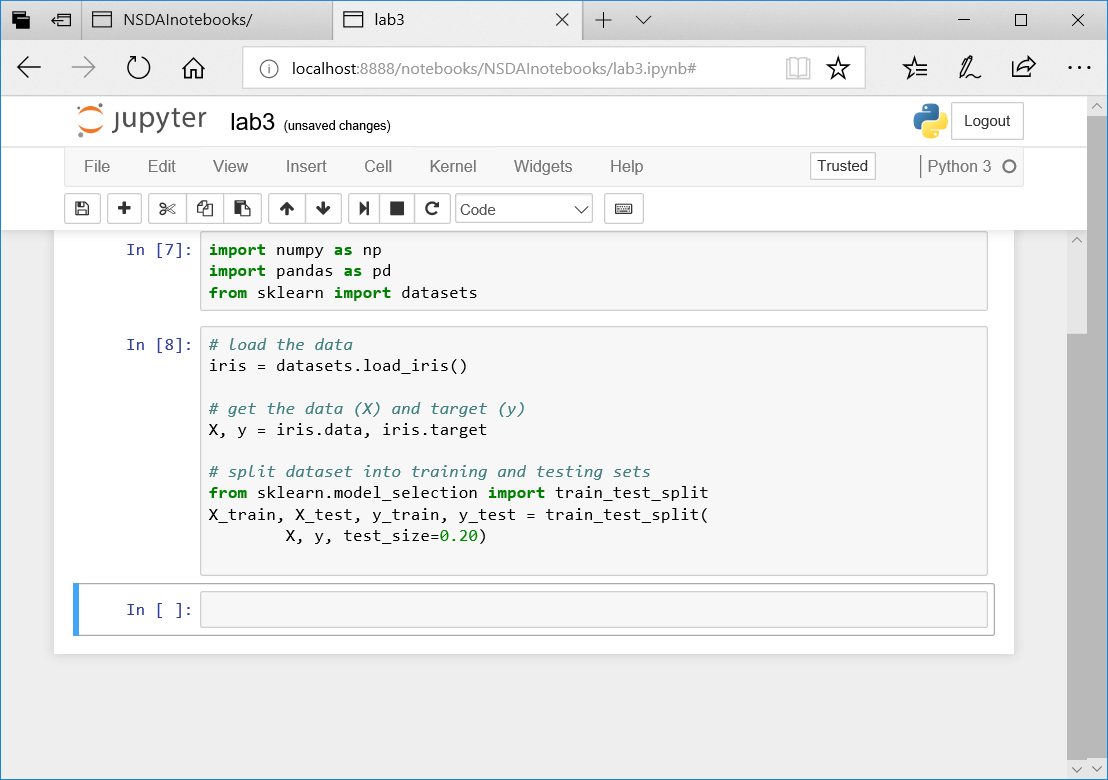
from sklearn.model\_selection import train\_test\_split

X\_train, X\_test, y\_train, y\_test = train\_test\_split(

X, y, test\_size=0.20)



We can perform all the same steps in the jupyter notebook.



# 3. ML Experiment with sKLEARN

We will load the iris data set and perform a similar experiment using a logistic regression classifier in scikit-learn. The main purpose is build a classifier to group a new sample of iris into the correct species.

## Data Preparation

The previous section has shown how we can load the iris data set and split it into a training and test set. We use X for the input data matrix and y for the target (label/output).

## Modelling

In this section we will show how to train the model using scikit-learn LogisticRegression estimator. In scikit-learn, an estimator for classification is a Python object that implements the methods fit(X, y) and predict(T).

# get the LogisticRegression estimator

from sklearn.linear\_model import LogisticRegression

# training the model

# apply algorithm to data using fit()

clf = LogisticRegression(solver='newton-cg',multi\_class='multinomial')

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

Model Persistence

It is possible to save a model in the scikit by using Python’s built-in persistence model, namely pickle:

# We can save the trained model clf using pickle (step 4)

import pickle

s = pickle.dumps(clf)

pickle.dump( clf, open( "iris\_trained\_model.p", "wb" ) )

clf2 = pickle.load( open( "iris\_trained\_model.p", "rb" ) )

# Alternatively, we can save the trained model clf with sklearn joblib

from sklearn.externals import joblib

joblib.dump(clf, 'iris\_trained\_model.pkl')

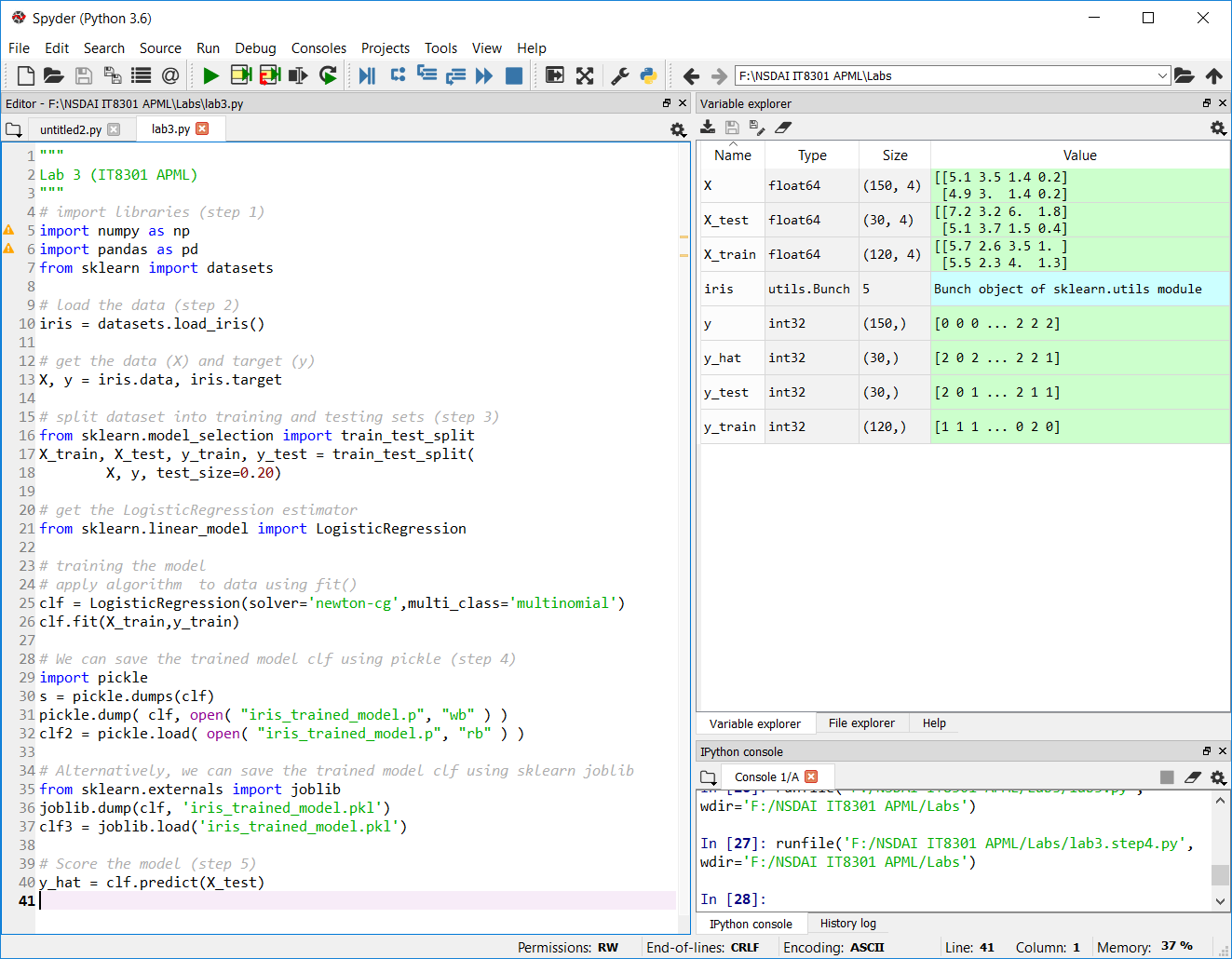
clf3 = joblib.load('iris\_trained\_model.pkl')

## Scoring/Predicting

Scoring the model is using the trained model to score/predict the target value of new data or test data.

# Score the model (step 5)

y\_hat = clf.predict(X\_test)



## Evaluation

The last step is to evaluate the performance of the model for the training and test set using the confusion matrix. (The target of the dataset is not binomial hence, we cannot use the ROC curve to evaluate it).

# Evaluation of the model (step 6)

from sklearn.metrics import confusion\_matrix

cm = confusion\_matrix(y\_test,y\_hat)

print(cm)