IT8302 APPLIED 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*

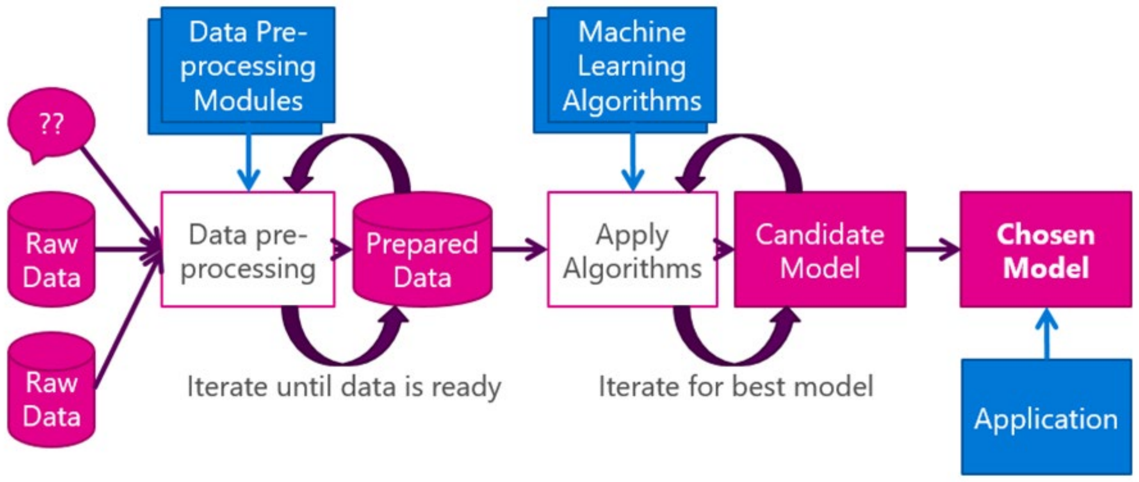
TABLE OF CONTENTS

1. OVERVIEW

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#

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)

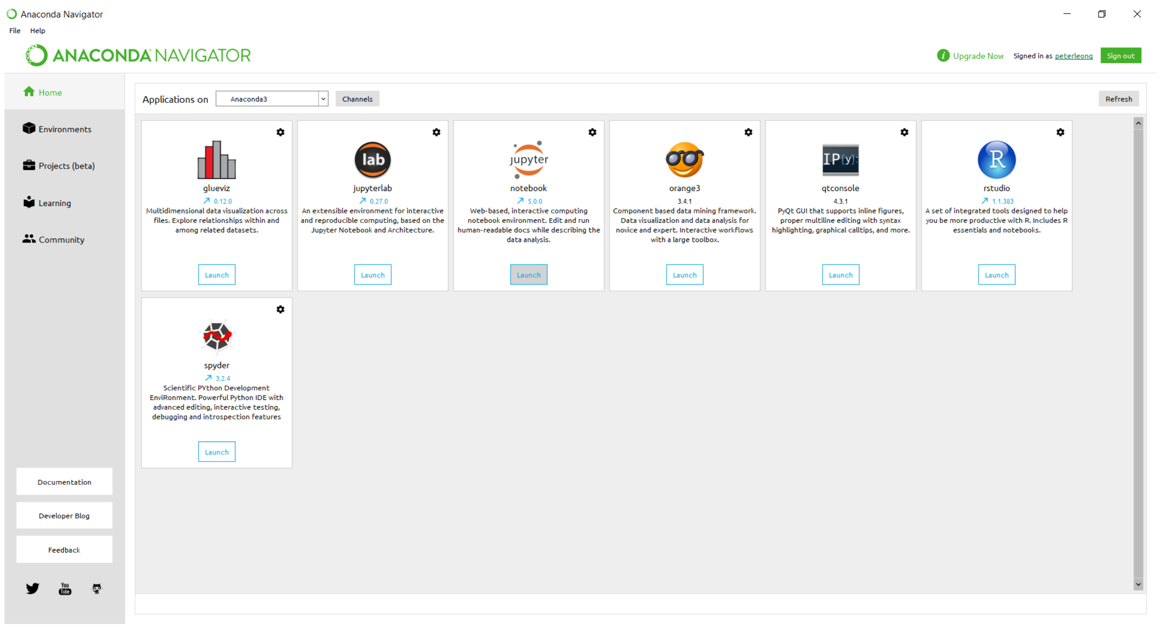
```
conda update scikit-learn
```

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



glueviz
0.12.0
Multidimensional data visualization across files. Explore relationships within and among related datasets.

jupyterlab
0.27.0
An extensible environment for interactive and reproducible computing, based on the Jupyter Notebook and Architecture.

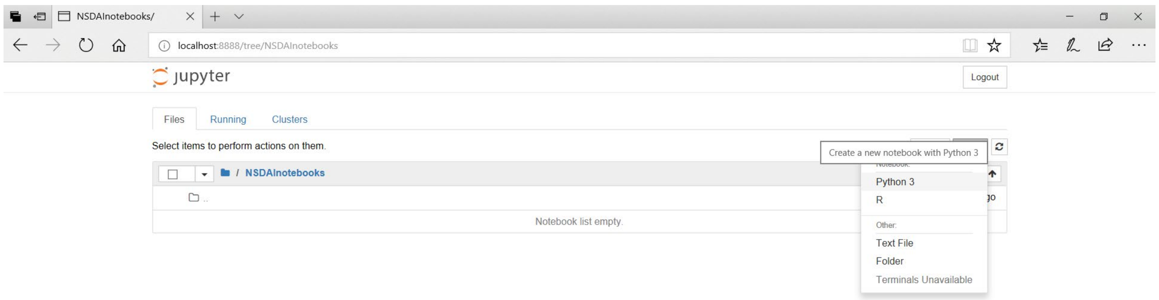
notebook
5.0.0
Web-based, interactive computing notebook environment. Edit and run human-readable docs while describing the data analysis.

orange3
3.4.1
Component based data mining framework. Data visualization and data analysis for novice and expert. Interactive workflows with a large toolbox.

qtconsole
4.3.1
PyQt GUI that supports inline figures, proper multiline editing with syntax highlighting, graphical calltips, and more.

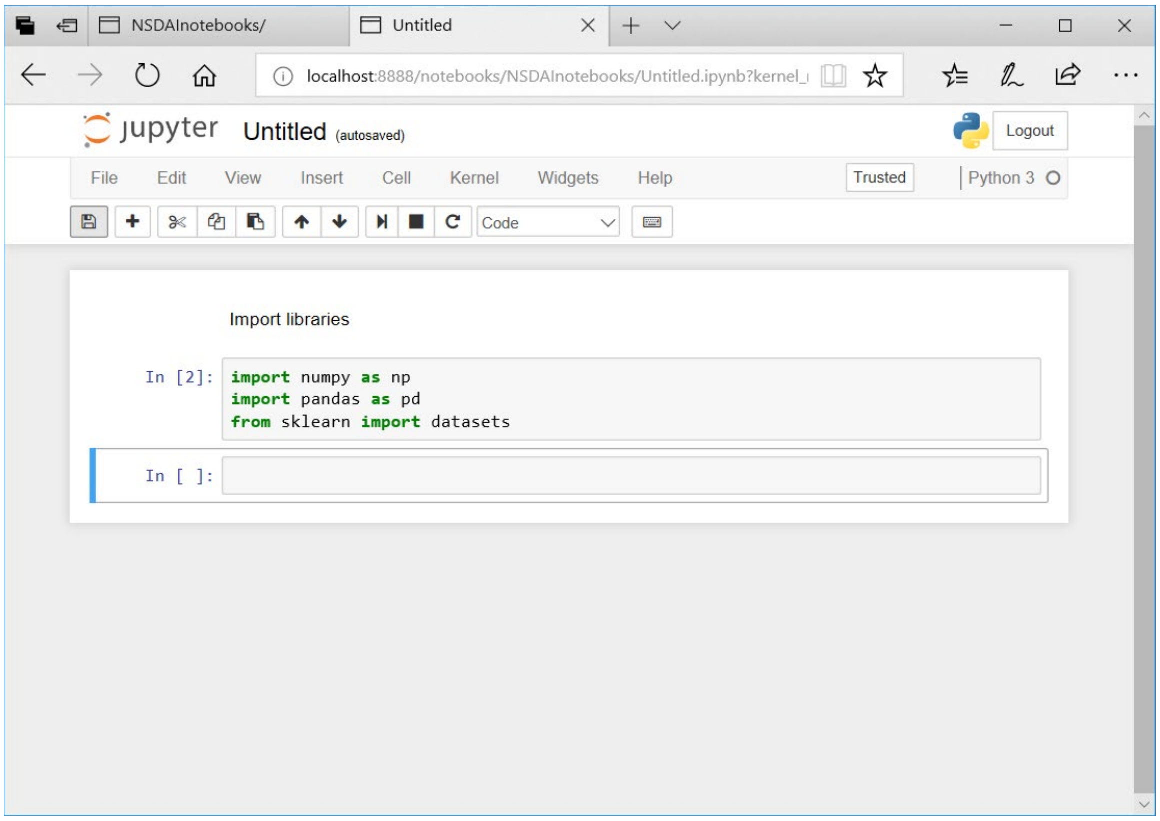
rstudio
1.1.383
A set of integrated tools designed to help you be more productive with R. Includes R essentials and notebooks.

spyder
3.2.4
Scientific PYthon Development EnviRonment. Powerful Python IDE with advanced editing, interactive testing, debugging and introspection features



Enter into a cell and run

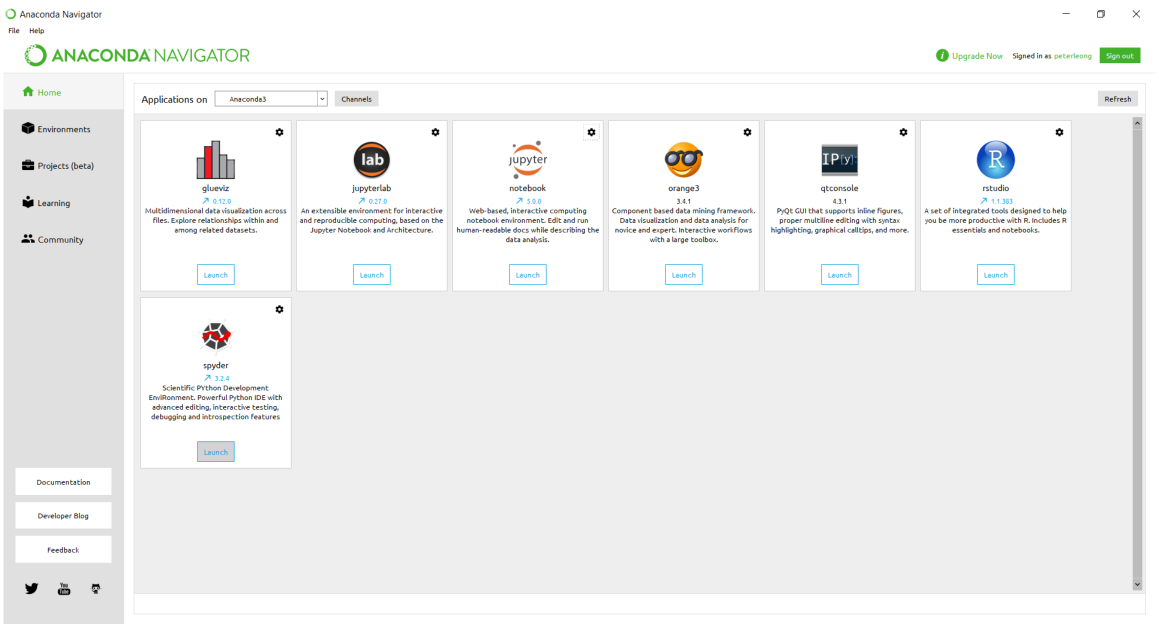
```
import numpy as np
import pandas as pd
from sklearn import datasets
```



```
In [2]: import numpy as np
        import pandas as pd
        from sklearn import datasets

In [ ]:
```

USING PYTHON SOURCE FILE

Launch the Spyder editor from the Anaconda Navigator



glueviz
0.12.0
Multidimensional data visualization across files. Explore relationships within and among related datasets.

jupyterlab
0.27.0
An extensible environment for interactive and reproducible computing, based on the Jupyter Notebook and Architecture.

notebook
5.0.0
Web-based, interactive computing notebook environment. Edit and run human-readable docs while describing the data analysis.

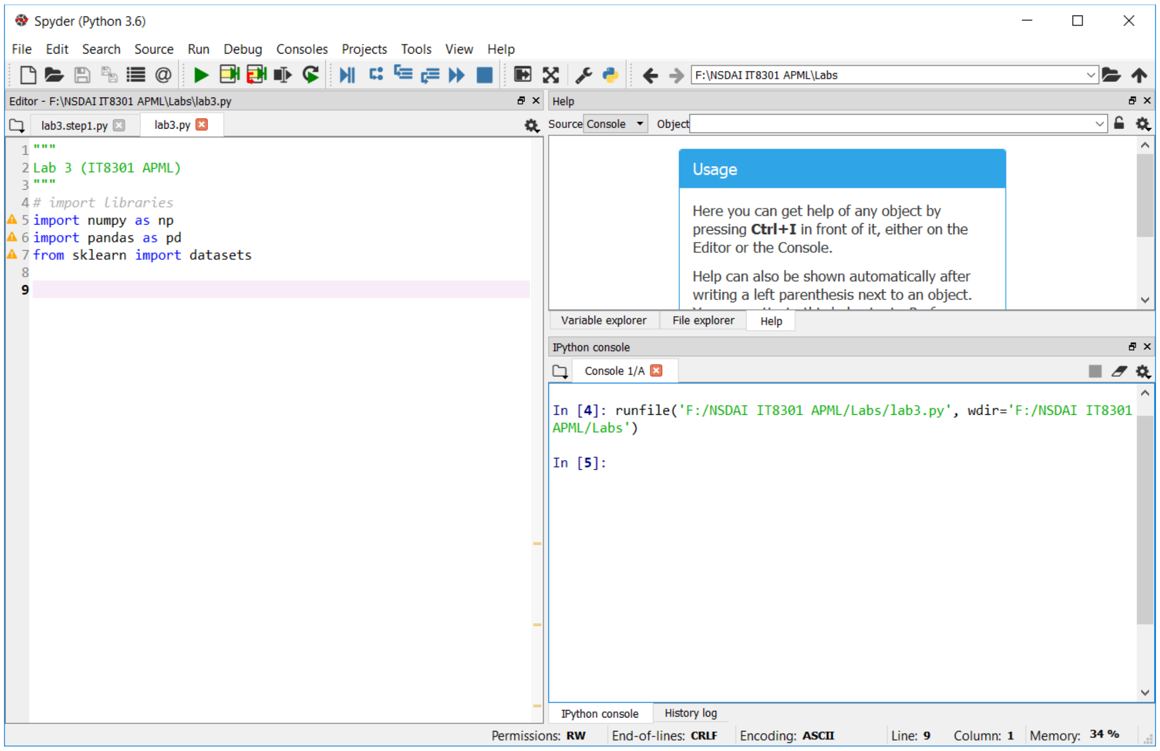
orange3
3.4.1
Component based data mining framework. Data visualization and data analysis for novice and expert. Interactive workflows with a large toolbox.

qtconsole
4.3.1
PyQt GUI that supports inline figures, proper multiline editing with syntax highlighting, graphical calltips, and more.

rstudio
1.1.383
A set of integrated tools designed to help you be more productive with R. Includes R essentials and notebooks.

spyder
3.2.4
Scientific PYthon Development EnviRonment. Powerful Python IDE with advanced editing, interactive testing, debugging and introspection features

Save the file as lab3.py
Enter the same code as before and run



```
"""
Lab 3 (IT8301 APML)
"""
# import libraries
import numpy as np
import pandas as pd
from sklearn import datasets

```

Usage

Here you can get help of any object by pressing **Ctrl+I** in front of it, either on the Editor or the Console.

Help can also be shown automatically after writing a left parenthesis next to an object.

```
In [4]: runfile('F:/NSDAI IT8301 APML/Labs/lab3.py', wdir='F:/NSDAI IT8301
APML/Labs')

In [5]:
```

Permissions: **RW** End-of-lines: **CRLF** Encoding: **ASCII** Line: **9** Column: **1** Memory: **34 %**

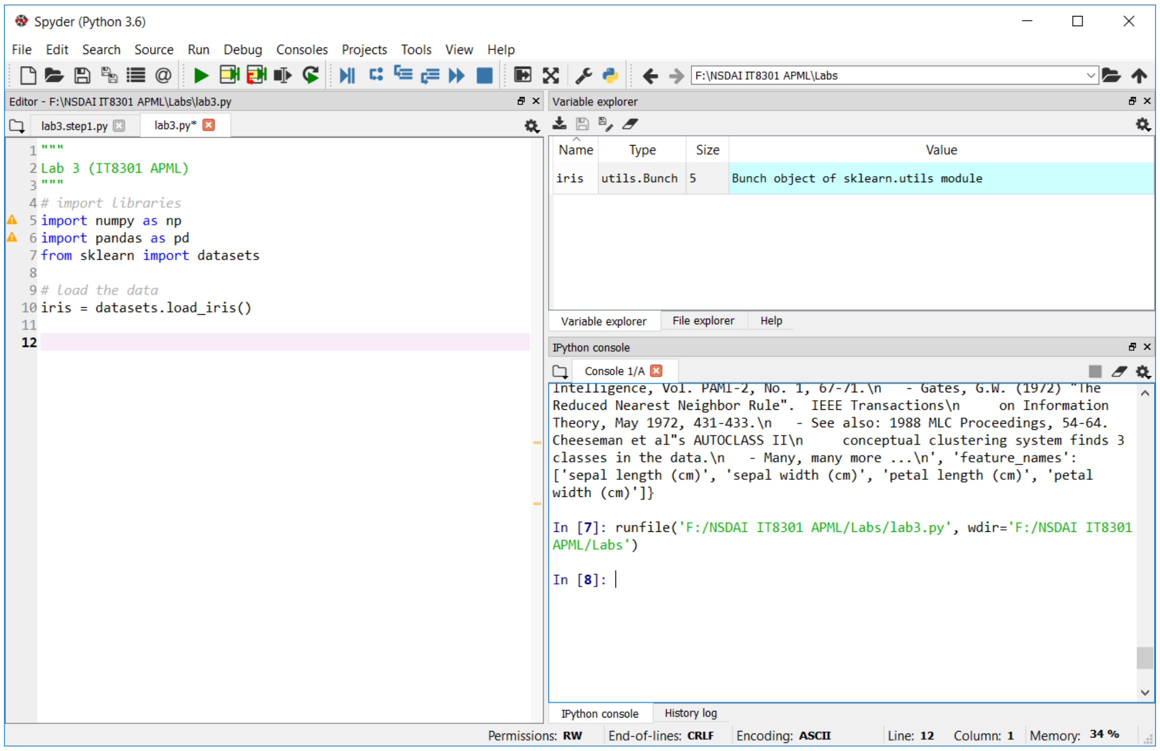
Note: the warning labels at the side are there because we have not actually used the imported modules.

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.

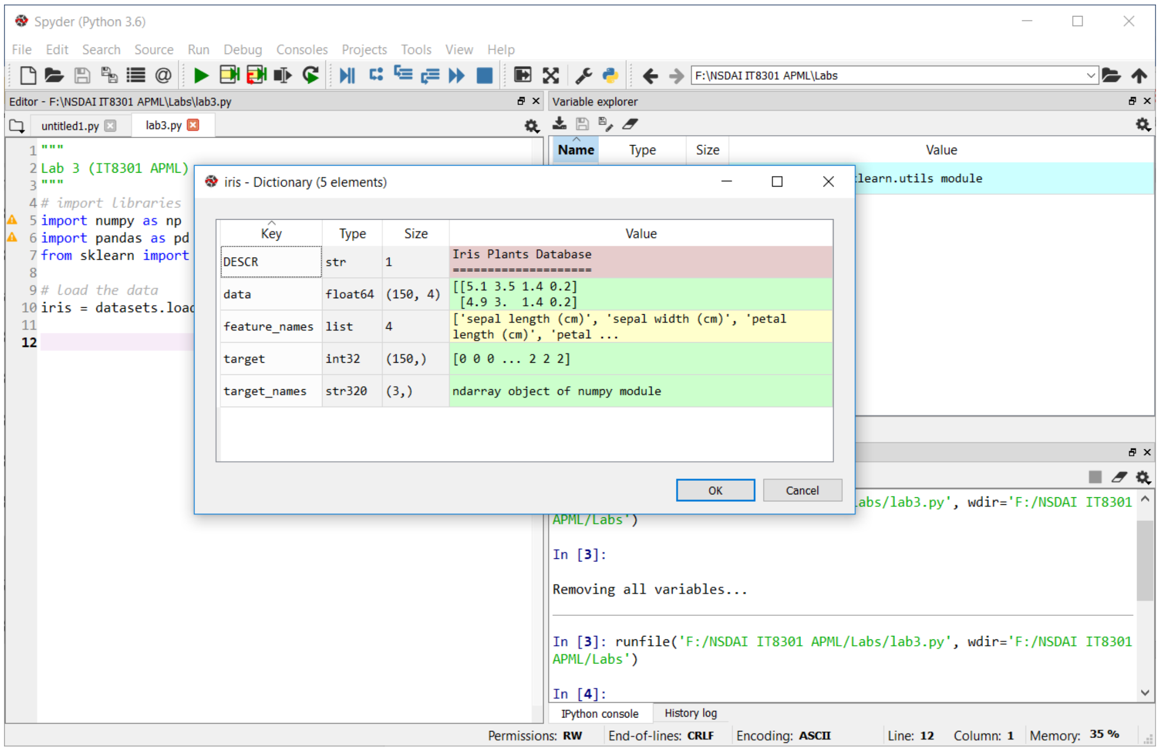


```
"""
Lab 3 (IT8301 APML)
"""
# import libraries
import numpy as np
import pandas as pd
from sklearn import datasets

# load the data
iris = datasets.load_iris()

```

Name	Type	Size	Value
iris	utils.Bunch	5	Bunch object of sklearn.utils module

```
Intelligence, Vol. PAMI-2, No. 1, 67-71.\n   - Gates, G.W. (1972) "The
Reduced Nearest Neighbor Rule".  IEEE Transactions\n     on Information
Theory, May 1972, 431-433.\n   - See also: 1988 MLC Proceedings, 54-64.
Cheeseman et al"s AUTOCLASS II\n     conceptual clustering system finds 3
classes in the data.\n   - Many, many more ...\n', 'feature_names':
['sepal length (cm)', 'sepal width (cm)', 'petal length (cm)', 'petal
width (cm)']}

In [7]: runfile('F:/NSDAI IT8301 APML/Labs/lab3.py', wdir='F:/NSDAI IT8301
APML/Labs')

In [8]:
```

Permissions: RW End-of-lines: CRLF Encoding: ASCII Line: 12 Column: 1 Memory: 34 %



Editor - F:\NSDAI IT8301 APML\Labs\lab3.py

```
"""
Lab 3 (IT8301 APML)
"""
# import libraries
import numpy as np
import pandas as pd
from sklearn import

# load the data
iris = datasets.load
```

iris - Dictionary (5 elements)

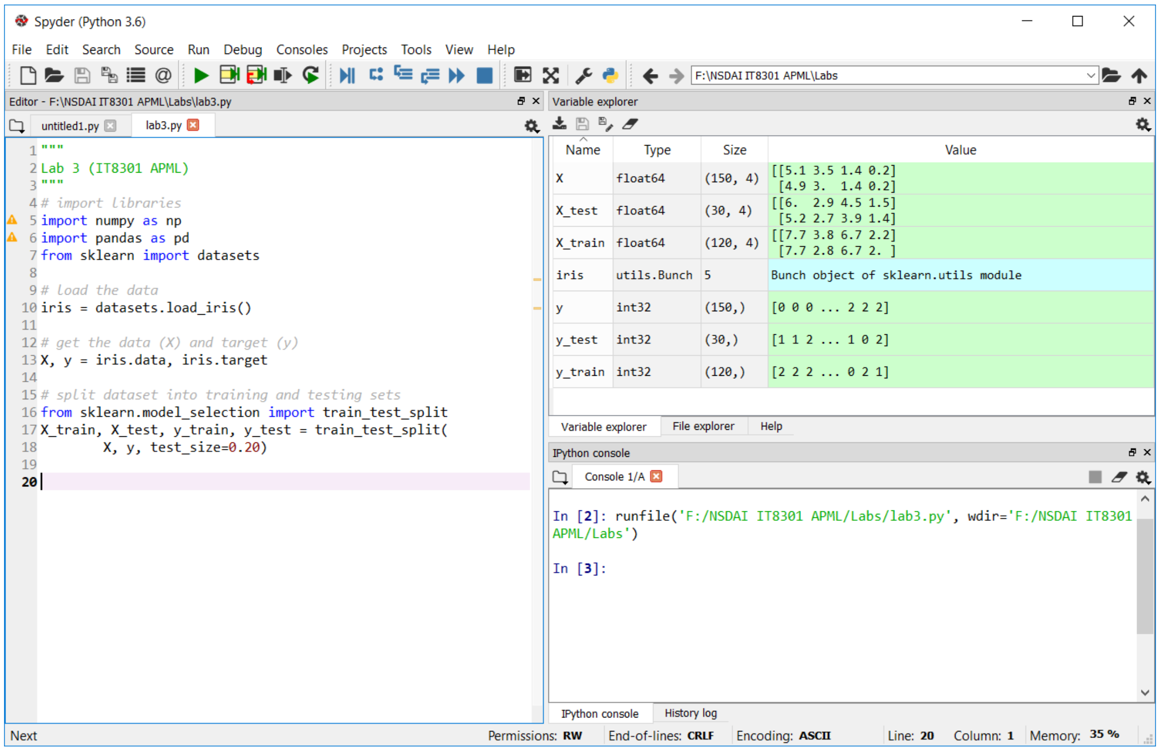
Key	Type	Size	Value
DESCR	str	1	Iris Plants Database ====================
data	float64	(150, 4)	[[5.1 3.5 1.4 0.2] [4.9 3. 1.4 0.2]
feature_names	list	4	['sepal length (cm)', 'sepal width (cm)', 'petal length (cm)', 'petal ...
target	int32	(150,)	[0 0 0 ... 2 2 2]
target_names	str320	(3,)	ndarray object of numpy module

```
abs/lab3.py', wdir='F:/NSDAI IT8301
APML/Labs')

In [3]:

Removing all variables...

In [3]: runfile('F:/NSDAI IT8301 APML/Labs/lab3.py', wdir='F:/NSDAI IT8301
APML/Labs')

In [4]:
```

Permissions: RW End-of-lines: CRLF Encoding: ASCII Line: 12 Column: 1 Memory: 35 %

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)
```

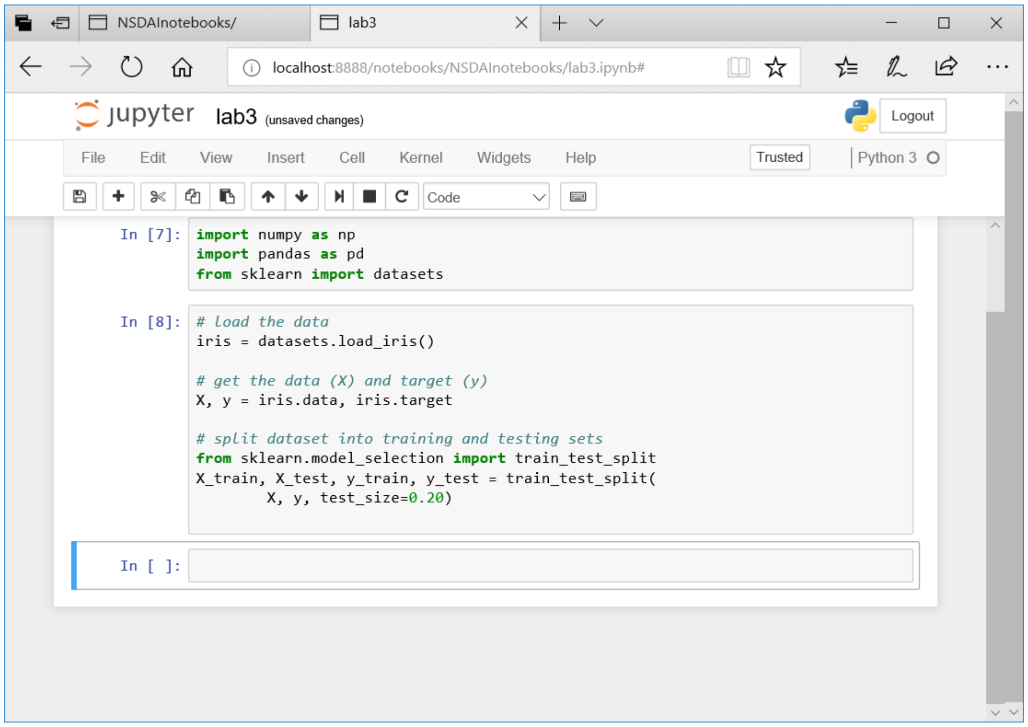

Spyder (Python 3.6)
File Edit Search Source Run Debug Consoles Projects Tools View Help
F:\NSDAI IT8301 APML\Labs
Editor - F:\NSDAI IT8301 APML\Labs\lab3.py
untitled1.py
lab3.py
Variable explorer
Variable explorer File explorer Help
IPython console
Console 1/A
IPython console History log
Next Permissions: RW End-of-lines: CRLF Encoding: ASCII Line: 20 Column: 1 Memory: 35 %

```
"""
Lab 3 (IT8301 APML)
"""
# import libraries
import numpy as np
import pandas as pd
from sklearn import datasets

# load the data
iris = datasets.load_iris()

# 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)

```

Name	Type	Size	Value
X	float64	(150, 4)	[[5.1 3.5 1.4 0.2] [4.9 3. 1.4 0.2]
X_test	float64	(30, 4)	[[6. 2.9 4.5 1.5] [5.2 2.7 3.9 1.4]
X_train	float64	(120, 4)	[[7.7 3.8 6.7 2.2] [7.7 2.8 6.7 2.]
iris	utils.Bunch	5	Bunch object of sklearn.utils module
y	int32	(150,)	[0 0 0 ... 2 2 2]
y_test	int32	(30,)	[1 1 2 ... 1 0 2]
y_train	int32	(120,)	[2 2 2 ... 0 2 1]

```
In [2]: runfile('F:/NSDAI IT8301 APML/Labs/lab3.py', wdir='F:/NSDAI IT8301
APML/Labs')

In [3]:
```

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


NSDAInotebooks/
lab3
localhost:8888/notebooks/NSDAInotebooks/lab3.ipynb#
jupyter lab3 (unsaved changes)
Logout
File Edit View Insert Cell Kernel Widgets Help
Trusted
Python 3
Code

```
In [7]: import numpy as np
        import pandas as pd
        from sklearn import datasets

In [8]: # load the data
        iris = datasets.load_iris()

        # 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)

In [ ]:
```

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

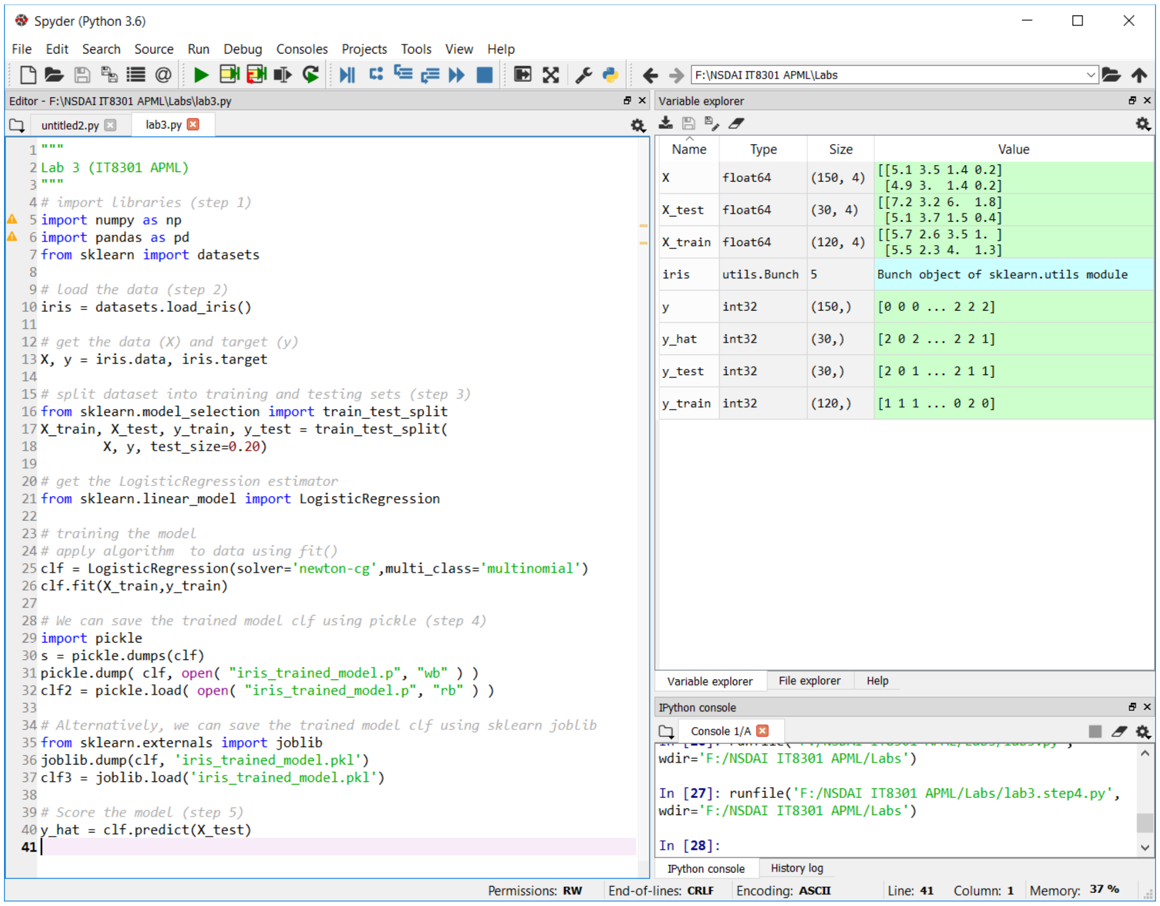
It is possible to save a model in the scikit by using Python’s built-in persistence model, namely `pickle`:

```
# We can save the trained model clf using pickle (step 4)
import pickle
pickle.dump(clf, open("iris_trained_model.p", "wb" ) )
clf2 = pickle.load(open("iris_trained_model.p", "rb" ) )
```

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)
```



```python
"""
Lab 3 (IT8301 APML)
"""
# import libraries (step 1)
import numpy as np
import pandas as pd
from sklearn import datasets

# load the data (step 2)
iris = datasets.load_iris()

# get the data (X) and target (y)
X, y = iris.data, iris.target

# split dataset into training and testing sets (step 3)
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)

# 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)

# 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 using sklearn joblib
from sklearn.externals import joblib
joblib.dump(clf, 'iris_trained_model.pkl')
clf3 = joblib.load('iris_trained_model.pkl')

# Score the model (step 5)
y_hat = clf.predict(X_test)

```

Name	Type	Size	Value
X	float64	(150, 4)	[[5.1 3.5 1.4 0.2] [4.9 3. 1.4 0.2]
X_test	float64	(30, 4)	[[7.2 3.2 6. 1.8] [5.1 3.7 1.5 0.4]
X_train	float64	(120, 4)	[[5.7 2.6 3.5 1.] [5.5 2.3 4. 1.3]
iris	utils.Bunch	5	Bunch object of sklearn.utils module
y	int32	(150,)	[0 0 0 ... 2 2 2]
y_hat	int32	(30,)	[2 0 2 ... 2 2 1]
y_test	int32	(30,)	[2 0 1 ... 2 1 1]
y_train	int32	(120,)	[1 1 1 ... 0 2 0]

```
wdir='F:/NSDAI IT8301 APML/Labs')

In [27]: runfile('F:/NSDAI IT8301 APML/Labs/lab3.step4.py',
wdir='F:/NSDAI IT8301 APML/Labs')

In [28]:
```

Permissions: **RW** End-of-lines: **CRLF** Encoding: **ASCII** Line: **41** Column: **1** Memory: **37 %**

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).

```python
# Evaluation of the model (step 6)
from sklearn.metrics import confusion_matrix
cm = confusion_matrix(y_test,y_hat)
print(cm)
```