

ST1511

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 |

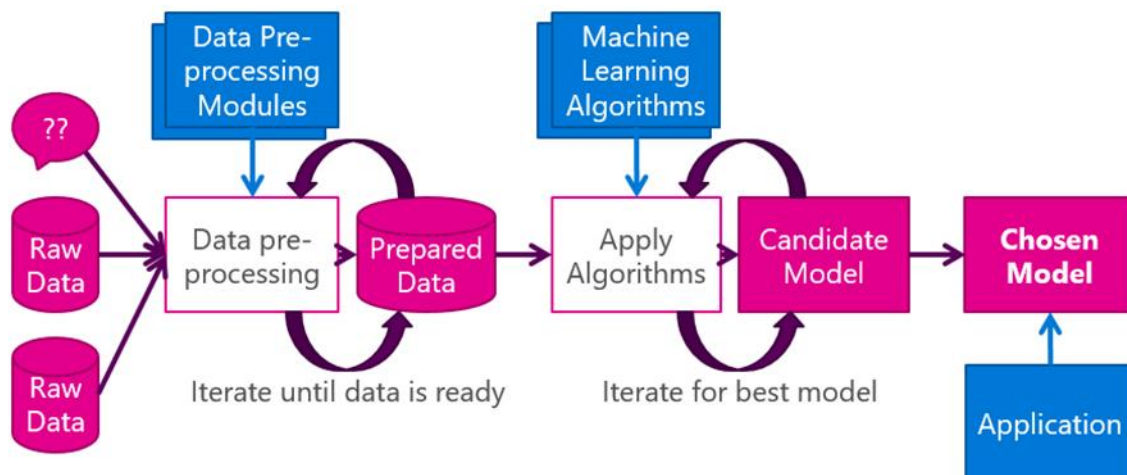
1.

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#>

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
```

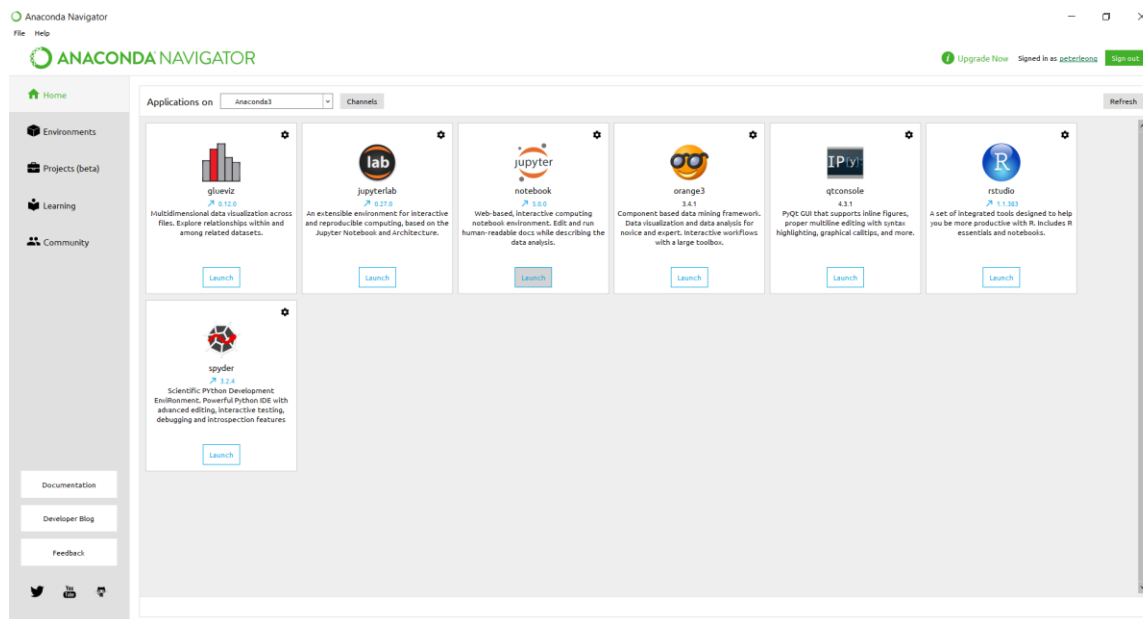
2.

USING SCIKIT-LEARN (SKLEARN)

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

USING JUPYTER

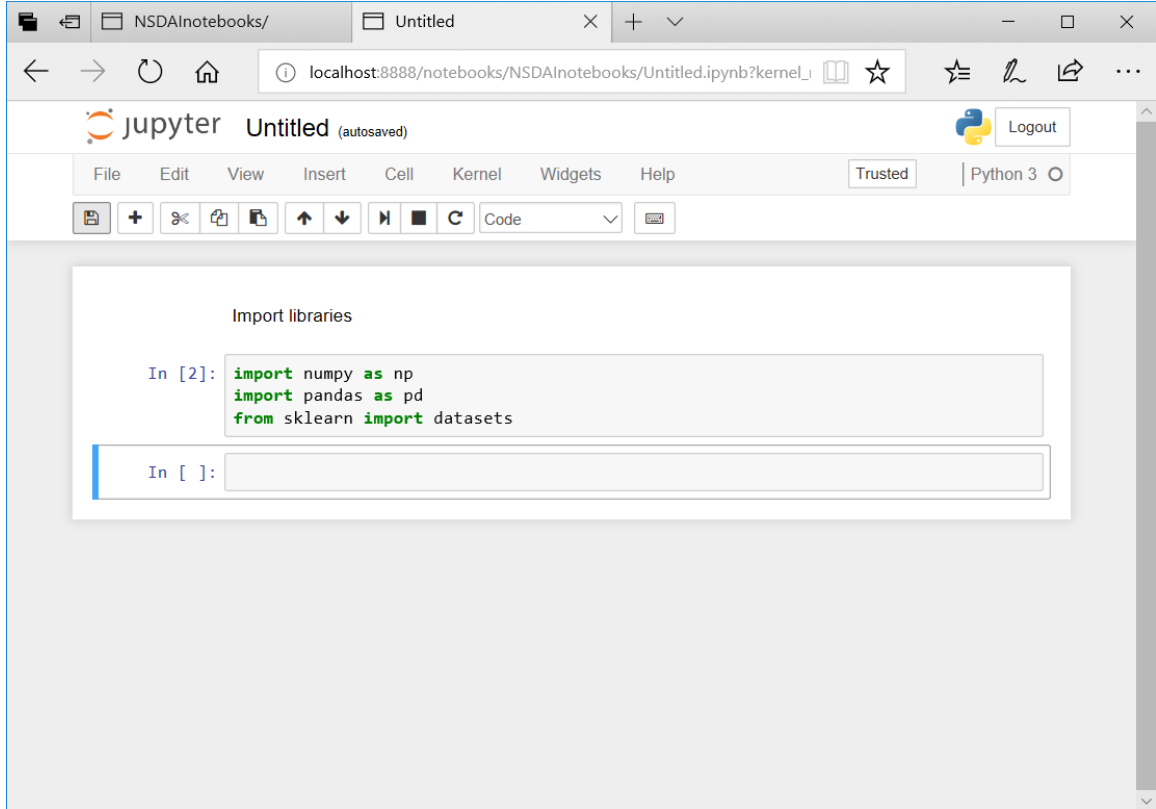
You can use scikit-learn within a jupyter notebook. Launch a python notebook from within the Anaconda Navigator





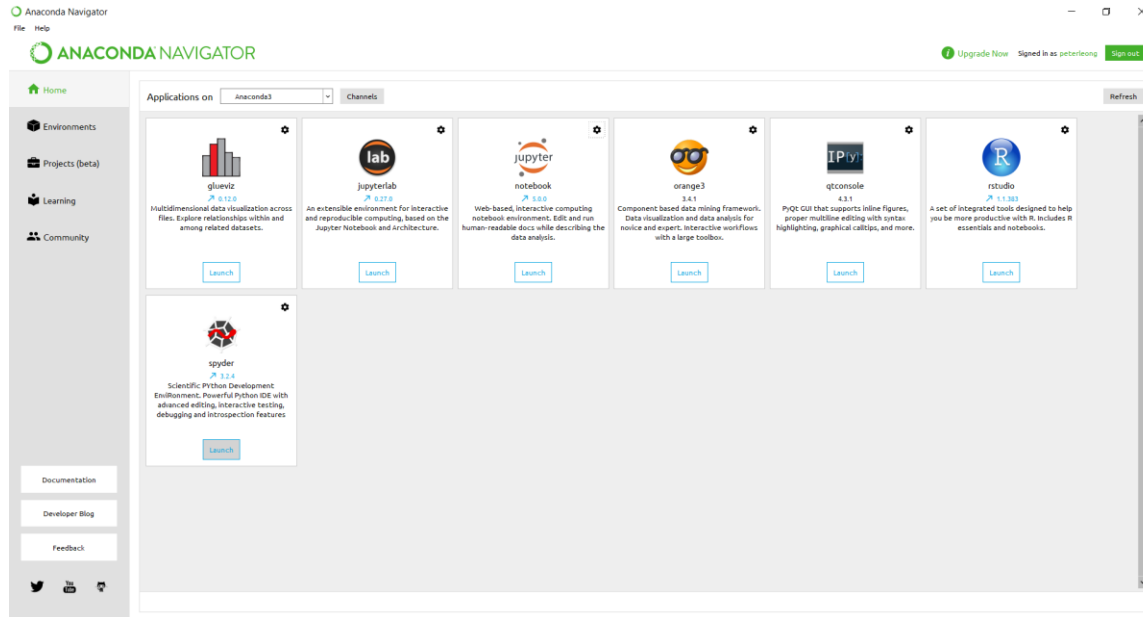
Enter into a cell and run

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

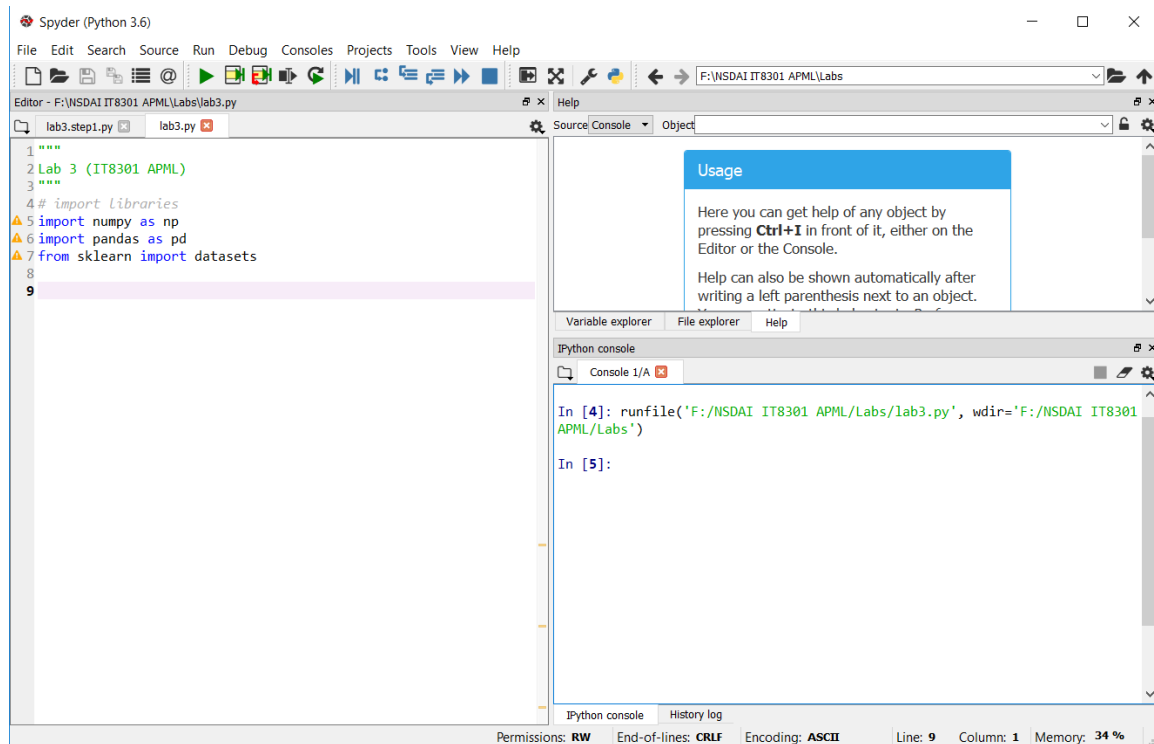


USING PYTHON SOURCE FILE

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.

The screenshot shows the Spyder Python IDE interface. The main editor window displays the following code:

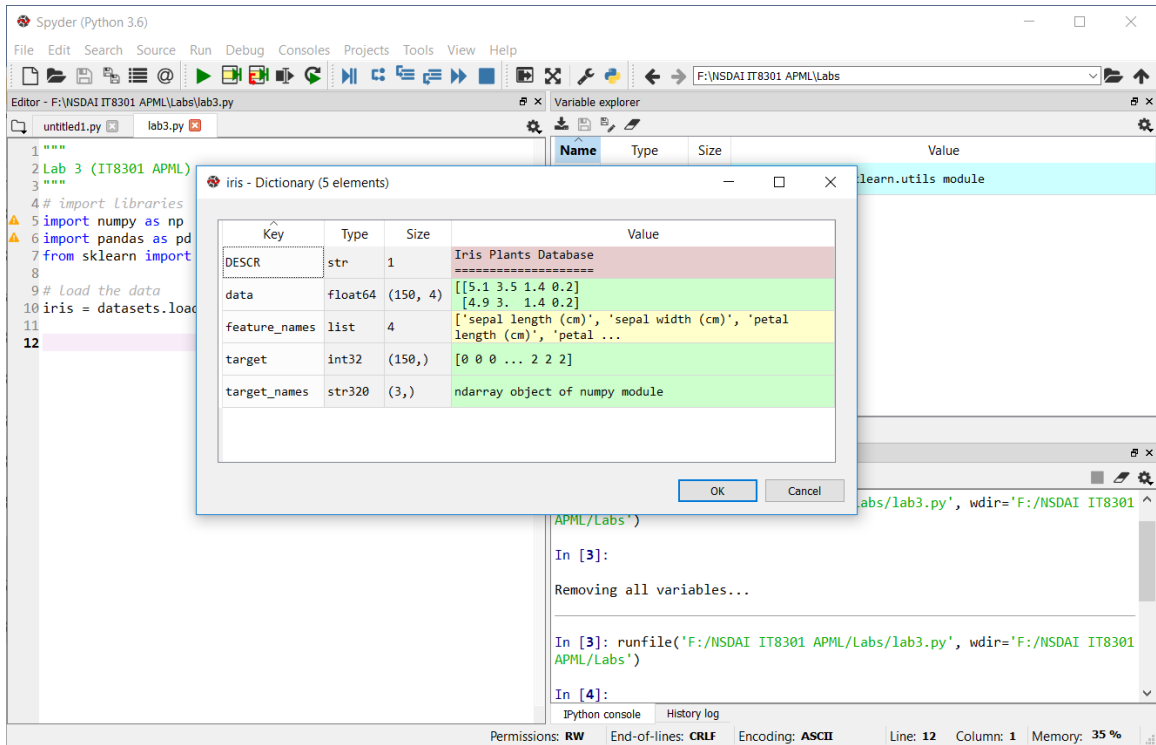
```
1 """
2 Lab 3 (IT8301 APML)
3 """
4 # import libraries
5 import numpy as np
6 import pandas as pd
7 from sklearn import datasets
8
9 # Load the data
10 iris = datasets.load_iris()
11
12
```

The Variable explorer pane on the right shows the variable 'iris' of type 'utils.Bunch' with a size of 5. The value is described as 'Bunch object of sklearn.utils module'.

The IPython console at the bottom shows the output of the code execution:

```
In [7]: runfile('F:/NSDAI IT8301 APML/Labs/lab3.py', wdir='F:/NSDAI IT8301 APML/Labs')
In [8]:
```

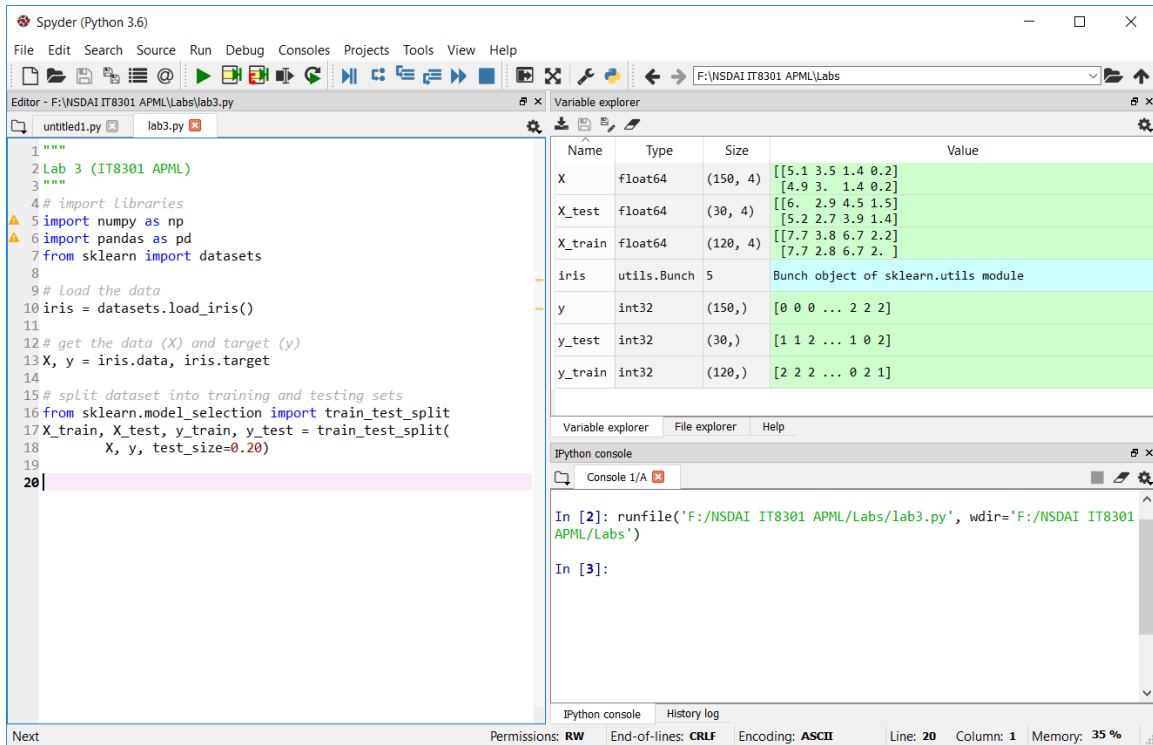
The status bar at the bottom indicates the file encoding is ASCII, the line is 12, the column is 1, and the memory usage is 34%.



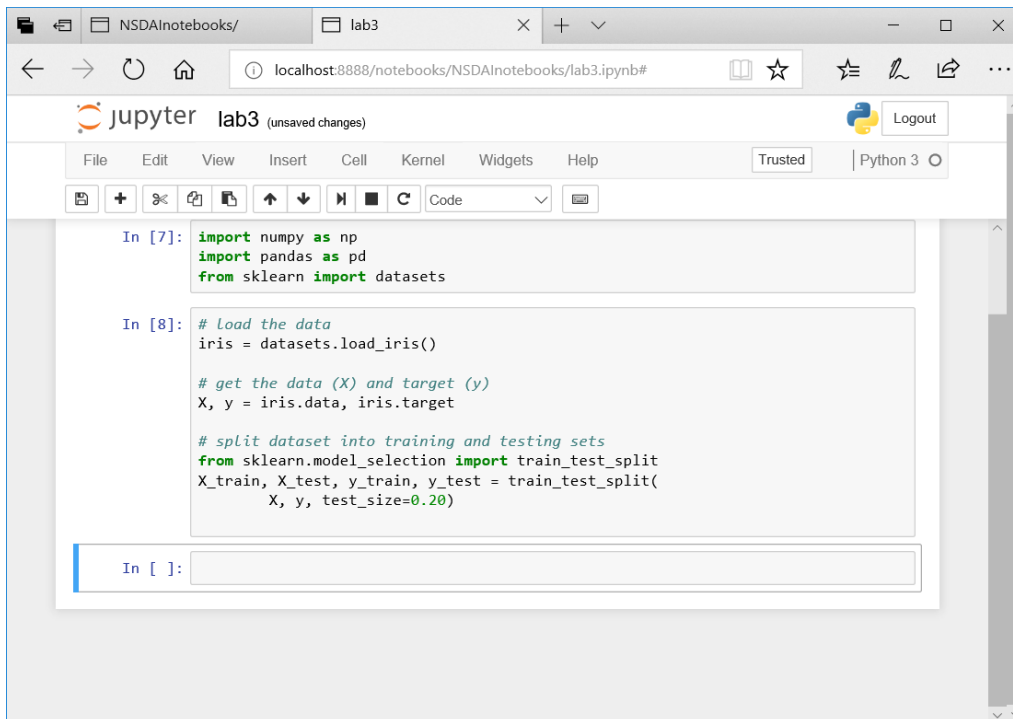
Next we load the data matrix in `X` and the target vector in `y`. After that, we split the dataset into a training set and a test set using the `train_test_split` function from `sklearn.model_selection`.

```
# 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)
```

The screenshot shows the Spyder Python IDE with a script named 'lab3.py' open. The script is titled 'Lab 3 (IT8301 APML)' and contains the following code:

```

1 """
2 Lab 3 (IT8301 APML)
3 """
4 # import libraries (step 1)
5 import numpy as np
6 import pandas as pd
7 from sklearn import datasets
8
9 # load the data (step 2)
10 iris = datasets.load_iris()
11
12 # get the data (X) and target (y)
13 X, y = iris.data, iris.target
14
15 # split dataset into training and testing sets (step 3)
16 from sklearn.model_selection import train_test_split
17 X_train, X_test, y_train, y_test = train_test_split(
18     X, y, test_size=0.20)
19
20 # get the LogisticRegression estimator
21 from sklearn.linear_model import LogisticRegression
22
23 # training the model
24 # apply algorithm to data using fit()
25 clf = LogisticRegression(solver='newton-cg', multi_class='multinomial')
26 clf.fit(X_train, y_train)
27
28 # We can save the trained model clf using pickle (step 4)
29 import pickle
30 s = pickle.dumps(clf)
31 pickle.dump( clf, open( "iris_trained_model.p", "wb" ) )
32 clf2 = pickle.load( open( "iris_trained_model.p", "rb" ) )
33
34 # Alternatively, we can save the trained model clf using sklearn joblib
35 from sklearn.externals import joblib
36 joblib.dump(clf, 'iris_trained_model.pkl')
37 clf3 = joblib.load('iris_trained_model.pkl')
38
39 # Score the model (step 5)
40 y_hat = clf.predict(X_test)
41

```

The Variable explorer on the right shows the following variables:

| Name | Type | Size | Value |
|---------|-------------|----------|--|
| X | float64 | (150, 4) | [[5.1 3.5 1.4 0.2] [4.9 3. 1.4 0.2] [7.2 3.2 6. 1.8] [5.1 3.7 1.5 0.4] [5.7 2.6 3.5 1.] [5.5 2.3 4. 1.3] |
| X_test | float64 | (30, 4) | [[7.2 3.2 6. 1.8] [5.1 3.7 1.5 0.4] [5.7 2.6 3.5 1.] [5.5 2.3 4. 1.3] |
| X_train | float64 | (120, 4) | [[5.1 3.5 1.4 0.2] [4.9 3. 1.4 0.2] [7.2 3.2 6. 1.8] [5.1 3.7 1.5 0.4] [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] |

The Python console shows the following output:

```

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

In [28]:

```

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)

```