http://scikit-learn.org

Overview

Machine Learning in Python

- Simple and efficient tools for data mining and data analysis
- Accessible to everybody, and reusable in various contexts
- Built on NumPy, SciPy, and matplotlib
- Open source, commercially usable - BSD license

This python library allows users to conduct statistical and machine learning operations on given datasets with minimal work. The library has uses for *classification, regression, clustering, dimensionality reduction,* and *data preprocessing*. These operations can allow you to extract features, normalize data, conduct forecasts, and various other (progressively more advanced) machine learning techniques.

Documentation: http://scikit-learn.org/stable/user_guide.html

Installation

Scikit-learn requires:

- Python (>= 2.7 or >= 3.3),
- NumPy (>= 1.8.2),
- SciPy (>= 0.13.3).

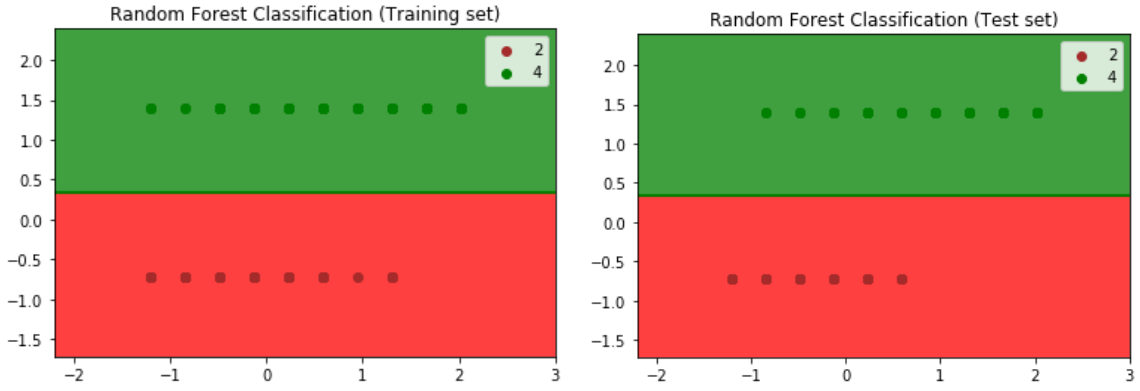
Installation can be completed from the command prompt using *pip* or *conda*.

```
pip install -U scikit-learn
```

```
conda install scikit-learn
```

Example: Predicting Malignancy of Breast Cancer Sample

Dataset : University of Wisconsin Hospitals, Madison from Dr. William H. Wolberg
https://www.kaggle.com/roustekbio/breast-cancer-csv/downloads/breastCancer.csv



Alexis Coffer
Samuel Peoples

http://scikit-learn.org

```
"""
Random Forest Classification
Predicting Malignancy of Breast Cancer data
2: Benign, 4: Malignant
"""
#       Importing the libraries
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
#       Importing the dataset
dataset = pd.read_csv('breastCancer.csv')
X = dataset.iloc[:, [1, 10]].values
y = dataset.iloc[:, 10].values
#       Splitting the dataset into the Training set and Test set
from sklearn.cross_validation import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.25)
#        Feature Scaling
from sklearn.preprocessing import StandardScaler
sc = StandardScaler()
X_train = sc.fit_transform(X_train)
X_test = sc.transform(X_test)
#       Fitting Random Forest Classification to the Training set
from sklearn.ensemble import RandomForestClassifier
classifier = RandomForestClassifier(n_estimators = 10, criterion = 'entropy')
classifier.fit(X_train, y_train)
#       Predicting the Test set results
y_pred = classifier.predict(X_test)
#       Making the Confusion Matrix
from sklearn.metrics import confusion_matrix
cm = confusion_matrix(y_test, y_pred)
print(cm)
#       Visualising the Test set results
from matplotlib.colors import ListedColormap
X_set, y_set = X_test, y_test
X1, X2 = np.meshgrid(np.arange(start = X_set[:, 0].min() - 1, stop = X_set[:, 0].max()
+ 1, step = 0.01), np.arange(start = X_set[:, 1].min() - 1, stop = X_set[:, 1].max() +
1, step = 0.01))
plt.contourf(X1, X2, classifier.predict(np.array([X1.ravel(),
X2.ravel()]).T).reshape(X1.shape), alpha = 0.75, cmap = ListedColormap(('red',
'green')))
plt.xlim(X1.min(), X1.max())
plt.ylim(X2.min(), X2.max())
for i, j in enumerate(np.unique(y_set)):
    plt.scatter(X_set[y_set == j, 0], X_set[y_set == j, 1],
                c = ListedColormap(('brown', 'green'))(i), label = j)
plt.title('Random Forest Classification (Test set)')
plt.legend()
plt.show()
```

Confusion matrix:

112	0
0	63

- Zero type I, or type II errors.

Alexis Coffer
Samuel Peoples