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*.

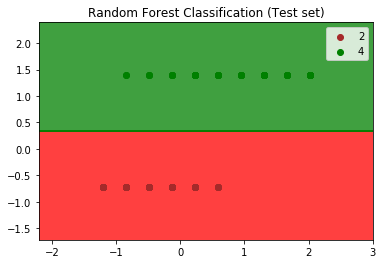
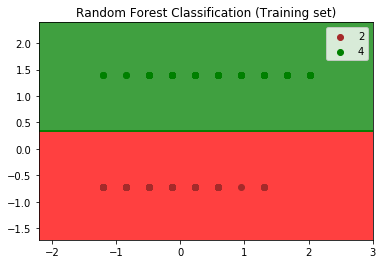
pip install -U scikit-learn

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>



1. """
2. Random Forest Classification
3. Predicting Malignancy of Breast Cancer data
4. 2: Benign, 4: Malignant
5. """
6. *# Importing the libraries*
7. **import** numpy as np
8. **import** matplotlib.pyplot as plt
9. **import** pandas as pd
10. *# Importing the dataset*
11. dataset = pd.read\_csv('breastCancer.csv')
12. X = dataset.iloc[:, [1, 10]].values
13. y = dataset.iloc[:, 10].values
14. ***# Splitting the dataset into the Training set and Test set***
15. **from** sklearn.cross\_validation **import** train\_test\_split
16. X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size = 0.25)
17. *# Feature Scaling*
18. **from** sklearn.preprocessing **import** StandardScaler
19. sc = StandardScaler()
20. X\_train = sc.fit\_transform(X\_train)
21. X\_test = sc.transform(X\_test)
22. *# Fitting Random Forest Classification to the Training set*
23. **from** sklearn.ensemble **import** RandomForestClassifier
24. classifier = RandomForestClassifier(n\_estimators = 10, criterion = 'entropy')
25. classifier.fit(X\_train, y\_train)
26. *# Predicting the Test set results*
27. y\_pred = classifier.predict(X\_test)
28. *# Making the Confusion Matrix*
29. **from** sklearn.metrics **import** confusion\_matrix
30. cm = confusion\_matrix(y\_test, y\_pred)
31. **print**(cm)
32. *# Visualising the Test set results*
33. **from** matplotlib.colors **import** ListedColormap
34. X\_set, y\_set = X\_test, y\_test
35. 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))
36. plt.contourf(X1, X2, classifier.predict(np.array([X1.ravel(), X2.ravel()]).T).reshape(X1.shape), alpha = 0.75, cmap = ListedColormap(('red', 'green')))
37. plt.xlim(X1.min(), X1.max())
38. plt.ylim(X2.min(), X2.max())
39. **for** i, j **in** enumerate(np.unique(y\_set)):
40. plt.scatter(X\_set[y\_set == j, 0], X\_set[y\_set == j, 1],
41. c = ListedColormap(('brown', 'green'))(i), label = j)
42. plt.title('Random Forest Classification (Test set)')
43. plt.legend()
44. plt.show()

Confusion matrix:

|  |  |
| --- | --- |
| 112 | 0 |
| 0 | 63 |

* Zero type I, or type II errors.