MedTeach_ML

April 1, 2018

In [222]:
```
import scipy.io as sio
import matplotlib.pyplot as plt
import numpy as np
import seaborn as sns
import pandas as pd
import sys
from sklearn import svm
from sklearn import tree
from sklearn.model_selection import GridSearchCV
from sklearn.ensemble import RandomForestClassifier
```

In [223]:
```
def convert_breast(label):
    """
    Function to convert the labels from string to numeric
    """
    if label == 2:
        return 1.0
    elif label == 4:
        return 0.0
    else:
        return label
def convert_heart(label):
    """
    Function to convert the labels from string to numeric
    """
    if label >=1:
        return 1.0
    elif label == 0:
        return 0.0
    else:
        return label
def convert_string(s):
    return float(s)
```

1 Parsing Data Sets

In [224]:
```
heart = pd.read_csv("http://archive.ics.uci.edu/ml/machine-learning-databases/heart-
heart = heart.replace({'?':np.nan}).dropna()
```

```
        heart['0'] = heart['0'].apply(convert_heart)
        for column in heart:
            heart[column] = heart[column].apply(convert_string)
```

```
In [225]: breast = pd.read_csv("http://archive.ics.uci.edu/ml/machine-learning-databases/breast
          breast['2.1'] = breast['2.1'].apply(convert_breast)
          breast.drop('1000025',axis = 1,inplace= True)
          breast = breast.replace({'?':np.nan}).dropna()
          for column in breast:
              breast[column] = breast[column].apply(convert_string)
```

2 Splitting

```
In [226]: heart_XY = heart.iloc[:,0:14].values
          np.random.shuffle(heart_XY)

          heart_X = heart_XY[:,0:13]
          heart_Y = heart_XY[:,13:14]

          num_training = int(0.8*heart_X.shape[0])
          num_testing = int(0.2*heart_X.shape[0])

          heart_X_train = heart_X[:num_training]
          heart_Y_train = heart_Y[:num_training]
          heart_X_test = heart_X[num_training:]
          heart_Y_test = heart_Y[num_training:]
```

```
In [227]: breast_XY = breast.iloc[:,0:10].values
          np.random.shuffle(breast_XY)

          breast_X = breast_XY[:,0:9]
          breast_Y = breast_XY[:,9:10]

          num_training = int(0.8*breast_X.shape[0])
          num_testing = int(0.2*breast_X.shape[0])

          breast_X_train = breast_X[:num_training]
          breast_Y_train = breast_Y[:num_training]
          breast_X_test = breast_X[num_training:]
          breast_Y_test = breast_Y[num_training:]
```

3 Train

```
In [228]: classifier = tree.DecisionTreeClassifier(
                                criterion='entropy')
          depth_list = [1, 2, 3, 4, 5]
          params = {"max_depth": depth_list}
```

```
grid_dt_heart = GridSearchCV(classifier, params,
                             return_train_score = True,
                             cv = 10)
grid_dt_heart.fit(heart_X_train, heart_Y_train)
```

```
Out[228]: GridSearchCV(cv=10, error_score='raise',
       estimator=DecisionTreeClassifier(class_weight=None, criterion='entropy', max_
            max_features=None, max_leaf_nodes=None,
            min_impurity_decrease=0.0, min_impurity_split=None,
            min_samples_leaf=1, min_samples_split=2,
            min_weight_fraction_leaf=0.0, presort=False, random_state=None,
            splitter='best'),
       fit_params=None, iid=True, n_jobs=1,
       param_grid={'max_depth': [1, 2, 3, 4, 5]}, pre_dispatch='2*n_jobs',
       refit=True, return_train_score=True, scoring=None, verbose=0)
```

```
In [229]: depth_list = [1, 2, 3, 4, 5]
          params = {"max_depth": depth_list}
          classifier = RandomForestClassifier(max_depth=5,
                                              random_state=0)
          grid_rf_breast = GridSearchCV(classifier, params,
                                        return_train_score = True, cv = 10)

          grid_rf_breast.fit(breast_X_train, breast_Y_train.ravel())
```

```
Out[229]: GridSearchCV(cv=10, error_score='raise',
       estimator=RandomForestClassifier(bootstrap=True, class_weight=None, criterion=
            max_depth=5, max_features='auto', max_leaf_nodes=None,
            min_impurity_decrease=0.0, min_impurity_split=None,
            min_samples_leaf=1, min_samples_split=2,
            min_weight_fraction_leaf=0.0, n_estimators=10, n_jobs=1,
            oob_score=False, random_state=0, verbose=0, warm_start=False),
       fit_params=None, iid=True, n_jobs=1,
       param_grid={'max_depth': [1, 2, 3, 4, 5]}, pre_dispatch='2*n_jobs',
       refit=True, return_train_score=True, scoring=None, verbose=0)
```

4 Test

```
In [230]: pred = grid_rf_heart.predict(heart_X_test)
          asserted = []
          for i in range(0,pred.shape[0]):
              if pred[i] == heart_Y_test[i]:
                  asserted.append(pred[i])
          test_accuracy = len(asserted)/pred.shape[0]
          print("Test Accuracy for heart: {0:f}".format(test_accuracy))
```

```
Test Accuracy for heart: 0.850000
```

```
In [231]: pred = grid_rf_breast.predict(breast_X_test)
          asserted = []
          for i in range(0,pred.shape[0]):
              if pred[i] == breast_Y_test[i]:
                  asserted.append(pred[i])
          test_accuracy = len(asserted)/pred.shape[0]
          print("Test Accuracy for breast: {0:f}".format(test_accuracy))

Test Accuracy for breast: 0.978102
```

5 Interact with heart disease features

Index For Features:

```
1.) age
2.) sex (1 = M, 0 = F)
3.) cp: chest pain type
    Value 1: typical angina
    Value 2: atypical angina
    Value 3: non-anginal pain
    Value 4: asymptomatic
4.) trestbps: resting blood pressure (in mm Hg on admission to the hospital)
5.) chol: serum cholestoral in mg/dl
6.) fbs: (fasting blood sugar > 120 mg/dl) (1 = true; 0 = false)
7.) estecg: resting electrocardiographic results
        Value 0: normal
        Value 1: having ST-T wave abnormality (T wave inversions and/or ST elevation
            or depression of > 0.05 mV)
        Value 2: showing probable or definite left ventricular hypertrophy by
            Estes' criteria
8.) thalach: maximum heart rate achieved
9.) exang: exercise induced angina (1 = yes; 0 = no)
10.) oldpeak = ST depression induced by exercise relative to rest
11.) slope: the slope of the peak exercise ST segment
        Value 1: upsloping
        Value 2: flat
        Value 3: downsloping
12.) ca: number of major vessels (0-3) colored by flourosopy
13.) thal: 3 = normal; 6 = fixed defect; 7 = reversable defect
```

```
In [232]: continue_1 = input("Time to predict if a patience has a heart disease! refer to the i
          age = int(input("Enter age: "))
          sex = int(input("Enter sex(1 for M, 0 for F): "))
          cp = int(input("Enter cp: "))
          trestbps = int(input("Enter trestbps: "))
          chol = int(input("Enter chol: "))
          fbs = int(input("Enter fbs: "))
```

```
estecg = int(input("Enter estecg: "))
thalach = int(input("Enter thalach: "))
exang = float(input("Enter exang: "))
oldpeak = float(input("Enter oldpeak: "))
slope = float(input("Enter slope: "))
ca = float(input("Enter ca: "))
thal = float(input("Enter thal: "))
label_to_predict = [[age,sex,cp,trestbps,chol,fbs,
                     estecg,thalach,exang,oldpeak,
                     slope,ca,thal]]
prediction = grid_rf_heart.predict(label_to_predict)
print()
if prediction == 1:
    print("The patient is highly likely to have a heart disease!")
else:
    print("The patient is highly unlikely to have a heart disease!")
```

```
Time to predict if a patience has a heart disease! refer to the index above for details on inpu
Enter age: 67
Enter sex(1 for M, 0 for F): 1
Enter cp: 4
Enter trestbps: 160
Enter chol: 286
Enter fbs: 0
Enter estecg: 2
Enter thalach: 108
Enter exang: 1
Enter oldpeak: 1.5
Enter slope: 2
Enter ca: 3
Enter thal: 3

The patient is highly likely to have a heart disease!
```

6 Interact with breast cancer features

Index For Features:

```
1.) Clump Thickness id
2.) Uniformity of Cell Size id
3.) Uniformity of Cell Shape id
4.) Maeginal Adhesion id
5.) Single Epithelial Cell Size id
6.) Bare Nuclei id
7.) Bland Chromatin id
8.) Normal Nucleoli id
9.) Mitoses id
```

```
In [233]: continue_1 = input("Time to predict if a patience has breast cancer! refer to the inc
          print()
          ct = int(input("Enter Clump Thickness: "))
          uocs = int(input("Uniformity of Cell Size: "))
          uocsh = int(input("Uniformity of Cell Shape: "))
          ma = int(input("Maeginal Adhesion id: "))
          secs = int(input("Single Epithelial Cell Size: "))
          bn = int(input("Bare Nuclei: "))
          bc = int(input("Bland Chromatin: "))
          nn = int(input("Normal Nucleoli: "))
          mi = float(input("Mitoses: "))
          label_to_predict = [[ct,uocs,uocsh,ma,
                               secs,bn,bc,nn,mi]]
          prediction = grid_rf_breast.predict(label_to_predict)
          print()
          if prediction == 1:
              print("The patient is highly likely to have breast cancer!")
          else:
              print("The patient is highly unlikely to have breast cancer!")
```

```
Time to predict if a patience has breast cancer! refer to the index above for details on input

Enter Clump Thickness: 8
Uniformity of Cell Size: 10
Uniformity of Cell Shape: 10
Maeginal Adhesion id: 8
Single Epithelial Cell Size: 7
Bare Nuclei: 10
Bland Chromatin: 9
Normal Nucleoli: 7
Mitoses: 1

The patient is highly unlikely to have breast cancer!
```