COMP809 – Data Mining and Machine Learning

Lab 7 – Classifier Models

- Two major objectives of this lab are to
  - configure Python's implementation of some of the most widely used classifiers
  - to evaluate these classifiers using a variety of different metrics.

- Configuring classifiers will be achieved using Python's sklearn library. Evaluation will also be done via sklearn but use a dedicated set of methods designed specifically for computing mterics.

1. Importing the libraries

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split,
cross_val_score
from sklearn.tree import DecisionTreeClassifier
from sklearn.naive_bayes import GaussianNB, MultinomialNB
from sklearn.metrics import accuracy_score
from sklearn.neighbors import KNeighborsClassifier
from sklearn.neural_network import MLPClassifier
from sklearn.metrics import precision_score, recall_score,auc
from sklearn.metrics import roc_curve,roc_auc_score,
plot_roc_curve
```

2. Loading and inspecting the dataset

```
path="P:\COMP809\Iris.xlsx"        #should change the path
accordingly
rawdata= pd.read_excel(path)    #pip install xlrd
print ("data summary")
print (rawdata.describe())
nrow, ncol = rawdata.shape
print (nrow, ncol)
print ("\n correlation Matrix")
print (rawdata.corr())
rawdata.hist()
plt.show()
```

3. **Display correlations between all pairs of features**

```
pd.plotting.scatter_matrix(rawdata,figsize=[8,8])
plt.show()

# boxplot
fig = plt.figure(1, figsize=(9, 6))
ax = fig.add_subplot(111)
ax.boxplot(rawdata.values)
ax.set_xticklabels(['Petal Length', 'Petal Width', 'Sepal
Length', 'Sepal Width', 'Class'])
plt.show()
```

4. **Get the predictors – all columns from 0 to last but one**

```
predictors = rawdata.iloc[:,:ncol-1]
print(predictors)
#index to last column to obtain class values
target = rawdata.iloc[:,-1]
print(target)
```

5. **Partition data using a train/test split**

By referring to https://scikit-learn.org/0.16/modules/generated/sklearn.cross_validation.train_test_split.html complete the right-hand side of the line below and set the training set size to 70% of the size of the dataset.

```
pred_train, pred_test, tar_train, tar_test =
train_test_split()
```

6. **Configure the Decision Tree Classifier**

```
split_threshold=4
fpr = dict()                        # store false positive
rate in a dictionary object
tpr = dict()   # likewise, store the true positive rate
roc_auc = dict()
```

By referring to https://scikit-learn.org/stable/modules/generated/sklearn.tree.DecisionTreeClassifier.html#sklearn.tree.DecisionTreeClassifier set the entropy criterion for splitting and set the minimum no of samples (objects) for splitting a decision node (Note: This threshold should be greater than 1).

```
for i in range(2, split_threshold):
     classifier = DecisionTreeClassifier() #configure the
     classifier

     classifier = classifier.fit(pred_train, tar_train) #
     train a decision tree model

     predictions = classifier.predict(pred_test) # deploy
     model and make predictions on test set

     prob = classifier.predict_proba(pred_test)     # obtain
     probability scores for each sample in test set

     print("Accuracy score of our model with Decision Tree:",
     i, accuracy_score(tar_test, predictions))

     precision = precision_score(y_true=tar_test,
     y_pred=predictions, average='micro')
     print("Precision score of our model with Decision Tree
     :", precision)

     recall = recall_score(y_true=tar_test,
     y_pred=predictions, average='micro')
     print("Recall score of our model with Decision Tree :",
     recall)

     for x in range(3):
          fpr[x], tpr[x], _ = roc_curve(tar_test[:], prob[:,
     x],pos_label=x)
          roc_auc[x] = auc(fpr[x], tpr[x])
          print("AUC values of the decision tree",roc_auc[x])
          plt.plot(fpr[x], tpr[x],
     color='darkorange',label='ROC curve (area = %0.2f)' %
     roc_auc[x])
          plt.show()
```