ComS573_Lab4_Q1

April 15, 2020

ComS 573

Lab 4

Kanak Choudhury

1 Problem 1

[15]:

```
import numpy as np
import pandas as pd
import sklearn.preprocessing
import matplotlib
import keras
import re
import sys
import gc
import time

print('python ' +sys.version)
print('numpy '+ np.__version__)
print('pandas '+ pd.__version__)
print('sklearn '+ sklearn.__version__)
print('matplotlib '+ matplotlib.__version__)
print('re '+ re.__version__)

from sklearn.preprocessing import StandardScaler
from sklearn.metrics import confusion_matrix
from matplotlib import pyplot as plt
from sklearn.ensemble import AdaBoostClassifier
from sklearn.ensemble import RandomForestClassifier

from itertools import product

def print_out(model, model_name, hyper_prem, x_dt_tr, y_dt_tr, x_dt_ts,␣
 ↪y_dt_ts):
    print("For "+model_name+" hyper-parameters:\n",hyper_prem)
    scores = model.score(x_dt_ts, y_dt_ts)
    print("\n Test Accuracy: %.2f%%" % (scores*100))
    A = model.predict(x_dt_tr)
    cm = confusion_matrix(y_dt_tr, A)
    print("\n Train confusion matrix: \n", cm)
    acc_train = np.diagonal(cm)/cm.sum(axis=1)
    print("\n Class Accuracy for Training Data is:")
    for i in range(2):
        print('Class %d: %.2f%%' %(i, acc_train[i]*100))

    A = model.predict(x_dt_ts)
    cm = confusion_matrix(y_dt_ts, A)
    print("\n Test confusion matrix: \n", cm)
    acc_test = np.diagonal(cm)/cm.sum(axis=1)
    print("\n Class Accuracy for Testing Data is:")
    for i in range(2):
        print('Class %d: %.2f%%' %(i, acc_test[i]*100))
    print("**********************************\n")
```

```
python 3.6.9 |Anaconda, Inc.| (default, Jul 30 2019, 14:00:49) [MSC v.1915 64
bit (AMD64)]
numpy 1.16.5
pandas 0.25.1
sklearn 0.21.3
matplotlib 3.1.1
re 2.2.1
```

[16]:

```
path  = 'D:/ISU/COMS 573 - Machine Learning/HW/Lab4/'

df_train = pd.read_csv(path + 'lab4-train.csv', sep=',', header=0)
df_test = pd.read_csv(path + 'lab4-test.csv', sep=',', header=0)
tr_size = df_train.shape
ts_size = df_test.shape

x_train = np.array(df_train[['R','F','M','T']])
y_train = np.array(df_train['Class'])

x_test = np.array(df_test[['R','F','M','T']])
y_test = np.array(df_test['Class'])
x_train12 = x_train
y_train12 = y_train
x_test12 = x_test
y_test12 = y_test
```

1.1 Random Forest

[17]:

```
n_estimators=[50, 100, 150, 200]
criterion=['gini', 'entropy']
max_depth=[1, 2, 3 ,4]
min_samples_split=[5, 7, 10, 12]
min_samples_leaf=[1, 2, 3]

def expand_grid(dictionary):
   return pd.DataFrame([row for row in product(*dictionary.values())],
                       columns=dictionary.keys())

dictionary = {'n_estimators': n_estimators,
              'criterion': criterion,
             'max_depth': max_depth,
             'min_samples_split': min_samples_split,
             'min_samples_leaf': min_samples_leaf}

prem1 = expand_grid(dictionary)
size_prem = prem1.shape[0]
prem = prem1
prem['train_acc'] = np.NaN
prem['test_acc'] = np.NaN

ll = 0
best_fit = None
best_ts_acc = 0

for i in range(prem.shape[0]):
    ts_acc1 = 0
    rf=RandomForestClassifier(n_estimators=prem.iloc[i,0], criterion=prem.
 ↪iloc[i,1],
                              max_depth=prem.iloc[i,2],
                              min_samples_split=prem.iloc[i,3],␣
 ↪min_samples_leaf=prem.iloc[i,4],
                              max_features='auto', bootstrap=True)
    model_rf = rf.fit(x_train, y_train)
    ts_acc1 = model_rf.score(x_test, y_test)*100
    if (ts_acc1 > best_ts_acc):
        best_ts_acc = ts_acc1
        best_fit = model_rf
    prem.loc[i,5:7] = [model_rf.score(x_train, y_train)*100, model_rf.
 ↪score(x_test, y_test)*100]
    ll = ll+1
    sys.stdout.write("\r Progress: %.2f%%" %round(float(ll)/size_prem*100,2))
    sys.stdout.flush()
```

```
Progress: 100.00%
```

[18]:
```
top10_mse = prem.nlargest(10,'test_acc')
print('\n Best 10 hyper-parameter combination for Random Forest:\n',␣
 ↪round(top10_mse, 4))

print_out(model = best_fit, model_name = 'Random Forest',
          hyper_prem = top10_mse.iloc[0,:], x_dt_tr = x_train,
          y_dt_tr = y_train, x_dt_ts = x_test, y_dt_ts = y_test)
```

```
 Best 10 hyper-parameter combination for Random Forest:
      n_estimators criterion  max_depth  min_samples_split  min_samples_leaf  \
335           200      gini          4                 12                 3
380           200   entropy          4                 10                 3
134           100      gini          4                  5                 3
230           150      gini          4                  5                 3
329           200      gini          4                  7                 3
36             50      gini          4                  5                 1
85             50   entropy          4                  5                 2
90             50   entropy          4                 10                 1
92             50   entropy          4                 10                 3
95             50   entropy          4                 12                 3

     train_acc  test_acc
335    79.6421   84.3854
380    80.3132   84.3854
134    79.8658   84.0532
230    80.0895   84.0532
329    80.0895   84.0532
36     79.4183   83.7209
85     80.3132   83.7209
90     79.6421   83.7209
92     79.8658   83.7209
95     79.1946   83.7209
For Random Forest hyper-parameters:
 n_estimators             200
criterion               gini
max_depth                  4
min_samples_split         12
min_samples_leaf           3
train_acc            79.6421
test_acc             84.3854
Name: 335, dtype: object

 Test Accuracy: 84.39%

 Train confusion matrix:
 [[313  19]
 [ 72  43]]

 Class Accuracy for Training Data is:
Class 0: 94.28%
Class 1: 37.39%

 Test confusion matrix:
 [[229   9]
 [ 38  25]]

 Class Accuracy for Testing Data is:
Class 0: 96.22%
Class 1: 39.68%
**********************************
```

1.2 AdaBoost

[19]:

```
n_estimators=[50, 100, 150, 200]
learning_rate=np.logspace(-5,0,30,base=10)


dictionary = {'n_estimators': n_estimators,
              'learning_rate': learning_rate}

prem1 = expand_grid(dictionary)
size_prem = prem1.shape[0]
prem = prem1
prem['train_acc'] = np.NaN
prem['test_acc'] = np.NaN

ll = 0
best_ts_acc = 0
best_fit = None
best_ts_acc = 0

for i in range(prem.shape[0]):
    ts_acc1 = 0
    adb=AdaBoostClassifier(n_estimators=prem.iloc[i,0], learning_rate=prem.
 ↪iloc[i,1],
                           algorithm='SAMME.R')
    model_adb = adb.fit(x_train, y_train)
    ts_acc1 = model_adb.score(x_test, y_test)*100
    if (ts_acc1 > best_ts_acc):
        best_ts_acc = ts_acc1
        best_fit = model_adb
    prem.loc[i,2:4] = [model_adb.score(x_train, y_train)*100, model_adb.
 ↪score(x_test, y_test)*100]
    ll = ll+1
    sys.stdout.write("\r Progress: %.2f%%" %round(float(ll)/size_prem*100,2))
    sys.stdout.flush()
```

```
 Progress: 100.00%
```

[20]:

```
top10_mse = prem.nlargest(10,'test_acc')
print('\n Best 10 hyper-parameter combination for AdaBoost:\n',␣
 ↪round(top10_mse, 4))

print_out(model = best_fit, model_name = 'AdaBoostn',
          hyper_prem = top10_mse.iloc[0,:], x_dt_tr = x_train,
          y_dt_tr = y_train, x_dt_ts = x_test, y_dt_ts = y_test)
```

```

 Best 10 hyper-parameter combination for AdaBoost:
      n_estimators  learning_rate  train_acc  test_acc
113          200         0.0924    79.1946   82.3920
84           150         0.1374    79.6421   82.0598
55           100         0.2043    79.4183   81.7276
85           150         0.2043    79.8658   81.7276
26            50         0.3039    79.4183   81.3953
54           100         0.1374    78.9709   81.3953
56           100         0.3039    79.6421   81.3953
83           150         0.0924    78.9709   81.3953
27            50         0.4520    79.1946   81.0631
28            50         0.6723    80.5369   81.0631
For AdaBoostn hyper-parameters:
 n_estimators     200.000000
learning_rate      0.092367
train_acc         79.194631
test_acc          82.392027
Name: 113, dtype: float64

 Test Accuracy: 82.39%

 Train confusion matrix:
 [[316  16]
 [ 77  38]]

 Class Accuracy for Training Data is:
Class 0: 95.18%
Class 1: 33.04%

 Test confusion matrix:
```

```
 [[229   9]
 [ 44  19]]

 Class Accuracy for Testing Data is:
Class 0: 96.22%
Class 1: 30.16%
**********************************
```

1.3 Comment:

Based on test accuracy, Random Forest (RF) model has highest (about 84.5%) accuracy than AdaBoost model (about 82.5%). Both models class 0 accuracy are about 96%. However, for class 1, RF has about 40% accuracy compare to AdaBoost (30%). That indicates that, for this data set AdaBoost model have higher bias for the mejority calss than the RM model.

```
[ ]:
```