Homework 3

两问均为分类问题，第一题为多分类，第二题为二分类。

对第一题在尝试了`LogisticRegression`，`RandomForestClassifier`，`SVC`，由`DecisionTreeClassifier`组成的`BaggingClassifier`，以及不同模型组成的`VotingClassifier`. 发现`RandomForestClassifier`相比之下效果较好且较为简单。于是选择`RandomForestClassifier`对其进行调参并拟合和预测。

Homework 3-1

1. 导入数据并归一化处理
2. 使用`GridSearchCV`对`RandomForestClassifier`调节参数，评价函数取RMSE的负值。得到：

```
{'max_features': 15, 'n_estimators': 300}
Lowest RMSE:0.764752
```

3. 选取最佳参数进行训练并预测。对测试集评估得到：

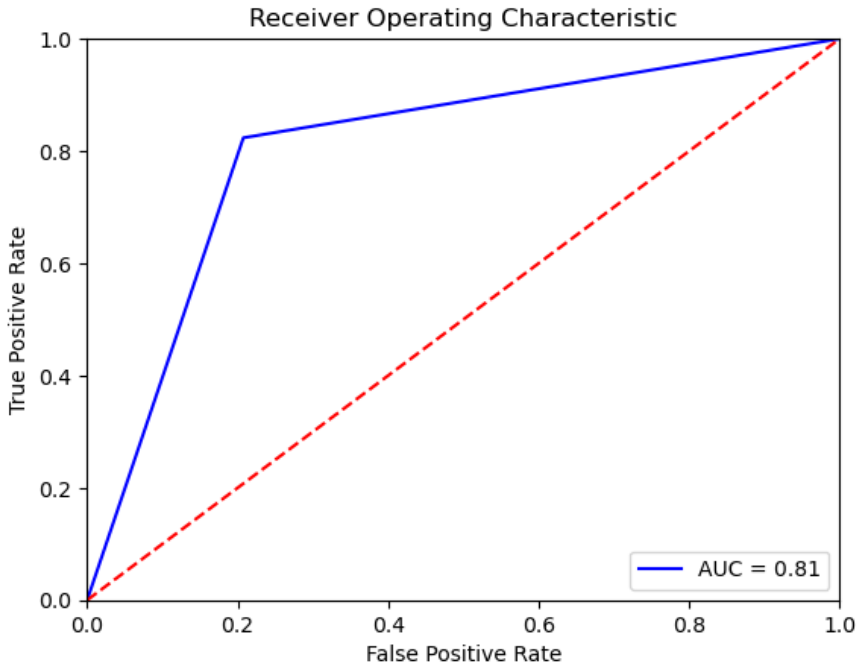
```
Prediction RMSE: 0.443
```

Homework 3-2

1. 导入数据并归一化处理
2. 使用`GridSearchCV`对`RandomForestClassifier`调节参数，评价函数取AUC。得到：

```
{'max_features': 2, 'n_estimators': 220}
best AUC:0.858329
```

3. 选取最佳参数进行训练并预测。将结果绘图得到：



assign3Code

November 6, 2022

[111]:
```
# import numpy as np
# import pandas as pd
# from sklearn.ensemble import RandomForestClassifier
# from sklearn.metrics import mean_squared_error
# from sklearn.ensemble import VotingClassifier
# from sklearn.linear_model import LogisticRegression
# from sklearn.svm import SVC
```

[112]:
```
# dataTrain = pd.read_csv("Homework-3-1/Homework-3-1-train_data.csv",index_col␣
 ↪= 0)
# dataTest = pd.read_csv("Homework-3-1/Homework-3-1-test_data.csv",index_col =␣
 ↪0)
# Train = dataTrain.values
# Test = dataTest.values
# aver=Train[:,:-1].mean(axis=0)
# std=Train[:,:-1].std(axis=0)
# xTrain = (Train[:,:-1]-aver)/std
# xTest = (Test[:,:-1]-aver)/std
# yTrain = Train[:,-1]
# yTest = Test[:,-1]
# yOneHot = np.zeros((len(yTrain),6))
# for i,val in enumerate(yTrain):
#     yOneHot[i,int(val-3)] = 1
```

[113]:
```
# rs = 42
# log_clf = LogisticRegression(solver = "lbfgs", random_state=rs,max_iter =␣
 ↪1000)
# rnd_clf = RandomForestClassifier(n_estimators=100, random_state=rs)
# svm_clf = SVC(gamma="scale", random_state=rs)
# voting_clf = VotingClassifier(estimators=[('rf',rnd_clf),('svc',svm_clf)],
#                               voting = 'hard')
# voting_clf.fit(xTrain,yTrain)
# y_pred = voting_clf.predict(xTest)
# len(yTest[voting_clf.predict(xTest)-yTest!=0])/len(yTest)#
```

[114]:
```
# len(yTrain[voting_clf.predict(xTrain)-yTrain!=0])/len(yTrain)
```

[115]:
```
# rnd_clf = RandomForestClassifier(oob_score=True,n_estimators=100,␣
 ↪random_state=rs)

# rnd_clf.fit(xTrain,yTrain)
# # len(yTest[rnd_clf.predict(xTest)-yTest!=0])/len(yTest)
# print("accuracy:%f"%rnd_clf.oob_score_)
```

[116]:
```
# from sklearn.ensemble import BaggingClassifier
# from sklearn.tree import DecisionTreeClassifier
```

[117]:
```
# bag_clf = BaggingClassifier(
#     DecisionTreeClassifier(random_state = rs),n_estimators=600,
#     max_samples=500,bootstrap=True,random_state=rs)
# bag_clf.fit(xTrain,yTrain)
# len(yTest[bag_clf.predict(xTest)-yTest!=0])/len(yTest)
```

[118]:
```
# voting_clf = VotingClassifier(estimators=[('rf',rnd_clf),('bg',bag_clf)],
#                               voting = 'hard')
# voting_clf.fit(xTrain,yTrain)
# y_pred = voting_clf.predict(xTest)
# len(yTest[voting_clf.predict(xTest)-yTest!=0])/len(yTest)
```

0.1 Homework 3-1

[64]:
```
import numpy as np
import pandas as pd
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import mean_squared_error
from sklearn.model_selection import GridSearchCV
```

[121]:
```
dataTrain = pd.read_csv("Homework-3-1/Homework-3-1-train_data.csv",index_col =␣
 ↪0)
dataTest = pd.read_csv("Homework-3-1/Homework-3-1-test_data.csv",index_col = 0)
Train = dataTrain.values
Test = dataTest.values
aver=Train[:,:-1].mean(axis=0)
std=Train[:,:-1].std(axis=0)
xTrain = (Train[:,:-1]-aver)/std
xTest = (Test[:,:-1]-aver)/std
yTrain = Train[:,-1]
yTest = Test[:,-1]
rs = 0
```

RandomForestClassifier

[122]:
```
param_grid = {"n_estimators":[100,200,300],"max_features":range(10,30,5)}
gsearch =␣
 ↪GridSearchCV(estimator=RandomForestClassifier(oob_score=True,random_state=rs),
                        ␣
 ↪param_grid=param_grid,scoring='neg_root_mean_squared_error',
                       cv=3,n_jobs = -1)
gsearch.fit(xTrain,yTrain)
print(gsearch.best_params_)
print("Lowest RMSE:%f" % -gsearch.best_score_)
```

```
{'max_features': 15, 'n_estimators': 300}
Lowest RMSE:0.764752
```

[148]:
```
rnd_clf = RandomForestClassifier(oob_score=True,n_estimators=300,
                                 random_state=rs,max_features = 15)
rnd_clf.fit(xTrain,yTrain)
test_pred = rnd_clf.predict(xTest)
test_rmse = mean_squared_error(test_pred,yTest)
print("Prediction RMSE: {}".format(float(test_rmse**0.5)))
```

```
Prediction RMSE: 0.4439767838698152
```

0.2 Homework 3-2

[134]:
```
import numpy as np
import pandas as pd
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import mean_squared_error
from sklearn.model_selection import GridSearchCV
from sklearn.metrics import roc_auc_score
```

[128]:
```
dataTrain = pd.read_csv("Homework-3-2/Homework-3-2-train_data.csv",index_col =␣
 ↪0)
dataTest = pd.read_csv("Homework-3-2/Homework-3-2-test_data.csv",index_col = 0)
Train = dataTrain.values
Test = dataTest.values
aver=Train[:,:-1].mean(axis=0)
std=Train[:,:-1].std(axis=0)
xTrain = (Train[:,:-1]-aver)/std
xTest = (Test[:,:-1]-aver)/std
yTrain = Train[:,-1]
yTest = Test[:,-1]
rs = 0
```

RandomForestClassifier

[141]:
```
param_grid = {"n_estimators":[150,210,220,230,250],"max_features":range(1,7,1)}
gsearch =␣
 ↪GridSearchCV(estimator=RandomForestClassifier(oob_score=True,random_state=rs),
                       param_grid=param_grid,scoring='roc_auc',
                       cv=3,n_jobs = -1)
gsearch.fit(xTrain,yTrain)
print(gsearch.best_params_)
print("best AUC:%f" % gsearch.best_score_)
```

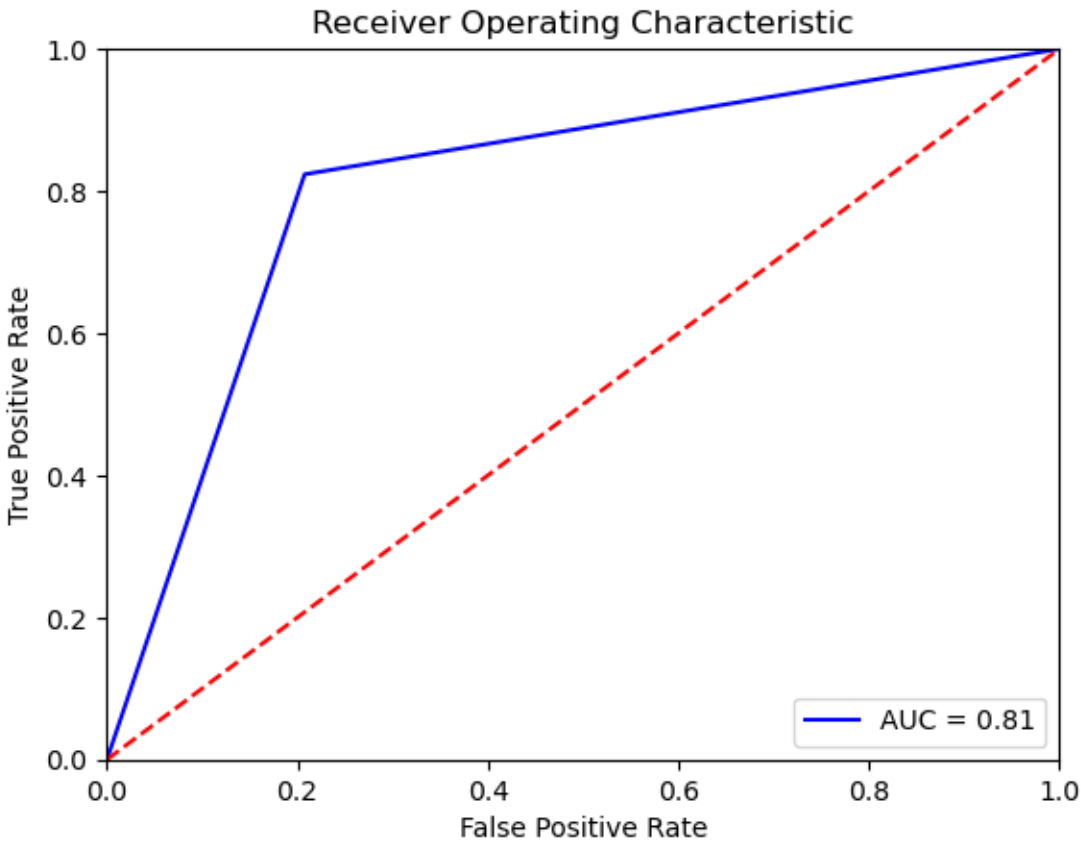
```
{'max_features': 2, 'n_estimators': 220}
best AUC:0.858329
```

[149]:
```
rnd_clf = RandomForestClassifier(oob_score=True,n_estimators=220,
                                 random_state=rs,max_features = 2)
rnd_clf.fit(xTrain,yTrain)
test_pred = rnd_clf.predict(xTest)
test_auc = roc_auc_score(yTest,test_pred)
print("Prediction AUC: {}%".format(float(test_auc)))
```

```
Prediction AUC: 0.8079911209766926%
```

[150]:
```
#ROC Curve
import matplotlib.pyplot as plt
from sklearn import metrics
def createROC(y_test, y_pred):
    fpr, tpr, thresholds = metrics.roc_curve(y_test, y_pred)
    roc_auc = metrics.auc(fpr,tpr)
    plt.title('Receiver Operating Characteristic')
    plt.plot(fpr, tpr, 'b',label='AUC = %0.2f'% roc_auc)
    plt.legend(loc='lower right')
    plt.plot([0,1],[0,1],'r--')
    plt.xlim([0,1])
    plt.ylim([0,1])
    plt.ylabel('True Positive Rate')
    plt.xlabel('False Positive Rate')
    plt.show()


createROC(yTest, test_pred)
```


Receiver Operating Characteristic
True Positive Rate
False Positive Rate
0.0
0.2
0.4
0.6
0.8
1.0
AUC = 0.81