Project 4

Group: P2eta

Members: Paul Rayment, Andrew Peters Roman Formicola

Q2)

In [55]:
```
from numpy import mean
from numpy import std
from sklearn import cluster
from sklearn import neighbors
from sklearn import metrics
from sklearn import tree
from sklearn import naive_bayes
from sklearn.preprocessing import MinMaxScaler
from sklearn.datasets import make_classification
from sklearn.model_selection import KFold
from sklearn.model_selection import GridSearchCV
from sklearn.ensemble import RandomForestClassifier
from sklearn.ensemble import AdaBoostClassifier
from sklearn.svm import SVC
from sklearn.metrics import accuracy_score
from sklearn.metrics import roc_auc_score
import numpy as np
import numpy.ma as ma
import pandas as pd

# Load Data
data = pd.read_csv("hit-movies.csv")

# Trim Data
data = pd.DataFrame(data)
data.pop("original_title")
data.pop("imdb_id")

# Seperate "Hit" as target
target = data.pop("Hit")

# Scale Data
scaler = MinMaxScaler()
scaler.fit(data)
data = pd.DataFrame(scaler.transform(data))

# Record Mean Accuracy, F1 Measure, and AUC
rows = ["KNN3", "KNN9", "KNN15", "DT1", "DT2", "NB", "best SVM", "best RF", "best AdaBoost"]
cols = ["Accuracy", "F1-Measure", "AUC"]
performance = [[0,0,0],[0,0,0],[0,0,0],
               [0,0,0],[0,0,0],[0,0,0],
               [0,0,0],[0,0,0],[0,0,0]]
split = 1
# Set up Cross Validation Splits
cv_outer = KFold(n_splits=10, shuffle=True, random_state=1)
for train_ix, test_ix in cv_outer.split(data):
    # Split the Data
    data_train, data_test = data.iloc[train_ix], data.iloc[test_ix]
    target_train, target_test = target[train_ix], target[test_ix]

    # Q2c - kNN
    # k = 3
    km = neighbors.KNeighborsClassifier(n_neighbors = 3)
    pred_test = km.fit(data_train, target_train).predict(data_test)
    performance[0][0] += metrics.accuracy_score(target_test, pred_test)
    performance[0][1] += metrics.f1_score(target_test, pred_test)
    performance[0][2] += metrics.roc_auc_score(target_test, pred_test)

    # k = 9
    km = neighbors.KNeighborsClassifier(n_neighbors = 9)
    pred_test = km.fit(data_train, target_train).predict(data_test)
    performance[1][0] += metrics.accuracy_score(target_test, pred_test)
    performance[1][1] += metrics.f1_score(target_test, pred_test)
    performance[1][2] += metrics.roc_auc_score(target_test, pred_test)

    # k = 15
    km = neighbors.KNeighborsClassifier(n_neighbors = 15)
    pred_test = km.fit(data_train, target_train).predict(data_test)
    performance[2][0] += metrics.accuracy_score(target_test, pred_test)
    performance[2][1] += metrics.f1_score(target_test, pred_test)
    performance[2][2] += metrics.roc_auc_score(target_test, pred_test)

    # Q2d - Decision Tree (Two Depths)
    # DT1 - depth 8
    dt = tree.DecisionTreeClassifier(max_depth=8, class_weight="balanced")
    pred_test = dt.fit(data_train, target_train).predict(data_test)
    performance[3][0] += metrics.accuracy_score(target_test, pred_test)
    performance[3][1] += metrics.f1_score(target_test, pred_test)
    performance[3][2] += metrics.roc_auc_score(target_test, pred_test)

    # DT2 - depth 12
    out = tree.DecisionTreeClassifier(max_depth=12, class_weight="balanced")
    pred_test = dt.fit(data_train, target_train).predict(data_test)
    performance[4][0] += metrics.accuracy_score(target_test, pred_test)
    performance[4][1] += metrics.f1_score(target_test, pred_test)
    performance[4][2] += metrics.roc_auc_score(target_test, pred_test)

    # Q2e - Naive bayes
    nb = naive_bayes.GaussianNB()
    pred_test = nb.fit(data_train, target_train).predict(data_test)
    performance[5][0] += metrics.accuracy_score(target_test, pred_test)
    performance[5][1] += metrics.f1_score(target_test, pred_test)
    performance[5][2] += metrics.roc_auc_score(target_test, pred_test)

    # set up the cross-validation procedure
    cv_inner = KFold(n_splits=5, shuffle=True, random_state=1)
    print("Split:", split)
    # Q2f(i) - SVM

    params = [{'kernel':['rbf'], 'C':[0.01, 0.1, 1]},
              {'kernel':['poly'], 'degree':[2], 'C':[0.01, 0.1, 1]},
              {'kernel':['poly'], 'degree':[3], 'C':[0.01, 0.1, 1]},
              {'kernel':['poly'], 'degree':[4], 'C':[0.01, 0.1, 1]}]
    svc = SVC()
    svc_search = GridSearchCV(svc, params, scoring='roc_auc', cv=cv_inner, refit=True)
    result = svc_search.fit(data_train, target_train)

    best_model = result.best_estimator_
    yhat = best_model.predict(data_test)
    performance[6][0] += metrics.accuracy_score(target_test, yhat)
    performance[6][1] += metrics.f1_score(target_test, yhat)
    performance[6][2] += metrics.roc_auc_score(target_test, yhat)

    print("SVC Best Param:", result.best_params_)

    # Q2f(ii) - Random Forests
    forest = RandomForestClassifier()
    space = dict()
    space['n_estimators'] = [25, 50, 100]
    space['max_features'] = [6, 10, 14]

    forest_search = GridSearchCV(forest, space, scoring='roc_auc', cv=cv_inner, refit=True)
    result = forest_search.fit(data_train, target_train)

    best_model = result.best_estimator_
    yhat = best_model.predict(data_test)
    performance[7][0] += metrics.accuracy_score(target_test, yhat)
    performance[7][1] += metrics.f1_score(target_test, yhat)
    performance[7][2] += metrics.roc_auc_score(target_test, yhat)

    print("Random Forest Best Param:", result.best_params_)

    # Q2f(iii) - AdaBoost

    ada = AdaBoostClassifier()
    space = dict()
    space['n_estimators'] = [25, 50]

    ada_search = GridSearchCV(ada, space, scoring='roc_auc', cv=cv_inner, refit=True)
    result = ada_search.fit(data_train, target_train)

    best_model = result.best_estimator_
    yhat = best_model.predict(data_test)
    performance[8][0] += metrics.accuracy_score(target_test, yhat)
    performance[8][1] += metrics.f1_score(target_test, yhat)
    performance[8][2] += metrics.roc_auc_score(target_test, yhat)

    print("Ada Boost Best Param:", result.best_params_)

    split += 1
    print()
# Average of Performance Measures
for i in range(9):
    for n in range(3):
        performance[i][n] = performance[i][n] / 10

performance = pd.DataFrame(performance, columns = cols, index = rows)
display(performance)
```

```
Split: 1
SVC Best Param: {'C': 0.1, 'degree': 2, 'kernel': 'poly'}
Random Forest Best Param: {'max_features': 6, 'n_estimators': 100}
Ada Boost Best Param: {'n_estimators': 50}

Split: 2
SVC Best Param: {'C': 0.1, 'degree': 2, 'kernel': 'poly'}
Random Forest Best Param: {'max_features': 6, 'n_estimators': 100}
Ada Boost Best Param: {'n_estimators': 50}

Split: 3
SVC Best Param: {'C': 0.1, 'degree': 2, 'kernel': 'poly'}
Random Forest Best Param: {'max_features': 6, 'n_estimators': 100}
Ada Boost Best Param: {'n_estimators': 25}

Split: 4
SVC Best Param: {'C': 1, 'degree': 2, 'kernel': 'poly'}
Random Forest Best Param: {'max_features': 14, 'n_estimators': 100}
Ada Boost Best Param: {'n_estimators': 50}

Split: 5
SVC Best Param: {'C': 0.1, 'degree': 2, 'kernel': 'poly'}
Random Forest Best Param: {'max_features': 6, 'n_estimators': 100}
Ada Boost Best Param: {'n_estimators': 50}

Split: 6
SVC Best Param: {'C': 1, 'degree': 2, 'kernel': 'poly'}
Random Forest Best Param: {'max_features': 10, 'n_estimators': 100}
Ada Boost Best Param: {'n_estimators': 50}

Split: 7
SVC Best Param: {'C': 0.1, 'degree': 2, 'kernel': 'poly'}
Random Forest Best Param: {'max_features': 6, 'n_estimators': 100}
Ada Boost Best Param: {'n_estimators': 25}

Split: 8
SVC Best Param: {'C': 0.1, 'degree': 2, 'kernel': 'poly'}
Random Forest Best Param: {'max_features': 14, 'n_estimators': 100}
Ada Boost Best Param: {'n_estimators': 50}

Split: 9
SVC Best Param: {'C': 0.1, 'degree': 2, 'kernel': 'poly'}
Random Forest Best Param: {'max_features': 10, 'n_estimators': 100}
Ada Boost Best Param: {'n_estimators': 50}

Split: 10
SVC Best Param: {'C': 1, 'degree': 2, 'kernel': 'poly'}
Random Forest Best Param: {'max_features': 6, 'n_estimators': 100}
Ada Boost Best Param: {'n_estimators': 50}
```

	Accuracy	F1-Measure	AUC
KNN3	0.783183	0.235911	0.550514
KNN9	0.823058	0.152285	0.532573
KNN15	0.829003	0.090940	0.518780
DT1	0.675859	0.390371	0.653901
DT2	0.675724	0.390251	0.653829
NB	0.586102	0.311041	0.574738
best SVM	0.831841	0.001724	0.500435
best RF	0.832653	0.083874	0.519027
best AdaBoost	0.829272	0.016123	0.501916