Beta

Siddhesh Mahadeshwar

Zachary Noel

Erin Dolson

```
import pandas as pd
import numpy as np
import matplotlib as mpl
import matplotlib.pyplot as plt
import seaborn as sb
import graphviz
%matplotlib inline
from numpy import mean
from numpy import std
from sklearn import datasets # import standard datasets
from sklearn import tree     # decision tree classifier
from sklearn import naive_bayes # naive bayes classifier
from sklearn import svm        # svm classifier
from sklearn.svm import SVC
from sklearn import ensemble   # ensemble classifiers
from sklearn import metrics    # performance evaluation metrics
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import OrdinalEncoder
from sklearn import model_selection
from sklearn import preprocessing
from sklearn import neighbors
from sklearn.preprocessing import MinMaxScaler
from sklearn.model_selection import cross_val_score
from sklearn.tree import DecisionTreeClassifier
from sklearn.model_selection import KFold
from sklearn.ensemble import RandomForestClassifier
from sklearn.datasets import make_classification
from sklearn.model_selection import GridSearchCV
from sklearn.svm import SVC
from sklearn.metrics import accuracy_score
from sklearn.metrics import roc_auc_score
```

a) Load in the hit-movies data. You will not use the original title and imdb id variables for prediction.

```
movies = pd.read_csv("hit-movies.csv",
                     engine='python')
movies = movies[movies.columns.difference(['original_title', 'imdb_id'])]
```

scaling to ensure all variables fall between values of [0,1].

```
# Feature Matrix and Target Array
X = movies[movies.columns.difference(['Hit'])]
Y = movies['Hit']

nFolds = 10
kf = model_selection.StratifiedKFold(n_splits=nFolds, shuffle=True,
random_state=3)
```

c) Use kNN to predict whether a movie is a hit. Estimate the generalization performance over the folds, report the mean accuracy, F1-measure, and AUC on the testing data fork values of 3, 9, and 15.

d) Use decision trees to predict whether a movie is a hit. Estimate the generalization performance over the folds, report the mean accuracy, F1-measure, and AUC on the testing data. Show the results for two different sized trees (consider different amounts of pruning).

e) Use a Naive Bayes classifier to predict whether a movie is a hit. Report the mean accuracy, F1-measure, and AUC on the testing data over the folds.

All three parts are combined in 1 loop below.

```
col_names = ['Knn3', 'Knn9', 'Knn15', 'Decision_Tree1', 'Decision_Tree2', 'NB']

df = pd.DataFrame(columns=['Accuracy','F1-Measure','AUC'],
index=['Knn3','Knn9', 'Knn15', 'Decision_Tree1', 'Decision_Tree2', 'NB',
       'best SVM', 'best RF', 'best AdaBoost'])

k_values=[3,9,15]
sizes = [10, 100]

for x in range(0,len(col_names)):
  knn_kcv_scores = []
  F1_measure_scores = []
  AUC_scores = []

  for tr_indx, te_indx in kf.split(X, Y):
      # scale the data
      x_train, x_test = X.iloc[tr_indx], X.iloc[te_indx]
      y_train, y_test = Y[tr_indx], Y[te_indx]
```

```
      clf.fit(x_train_transformed, y_train)
      y_pred_te = clf.predict(x_test_transformed)
    #Q2d -NB
    else:
      gnb = naive_bayes.GaussianNB()
      y_pred_test = gnb.fit(x_train_transformed, y_train)
      y_pred_te = gnb.predict(x_test_transformed)

    # calculate scores or values
    knn_kcv_scores.append(metrics.accuracy_score(y_test, y_pred_te))
    auc_score1 = roc_auc_score(y_test, y_pred_te)
    f1_val = metrics.f1_score(y_test, y_pred_te, average='binary')
    F1_measure_scores.append(f1_val)
    AUC_scores.append(auc_score1)

knn_value = np.mean(knn_kcv_scores)
f1_value = np.mean(F1_measure_scores)
auc_value = np.mean(AUC_scores)

df.loc[col_names[x]] = pd.Series({'Accuracy':knn_value, 'F1-Measure':f1_value,
                                  'AUC':auc_value})
```

f) Perform a second layer of cross-validation (k=5), an inner loop, to estimate the parameters of the following classifiers. The inner loop of the cross-validation can make use of the methods of grid search to select the best parameterization of the following classifiers. Or, you may elect to use the do-it-yourself approach with a nested loop.

i) Learn support vector machine (SVM) models to predict whether a movie is a hit. You will consider multiple classifiers using both the RBF kernel (with default values) and polynomial kernel with degree 2, 3, and 4. Consider values for cost penalty parameter of {0.01,0.1,1}. Report the best parameter values (kernel + cost) for each outer fold (selected by AUC).

ii) Use Random Forests to predict whether a movie is a hit. Consider multiple random forests with the number of trees in the forest to be {25,50,100} and the maximum number of features to

```
    _
i = 0
for train_ix, test_ix in cv_outer.split(X):
  accuracy_scores = []
  F1_measure_scores = []
  AUC_scores = []
  # split data
  X_train = X.iloc[train_ix, :]
  X_test = X.iloc[test_ix, :]
  y_train = Y[train_ix]
  y_test = Y[test_ix]

  # scale the data
  y = MinMaxScaler().fit(X_train)
  x_train_transformed = y.transform(X_train)
  x_test_transformed = y.transform(X_test)

  # set up the cross-validation procedure
  cv_inner = KFold(n_splits=5, shuffle=True, random_state=1)
  # Q2f i - SVM
  if num == 0:
    # define the model
    svmModel = SVC(random_state=1)

    # defining parameter range
    param_grid = {'C': [0.01, 0.1, 1],
                  'kernel': ('rbf', 'poly'),
                  'degree': [2, 3, 4]
                  }

    svmSearch = GridSearchCV(svmModel, param_grid, scoring='roc_auc',
                            cv=cv_inner, refit=True, n_jobs=5)
    svmResult = svmSearch.fit(x_train_transformed, y_train)
```

```
      rfSearch = GridSearchCV(rfModel, rfSpace, scoring='roc_auc',
                              cv=cv_inner,
                              refit=True)
      rfResult = rfSearch.fit(x_train_transformed, y_train)

      # evaluate performance of best model
      best_model_rf = rfResult.best_estimator_
      yhat = best_model_rf.predict(x_test_transformed)
      acc = accuracy_score(y_test, yhat)
      auc = roc_auc_score(y_test, yhat)

      # store the results
      outer_results.append(auc)

      # report progress
      print(' split=%d, acc=%.3f, auc=%.3f, est=%.3f, cfg=%s' %
(i, acc, auc, rfResult.best_score_, rfResult.best_params_))
      i += 1
    # Q2f iii - Adaboost
    elif num == 2:
      adaModel = ensemble.AdaBoostClassifier(random_state=1)

      # define search space, complete the search over the inner cv loop
      adaSpace = dict()
      adaSpace['n_estimators'] = [25, 50]
```

```
    auc_value = np.mean(AUC_scores)

    df.loc[gridsearch[num]] = pd.Series({'Accuracy':accuracy_value,
                                        'F1-Measure':f1_value, 'AUC':auc_value})
```

```
best SVM
 split=0, auc=0.500, est=0.675, cfg={'C': 0.1, 'degree': 2, 'kernel': 'poly'}
 split=1, auc=0.500, est=0.686, cfg={'C': 0.1, 'degree': 2, 'kernel': 'poly'}
 split=2, auc=0.500, est=0.662, cfg={'C': 0.1, 'degree': 2, 'kernel': 'poly'}
 split=3, auc=0.500, est=0.682, cfg={'C': 0.1, 'degree': 2, 'kernel': 'poly'}
 split=4, auc=0.500, est=0.678, cfg={'C': 0.1, 'degree': 2, 'kernel': 'poly'}
 split=5, auc=0.504, est=0.681, cfg={'C': 1, 'degree': 2, 'kernel': 'poly'}
 split=6, auc=0.500, est=0.676, cfg={'C': 0.01, 'degree': 2, 'kernel': 'poly'
 split=7, auc=0.500, est=0.674, cfg={'C': 0.1, 'degree': 2, 'kernel': 'poly'}
 split=8, auc=0.500, est=0.674, cfg={'C': 0.1, 'degree': 2, 'kernel': 'poly'}
 split=9, auc=0.500, est=0.679, cfg={'C': 1, 'degree': 2, 'kernel': 'poly'}
best RF
 split=0, acc=0.845, auc=0.509, est=0.694, cfg={'max_features': 6, 'n_estimat
 split=1, acc=0.861, auc=0.534, est=0.693, cfg={'max_features': 14, 'n_estima
 split=2, acc=0.808, auc=0.514, est=0.694, cfg={'max_features': 6, 'n_estimat
 split=3, acc=0.835, auc=0.502, est=0.692, cfg={'max_features': 10, 'n_estima
```