**#Simple K-fold Cross-Validation**

from sklearn.model\_selection import cross\_val\_score

from sklearn import tree

from sklearn.preprocessing import StandardScaler

from sklearn.model\_selection import train\_test\_split

# read the data in

df = pd.read\_csv("F:/Jan-May\_\_\_2023/CS317\_\_slides/Python\_Programs\_tutorials/Diabetes.csv")

X = df.iloc[:,:8].values # independent variables

y = df['class'].values # dependent variables

# Normalize Data

sc = StandardScaler()

sc.fit(X)

X = sc.transform(X)

# evaluate the model by splitting into train and test sets

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.3,

random\_state=2017)

# build a decision tree classifier

clf = tree.DecisionTreeClassifier(random\_state=2017)

# evaluate the model using 10-fold cross-validation

train\_scores = cross\_val\_score(clf, X\_train, y\_train, scoring='accuracy', cv=5)

test\_scores = cross\_val\_score(clf, X\_test, y\_test, scoring='accuracy', cv=5)

print ("Train Fold AUC Scores: ", train\_scores)

print ("Train CV AUC Score: ", train\_scores.mean())

print ("\nTest Fold AUC Scores: ", test\_scores)

print ("Test CV AUC Score: ", test\_scores.mean())

**Output:**

Train Fold AUC Scores: [0.7037037 0.63888889 0.64485981 0.6635514 0.70093458]

Train CV AUC Score: 0.6703876773970231

Test Fold AUC Scores: [0.70212766 0.67391304 0.76086957 0.65217391 0.67391304]

Test CV AUC Score: 0.6925994449583719

**#Stratified K-fold Cross-Validation**

from sklearn import model\_selection

import numpy as np

kfold = model\_selection.StratifiedKFold(n\_splits=5, random\_state=2019, shuffle=True)

train\_scores = []

test\_scores = []

k = 0

for (train, test) in kfold.split(X\_train, y\_train):

clf.fit(X\_train[train], y\_train[train])

train\_score = clf.score(X\_train[train], y\_train[train])

train\_scores.append(train\_score)

# score for test set

test\_score = clf.score(X\_train[test], y\_train[test])

test\_scores.append(test\_score)

k += 1

print('Fold: %s, Class dist.: %s, Train Acc: %.3f, Test Acc: %.3f'

% (k, np.bincount(y\_train[train]), train\_score, test\_score))

print('\nTrain CV accuracy: %.3f' % (np.mean(train\_scores)))

print('Test CV accuracy: %.3f' % (np.mean(test\_scores)))

**Output:**

Fold: 1, Class dist.: [277 152], Train Acc: 1.000, Test Acc: 0.630

Fold: 2, Class dist.: [277 152], Train Acc: 1.000, Test Acc: 0.694

Fold: 3, Class dist.: [278 152], Train Acc: 1.000, Test Acc: 0.757

Fold: 4, Class dist.: [278 152], Train Acc: 1.000, Test Acc: 0.701

Fold: 5, Class dist.: [278 152], Train Acc: 1.000, Test Acc: 0.673

Train CV accuracy: 1.000

Test CV accuracy: 0.691

**#Plotting the ROC Curve for Stratified K-fold Cross-Validation**

import warnings

warnings.filterwarnings('ignore')

from sklearn.metrics import roc\_curve, auc

from itertools import cycle

from scipy import interp

import matplotlib.pyplot as plt

kfold = model\_selection.StratifiedKFold(n\_splits=5, random\_state=2019, shuffle=True)

mean\_tpr = 0.0

mean\_fpr = np.linspace(0, 1, 100)

colors = cycle(['cyan', 'indigo', 'seagreen', 'yellow', 'blue', 'darkorange'])

lw = 2

i = 0

for (train, test), color in zip(kfold.split(X, y), colors):

probas\_ = clf.fit(X[train], y[train]).predict\_proba(X[test])

# Compute ROC curve and area the curve

fpr, tpr, thresholds = roc\_curve(y[test], probas\_[:, 1])

mean\_tpr += interp(mean\_fpr, fpr, tpr)

mean\_tpr[0] = 0.0

roc\_auc = auc(fpr, tpr)

plt.plot(fpr, tpr, lw=lw, color=color,

label='ROC fold %d (area = %0.2f)' % (i, roc\_auc))

i += 1

plt.plot([0, 1], [0, 1], linestyle='--', lw=lw, color='k', label='Luck')

mean\_tpr /= kfold.get\_n\_splits(X, y)

mean\_tpr[-1] = 1.0

mean\_auc = auc(mean\_fpr, mean\_tpr)

plt.plot(mean\_fpr, mean\_tpr, color='g', linestyle='--',

label='Mean ROC (area = %0.2f)' % mean\_auc, lw=lw)

plt.xlim([-0.05, 1.05])

plt.ylim([-0.05, 1.05])

plt.xlabel('False Positive Rate')

plt.ylabel('True Positive Rate')

plt.title('Receiver operating characteristic example')

plt.legend(loc="lower right")

plt.show()

**Output:**

