**A)**

import pandas as pd

import numpy as np

from sklearn.ensemble import RandomForestClassifier

import matplotlib.pyplot as plt

from sklearn.metrics import roc\_curve

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

from sklearn.model\_selection import StratifiedKFold, LeaveOneOut

get\_ipython().run\_line\_magic('matplotlib', 'inline')

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

from sklearn.model\_selection import LeaveOneOut

from sklearn.metrics import confusion\_matrix

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

from sklearn.model\_selection import GridSearchCV

from sklearn.model\_selection import ShuffleSplit

from sklearn.ensemble import RandomForestRegressor

import operator

import pandas as pd

import math

import matplotlib.pyplot as plt

from sklearn.model\_selection import KFold

from sklearn.gaussian\_process import GaussianProcessClassifier

from sklearn.ensemble import RandomForestClassifier

from sklearn import svm

from sklearn.decomposition import PCA

from sklearn import svm, datasets

from sklearn.metrics import roc\_curve, auc

from sklearn.model\_selection import StratifiedKFold, LeaveOneOut

from sklearn.ensemble import RandomForestClassifier

from sklearn.decomposition import PCA

from numpy import interp

import scipy.stats as stats

**B)**

data = pd.read\_csv(r'/Users/neginkazemian/Desktop/Book.csv', index\_col='#SampleID')

y = pd.read\_csv(r'/Users/neginkazemian/Desktop/Book.csv')['ACE']

data.drop('ACE', axis = 1, inplace=True)

data.shape

**C)**

# PCA + RF Analysis to find top features

# 1) PCA

X = data

pca = PCA(n\_components=0.98)

X\_reduced = pca.fit\_transform(data)

pca.explained\_variance\_ratio\_

Comp\_p2 = pd.DataFrame(pca.components\_, columns=X.columns).T\*\*2

Comp\_p2['sum'] = Comp\_p2.sum(axis=1)

X\_20PCA\_Reduced = X[Comp\_p2.sort\_values(by='sum',ascending=False).index[0:20]]

# 2) RF

p = X\_20PCA\_Reduced

m = np.zeros(p.shape[1]).tolist()

for i in range(100):

rf = RandomForestClassifier(n\_jobs=-1, n\_estimators=200)

model = rf.fit(p, y)

m1 = model.feature\_importances\_.tolist()

m = list(map(operator.add, m,m1))

# to make a table for important features

n = p.columns.tolist()

np.append([m],[n], axis=0)

fe\_imp= pd.DataFrame(np.append([m],[n], axis=0)).T

fe\_imp[0] = fe\_imp[0].astype(float)#\*\*6

fe\_imprt = fe\_imp.sort\_values(by=[0], ascending=False)#.iloc[0:20]

**D)**

fe\_imprt.to\_csv('features.csv')

**E)**

%pwd

X\_20PCA\_Reduced.shape

pd.DataFrame(X\_reduced)

**F)**

## Plot ROC Curve and measure the model accuracy

x = data

cv= 1

scores = []

y\_tests = []

y\_predicts = []

y\_predictions = []

tprs = []

aucs = []

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

plt.figure(figsize=(10, 7.5))

for i in range(10):

rf = RandomForestClassifier(n\_jobs=-1, n\_estimators=200, max\_features = 'auto')

rf\_acc = cross\_val\_score(rf, x, y, cv=cv, n\_jobs=-1)

rf\_proba = cross\_val\_predict(rf, x,y, cv=cv, method='predict\_proba')

rf\_confusion = cross\_val\_predict(rf, x, y, cv=cv)

rf\_conf = confusion\_matrix(y,rf\_confusion)

rf\_scores = rf\_proba[:, 1] # Get the probability of the positive class

fpr, tpr, thresholds = roc\_curve(y,rf\_scores)

tprs.append(interp(mean\_fpr, fpr, tpr))

tprs[-1][0] = 0.0

roc\_auc = auc(fpr, tpr)

aucs.append(roc\_auc)

plt.plot(fpr, tpr, lw=1, alpha=0.5,c='steelblue',linestyle='--')

i += 1

scores.append(rf\_acc)

y\_tests.append(y)

y\_predicts.append(rf\_scores)

y\_predictions.append(rf\_confusion)

plt.plot([0, 1], [0, 1], linestyle='-.', lw=2, color='black',

label='Luck', alpha=.8)

mean\_tpr = np.mean(tprs, axis=0)

mean\_tpr[-1] = 1.0

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

std\_auc = np.std(aucs)

plt.plot(mean\_fpr, mean\_tpr, color='teal', label=r'Mean ROC' ,lw=3, alpha=1)

plt.xlim([-0.05, 1.05])

plt.ylim([-0.05, 1.05])

plt.xlabel('False Positive Rate')

plt.ylabel('True Positive Rate')

plt.title(r'ROC Curve -- Classifier: Random Forests -- AUC = %0.2f ' % (mean\_auc))

plt.legend(loc="lower right")

plt.show()