## **Lab Assignment-6**

**Afsarul Islam Meraj**

**18301239**

**Sec-3**

# Importing the libraries

import numpy as np

import pandas as pd

# Loading the dataset as dataframe using pandas

heart\_df=pd.read\_csv('/content/sample\_data/heart failur classification dataset.csv')

heart\_df

# Dropping the unnecessary column

heart\_df.drop("Unnamed: 0", axis=1, inplace=True)

heart\_df.shape

# Handling the missing values

heart\_df.isnull().sum()

from sklearn.impute import SimpleImputer

simple\_imputer = SimpleImputer(missing\_values=np.nan, strategy='mean')

simple\_imputer.fit(heart\_df[['serum\_sodium','time']])

heart\_df[['serum\_sodium','time']]= simple\_imputer.transform(heart\_df[['serum\_sodium','time']])

heart\_df

# Encoding categorical features

heart\_df.info()

heart\_df['sex'].unique()

heart\_df['smoking'].unique()

from sklearn.preprocessing import LabelEncoder

label\_encoder = LabelEncoder()

heart\_df['smoking'] = label\_encoder.fit\_transform(heart\_df['smoking'])

heart\_df['sex'] = label\_encoder.fit\_transform(heart\_df['sex'])

heart\_df.head()

# Scaling all the values between 0-1 using MinMax Scaler

from sklearn.preprocessing import MinMaxScaler

minmax\_scaler = MinMaxScaler()

minmax\_scaler.fit(heart\_df)

heart\_df=pd.DataFrame(minmax\_scaler.transform(heart\_df))

# Splitting the dataset into features and labels

features= heart\_df.iloc[:,:-1]

features.shape

labels= heart\_df.iloc[:,12]

labels.shape

# Training set and testing set splitting (8:2)

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

features\_train, features\_test, labels\_train, labels\_test = train\_test\_split(features, labels, test\_size=0.2, random\_state=1)

# SVC

from sklearn.svm import SVC

svc= SVC(kernel="linear")

svc.fit(features\_train, labels\_train)

labels\_predict\_svc= svc.predict(features\_test)

accuracy\_svc= svc.score(features\_test, labels\_test)

accuracy\_svc

# Neural Network (MLPClassifier)

from sklearn.neural\_network import MLPClassifier

mlpc= MLPClassifier(activation="relu",hidden\_layer\_sizes=7,max\_iter=10000)

mlpc.fit(features\_train, labels\_train)

labels\_predict\_mlpc=mlpc.predict(features\_test)

accuracy\_mlpc=mlpc.score(features\_test, labels\_test)

accuracy\_mlpc

# Random Forest

from sklearn.ensemble import RandomForestClassifier

rfc= RandomForestClassifier(n\_estimators=50)

rfc.fit(features\_train, labels\_train)

labels\_predict\_rfc=rfc.predict(features\_test)

accuracy\_rfc= rfc.score(features\_test, labels\_test)

accuracy\_rfc

# Performing dimensionality reduction using PCA

from sklearn.decomposition import PCA

pca = PCA(n\_components=6)

principal\_components= pca.fit\_transform(features)

column\_names=["Principal Comp.-"+str(i) for i in range(1,7)]

after\_pca\_features= pd.DataFrame(data=principal\_components, columns=column\_names)

after\_pca\_labels=labels

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

after\_pca\_features\_train, after\_pca\_features\_test, after\_pca\_labels\_train, after\_pca\_labels\_test = train\_test\_split(after\_pca\_features, after\_pca\_labels, test\_size=0.2, random\_state=1)

# Post-PCA SVC

from sklearn.svm import SVC

svc= SVC(kernel="linear")

svc.fit(after\_pca\_features\_train, after\_pca\_labels\_train)

after\_pca\_labels\_predict\_svc= svc.predict(after\_pca\_features\_test)

after\_pca\_accuracy\_svc= svc.score(after\_pca\_features\_test, after\_pca\_labels\_test)

after\_pca\_accuracy\_svc

# Post-PCA Neural Network (MLPClassifier)

from sklearn.neural\_network import MLPClassifier

mlpc= MLPClassifier(activation="relu",hidden\_layer\_sizes=7,max\_iter=10000)

mlpc.fit(after\_pca\_features\_train, after\_pca\_labels\_train)

after\_pca\_labels\_predict\_mlpc=mlpc.predict(after\_pca\_features\_test)

after\_pca\_accuracy\_mlpc= mlpc.score(after\_pca\_features\_test, after\_pca\_labels\_test)

after\_pca\_accuracy\_mlpc

# Post-PCA Random Forest

from sklearn.ensemble import RandomForestClassifier

rfc= RandomForestClassifier(n\_estimators=50)

rfc.fit(after\_pca\_features\_train, after\_pca\_labels\_train)

after\_pca\_labels\_predict\_rfc=rfc.predict(after\_pca\_features\_test)

after\_pca\_accuracy\_rfc= rfc.score(after\_pca\_features\_test, after\_pca\_labels\_test)

after\_pca\_accuracy\_rfc

# Comparing Pre-PCA SVC & Post\_PCA SVC

import matplotlib.pyplot as plt

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

svcs = ['Pre-PCA SVC', 'Post-PCA SVC']

accuracies\_svc= [accuracy\_svc, after\_pca\_accuracy\_svc]

plt.bar(svcs, accuracies\_svc)

plt.show()

# Comparing Pre-PCA Neural Network & Post\_PCA Neural Network

import matplotlib.pyplot as plt

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

mlpcs = ['Pre-PCA Neural Network', 'Post-PCA Neural Network']

accuracies\_mlpc= [accuracy\_mlpc, after\_pca\_accuracy\_mlpc]

plt.bar(mlpcs, accuracies\_mlpc)

plt.show()

# Comparing Pre-PCA Random Forest & Post\_PCA Random Forest

import matplotlib.pyplot as plt

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

rfcs = ['Pre-PCA Random Forest', 'Post-PCA Random Forest']

accuracies\_rfc= [accuracy\_rfc, after\_pca\_accuracy\_rfc]

plt.bar(rfcs, accuracies\_rfc)

plt.show()