import matplotlib.pyplot as plt

import pandas as pd

import numpy as np

data=pd.read\_csv('/content/breast\_cancer\_classification\_dataset.csv')

data.head()

data.isnull().sum()

data=data.drop(['id','Unnamed: 32'],1)

from sklearn.impute import SimpleImputer

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

impute.fit(data[['radius\_mean']])

data[['radius\_mean']]=impute.transform(data[['radius\_mean']])

impute.fit(data[['fractal\_dimension\_worst']])

data[['fractal\_dimension\_worst']]=impute.transform(data[['fractal\_dimension\_worst']])

data.isnull().sum()

data.info()

print(data['diagnosis'].unique())

from sklearn.preprocessing import LabelEncoder

enc = LabelEncoder()

data['diagnosis'] = enc.fit\_transform(data['diagnosis'])

data.info()

features=['radius\_mean','texture\_mean','perimeter\_mean','area\_mean','smoothness\_mean','compactness\_mean','concavity\_mean','concave points\_mean','symmetry\_mean','fractal\_dimension\_mean','radius\_se','texture\_se','perimeter\_se','area\_se','smoothness\_se','compactness\_se','concavity\_se','concave points\_se','symmetry\_se','fractal\_dimension\_se','radius\_worst','texture\_worst','perimeter\_worst','area\_worst','smoothness\_worst','compactness\_worst','concavity\_worst','concave points\_worst','symmetry\_worst','fractal\_dimension\_worst']

label=['diagnosis']

X=data[features]

y=data[label]

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

X\_train,X\_test,y\_train,y\_test=train\_test\_split(X,y,random\_state=1)

from sklearn.svm import SVC

svc = SVC(kernel="linear")

svc.fit(X\_train, y\_train)

prePcaSVM\_train=round(svc.score(X\_train, y\_train),2)\*100

prePcaSVM\_test=round(svc.score(X\_test, y\_test),2)\*100

print("Training accuracy of the model is {:.2f}".format(svc.score(X\_train, y\_train)))

print("Testing accuracy of the model is {:.2f}".format(svc.score(X\_test, y\_test)))

from sklearn.ensemble import RandomForestClassifier

rfc = RandomForestClassifier(n\_estimators=100)

rfc.fit(X\_train, y\_train)

prePcaRFC\_train=round(rfc.score(X\_train, y\_train),2)\*100

prePcaRFC\_test=round(rfc.score(X\_test, y\_test),2)\*100

print("The Training accuracy of the model is {:.2f}".format(rfc.score(X\_train, y\_train)))

print("The Testing accuracy of the model is {:.2f}".format(rfc.score(X\_test, y\_test)))

from sklearn.neural\_network import MLPClassifier

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

nnc.fit(X\_train, y\_train)

prePcaNNC\_train=round(nnc.score(X\_train, y\_train),2)\*100

prePcaNNC\_test=round(nnc.score(X\_test, y\_test),2)\*100

print("The Training accuracy of the model is {:.2f}".format(nnc.score(X\_train, y\_train)))

print("The Testing accuracy of the model is {:.2f}".format(nnc.score(X\_test, y\_test)))

from sklearn.decomposition import PCA

pca = PCA(n\_components=5)

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

principal\_df = pd.DataFrame(data=principal\_components, columns=["principle component 1", "principle component 2","principle component 3","principle component 4","principle component 5"])

main\_df=pd.concat([principal\_df, data[["diagnosis"]]], axis=1)

main\_df.head()

x = main\_df.iloc[:,1:-1]

y = main\_df.iloc[:,-1]

X\_train,X\_test,y\_train,y\_test = train\_test\_split(X,y,test\_size=0.2,random\_state=12,stratify=y)

svc = SVC(kernel="linear")

svc.fit(X\_train, y\_train)

postPcaSVM\_train=round(svc.score(X\_train, y\_train),2)\*100

postPcaSVM\_test=round(svc.score(X\_test, y\_test),2)\*100

print("Training accuracy of the model is {:.2f}".format(svc.score(X\_train, y\_train)))

print("Testing accuracy of the model is {:.2f}".format(svc.score(X\_test, y\_test)))

rfc = RandomForestClassifier(n\_estimators=100)

rfc.fit(X\_train, y\_train)

postPcaRFC\_train=round(rfc.score(X\_train, y\_train),2)\*100

postPcaRFC\_test=round(rfc.score(X\_test, y\_test),2)\*100

print("The Training accuracy of the model is {:.2f}".format(rfc.score(X\_train, y\_train)))

print("The Testing accuracy of the model is {:.2f}".format(rfc.score(X\_test, y\_test)))

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

nnc.fit(X\_train, y\_train)

postPcaNNC\_train=round(nnc.score(X\_train, y\_train),2)\*100

postPcaNNC\_test=round(nnc.score(X\_test, y\_test),2)\*100

print("The Training accuracy of the model is {:.2f}".format(nnc.score(X\_train, y\_train)))

print("The Testing accuracy of the model is {:.2f}".format(nnc.score(X\_test, y\_test)))

data = [[prePcaSVM\_test, prePcaRFC\_test, prePcaNNC\_test],

[postPcaSVM\_test, postPcaRFC\_test, postPcaNNC\_test],

[prePcaSVM\_train, prePcaRFC\_train, prePcaNNC\_train],

[postPcaSVM\_train, postPcaRFC\_train, postPcaNNC\_train]]

X = np.arange(3)

figure = plt.figure()

a = figure.add\_axes([0,0,1,1])

a.bar(X + 0.00, data[0], color = 'blue', width = 0.20)

a.bar(X + 0.20, data[1], color = 'green', width = 0.20)

a.bar(X + 0.40, data[2], color = 'red', width = 0.20)

a.bar(X + 0.60, data[3], color = 'yellow', width = 0.20)

a.legend(labels=['Pre PCA Train','Post PCA Train','Pre PCA Test', 'Post PCA Test'],loc='upper left',bbox\_to\_anchor=(1, 1))

plt.ylabel('Percentage')

plt.xlabel('SVM (left), Random Forest (middle), NNC (right)')

plt.title('Test Score')