CSE422 Lab Assignment 4

Md. Shamiul Islam

ID: 17301108

Theory Section: 8

Lab Section: 4

import numpy as np

import pandas as pd

from sklearn import svm

import matplotlib.pyplot as plt

%matplotlib inline

shamiul = pd.read\_csv("/content/mushroom edibility classification dataset.csv")

shamiul.head()

shamiul = shamiul.drop(['Unnamed: 0'],axis = 1)

shamiul.describe().T

shamiul.isnull().sum()

#Encoding part

from sklearn import preprocessing

label\_encoder = preprocessing.LabelEncoder()

shamiul['class'].value\_counts()

Class = label\_encoder.fit\_transform(shamiul['class'])

Class = pd.DataFrame({'class': Class})

t = label\_encoder.fit\_transform(shamiul['bruises'])

t = pd.DataFrame({'bruises': t})

shamiul['class'] = Class['class'].values

shamiul['bruises'] = t['bruises'].values

shamiul.head()

shamiul.fillna(shamiul.mean(), inplace=True)

shamiul.isnull().sum()

from pandas.plotting import scatter\_matrix

scatter\_matrix(shamiul[['cap-shape','odor','veil-type','ring-type','habitat']],

s=shamiul["class"] , figsize=(12,12), alpha=1)

X = shamiul.drop(columns =['class'])

y = shamiul['class']

#Split dataset in 80/20

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

x\_train, x\_test, y\_train, y\_test= train\_test\_split(X, y, stratify=y, test\_size=0.2)

print("training set: x->{} , y->{} \n testing set: x->{} , y->{}".format(x\_train.shape, y\_train.shape, x\_test.shape, y\_test.shape))

nameOfClassifiers=[ ]

Accuracy=[ ]

#Support vector machine

from sklearn.svm import SVC

svc = SVC(kernel="linear")

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

print("t accuracy of SVC model {:.2f}".format(svc.score(x\_train, y\_train)))

nameOfClassifiers.append("SVC before PCA\_Train")

Accuracy.append(svc.score(x\_train, y\_train))

print("t accuracy of SVC model {:.2f}".format(svc.score(x\_test, y\_test)))

nameOfClassifiers.append("SVC before PCA\_Test")

Accuracy.append(svc.score(x\_test, y\_test))

#Neural Network

from sklearn.neural\_network import MLPClassifier

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

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

print("training accuracy of the model {:.2f}".format(nnc.score(x\_train, y\_train)))

nameOfClassifiers.append("NNC before PCA\_Train")

Accuracy.append(nnc.score(x\_train, y\_train))

print("testing accuracy of the model {:.2f}".format(nnc.score(x\_test, y\_test)))

nameOfClassifiers.append("NNC before PCA\_Test")

Accuracy.append(nnc.score(x\_test, y\_test))

#Random Forest

from sklearn.ensemble import RandomForestClassifier

rfc = RandomForestClassifier(n\_estimators=50)

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

print("training accuracy of the model {:.2f}".format(rfc.score(x\_train, y\_train)))

nameOfClassifiers.append("RFC before PCA\_train")

Accuracy.append(rfc.score(x\_train, y\_train))

print("testing accuracy of the model {:.2f}".format(rfc.score(x\_test, y\_test)))

nameOfClassifiers.append("RFC before PCA\_test")

Accuracy.append(rfc.score(x\_test, y\_test))

import matplotlib.pyplot as plt

fig = plt.figure()

ax = fig.add\_axes([0,0,1.75,1])

print("training & testing accuracy check before applying PCA")

y\_pos = range(len(nameOfClassifiers))

ax.bar(nameOfClassifiers,Accuracy)

shamiul.columns.shape

type(shamiul)

shamiul.head()

#SVC

from sklearn.decomposition import PCA

pca = PCA(n\_components=10)

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

print(principal\_components)

pca.explained\_variance\_ratio\_

sum(pca.explained\_variance\_ratio\_)

cols=["PC"+str(i) for i in range(1,11)]

print(cols)

principal\_shamiul = pd.DataFrame(data=principal\_components, columns=cols)

main\_shamiul=pd.concat([principal\_shamiul, shamiul[["habitat"]]], axis=1)

main\_shamiul.head()

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

ax = fig.add\_subplot(1,1, 1)

ax.set\_xlabel('10 principal component', fontsize = 12)

ax.set\_ylabel('10 principal component' , fontsize = 12)

ax.set\_title('10 component pca', fontsize = 12)

targets = [1, 0]

colors = ['b', 'y']

for target, color in zip(targets,colors):

indicesToKeep = main\_shamiul['habitat'] == target

#print(indicesToKeep)

ax.scatter(main\_shamiul.loc[indicesToKeep, 'PC1'], main\_shamiul.loc[indicesToKeep, 'PC2'], c = color, s = 50)

ax.legend(["poisonus", "edible"])

ax.grid()

X= main\_shamiul.drop("habitat", axis=1)

y=main\_shamiul["habitat"]

x\_train, x\_test, y\_train, y\_test = train\_test\_split(X , y , test\_size=0.2,random\_state=42)

from sklearn.svm import SVC

svc = SVC(kernel="linear")

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

print("now after PCA Training accuracy of SVC model {:.2f}".format(svc.score(x\_train, y\_train)))

nameOfClassifiers.append("SVC after PCA\_Train")

Accuracy.append(svc.score(x\_train, y\_train))

print("now after PCA Testing accuracy of SVC model {:.2f}".format(svc.score

(x\_test, y\_test)))

nameOfClassifiers.append("SVC after PCA\_Test")

Accuracy.append(svc.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)

print("After PCA The Training accuracy of the model is {:.2f}".format(nnc.

score(x\_train, y\_train)))

nameOfClassifiers.append("NNC after PCA\_train")

Accuracy.append(nnc.score(x\_train, y\_train))

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

nameOfClassifiers.append("NNC after PCA\_test")

Accuracy.append(nnc.score(x\_test, y\_test))

from sklearn.ensemble import RandomForestClassifier

rfc = RandomForestClassifier(n\_estimators=50)

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

print("after PCA training accuracy of the model {:.2f}".format(rfc.score(x\_train, y\_train)))

nameOfClassifiers.append("RFC after PCA\_Train")

Accuracy.append(rfc.score(x\_train, y\_train))

print("ater PCA testing accuracy of the model {:.2f}".format(rfc.score(x\_test, y\_test)))

nameOfClassifiers.append("RFC after PCA\_Test")

Accuracy.append(rfc.score(x\_test, y\_test))

import matplotlib.pyplot as plt

fig = plt.figure()

ax = fig.add\_axes([0,0,4,1])

print("training and testing accuracy before and after doing PCA:")

y\_pos = range(len(nameOfClassifiers))

ax.bar(nameOfClassifiers,Accuracy)