CSE422 Lab Assignment 5

Md. Shamiul Islam

ID: 17301108  
Theory Section: 8

Lab Section: 4

import numpy as np

import pandas as pd

import sklearn as skl

import matplotlib.pyplot as plt

%matplotlib inline

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

shamiul.head()

shamiul.shape

shamiul.isnull().sum()

columns = ['cap-shape', 'cap-color']

shamiul[columns] = shamiul[columns].fillna(shamiul.mode().iloc[0])

shamiul.isnull().sum()

from sklearn.preprocessing import LabelEncoder

enc = LabelEncoder()

shamiul['class'] = enc.fit\_transform(shamiul['class'])

shamiul['bruises'] = enc.fit\_transform(shamiul['bruises'])

shamiul.head()

shamiul = shamiul.drop("Unnamed: 0", axis = 1)

shamiul.head()

type(shamiul)

x = shamiul.drop("class", axis = 1)

y = shamiul["class"]

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

#from sklearn.metrics import classification\_report

#from sklearn.metrics import accuracy\_score

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

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

#split the data into 80% training and 20% testing

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

#target = shamiul["class"].values

#dataset = shamiul.drop(["class"], axis=1).values

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

from sklearn.preprocessing import StandardScaler

sc=StandardScaler()

x\_train = sc.fit\_transform(x\_train)

x\_test = sc.fit\_transform(x\_test)

d = sc.fit\_transform(x)

d = pd.DataFrame(x)

from sklearn.metrics import accuracy\_score, confusion\_matrix

import matplotlib.pyplot as plt

from sklearn.linear\_model import LogisticRegression

from sklearn.metrics import f1\_score, auc

from sklearn.metrics import roc\_auc\_score

from sklearn.linear\_model import LogisticRegression

clf = LogisticRegression()

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

y\_pred = clf.predict(x\_test)

print(confusion\_matrix(y\_test, y\_pred))

m = (accuracy\_score(y\_test,y\_pred))

print(m)

from sklearn.tree import DecisionTreeClassifier

clf1 = DecisionTreeClassifier(random\_state=50, min\_samples\_split= 2)

clf1.fit(x\_train, y\_train)

y\_pred = clf1.predict(x\_test)

print(confusion\_matrix(y\_test, y\_pred))

n = (accuracy\_score(y\_test,y\_pred))

print(n)

plt.figure(figsize=(8, 6))

classifiers = ['Logistic Regression', 'Decision Tree']

Accuracy = [m, n]

plt.bar(classifiers, Accuracy)

plt.show()

#random forest

from sklearn.ensemble import RandomForestClassifier

rfc = RandomForestClassifier(n\_estimators=50)

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

predictionsRFC = rfc.predict(x\_test)

print("Accuracy using Random Forest: {}%".format(round(accuracy\_score(y\_test, predictionsRFC)\*100,2)) )

prePCArfc=accuracy\_score(y\_test, predictionsRFC)\*100

from sklearn.ensemble import RandomForestClassifier

clf = RandomForestClassifier()

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

y\_pred = clf.predict(x\_test)

RF = print(accuracy\_score(y\_test,y\_pred))

from sklearn import svm

clf2 = svm.SVC()

clf2.fit(x\_train, y\_train)

y\_pred = clf2.predict(x\_test)

SV = print(accuracy\_score(y\_test,y\_pred))

from sklearn.neural\_network import MLPClassifier

clf3 = MLPClassifier(solver='lbfgs', alpha=1e-5,max\_iter=30,hidden\_layer\_sizes=(3, 2), random\_state=1)

clf3.fit(x\_train, y\_train)

y\_pred = clf3.predict(x\_test)

ML = print(accuracy\_score(y\_test,y\_pred))

from sklearn.decomposition import PCA

pca = PCA(n\_components = 6)

x\_train = pca.fit\_transform(x\_train)

x\_test = pca.transform(x\_test)

from sklearn.ensemble import RandomForestClassifier

clf = RandomForestClassifier()

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

y\_pred = clf.predict(x\_test)

RF1=print(accuracy\_score(y\_test,y\_pred))

from sklearn import svm

clf2 = svm.SVC()

clf2.fit(x\_train, y\_train)

y\_pred = clf2.predict(x\_test)

SV1 = print(accuracy\_score(y\_test,y\_pred))

from sklearn.neural\_network import MLPClassifier

clf3 = MLPClassifier(solver='lbfgs', alpha=1e-5,max\_iter=30,hidden\_layer\_sizes=(3, 2), random\_state=1)

clf3.fit(x\_train, y\_train)

y\_pred = clf3.predict(x\_test)

ML1 = print(accuracy\_score(y\_test,y\_pred))

A = 2

ind = np.arange(A)

width = 0.27

fig = plt.figure()

ax = fig.add\_subplot(111)

yvals = [1, 0.998]

rects1 = ax.bar(ind, yvals, width, color='y')

zvals = [1, 0.992]

rects2 = ax.bar(ind+width, zvals, width, color='r')

kvals = [0.9136, 0.9216]

rects3 = ax.bar(ind+width\*2, kvals, width, color='g')

ax.set\_ylabel('Accuracy')

ax.set\_xticks(ind+width)

ax.set\_xticklabels( ('Pre-PCA', 'Post-PCA') )

ax.legend( (rects1[0], rects2[0], rects3[0]), ('Random Forest', 'SVM', 'MLP') )

def autolabel(rects):

for rect in rects:

h = rect.get\_height()

ax.text(rect.get\_x()+rect.get\_width()/2., 1.05\*h, '%d'%int(h),

ha='center', va='bottom')

autolabel(rects1)

autolabel(rects2)

autolabel(rects3)

plt.show()