Machine Learning Assignment 5

Name : Viraj Patil

PRN : 21510097

Batch : T5

#!/usr/bin/env python

# coding: utf-8

# In[36]:

pip install nose

# In[37]:

#Perform logistic regression on diabetes dataset

# In[38]:

#Load diabetes dataset here and return shape of dataset

from cz\_utility import dataset

# diabetes dataset

df = dataset.load('63ea33b29ac5f900139b8d07')

def returnShape(df):

return df.shape

raise NotImplementedError

# In[39]:

from nose.tools import assert\_equal

assert\_equal(returnShape(df), (768, 9))

# In[40]:

#separate input and output from dataset and remove unwanted data columns

#take x for input and y for output

def sepAndRemove(df):

inp = df.drop('Outcome', axis = 1)

out = df['Outcome']

inp.drop('SkinThickness', axis=1, inplace = True)

return [inp, out]

raise NotImplementedError

# In[41]:

trainAndTestData = sepAndRemove(df)

assert\_equal(trainAndTestData[0].shape, (768, 7))

assert\_equal(trainAndTestData[1].shape, (768,))

# In[42]:

#separate training and testing dataset with test\_size = 0.3

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

def trainTestSplit(trainAndTestData):

X\_train, X\_test, y\_train, y\_test = train\_test\_split(trainAndTestData[0], trainAndTestData[1], test\_size=0.3, random\_state=42)

return [X\_train, X\_test, y\_train, y\_test]

raise NotImplementedError

# In[44]:

trainAndTestData=trainTestSplit(sepAndRemove(df))

assert\_equal(trainAndTestData[0].shape,(537, 7))

assert\_equal(trainAndTestData[1].shape,(231, 7))

assert\_equal(trainAndTestData[2].shape,(537,))

assert\_equal(trainAndTestData[3].shape,(231,))

# In[46]:

#use MinMaxScaler to scale the data to a fixed range

from sklearn.preprocessing import MinMaxScaler

def scalar(trainAndTestData):

scaler = MinMaxScaler()

X\_train\_scaled = scaler.fit\_transform(trainAndTestData[0])

X\_test\_scaled = scaler.transform(trainAndTestData[1])

return [X\_train\_scaled, X\_test\_scaled]

raise NotImplementedError

# In[47]:

trainAndTestData=scalar(trainTestSplit(sepAndRemove(df)))

assert\_equal(trainAndTestData[0].shape, (537, 7))

assert\_equal(trainAndTestData[1].shape, (231, 7))

# In[48]:

#perform logistic regression here and return accuracy between predected output for testx and testy

from sklearn.linear\_model import LogisticRegression

from sklearn.metrics import accuracy\_score

def logisticReg(trainAndTestData, trainy, testy):

model = LogisticRegression()

model.fit(trainAndTestData[0], trainy)

predicted\_output = model.predict(trainAndTestData[1])

accuracy = accuracy\_score(testy, predicted\_output)

return accuracy

raise NotImplementedError

# In[49]:

#there might be difference between student answer and faculty answer

trainAndTestData1 = trainTestSplit(sepAndRemove(df))

accuracyScore = logisticReg(scalar(trainTestSplit(sepAndRemove(df))), trainAndTestData1[2], trainAndTestData1[3])

assert\_equal(accuracyScore>0.60, True)

# In[51]:

df

# In[ ]: