pr3

# for building the model

import pandas as pd

from sklearn.tree import DecisionTreeClassifier

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

from sklearn import metrics

# for plotting tree

from sklearn.tree import export\_graphviz

from sklearn.externals.six import StringIO

from IPython.display import Image

import pydotplus

col\_names = ['Reservation', 'Raining', 'BadService','Satur','Result']

hoteldata = pd.read\_csv("hotel for dtree.csv", header=None, names=col\_names)

feature\_cols = ['Reservation', 'Raining', 'BadService','Satur']

X = hoteldata[feature\_cols] # Feature Columns

y = hoteldata.Result # Target variable

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y,

test\_size=0.3, random\_state=1)

# 70% training and 30% test

#clf = DecisionTreeClassifier()

clf = DecisionTreeClassifier(criterion="entropy", max\_depth=5)

clf = clf.fit(X\_train,y\_train)

#Predict the response for test dataset

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

print("ytest = ", y\_test)

print("ypred = ", y\_pred)

# Accuracy of the model

print("Accuracy:",metrics.accuracy\_score(y\_test, y\_pred))

# install graphviz library from https://graphviz.gitlab.io/\_pages/Download/Download\_windows.html

# update your path variable to include C:\Program Files (x86)\Graphviz2.38\bin

dot\_data = StringIO()

export\_graphviz(clf, out\_file=dot\_data, filled=True, rounded=True,

feature\_names = feature\_cols,

class\_names=['Leave','Wait'])

graph = pydotplus.graph\_from\_dot\_data(dot\_data.getvalue())

graph.write\_png('hotel.png')

pr4-a

import numpy as np

def activfun\_sigmoid(x,deriv=False): # sigmoid function

if(deriv==True):

return x\*(1-x) #g'(z) Pg 727

return 1/(1+np.exp(-x)) #Log(z) Pg 725

#input data from hotel.csv with 4 feature columns & first 8 rows, 1-True, 0-False

X = np.array([ [1,1,0,1],

[0,0,0,0],

[0,1,1,1],

[1,1,1,1] ,

[0,1,0,1],

[0,1,0,0],

[1,0,0,1],

[0,1,0,0] ])

# result column, wait -> 0, Leave -> 1

y = np.array([[0,0,1,0,1,0,1,0]]).T

np.random.seed(1) #add this line if you want same output in multliple executions

# initialize weights randomly with values between -1 to 1 with mean 0

w0 = 2\*np.random.random((4,1)) - 1

#OR you may just set them to random values

#w0 = np.random.random((4,1))

print("initial weights - \n", w0)

for n in range(1000):

l\_input = X # x

#feed forward

l\_output = activfun\_sigmoid(l\_input.dot(w0)) #h(x)

#Error calculation

l\_output\_error = y - l\_output #(y-h(x))

l\_output\_delta = l\_output\_error \* activfun\_sigmoid(l\_output,True) # (y-h(x)) \* ( h(x)\*(1-h(x)))

#Backpropagation and updating w

w0 = w0 + 2\*l\_input.T.dot(l\_output\_delta) # Eqn 18.8 Pg-727 where alpha is 2

print("final weights - \n", w0)

print( "Output After Training:")

print (l\_output)

print ("Loss: \n" + str(np.mean(np.square(y - l\_output))))

pr4-b

import numpy as np

def activfun\_sigmoid(x,deriv=False): # sigmoid function

if(deriv==True):

return x\*(1-x) #g'(z) Pg 727

return 1/(1+np.exp(-x)) #Log(z) Pg 725

#input data from hotel.csv with 4 feature columns & first 8 rows, 1-True, 0-False

X = np.array([ [1,1,0,1],

[0,0,0,0],

[0,1,1,1],

[1,1,1,1] ,

[0,1,0,1],

[0,1,0,0],

[1,0,0,1],

[0,1,0,0] ])

# result column, wait -> 0, Leave -> 1

y = np.array([[0,0,1,0,1,0,1,0]]).T

np.random.seed(1) #add this line if you want same output in multliple executions

# initialize weights randomly with values between -1 to 1 with mean 0

w0 = 2\*np.random.random((4,4)) - 1

w1 = 2\*np.random.random((4,1)) - 1

for n in range(140):

l\_input = X # x

#feed forward

l1 = activfun\_sigmoid(l\_input.dot(w0))

l\_output = activfun\_sigmoid(l1.dot(w1))

#Error calculation

l\_output\_error = y - l\_output

l\_output\_delta = l\_output\_error \* activfun\_sigmoid(l\_output,True) # (y-h(x)) \* ( h(x)\*(1-h(x)))

l1\_error = l\_output\_delta.dot(w1.T)

l1\_delta = l1\_error \* activfun\_sigmoid(l1,deriv=True)

#backpropagation and up

w1 = w1 + 2\*l1.T.dot(l\_output\_delta)

w0 = w0 + 2\*l\_input.T.dot(l1\_delta)

print( "Output After Training:")

print (l\_output)

print ("Loss: \n" + str(np.mean(np.square(y - l\_output))))

pr5

import pandas as pd

from sklearn.ensemble import AdaBoostClassifier

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

from sklearn import metrics

col\_names = ['Reservation', 'Raining', 'BadService','Satur','Result']

hoteldata = pd.read\_csv("hotelforada.csv", header=None, names=col\_names)

feature\_cols = ['Reservation', 'Raining', 'BadService','Satur']

X = hoteldata[feature\_cols] # Feature Columns

y = hoteldata.Result # Target variable

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y,

test\_size=0.3, random\_state=1)

# 70% training and 30% test

adahotel = AdaBoostClassifier(n\_estimators=6, learning\_rate=2)

adahotel= adahotel.fit(X\_train,y\_train)

#Predict the response for test dataset

y\_pred = adahotel.predict(X\_test)

print("ytest = \n", y\_test)

print("ypred = \n", y\_pred)

# Accuracy of the model

print("Accuracy:",metrics.accuracy\_score(y\_test, y\_pred))