import numpy as np  
import pandas as pd  
import tensorflow as tf  
import matplotlib.pyplot as plt  
from sklearn.model\_selection import train\_test\_split from sklearn.preprocessing import LabelEncoder from sklearn.utils import shuffle

def read\_dataset(): df=pd.read\_csv('/Users/poorva/Desktop/threat.csv')

X=df[df.columns[0:7]].values y=df[df.columns[4]]

encoder=LabelEncoder() encoder.fit(y) y=encoder.transform(y) Y=one\_hot\_encode(y) print(X.shape) return(X,Y)

def one\_hot\_encode(labels):  
n\_labels=len(labels) n\_unique\_labels=len(np.unique(labels)) one\_hot\_encode=np.zeros((n\_labels,n\_unique\_labels)) one\_hot\_encode[np.arange( n\_labels), labels] = 1 return one\_hot\_encode

X,Y= read\_dataset()  
X,Y = shuffle(X,Y, random\_state=1)  
train\_x, test\_x, train\_y, test\_y= train\_test\_split(X, Y, test\_size = 0.20, random\_state = 0)

print(train\_x.shape) print(train\_y.shape) print(test\_x.shape)

learning\_rate=0.3 epochs=100

cost=np.empty(shape=[1], dtype=float) n\_dim=X.shape[1]

print("n\_dim", n\_dim)  
n\_class = 1 model\_path=“/Users/poorva/Desktop/threatd.csv"

n\_hidden1 =10 n\_hidden2 =10 n\_hidden3 =10 n\_hidden4 =10

x=tf.placeholder(tf.float32, [None, n\_dim]) print(x)  
W=tf.Variable(tf.zeros([n\_dim, n\_class])) b=tf.Variable(tf.zeros([n\_class])) y\_=tf.placeholder(tf.float32, [None, n\_class]) print(y\_)

def multilayer\_perceptron(x, weights, biases):

layer1=tf.add(tf.matmul(x,weights['h1']), biases['b1']) layer1=tf.nn.sigmoid(layer1)

layer2=tf.add(tf.matmul(layer1,weights['h2']), biases['b2']) layer2=tf.nn.sigmoid(layer2)

layer3=tf.add(tf.matmul(layer2,weights['h3']), biases['b3']) layer3=tf.nn.sigmoid(layer3)

layer4=tf.add(tf.matmul(layer3,weights['h4']), biases['b4']) layer4=tf.nn.relu(layer4)

out\_layer=tf.matmul(layer4,weights['out']) + biases['out'] return out\_layer

weights={  
'h1': tf.Variable(tf.truncated\_normal([n\_dim,n\_hidden1])), 'h2': tf.Variable(tf.truncated\_normal([n\_hidden1,n\_hidden2])), 'h3': tf.Variable(tf.truncated\_normal([n\_hidden2,n\_hidden3])), 'h4': tf.Variable(tf.truncated\_normal([n\_hidden3,n\_hidden4])), 'out': tf.Variable(tf.truncated\_normal([n\_hidden4,n\_class]))

} biases={

'b1': tf.Variable(tf.truncated\_normal([n\_hidden1])),

'b2': tf.Variable(tf.truncated\_normal([n\_hidden2])), 'b3': tf.Variable(tf.truncated\_normal([n\_hidden3])), 'b4': tf.Variable(tf.truncated\_normal([n\_hidden4])), 'out': tf.Variable(tf.truncated\_normal([n\_class]))

}

init= tf.global\_variables\_initializer()

saver = tf.train.Saver()

y=multilayer\_perceptron(x,weights,biases)

cost = tf.reduce\_mean(tf.nn.softmax\_cross\_entropy\_with\_logits(logits=y, labels=y\_)) training\_step = tf.train.GradientDescentOptimizer(learning\_rate).minimize(cost)

sess=tf.Session() sess.run(init)

mse\_history=[] accuracy\_history= [] cost\_history = []

for epoch in range(epochs):  
sess.run(training\_step, feed\_dict={x: train\_x, y\_: train\_y}) cst=sess.run(cost, feed\_dict={x: train\_x, y\_: train\_y}) cost\_history=np.append(cost\_history, cst)  
correct\_prediction= tf.equal(tf.argmax(y,1), tf.argmax(y\_, 1)) accuracy= tf.reduce\_mean(tf.cast(correct\_prediction, tf.float32))

pred\_y = sess.run(y, feed\_dict={x: test\_x})  
mse= tf.reduce\_mean(tf.square(pred\_y - test\_y))  
mse\_ = sess.run(mse)  
mse\_history.append(mse\_)  
accuracy = (sess.run(accuracy, feed\_dict={x: train\_x, y\_: train\_y})) accuracy\_history.append(accuracy)

print('epoch:', epoch, '-', 'cst: ',cst, "- MSE:", mse\_, "- Tra")

#save\_path= saver.save(sess,model\_path) #print("Model saved in file: %s" % save\_path)

plt.plot(accuracy\_history) plt.xlabel('Epoch') plt.ylabel('Accuracy') plt.show()

correct\_prediction= tf.equal(tf.argmax(y,1), tf.argmax(y\_, 1))  
accuracy= tf.reduce\_mean(tf.cast(correct\_prediction, tf.float32))  
print("Test accuracy: ", (sess.run(accuracy, feed\_dict={x: train\_x, y\_: train\_y})))

pred\_y = sess.run(y, feed\_dict={x: test\_x}) mse= tf.reduce\_mean(tf.square(pred\_y - test\_y)) print("Mse: % 4f" % sess.run(mse))