#Dependencies

import numpy as np

import pandas as pd

from sklearn.preprocessing import OneHotEncoder

from google.colab import drive

drive.mount('/content/drive')

path ="drive/My Drive/Data/Bitcoin Trust/"

dataset = pd.read\_csv(path+'NNInput.csv', header = None)

#path ="drive/My Drive/Data/Digg/"

#dataset = pd.read\_csv(path+'NNInput.txt',sep = '\t', header = None)

#dataset = dataset.astype(int)

dataset.head()

input\_cols = len(dataset.columns) - 1

output\_col = len(dataset.columns)

output\_classes = len(set(dataset.iloc[:,-1]))

X = dataset.iloc[:,:input\_cols].values

y = dataset.iloc[:,input\_cols:output\_col].values

from sklearn.preprocessing import StandardScaler

sc = StandardScaler()

X = sc.fit\_transform(X)

ohe = OneHotEncoder()

y = ohe.fit\_transform(y).toarray()

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

X\_train,X\_test,y\_train,y\_test = train\_test\_split(X,y,test\_size = 0.1)

#from sklearn.decomposition import PCA

#from sklearn.utils.extmath import randomized\_svd

#pca = PCA(n\_components = 500)

#X\_train = pca.fit\_transform(X\_train)

#X\_test = pca.transform(X\_test)

#explained\_variance = pca.explained\_variance\_ratio\_

input\_cols = X\_train.shape[1]

#!pip install absl-py

#Dependencies

import keras

from keras.models import Sequential

from keras import regularizers

#from keras.layers.core import Dropout

from keras.layers import Dense

#Dropout(0.25),

#num\_Rows \* AbsolongE / input\_cols

model = Sequential()

model.add(Dense(16, input\_dim=input\_cols , activation='relu',  kernel\_regularizer=regularizers.l2(0.1)))

model.add(Dense(12, activation='relu',  kernel\_regularizer=regularizers.l2(0.1)))

model.add(Dense(output\_classes, activation='softmax'))

model.compile(loss='categorical\_crossentropy', optimizer='adam', metrics=["accuracy"])

from keras.callbacks import EarlyStopping

es = keras.callbacks.EarlyStopping(monitor='val\_loss',

                                   mode='min',

                                   patience=10,

                                   restore\_best\_weights=True)

model.summary()

istory = model.fit(X\_train, y\_train, epochs=50,  shuffle=True, validation\_split=0.2)

#y\_pred = ridgereg.predict(X\_test)

y\_pred = model.predict(X\_test)

#Converting predictions to label

pred = list()

for i in range(len(y\_pred)):

    pred.append(np.argmax(y\_pred[i]))

#Converting one hot encoded test label to label

test = list()

for i in range(len(y\_test)):

    test.append(np.argmax(y\_test[i]))

from sklearn.metrics import accuracy\_score

a = accuracy\_score(pred,test)

print('Accuracy is:', a)

#Accuracy is: 0.8818629307080651

import matplotlib.pyplot as plt

plt.plot(history.history['accuracy'])

plt.plot(history.history['val\_accuracy'])

plt.title('Model Accuracy')

plt.ylabel('Accuracy')

plt.xlabel('Epoch')

plt.legend(['Training', 'Validation'], loc='lower right')

plt.show()

plt.plot(history.history['loss'])

plt.plot(history.history['val\_loss'])

plt.title('Model loss')

plt.ylabel('Loss')

plt.xlabel('Epoch')

plt.legend(['Train', 'Test'], loc='upper left')

plt.show()

from sklearn.metrics import confusion\_matrix

from sklearn.metrics import classification\_report 

matrix = confusion\_matrix(test, pred)

matrix

print(classification\_report(test, pred))

from sklearn.metrics import roc\_auc\_score

from sklearn.metrics import average\_precision\_score

lr\_roc\_auc\_multiclass = roc\_auc\_score\_multiclass(test, pred)

print(lr\_roc\_auc\_multiclass)

#{0: 0.986615385994371, 1: 0.9777492025169555, 2: 0.996407438715131, 3: 0.9573247698247698, 4: 0.8809423692873043, 5: 0.9967375897219373, 6: 0.9490835391998182, 7: 0.7583609746539587, 8: 0.9991564740615774, 9: 0.49943117178612056, 11: 0.5, 12: 0.5, 13: 0.5, 14: 0.5, 15: 0.5, 16: 0.5, 17: 0.5, 18: 0.49962121212121213, 20: 0.5, 21: 0.5, 23: 0.5, 24: 0.5, 25: 0.5, 26: 0.5, 27: 0.5, 35: 0.5, 45: 0.5, 62: 0.5, 75: 0.5}