**Module 11**

1.Write an LSTM program to predict next alphabet in the sequence “A B C D E F G H I J K L M N O P

Q R S T U V W X Y Z”

import numpy

from keras.models import Sequential

from keras.layers import Dense

from keras.layers import LSTM

from keras.utils import np\_utils

# fix random seed for reproducibility

numpy.random.seed(7)

# define the raw dataset

alphabet = "ABCDEFGHIJKLMNOPQRSTUVWXYZ"

# create mapping of characters to integers (0-25) and the reverse

char\_to\_int = dict((c, i) for i, c in enumerate(alphabet))

int\_to\_char = dict((i, c) for i, c in enumerate(alphabet))

# prepare the dataset of input to output pairs encoded as integers

seq\_length = 1

dataX = []

dataY = []

for i in range(0, len(alphabet) - seq\_length, 1):

seq\_in = alphabet[i:i + seq\_length]

seq\_out = alphabet[i + seq\_length]

dataX.append([char\_to\_int[char] for char in seq\_in])

dataY.append(char\_to\_int[seq\_out])

print(seq\_in, '->', seq\_out)

# reshape X to be [samples, time steps, features]

X = numpy.reshape(dataX, (len(dataX), seq\_length, 1))

# normalize

X = X / float(len(alphabet))

# one hot encode the output variable

y = np\_utils.to\_categorical(dataY)

# create and fit the model

model = Sequential()

model.add(LSTM(32, input\_shape=(X.shape[1], X.shape[2])))

model.add(Dense(y.shape[1], activation='softmax'))

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

model.fit(X, y, epochs=500, batch\_size=1, verbose=2)

# summarize performance of the model

scores = model.evaluate(X, y, verbose=0)

print("Model Accuracy: %.2f%%" % (scores[1]\*100))

# demonstrate some model predictions

for pattern in dataX:

x = numpy.reshape(pattern, (1, len(pattern), 1))

x = x / float(len(alphabet))

prediction = model.predict(x, verbose=0)

index = numpy.argmax(prediction)

result = int\_to\_char[index]

seq\_in = [int\_to\_char[value] for value in pattern]

print(seq\_in, "->", result)

2.Build a LSTM to predict the stock price for multivariate data(use multiple features).Use any online website or tiingo to get the dataset.

#Dataset is APPLE Company Sock Price Prediction Data

import pandas as pd

df=pd.read\_csv('C:/Users/DELL/Downloads/stock price/Stock-MArket-Forecasting-master/AAPL.csv')

df.head()

df.tail()

df1=df.reset\_index()['close']

df1

import matplotlib.pyplot as plt

plt.plot(df1)

import numpy as np

from sklearn.preprocessing import MinMaxScaler

scaler=MinMaxScaler(feature\_range=(0,1))

df1=scaler.fit\_transform(np.array(df1).reshape(-1,1))

print(df1)

##splitting dataset into train and test split

training\_size=int(len(df1)\*0.65)

test\_size=len(df1)-training\_size

train\_data,test\_data=df1[0:training\_size,:],df1[training\_size:len(df1),:1]

training\_size,test\_size

train\_data

import numpy

# convert an array of values into a dataset matrix

def create\_dataset(dataset, time\_step=1):

dataX, dataY = [], []

for i in range(len(dataset)-time\_step-1):

a = dataset[i:(i+time\_step), 0] ###i=0, 0,1,2,3-----99 100

dataX.append(a)

dataY.append(dataset[i + time\_step, 0])

return numpy.array(dataX), numpy.array(dataY)

# reshape into X=t,t+1,t+2,t+3 and Y=t+4

time\_step = 100

X\_train, y\_train = create\_dataset(train\_data, time\_step)

X\_test, ytest = create\_dataset(test\_data, time\_step)

print(X\_train.shape), print(y\_train.shape)

print(X\_test.shape), print(ytest.shape)

# reshape input to be [samples, time steps, features] which is required for LSTM

X\_train =X\_train.reshape(X\_train.shape[0],X\_train.shape[1] , 1)

X\_test = X\_test.reshape(X\_test.shape[0],X\_test.shape[1] , 1)

### Create the Stacked LSTM model

from tensorflow.keras.models import Sequential

from tensorflow.keras.layers import Dense

from tensorflow.keras.layers import LSTM

model=Sequential()

model.add(LSTM(50,return\_sequences=True,input\_shape=(100,1)))

model.add(LSTM(50,return\_sequences=True))

model.add(LSTM(50))

model.add(Dense(1))

model.compile(loss='mean\_squared\_error',optimizer='adam')

model.fit(X\_train,y\_train,validation\_data=(X\_test,ytest),epochs=100,batch\_size=64,verbose=1)

import tensorflow as tf

tf.\_\_version\_\_

### Lets Do the prediction and check performance metrics

train\_predict=model.predict(X\_train)

test\_predict=model.predict(X\_test)

##Transformback to original form

train\_predict=scaler.inverse\_transform(train\_predict)

test\_predict=scaler.inverse\_transform(test\_predict)

### Calculate RMSE performance metrics

import math

from sklearn.metrics import mean\_squared\_error

math.sqrt(mean\_squared\_error(y\_train,train\_predict))

### Test Data RMSE

math.sqrt(mean\_squared\_error(ytest,test\_predict))

### Plotting

# shift train predictions for plotting

look\_back=100

trainPredictPlot = numpy.empty\_like(df1)

trainPredictPlot[:, :] = np.nan

trainPredictPlot[look\_back:len(train\_predict)+look\_back, :] = train\_predict

# shift test predictions for plotting

testPredictPlot = numpy.empty\_like(df1)

testPredictPlot[:, :] = numpy.nan

testPredictPlot[len(train\_predict)+(look\_back\*2)+1:len(df1)-1, :] = test\_predict

# plot baseline and predictions

plt.plot(scaler.inverse\_transform(df1))

plt.plot(trainPredictPlot)

plt.plot(testPredictPlot)

plt.show()

len(test\_data)

x\_input=test\_data[341:].reshape(1,-1)

x\_input.shape

temp\_input=list(x\_input)

temp\_input=temp\_input[0].tolist()

temp\_input

# demonstrate prediction for next 10 days

from numpy import array

lst\_output=[]

n\_steps=100

i=0

while(i<30):

if(len(temp\_input)>100):

#print(temp\_input)

x\_input=np.array(temp\_input[1:])

print("{} day input {}".format(i,x\_input))

x\_input=x\_input.reshape(1,-1)

x\_input = x\_input.reshape((1, n\_steps, 1))

#print(x\_input)

yhat = model.predict(x\_input, verbose=0)

print("{} day output {}".format(i,yhat))

temp\_input.extend(yhat[0].tolist())

temp\_input=temp\_input[1:]

#print(temp\_input)

lst\_output.extend(yhat.tolist())

i=i+1

else:

x\_input = x\_input.reshape((1, n\_steps,1))

yhat = model.predict(x\_input, verbose=0)

print(yhat[0])

temp\_input.extend(yhat[0].tolist())

print(len(temp\_input))

lst\_output.extend(yhat.tolist())

i=i+1

print(lst\_output)

#predicting nexture future 30days

day\_new=np.arange(1,101)

day\_pred=np.arange(101,131)

import matplotlib.pyplot as plt

len(df1)

plt.plot(day\_new,scaler.inverse\_transform(df1[1158:]))

plt.plot(day\_pred,scaler.inverse\_transform(lst\_output))

df3=df1.tolist()

df3.extend(lst\_output)

plt.plot(df3[1200:])

df3=scaler.inverse\_transform(df3).tolist()

plt.plot(df3)

#################################### END ####################