NAME: AYUSH CHANCHAL

SAP ID:500097569

BATCH: B6

EXPERIMENT:09

Q). Explain LSTM?

: - Traditional RNNs may struggle to capture long-range dependencies in sequences because they suffer from the vanishing gradient problem. This means that as information passes through many time steps, it can quickly become small and eventually disappear, making it difficult for the network to learn from distant events. LSTM solves this issue by introducing a more complex structure compared to simple RNN. They have memory cells, which are like conveyor belts that can store information and move it over several time steps. This allows LSTMs to selectively remember or forget information over time.

Key Components of LSTM:

1. Cell state
2. Hidden State
3. Gates
4. Gate Activation function

Source code:

import numpy as np

import pandas as  pd

import matplotlib.pyplot as plt

from keras.preprocessing.sequence import TimeseriesGenerator

from sklearn.preprocessing import MinMaxScaler

from keras.models import Sequential

from keras.layers import Dense

from keras.layers import LSTM

from sklearn.metrics import mean\_squared\_error

from math import sqrt

import io

df = pd.read\_csv(io.BytesIO(uploaded['airline-passengers.csv']),index\_col='Month',parse\_dates=True)

# df=pd.read\_csv('airline-passengers.csv',index\_col='Month',parse\_dates=True)

df.index.freq='MS'

print(df.shape)

print(df.columns)

plt.figure(figsize=(20,40))

plt.plot(df.Passengers,linewidth=2)

plt.show()

nobs=12

df\_train=df.iloc[:-nobs]

df\_test=df.iloc[-nobs:]

print(df\_train)

print(df\_test)

scaler=MinMaxScaler()

scaler.fit(df\_train)

scaled\_train=scaler.transform(df\_train)

scaled\_test=scaler.transform(df\_test)

n\_inputs=12

n\_features=1

generator=TimeseriesGenerator(scaled\_train,scaled\_train,length=n\_inputs,batch\_size=1)

for i in range(len(generator)):

    x,y=generator[i]

    print(f'\n{x.flatten()} and {y}')

print(x.shape)

model=Sequential()

model.add(LSTM(200,activation='relu',input\_shape=(n\_inputs,n\_features)))

model.add(Dense(1))

model.compile(optimizer='adam',loss='mse')

print(model.summary())

model.fit(generator,epochs=50)

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

last\_train\_batch=scaled\_train[-12:]

last\_train\_batch=last\_train\_batch.reshape(1,12,1)

print(last\_train\_batch)

model.predict(last\_train\_batch)

scaled\_test[0]

y\_pred=[]

first\_batch=scaled\_train[-n\_inputs:]

current\_batch=first\_batch.reshape(1,n\_inputs,n\_features)

for i in range(len(scaled\_test)):

    batch=current\_batch

    pred=model.predict(batch)[0]

    y\_pred.append(pred)

    current\_batch=np.append(current\_batch[:,1:, :],[[pred]],axis=1)

print(y\_pred)

print(scaled\_test)

print(df\_test)

y\_pred\_transformed=scaler.inverse\_transform(y\_pred)

y\_pred\_transformed=np.round(y\_pred\_transformed,0)

y\_pred\_final=y\_pred\_transformed.astype(int)

print(y\_pred\_final)

print(df\_test.values)

print(y\_pred\_final)

df\_test['Predictions']=y\_pred\_final

print(df\_test)

plt.figure(figsize=(15,6))

plt.plot(df\_train.index,df\_train.Passengers,linewidth=2,color='black',label='Train Values')

plt.plot(df\_test.index,df\_test.Passengers,linewidth=2,color='green',label='True Values')

plt.plot(df\_test.index,df\_test.Predictions,linewidth=2,color='red',label='Predicted Values')

plt.legend()

plt.show()

squareroot=sqrt(mean\_squared\_error(df\_test.Passengers,df\_test.Predictions))

Screenshot:

