**Recurrent Neural Network (RNN)**

**Instructions:**

Please share your answers filled in-line in the word document. Submit code separately wherever applicable.

Please ensure you update all the details:

**Name: Vishvash C Batch ID:** 23012024

**Topic: Recurrent Neural Network.**

**Guidelines:**

**1. An assignment submission is considered complete only when the correct and executable code(s) and documentation explaining the method and results are submitted. Failing to submit either of those will be considered an invalid submission and not a correct submission.**

**2. Ensure that you submit your assignments correctly and in full. Resubmission is not allowed.**

**3. Post the submission you can evaluate your work by referring to the keys provided. (will be available only post the submission).**

**Hints:**

1. **Business Problem**
   1. **What is the business objective?**
   2. **Are there any constraints?**
2. **Work on each feature of the dataset to create a data dictionary.**

**Using Python to perform the following:**

1. **Data Pre-processing**

**3.1 Data Cleaning, Feature Engineering, etc.**

**3.2 Outlier treatment.**

1. **Model Building**

**4.1 Build a Recurrent Neural Network.**

**4.2 Train and test the model.**

**4.3 Briefly explain the model output in the documentation.**

1. **Write about the benefits/impact of the solution - in what way does the business (client) benefit from the solution provided?**
2. **Use Tensorflow for this assignment. Depending on your system configuration, use either Tensorflow GPU or Tensorflow CPU versions.**

**Problem Statement: -**

1. Here is the time series data [110,125,133,146,158,172,187,196,210]. Build an RNN/LSTM model to predict the next 10 digits. similarly show the next 10 Forecasting in streamlit Framework
2. Write down the applications of RNN.

Natural Language Processing (NLP) tasks like language translation, sentiment analysis, and text generation.

Time series prediction and forecasting.

Speech recognition and synthesis.

Handwriting recognition.

Video analysis and action recognition.

1. Write about how the inputs are selected for LSTM/RNN models. Explain in terms of timesteps, samples, and features.

**Timesteps:** Timesteps represent the number of time steps in each sample of input data. For example, in a time series forecasting problem, each timestep could represent a day, hour, or any other unit of time.

**Samples:** Samples represent the number of data points or sequences in your dataset. Each sample consists of one or more timesteps.

**Features:** Features represent the number of features or variables at each timestep. For example, in a multivariate time series problem, each timestep may have multiple features such as temperature, humidity, etc.

1. What are the disadvantages of MLP when dealing with sequence data?

MLPs don't have memory, meaning they treat each input independently and don't consider the sequence or temporal dependencies in the data.

They can't handle variable length sequences directly, as they require fixed-size inputs.

MLPs are prone to overfitting on sequence data, especially when dealing with long sequences, due to the large number of parameters and the lack of regularization inherent in the architecture.

MLPs struggle with capturing long-term dependencies in the data, which is crucial in many sequence modeling tasks.

**Code:**

import numpy as np

import tensorflow as tf

from tensorflow.keras.models import Sequential

from tensorflow.keras.layers import LSTM, Dense

# Define the time series data

data = np.array([110,125,133,146,158,172,187,196,210])

# Prepare the data for LSTM

def prepare\_data(data, time\_steps):

X, y = [], []

for i in range(len(data)-time\_steps):

X.append(data[i:i+time\_steps])

y.append(data[i+time\_steps])

return np.array(X), np.array(y)

time\_steps = 3

X, y = prepare\_data(data, time\_steps)

# Reshape input data to fit LSTM model

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

# Build LSTM model

model = Sequential()

model.add(LSTM(units=50, activation='relu', input\_shape=(time\_steps, 1)))

model.add(Dense(units=1))

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

# Fit the model

model.fit(X, y, epochs=100, verbose=0)

# Forecast the next 10 digits

forecast = []

last\_sequence = X[-1]

for \_ in range(10):

next\_digit = model.predict(last\_sequence.reshape(1, time\_steps, 1))[0][0]

forecast.append(next\_digit)

last\_sequence = np.append(last\_sequence[1:], next\_digit)

print("Forecasted digits:", forecast)

**Output:**

for \_ in range(10):

next\_digit = model.predict(last\_sequence.reshape(1, time\_steps, 1))[0][0]

forecast.append(next\_digit)

last\_sequence = np.append(last\_sequence[1:], next\_digit)

print("Forecasted digits:", forecast)

1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 262ms/step

1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 241ms/step

1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 37ms/step

1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 70ms/step

1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 51ms/step

1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 48ms/step

1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 35ms/step

1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 31ms/step

1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 34ms/step

1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 36ms/step

Forecasted digits: [214.93652, 231.6777, 248.70184, 269.3247, 290.37244, 313.03897, 337.99023, 364.54718, 393.2335, 424.26877]