Experiment No.: 8

Title: LSTM

Batch: HO–DL 1 **Roll No.: 16010422234** **Experiment No. 08**

Aim: To implement LSTM network

Resources needed: Python/Matlab

Theory:

The most effective sequence models used in practical applications are called gated RNNs. These include the long short-term memory and networks based on the gated recurrent unit. Long Short Term Memory Networks is an advanced RNN, a sequential network, that allows information to persist. It is capable of handling the vanishing gradient problem faced by RNN. A recurrent neural network also known as RNN is used for persistent memory.

Let's say while watching a video you remember the previous scene or while reading a book you know what happened in the earlier chapter. Similarly, RNNs work, they remember the previous information and use it for processing the current input. The shortcoming of RNN is, they cannot remember Long term dependencies due to vanishing gradients. LSTMs are explicitly designed to avoid long-term dependency problems.

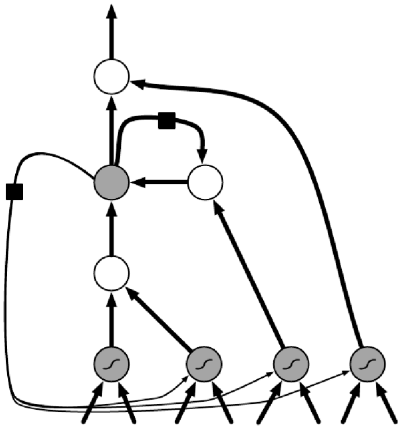
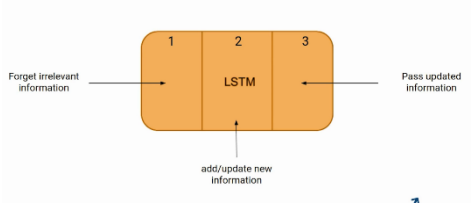


Fig-1: Block diagram of the LSTM recurrent network "cell."

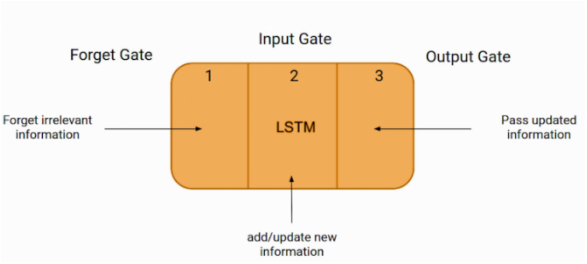
The clever idea of introducing self-loops to produce paths where the gradient can flow for long durations is a core contribution of the initial long short-term memory (LSTM) model (Hochreiter and Schmidhuber, 1997).

At a high-level LSTM works very much like an RNN cell. Here is the internal functioning of the LSTM network. The LSTM consists of three parts, as shown in the image below and each part performs an individual function.



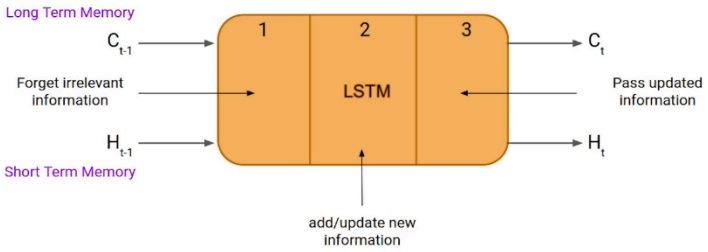
The first part chooses whether the information coming from the previous timestamp is to be remembered or is irrelevant and can be forgotten. In the second part, the cell tries to learn new information from the input to this cell. At last, in the third part, the cell passes the updated information from the current timestamp to the next timestamp.

These three parts of an LSTM cell are known as gates. The first part is called Forget gate, the second part is known as the Input gate and the last one is the Output gate.

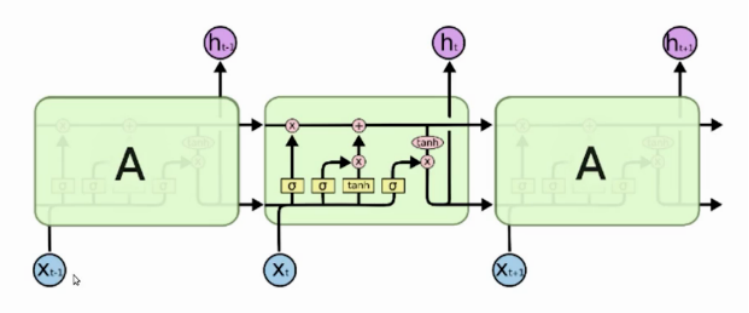


LSTM gates- Just like a simple RNN, an LSTM also has a hidden state where H(t-1) represents the hidden state of the previous timestamp and Ht is the hidden state of the current timestamp. In addition to that, LSTM also has a cell state represented by C(t-1) and C(t) for previous and current timestamps respectively.

Here the hidden state is known as Short term memory and the cell state is known as Long term memory. Refer to the following image.



This is the More intuitive diagram of the LSTM network.



The LSTM has been found extremely successful in many applications, such as unconstrained handwriting recognition (Graves et al., 2009), speech recognition (Graves et al., 2013; Graves and Jaitly, 2014), handwriting generation (Graves, 2013), machine translation (Sutskever et al., 2014), image captioning (Kiros et al., 2014b; Vinyals et al., 2014b; Xu et al., 2015) and parsing (Vinyals et al., 2014a).

Activity:

• Load any time series dataset.

• Pre-process and visualize the dataset.

• Form the Training and Testing Data.

• Develop and train LSTM model.

• Plot the predictions for training and testing data.

• Comment on the output.

Program:

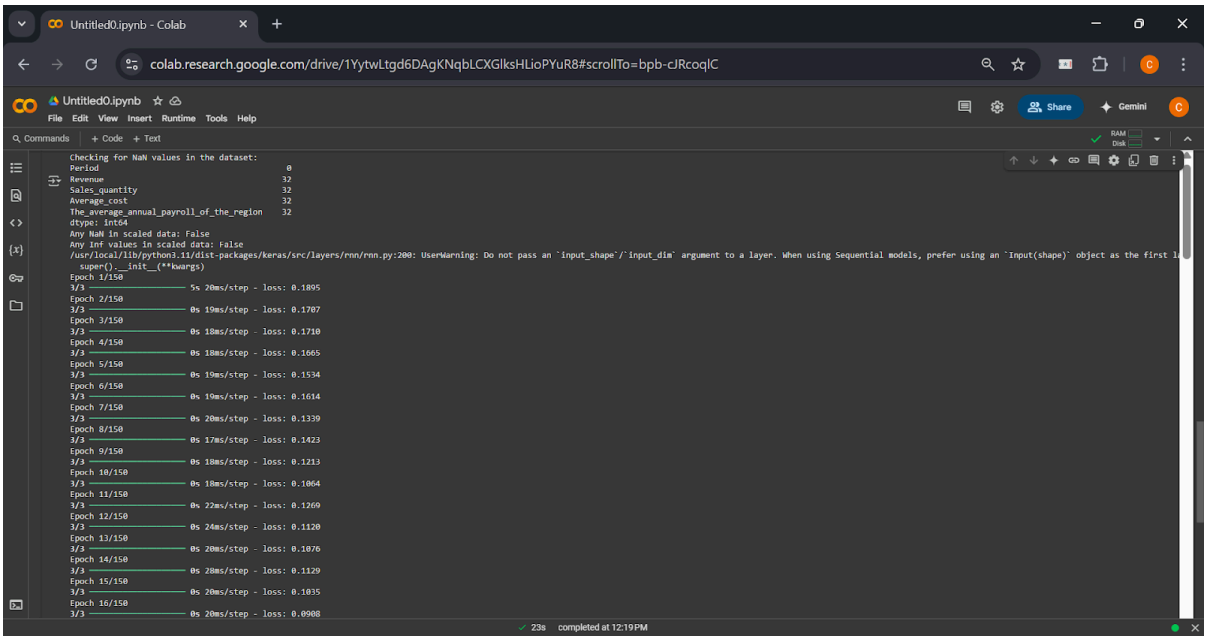
```
# Import necessary libraries
import numpy as np
import matplotlib.pyplot as plt
from sklearn.preprocessing import MinMaxScaler
from keras.models import Sequential
from keras.layers import LSTM, GRU, Dense, Dropout
from keras.optimizers import Adam
import pandas as pd

# Load the dataset
file_path = '/content/Month_Value_1.csv'
data = pd.read_csv(file_path)
```

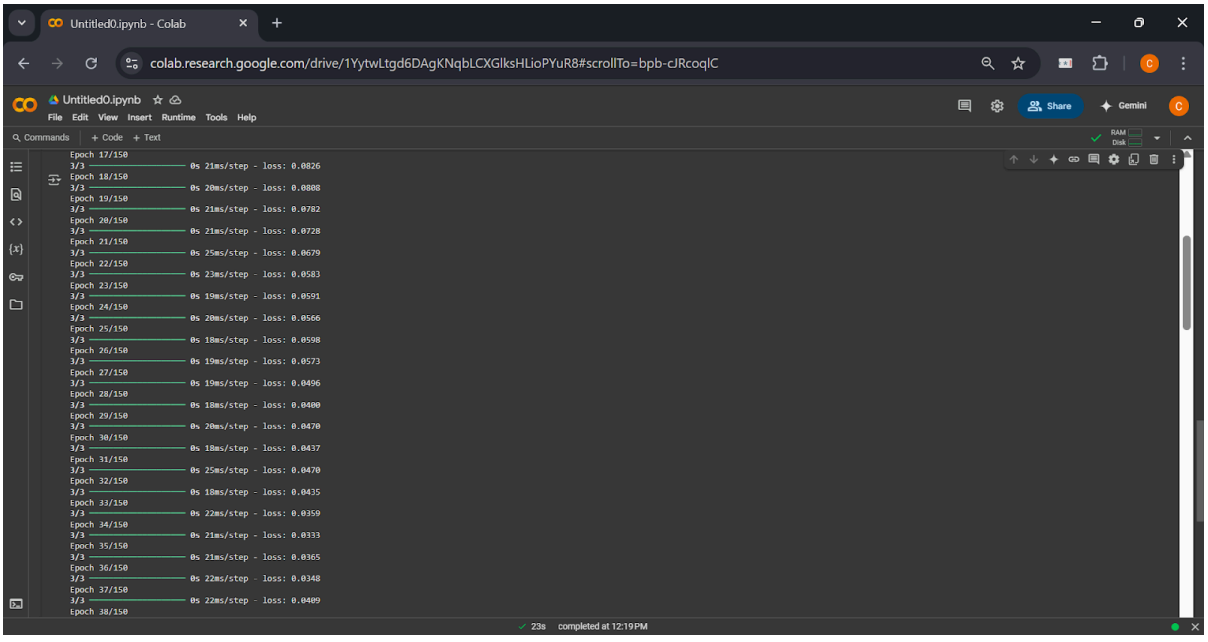
```
# Convert 'Period' to datetime
data['Period'] = pd.to_datetime(data['Period'], format='%d.%m.%Y')

# Check for NaN values in the dataset
print("Checking for NaN values in the dataset:")
print(data.isna().sum())  # Check how many NaN values there are in each
column

# Handle NaN values: Drop rows with NaN values
data = data.dropna()

# Replace zeros in numerical columns with small positive values to
avoid issues during scaling
data[["Sales_quantity", "Average_cost",
"The_average_annual_payroll_of_the_region"]] = data[["Sales_quantity",
"Average_cost", "The_average_annual_payroll_of_the_region"]].replace(0,
0.0001)

# We will use 'Revenue' as the target and the other columns as features
features = ['Sales_quantity', 'Average_cost',
'The_average_annual_payroll_of_the_region']
target = 'Revenue'

# Scaling the data (MinMax scaling to range between 0 and 1)
scaler = MinMaxScaler(feature_range=(0, 1))
scaled_data = scaler.fit_transform(data[features + [target]])

# Check for NaN or Inf values in the scaled data after scaling
print(f"Any NaN in scaled data: {np.any(np.isnan(scaled_data))}")
print(f"Any Inf values in scaled data:
{np.any(np.isinf(scaled_data))}")

# Prepare the data for LSTM: create time series sequences
time_step = 3  # Using 3 time steps as the window size
x_data, y_data = [], []

for i in range(time_step, len(scaled_data)):
    x_data.append(scaled_data[i-time_step:i, :-1])  # Features
(excluding target)
    y_data.append(scaled_data[i, -1])  # Target ('Revenue')

x_data, y_data = np.array(x_data), np.array(y_data)
```

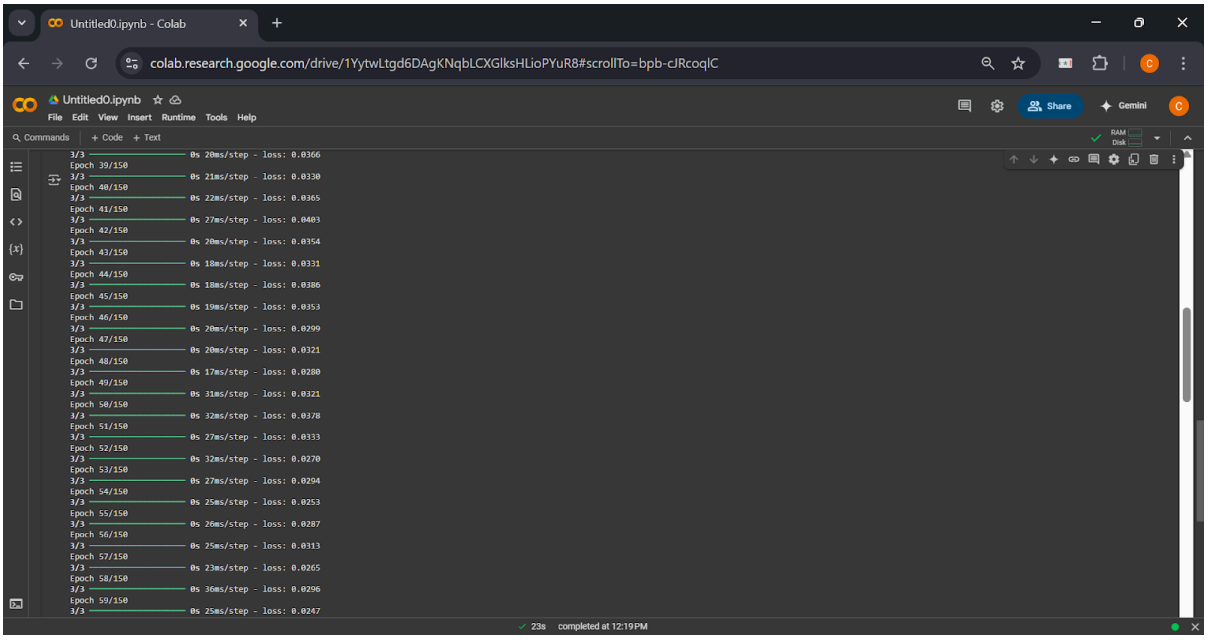
```
# Split into training and testing sets
train_size = int(len(x_data) * 0.8)
x_train, x_test = x_data[:train_size], x_data[train_size:]
y_train, y_test = y_data[:train_size], y_data[train_size:]

# Reshape data for LSTM (LSTM expects 3D input: [samples, time steps,
features])
x_train = x_train.reshape(x_train.shape[0], x_train.shape[1],
x_train.shape[2])
x_test = x_test.reshape(x_test.shape[0], x_test.shape[1],
x_test.shape[2])

# Build the LSTM model with more units and layers for better learning
model = Sequential()
model.add(LSTM(units=100, return_sequences=True,
input_shape=(x_train.shape[1], x_train.shape[2])))
model.add(Dropout(0.2))  # Add dropout for regularization
model.add(LSTM(units=100, return_sequences=True))
model.add(Dropout(0.2))  # Add another dropout layer
model.add(GRU(units=50))  # Adding a GRU layer as an alternative
model.add(Dense(units=1))  # Output layer with 1 unit (for prediction
of 'Revenue')

# Compile the model with a lower learning rate
model.compile(optimizer=Adam(learning_rate=0.0001),
loss='mean_squared_error')

# Train the model
model.fit(x_train, y_train, epochs=150, batch_size=16)  # Increased
epochs

# Predict the results
train_pred = model.predict(x_train)
test_pred = model.predict(x_test)

# Inverse transform the predictions and actual values
train_pred =
scaler.inverse_transform(np.concatenate((np.zeros((train_pred.shape[0],
3)), train_pred), axis=1))[:, -1]
y_train_actual =
scaler.inverse_transform(np.concatenate((np.zeros((y_train.shape[0],
3)), y_train.reshape(-1, 1)), axis=1))[:, -1]
```

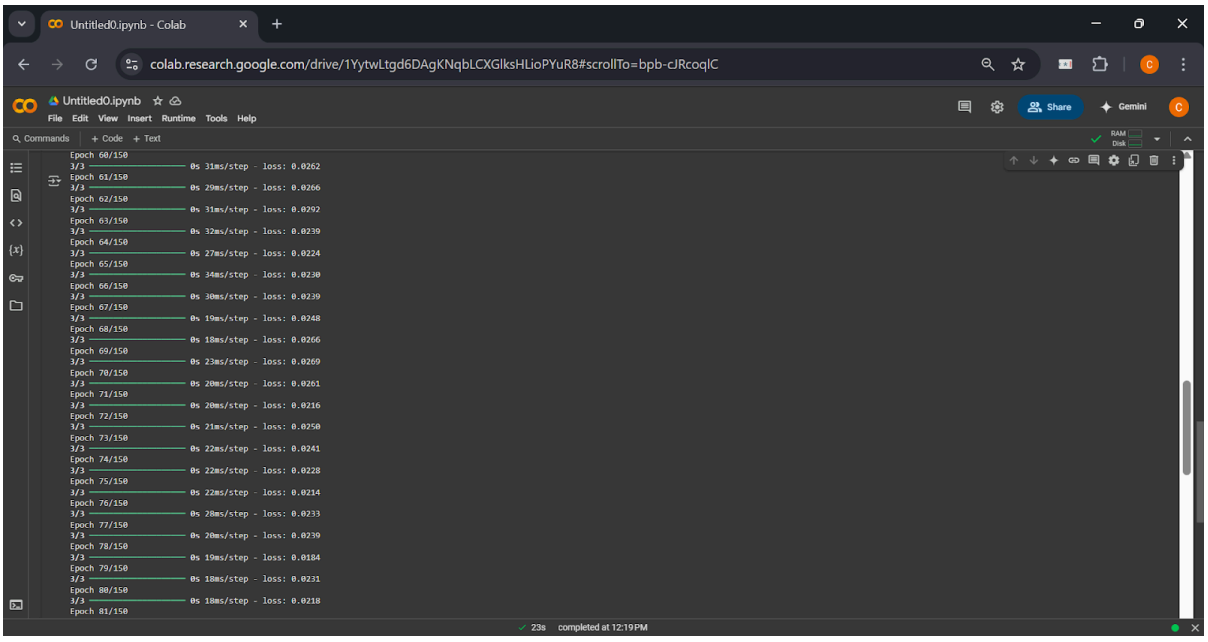
```
test_pred =
scaler.inverse_transform(np.concatenate((np.zeros((test_pred.shape[0],
3)), test_pred), axis=1))[:, -1]
y_test_actual =
scaler.inverse_transform(np.concatenate((np.zeros((y_test.shape[0],
3)), y_test.reshape(-1, 1)), axis=1))[:, -1]

# Plotting the predictions
plt.figure(figsize=(12, 6))

# Plot the actual training data (up to the actual training length)
plt.plot(data['Period'][:len(y_train_actual)], y_train_actual,
color='blue', label='Actual Training Data')

# Plot the actual testing data (from train_size + time_step onward)
plt.plot(data['Period'][train_size + time_step:], y_test_actual,
color='green', label='Actual Testing Data')

# Plot the predicted training data (up to the actual training length)
plt.plot(data['Period'][:len(train_pred)], train_pred, color='red',
label='Predicted Training Data')

# Plot the predicted testing data (from train_size + time_step onward)
plt.plot(data['Period'][train_size + time_step:], test_pred,
color='orange', label='Predicted Testing Data')

plt.title('Revenue Prediction (LSTM)')
plt.xlabel('Date')
plt.ylabel('Revenue')
plt.legend()
plt.show()
```

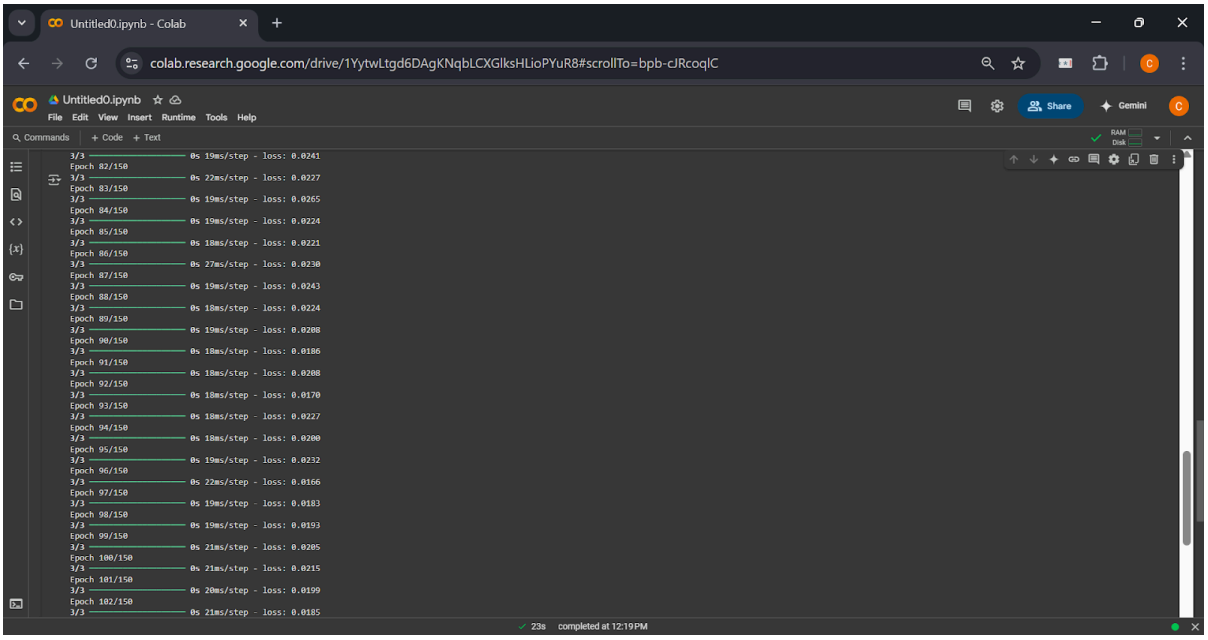
Output:

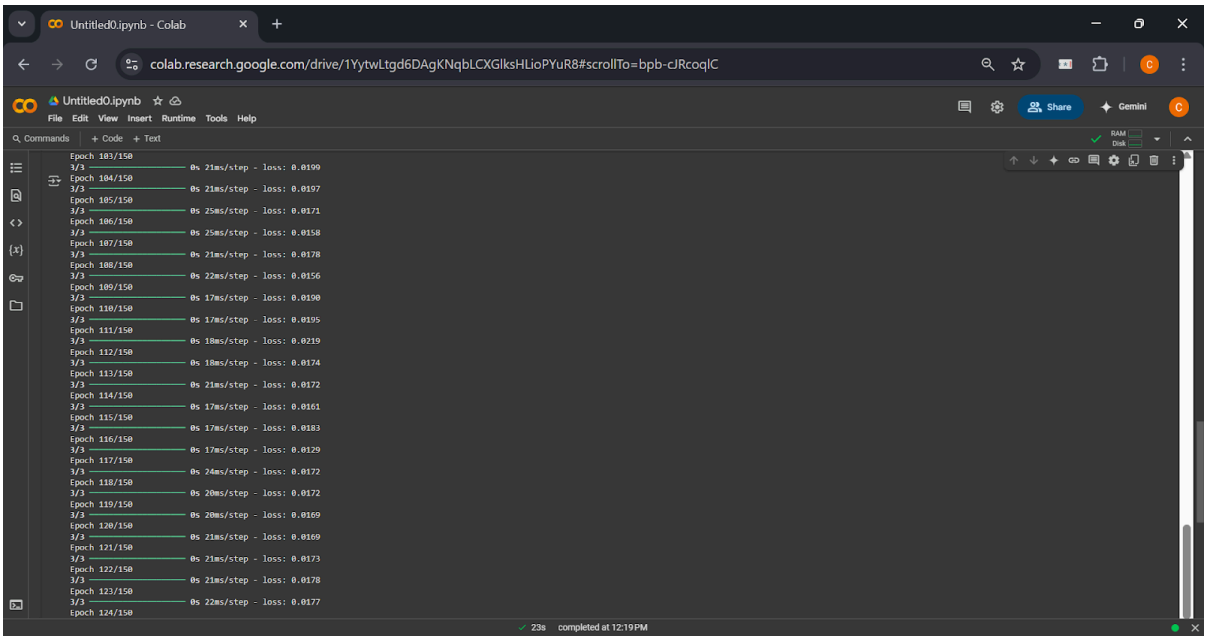

Untitled0.ipynb - Colab
colab.research.google.com/drive/1YytwLtgd6DAgKNqbLCXGlksHLioPYuR8#scrollTo=bpb-cJRcoqlC
Untitled0.ipynb
File Edit View Insert Runtime Tools Help
Share
Gemini
Commands + Code + Text
Checking for NaN values in the dataset:
Period 0
Revenue 32
Sales_quantity 32
Average_cost 32
The_average_annual_payroll_of_the_region 32
dtype: int64
Any NaN in scaled data: False
Any Inf values in scaled data: False
/usr/local/lib/python3.11/dist-packages/keras/src/layers/rnn/rnn.py:200: UserWarning: Do not pass an `input_shape`/`input_dim` argument to a layer. When using Sequential models, prefer using an `Input(shape)` object as the first l
super().__init__(**kwargs)
Epoch 1/150
3/3 5s 20ms/step - loss: 0.1895
Epoch 2/150
3/3 0s 19ms/step - loss: 0.1707
Epoch 3/150
3/3 0s 18ms/step - loss: 0.1710
Epoch 4/150
3/3 0s 18ms/step - loss: 0.1665
Epoch 5/150
3/3 0s 19ms/step - loss: 0.1534
Epoch 6/150
3/3 0s 19ms/step - loss: 0.1614
Epoch 7/150
3/3 0s 20ms/step - loss: 0.1339
Epoch 8/150
3/3 0s 17ms/step - loss: 0.1423
Epoch 9/150
3/3 0s 18ms/step - loss: 0.1213
Epoch 10/150
3/3 0s 18ms/step - loss: 0.1064
Epoch 11/150
3/3 0s 22ms/step - loss: 0.1269
Epoch 12/150
3/3 0s 24ms/step - loss: 0.1120
Epoch 13/150
3/3 0s 20ms/step - loss: 0.1076
Epoch 14/150
3/3 0s 28ms/step - loss: 0.1129
Epoch 15/150
3/3 0s 20ms/step - loss: 0.1035
Epoch 16/150
3/3 0s 20ms/step - loss: 0.0908
23s completed at 12:19PM

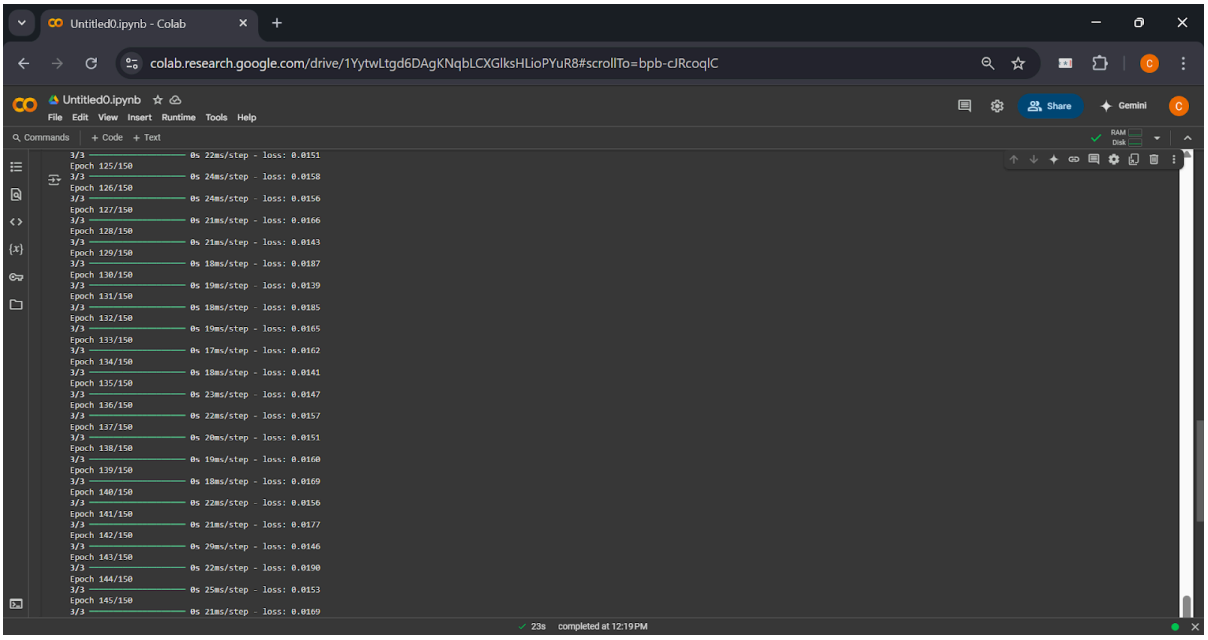

Untitled0.ipynb - Colab
colab.research.google.com/drive/1YytwLtgd6DAgKNqbLCXGlksHLioPYuR8#scrollTo=bpb-cJRcoqlC
Untitled0.ipynb
File Edit View Insert Runtime Tools Help
Share
Gemini
Commands + Code + Text
Epoch 17/150
3/3 0s 21ms/step - loss: 0.0826
Epoch 18/150
3/3 0s 20ms/step - loss: 0.0808
Epoch 19/150
3/3 0s 21ms/step - loss: 0.0782
Epoch 20/150
3/3 0s 21ms/step - loss: 0.0728
Epoch 21/150
3/3 0s 25ms/step - loss: 0.0679
Epoch 22/150
3/3 0s 23ms/step - loss: 0.0583
Epoch 23/150
3/3 0s 19ms/step - loss: 0.0591
Epoch 24/150
3/3 0s 20ms/step - loss: 0.0566
Epoch 25/150
3/3 0s 18ms/step - loss: 0.0598
Epoch 26/150
3/3 0s 19ms/step - loss: 0.0573
Epoch 27/150
3/3 0s 19ms/step - loss: 0.0496
Epoch 28/150
3/3 0s 18ms/step - loss: 0.0400
Epoch 29/150
3/3 0s 20ms/step - loss: 0.0470
Epoch 30/150
3/3 0s 18ms/step - loss: 0.0437
Epoch 31/150
3/3 0s 25ms/step - loss: 0.0470
Epoch 32/150
3/3 0s 18ms/step - loss: 0.0435
Epoch 33/150
3/3 0s 22ms/step - loss: 0.0359
Epoch 34/150
3/3 0s 21ms/step - loss: 0.0333
Epoch 35/150
3/3 0s 21ms/step - loss: 0.0365
Epoch 36/150
3/3 0s 22ms/step - loss: 0.0348
Epoch 37/150
3/3 0s 22ms/step - loss: 0.0409
Epoch 38/150
23s completed at 12:19PM

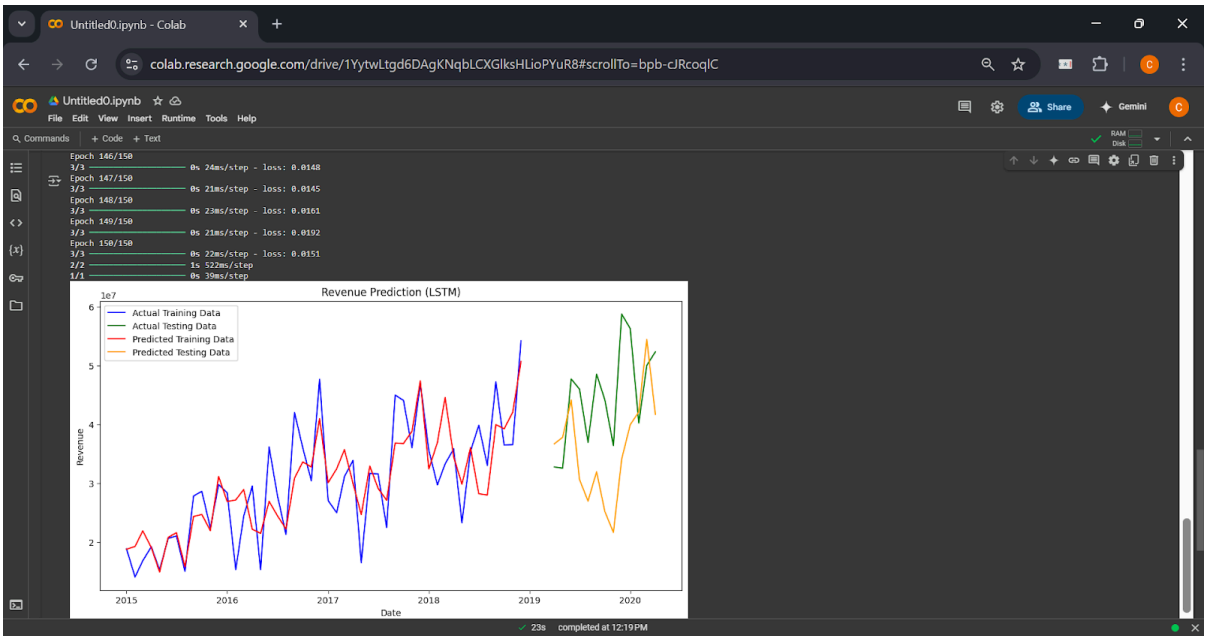

Untitled0.ipynb - Colab
colab.research.google.com/drive/1YytwLtgd6DAgKNqbLCXGlksHLioPYuR8#scrollTo=bpb-cJRcoqlC
Untitled0.ipynb
File Edit View Insert Runtime Tools Help
Share
Gemini
Commands
+ Code
+ Text
RAM
Disk
3/3 0s 20ms/step - loss: 0.0366
Epoch 39/150
3/3 0s 21ms/step - loss: 0.0330
Epoch 40/150
3/3 0s 22ms/step - loss: 0.0365
Epoch 41/150
3/3 0s 27ms/step - loss: 0.0403
Epoch 42/150
3/3 0s 20ms/step - loss: 0.0354
Epoch 43/150
3/3 0s 18ms/step - loss: 0.0331
Epoch 44/150
3/3 0s 18ms/step - loss: 0.0386
Epoch 45/150
3/3 0s 19ms/step - loss: 0.0353
Epoch 46/150
3/3 0s 20ms/step - loss: 0.0299
Epoch 47/150
3/3 0s 20ms/step - loss: 0.0321
Epoch 48/150
3/3 0s 17ms/step - loss: 0.0280
Epoch 49/150
3/3 0s 31ms/step - loss: 0.0321
Epoch 50/150
3/3 0s 32ms/step - loss: 0.0378
Epoch 51/150
3/3 0s 27ms/step - loss: 0.0333
Epoch 52/150
3/3 0s 32ms/step - loss: 0.0270
Epoch 53/150
3/3 0s 27ms/step - loss: 0.0294
Epoch 54/150
3/3 0s 25ms/step - loss: 0.0253
Epoch 55/150
3/3 0s 26ms/step - loss: 0.0287
Epoch 56/150
3/3 0s 25ms/step - loss: 0.0313
Epoch 57/150
3/3 0s 23ms/step - loss: 0.0265
Epoch 58/150
3/3 0s 36ms/step - loss: 0.0296
Epoch 59/150
3/3 0s 25ms/step - loss: 0.0247
23s completed at 12:19PM


Untitled0.ipynb - Colab
colab.research.google.com/drive/1YytwLtgd6DAgKNqbLCXGlksHLioPYuR8#scrollTo=bpb-cJRcoqlC
Untitled0.ipynb
File Edit View Insert Runtime Tools Help
Share
Gemini
Commands
+ Code
+ Text
RAM
Disk
Epoch 60/150
3/3 0s 31ms/step - loss: 0.0262
Epoch 61/150
3/3 0s 29ms/step - loss: 0.0266
Epoch 62/150
3/3 0s 31ms/step - loss: 0.0292
Epoch 63/150
3/3 0s 32ms/step - loss: 0.0239
Epoch 64/150
3/3 0s 27ms/step - loss: 0.0224
Epoch 65/150
3/3 0s 34ms/step - loss: 0.0230
Epoch 66/150
3/3 0s 30ms/step - loss: 0.0239
Epoch 67/150
3/3 0s 19ms/step - loss: 0.0248
Epoch 68/150
3/3 0s 18ms/step - loss: 0.0266
Epoch 69/150
3/3 0s 23ms/step - loss: 0.0269
Epoch 70/150
3/3 0s 20ms/step - loss: 0.0261
Epoch 71/150
3/3 0s 20ms/step - loss: 0.0216
Epoch 72/150
3/3 0s 21ms/step - loss: 0.0250
Epoch 73/150
3/3 0s 22ms/step - loss: 0.0241
Epoch 74/150
3/3 0s 22ms/step - loss: 0.0228
Epoch 75/150
3/3 0s 22ms/step - loss: 0.0214
Epoch 76/150
3/3 0s 28ms/step - loss: 0.0233
Epoch 77/150
3/3 0s 20ms/step - loss: 0.0239
Epoch 78/150
3/3 0s 19ms/step - loss: 0.0184
Epoch 79/150
3/3 0s 18ms/step - loss: 0.0231
Epoch 80/150
3/3 0s 18ms/step - loss: 0.0218
Epoch 81/150
23s completed at 12:19PM


Untitled0.ipynb - Colab
colab.research.google.com/drive/1YytwLtgd6DAgKNqbLCXGlksHLioPYuR8#scrollTo=bpb-cJRcoqlC
Untitled0.ipynb
File Edit View Insert Runtime Tools Help
Share
Gemini
Commands
+ Code
+ Text
RAM
Disk
3/3 0s 19ms/step - loss: 0.0241
Epoch 82/150
3/3 0s 22ms/step - loss: 0.0227
Epoch 83/150
3/3 0s 19ms/step - loss: 0.0265
Epoch 84/150
3/3 0s 19ms/step - loss: 0.0224
Epoch 85/150
3/3 0s 18ms/step - loss: 0.0221
Epoch 86/150
3/3 0s 27ms/step - loss: 0.0230
Epoch 87/150
3/3 0s 19ms/step - loss: 0.0243
Epoch 88/150
3/3 0s 18ms/step - loss: 0.0224
Epoch 89/150
3/3 0s 19ms/step - loss: 0.0208
Epoch 90/150
3/3 0s 18ms/step - loss: 0.0186
Epoch 91/150
3/3 0s 18ms/step - loss: 0.0208
Epoch 92/150
3/3 0s 18ms/step - loss: 0.0170
Epoch 93/150
3/3 0s 18ms/step - loss: 0.0227
Epoch 94/150
3/3 0s 18ms/step - loss: 0.0200
Epoch 95/150
3/3 0s 19ms/step - loss: 0.0232
Epoch 96/150
3/3 0s 22ms/step - loss: 0.0166
Epoch 97/150
3/3 0s 19ms/step - loss: 0.0183
Epoch 98/150
3/3 0s 19ms/step - loss: 0.0193
Epoch 99/150
3/3 0s 21ms/step - loss: 0.0205
Epoch 100/150
3/3 0s 21ms/step - loss: 0.0215
Epoch 101/150
3/3 0s 20ms/step - loss: 0.0199
Epoch 102/150
3/3 0s 21ms/step - loss: 0.0185
23s completed at 12:19PM


Untitled0.ipynb - Colab
colab.research.google.com/drive/1YytwLtgd6DAgKNqbLCXGlksHLioPYuR8#scrollTo=bpb-cJRcoqlC
Untitled0.ipynb
File Edit View Insert Runtime Tools Help
Share
Gemini
Commands
+ Code
+ Text
RAM
Disk
Epoch 103/150
3/3 0s 21ms/step - loss: 0.0199
Epoch 104/150
3/3 0s 21ms/step - loss: 0.0197
Epoch 105/150
3/3 0s 25ms/step - loss: 0.0171
Epoch 106/150
3/3 0s 25ms/step - loss: 0.0158
Epoch 107/150
3/3 0s 21ms/step - loss: 0.0178
Epoch 108/150
3/3 0s 22ms/step - loss: 0.0156
Epoch 109/150
3/3 0s 17ms/step - loss: 0.0190
Epoch 110/150
3/3 0s 17ms/step - loss: 0.0195
Epoch 111/150
3/3 0s 18ms/step - loss: 0.0219
Epoch 112/150
3/3 0s 18ms/step - loss: 0.0174
Epoch 113/150
3/3 0s 21ms/step - loss: 0.0172
Epoch 114/150
3/3 0s 17ms/step - loss: 0.0161
Epoch 115/150
3/3 0s 17ms/step - loss: 0.0183
Epoch 116/150
3/3 0s 17ms/step - loss: 0.0129
Epoch 117/150
3/3 0s 24ms/step - loss: 0.0172
Epoch 118/150
3/3 0s 20ms/step - loss: 0.0172
Epoch 119/150
3/3 0s 20ms/step - loss: 0.0169
Epoch 120/150
3/3 0s 21ms/step - loss: 0.0169
Epoch 121/150
3/3 0s 21ms/step - loss: 0.0173
Epoch 122/150
3/3 0s 21ms/step - loss: 0.0178
Epoch 123/150
3/3 0s 22ms/step - loss: 0.0177
Epoch 124/150
23s completed at 12:19PM





Post Lab Question- Answers (If Any):

1. How does LSTM solve the vanishing gradient challenge?

LSTM solves the vanishing gradient problem by introducing memory cells that can maintain information over long periods. The gradient can flow through these cells without decaying, as they have self-loops allowing the gradient to be passed through longer sequences.

2. What are the practical applications of LSTM?

LSTM networks are used in a variety of fields, including:

• Time series forecasting (e.g., stock market prediction)

• Speech recognition and synthesis

• Handwriting recognition and generation

• Language translation

• Image captioning

• Parsing

CO – 4: Underhand the essentials of Recurrent and Recursive Nets.

Conclusion:

LSTM networks are powerful tools for solving sequence-related problems by remembering long-term dependencies. The lab demonstrated how to build, train, and evaluate an LSTM model on time series data.

Grade: AA / AB / BB / BC / CC / CD / DD

Signature of faculty in-charge with date

References:

Books/ Journals/ Websites:

1. Josh Patterson and Adam Gibson, “Deep Learning A Practitioner’s Approach”, O’Reilly Media 2017
2. https://www.ibm.com/cloud/learn/recurrent-neural-networks
3. https://searchenterpriseai.techtarget.com/definition/recurrent-neural-networks
4. https://www.sciencedirect.com/science/article/pii/B9780128161760000260
