import numpy as np

import tensorflow as tf

from tensorflow.keras.models import Sequential

from tensorflow.keras.layers import LSTM, Dense

import matplotlib.pyplot as plt

# 1. Generate Sample Data

# Simulating monthly spending data (e.g., salary utilization).

spending\_data = np.array([2500, 2600, 2700, 2900, 3100, 3000, 3200, 3400, 3600, 3800,

3700, 3900, 4100, 4200, 4000, 4300, 4500, 4700, 4900, 5000])

# Hyperparameters

seq\_len = 3 # Sequence length for the LSTM input

# 2. Prepare Data for Training

# Create sequences and targets

X, y = [], []

for i in range(len(spending\_data) - seq\_len):

X.append(spending\_data[i:i+seq\_len])

y.append(spending\_data[i+seq\_len])

X = np.array(X).reshape(-1, seq\_len, 1) # Reshape to 3D for LSTM [samples, time steps, features]

y = np.array(y).reshape(-1, 1) # Reshape to 2D for Keras

# 3. Build LSTM Model

model = Sequential([

LSTM(50, activation='relu', input\_shape=(seq\_len, 1)), # LSTM layer

Dense(1) # Output layer (predict next value)

])

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

print(model.summary())

# 4. Train the Model

history = model.fit(X, y, epochs=100, batch\_size=1, verbose=1)

# 5. Visualize Training Loss

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

plt.title('Model Loss Over Epochs')

plt.xlabel('Epochs')

plt.ylabel('Loss')

plt.show()

# 6. Make a Prediction

# Input the most recent sequence for prediction (e.g., last 3 months of data)

latest\_input = np.array([4700, 4900, 5000]).reshape(1, seq\_len, 1)

predicted\_spending = model.predict(latest\_input)

print(f"Predicted Next Month's Spending: {predicted\_spending[0][0]:.2f}")

# 7. Save the Model

model.save('lstm\_financial\_model.h5')

print("Model saved as 'lstm\_financial\_model.h5'!")

**Explanation of the Code:**

1. **Data Preparation**:

* The spending data is split into sequences (X) and corresponding next values (y).
* For example, for a sequence length of 3:
  + - Input (X): [2500, 2600, 2700], Target (y): 2900.

1. **Model**:
   * + The LSTM layer processes the time-series data, and a Dense layer predicts the next spending value.
2. **Training**:
   * + The model is trained for 100 epochs to minimize the Mean Squared Error (MSE) loss.
3. **Prediction**:
   * + After training, the most recent sequence is used to predict the next month's spending.
4. **Visualization**:
   * + The loss progression is plotted to analyze how the model learns over epochs.
5. **Saving the Model**:
   * + The trained model is saved as a file (.h5) for future use.