**Develop neural network-based time series forecasting model.**

**Aim:**

To develop an **neural network model** for forecasting the **rank trends** in covid 19 dataset using time series analysis.

**Procedure:**

1. Load the Google Trends dataset, convert the year column to datetime, and **Install TensorFlow (if not already installed)**:
   * Install TensorFlow using !pip install tensorflow --quiet (if needed).
2. **Import Libraries**:
   * Import the necessary libraries: pandas, numpy, matplotlib.pyplot, tensorflow, MinMaxScaler from sklearn.preprocessing, and train\_test\_split from sklearn.model\_selection.
3. **Load the Dataset**:
   * Load the COVID-19 dataset using pandas.read\_csv().
   * Clean column names by stripping any leading or trailing spaces.
   * Convert the Date column to datetime format and set it as the index.
4. **Calculate Daily New Global Cases**:
   * Group the dataset by Date and sum the confirmed cases globally.
   * Calculate **daily new cases** by taking the difference between consecutive days using .diff().
   * Drop any NaN values that arise from the first row due to the diff() operation.
5. **Resample to Monthly Average Daily Cases**:
   * Resample the daily data to monthly frequency using .resample("M") and compute the **monthly average daily cases**.
   * Set Date as the index of the resampled data.
6. **Normalize the Data**:
   * Apply **MinMax scaling** using MinMaxScaler to scale the daily cases between 0 and 1 for better performance with LSTM models.
   * Store the scaled data in a new column cases\_scaled.
7. **Create Dataset for LSTM**:
   * Define a function create\_dataset() to prepare the data for LSTM training.
   * Create sequences of data (X) and corresponding labels (y) using the specified time\_steps (3 in this case).
   * Reshape the input data to the format [samples, time steps, features] required by LSTM.
8. **Split Data into Train and Test Sets**:
   * Use train\_test\_split() to split the data into training and testing sets (80% train, 20% test), without shuffling to maintain the temporal order.
9. **Build and Train the LSTM Model**:
   * Create a sequential LSTM model with two LSTM layers and one Dense layer.
   * Compile the model using the Adam optimizer and Mean Squared Error (MSE) loss.
   * Train the model for 50 epochs with a batch size of 8, validating on the test data.
10. **Make Predictions and Inverse Transform**:
    * Use the trained model to make predictions on the test data.
    * Inverse transform the predicted values and actual test values back to the original scale using the scaler.
11. **Plot the Results**:
    * Plot the actual vs predicted monthly average daily cases for the test set.
    * Use markers and different line styles to differentiate between actual and predicted values.
    * Customize the plot with labels, a title, and a legend, and display it.
12. **Code :**

# Install TensorFlow if not installed

!pip install tensorflow --quiet

import pandas as pd

import numpy as np

import matplotlib.pyplot as plt

import tensorflow as tf

from tensorflow.keras.models import Sequential

from tensorflow.keras.layers import LSTM, Dense

from sklearn.preprocessing import MinMaxScaler

from sklearn.model\_selection import train\_test\_split

df = pd.read\_csv("/time-series-19-covid-combined.csv")

df.columns = df.columns.str.strip()

df["Date"] = pd.to\_datetime(df["Date"])

df.set\_index("Date", inplace=True)

daily\_df = df.groupby(df.index).sum(numeric\_only=True)

daily\_df["daily\_cases"] = daily\_df["Confirmed"].diff()

daily\_df.dropna(subset=["daily\_cases"], inplace=True)

# Resample to monthly average daily cases

monthly\_avg = daily\_df["daily\_cases"].resample("M").mean().reset\_index()

monthly\_avg.set\_index("Date", inplace=True)

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

monthly\_avg["cases\_scaled"] = scaler.fit\_transform(monthly\_avg[["daily\_cases"]])

def create\_dataset(data, time\_steps=3):

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 = create\_dataset(monthly\_avg["cases\_scaled"].values, time\_steps)

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

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.2, shuffle=False)

model = Sequential([

LSTM(50, activation='relu', return\_sequences=True, input\_shape=(time\_steps, 1)),

LSTM(50, activation='relu'),

Dense(1)

])

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

model.fit(X\_train, y\_train, epochs=50, batch\_size=8, validation\_data=(X\_test, y\_test), verbose=1)

y\_pred = model.predict(X\_test)

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

y\_test\_actual = scaler.inverse\_transform(y\_test.reshape(-1, 1))

test\_dates = monthly\_avg.index[-len(y\_test):]

plt.figure(figsize=(10, 5))

plt.plot(test\_dates, y\_test\_actual, label="Actual Avg Daily Cases", marker='o')

plt.plot(test\_dates, y\_pred\_actual, label="Predicted Avg Daily Cases", linestyle='dashed', marker='s')

plt.xlabel("Month")

plt.ylabel("Avg Daily Cases")

plt.title("LSTM Forecasting of COVID-19 Avg Daily Cases (Monthly)")

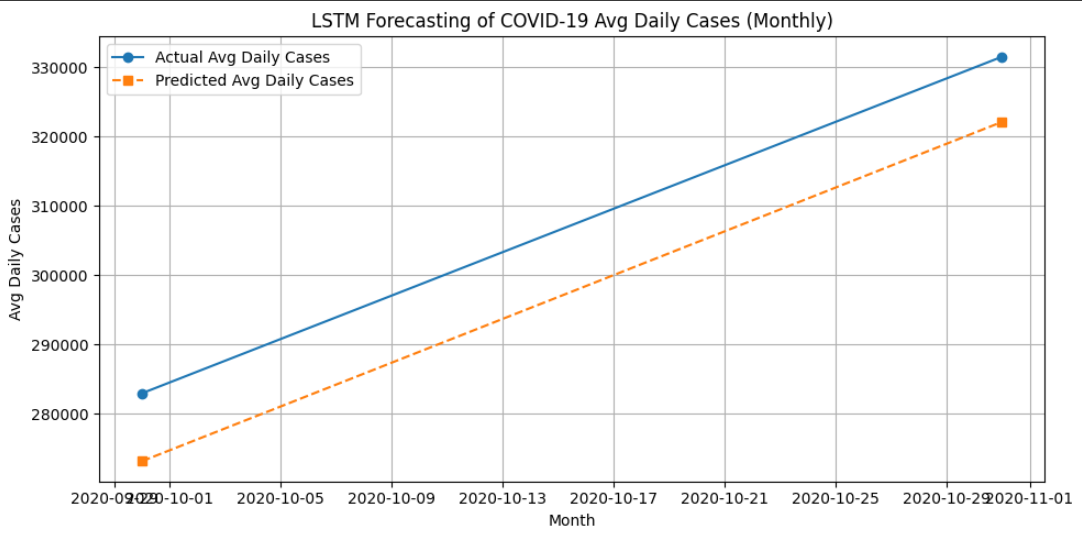
plt.legend()

plt.grid(True)

plt.tight\_layout()

plt.show()

**Output:**



**Result:**

The program to develop a neural network based time series forecasting model has been successfully implemented