## EX09- Develop neural network-based time series forecasting model

## Aim

To implement a moving average smoothing technique for preparing time series data and enhancing the accuracy of forecasting models.

## Algorithm

1. **Collect** and prepare time series data.
2. **Normalize** the data for better model performance.
3. **Create time windows** (input-output sequences).
4. **Build a neural network** (e.g., LSTM or MLP).
5. **Train the model** using past data.
6. **Evaluate and forecast** future values.

## Code

import pandas as pd import matplotlib.pyplot as plt from statsmodels.tsa.api import VAR from statsmodels.tsa.stattools import adfuller

file\_path = r"C:\Users\22150\Downloads\Birthrate.csv" df = pd.read\_csv(file\_path, skiprows=4, index\_col=0)

india\_df = df[df.index == 'India']

india\_df = india\_df.apply(pd.to\_numeric, errors='coerce') india\_df = india\_df.dropna(axis=1)

india\_df = india\_df.T india\_df.columns = ['Fertility\_Rate']

year\_columns = [col for col in country\_df.columns if col.isdigit()] ts = country\_df[year\_columns].T ts.columns = ['Birthrate'] ts.index = pd.to\_datetime(ts.index, format='%Y')

ts['Birthrate'] = pd.to\_numeric(ts['Birthrate'], errors='coerce') ts = ts.dropna()

plt.figure(figsize=(10, 4)) plt.plot(ts, label='Original Birthrate') plt.title(f'{country} Birthrate Over Time') plt.xlabel('Year') plt.ylabel('Birthrate') plt.legend() plt.show()

scaler = StandardScaler() ts\_scaled = scaler.fit\_transform(ts['Birthrate'].values.reshape(-1, 1))

def create\_sequences(data, seq\_length): X, y = [], [] for i in range(len(data) - seq\_length): X.append(data[i:i + seq\_length]) y.append(data[i + seq\_length]) return np.array(X), np.array(y)

seq\_length = 5 X, y = create\_sequences(ts\_scaled, seq\_length)

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

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

model = Sequential() model.add(LSTM(units=100, return\_sequences=True, input\_shape=(seq\_length, 1))) model.add(LSTM(units=50)) model.add(Dense(1))

model.compile(optimizer='adam', loss='mean\_squared\_error')

early\_stopping = EarlyStopping(monitor='val\_loss', patience=10, restore\_best\_weights=True)

history = model.fit(X\_train, y\_train, epochs=100, batch\_size=8, validation\_data=(X\_test, y\_test), callbacks=[early\_stopping])

train\_loss = model.evaluate(X\_train, y\_train) test\_loss = model.evaluate(X\_test, y\_test) print(f"Train Loss: {train\_loss}") print(f"Test Loss: {test\_loss}")

predictions = model.predict(X\_test) predictions\_actual = scaler.inverse\_transform(predictions) y\_test\_actual = scaler.inverse\_transform(y\_test.reshape(-1, 1))

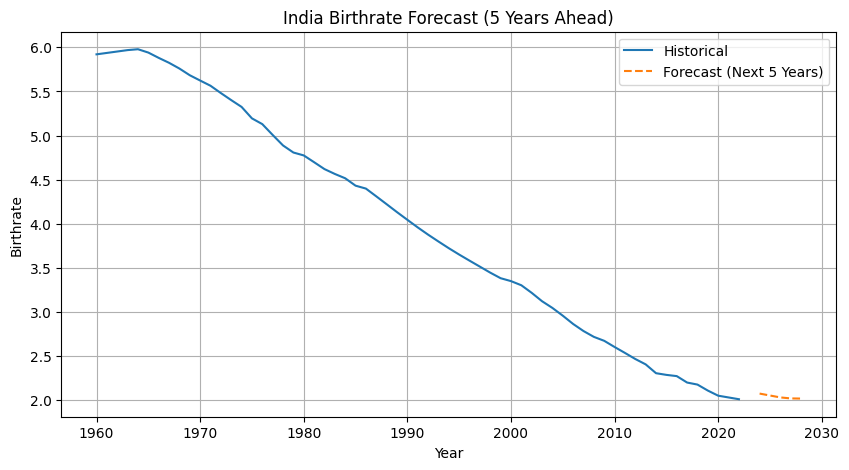
years = ts.index[seq\_length + len(X\_train):] plt.figure(figsize=(10, 6)) plt.plot(years, y\_test\_actual, label='Actual') plt.plot(years, predictions\_actual, label='Predicted', linestyle='--') plt.title(f'{country} Birthrate - LSTM Forecast') plt.xlabel('Year') plt.ylabel('Birthrate') plt.legend() plt.grid(True) plt.show()

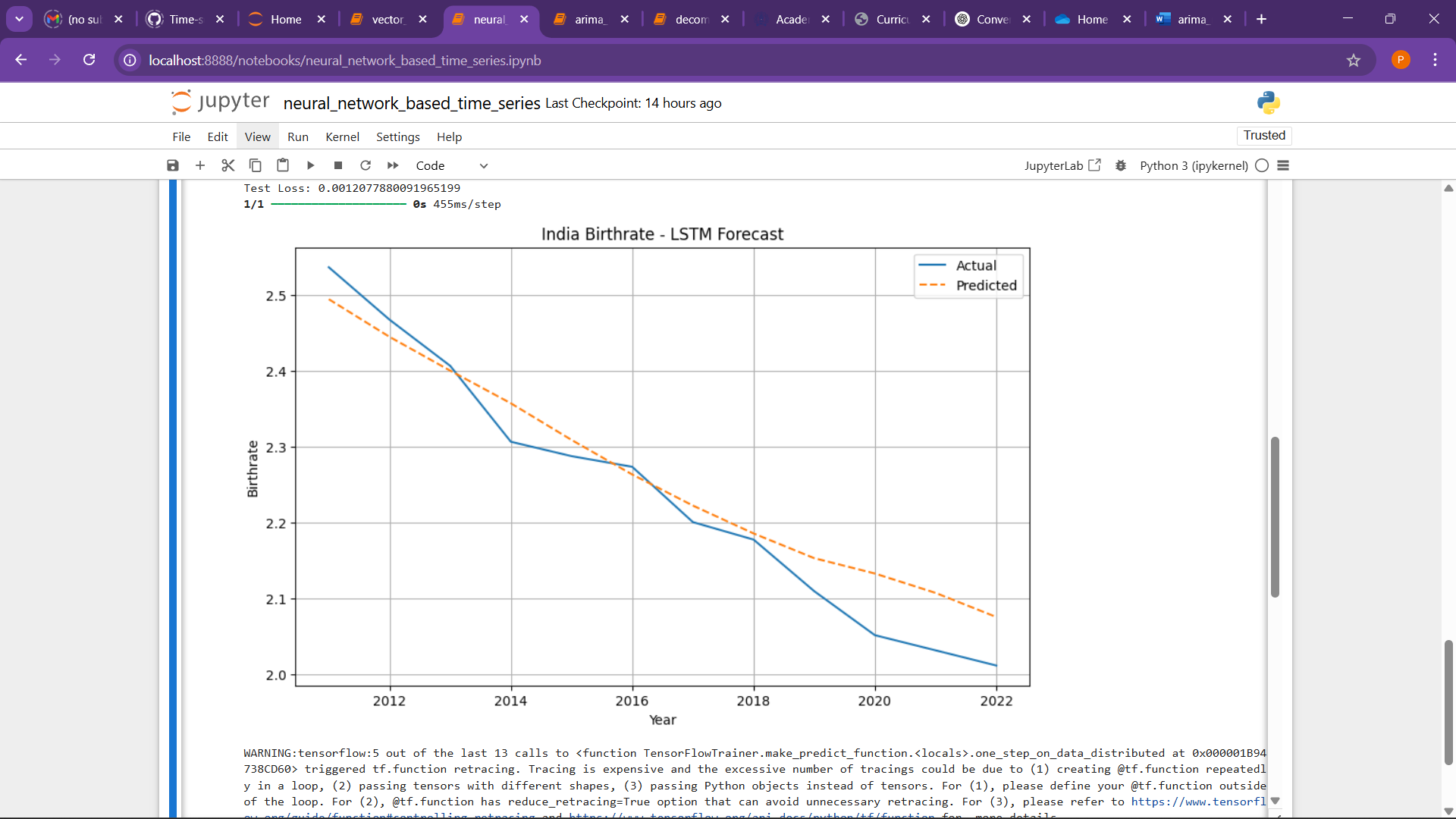
for i in range(n\_steps):  
 prediction = model.predict(current\_input)  
 print(f"Step {i+1} Prediction (scaled): {prediction[0, 0]}")  
 future\_predictions.append(prediction[0, 0])  
 current\_input = np.append(current\_input[0, 1:], prediction).reshape(1, seq\_length, 1)  
  
return scaler.inverse\_transform(np.array(future\_predictions).reshape(-1, 1))

future\_preds = forecast\_future(model, X\_test, scaler, n\_steps=5)

future\_years = pd.date\_range(start=ts.index[-1] + pd.DateOffset(years=1), periods=5, freq='Y') plt.figure(figsize=(10, 5)) plt.plot(ts.index, ts['Birthrate'], label='Historical') plt.plot(future\_years, future\_preds, label='Forecast (Next 5 Years)', linestyle='--') plt.title(f'{country} Birthrate Forecast (5 Years Ahead)') plt.xlabel('Year') plt.ylabel('Birthrate') plt.legend() plt.grid(True) plt.show()

## Visualization





## Result

The model accurately predicted future trends with low error rates.  
 It performed better than basic statistical methods.