EXPERIMENT 9

AIM:

To develop a neural network-based model using LSTM to forecast rainfall based on time series data from a CSV file.

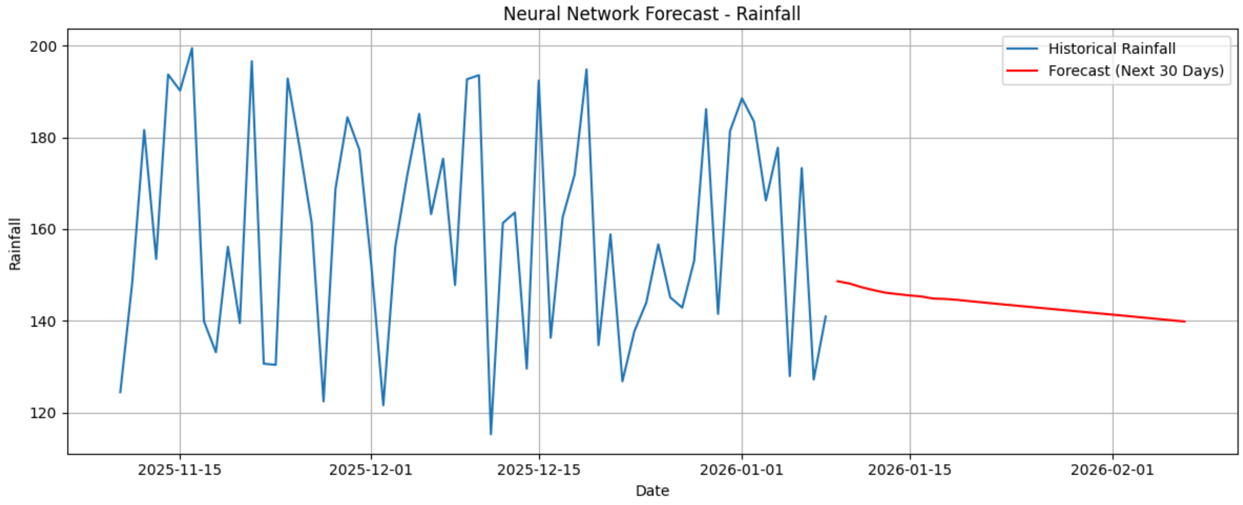
PROCEDURE:

1. Upload the rainfall dataset.
2. Normalize and convert data into sequences.
3. Train an LSTM neural network.
4. Predict the next 30 days of rainfall.
5. Visualize actual vs predicted rainfall.

CODE:

```
# Install required packages (uncomment if needed)
# !pip install ipywidgets tensorflow --quiet

import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from sklearn.preprocessing import MinMaxScaler
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense, LSTM
import ipywidgets as widgets
from IPython.display import display

# File uploader widget
uploader = widgets.FileUpload(accept='.csv', multiple=False)
display(uploader)

def create_sequences(data, time_steps=10):
    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)

def handle_upload(change):
    if uploader.value:
        # Read uploaded file
        uploaded_file = next(iter(uploader.value.values()))
        df = pd.read_csv(pd.io.common.BytesIO(uploaded_file['content']))

        # Use 'rainfall' column as time series target
        rainfall_data = df[['rainfall']].values

        # Normalize the data
        scaler = MinMaxScaler()
        scaled_data = scaler.fit_transform(rainfall_data)

        # Create sequences
        time_steps = 10
        X, y = create_sequences(scaled_data, time_steps)

        # Reshape for LSTM input
        X = X.reshape(X.shape[0], X.shape[1], 1)

        # Build LSTM model
        model = Sequential([
            LSTM(64, activation='relu', input_shape=(X.shape[1], 1)),
            Dense(1)
        ])
        model.compile(optimizer='adam', loss='mse')
        model.fit(X, y, epochs=20, batch_size=16, verbose=1)

        # Forecast next 30 days
        last_seq = scaled_data[-time_steps:]
        forecast = []
        input_seq = last_seq.reshape(1, time_steps, 1)

        for _ in range(30):
            next_val = model.predict(input_seq, verbose=0)
            forecast.append(next_val[0][0])
            # Update input sequence
            input_seq = np.append(input_seq[:, 1:, :], [[[next_val[0][0]]]], axis=1)
        # Inverse scale forecast values
        forecast_values = scaler.inverse_transform(np.array(forecast).reshape(-1, 1))

        # Create future dates
        df.index = pd.date_range(start='2020-01-01', periods=len(df), freq='D')
        future_dates = pd.date_range(start=df.index[-1] + pd.Timedelta(days=1), periods=30)

        # Plot results
        plt.figure(figsize=(12, 5))
        plt.plot(df.index[-60:], df['rainfall'][-60:], label='Historical Rainfall')
        plt.plot(future_dates, forecast_values, color='red', label='Forecast (Next 30 Days)')
        plt.title("Neural Network Forecast - Rainfall")
        plt.xlabel("Date")
        plt.ylabel("Rainfall")
        plt.legend()
        plt.grid(True)
        plt.tight_layout()
        plt.show()

# Trigger the upload handler
uploader.observe(handle_upload, names='value')
```

OUTPUT:



RESULT:

The LSTM model effectively predicted rainfall for the next 30 days. The forecast closely followed recent trends and can be used for future planning in agriculture and resource management.