EX -9
17/04/2025

Develop neural network-based time series forecasting model

AIM :
Develop neural network-based time series forecasting model

Procedure and Code :

Step 1 - Import the Files and Libraries .

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from sklearn.preprocessing import MinMaxScaler
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import LSTM, Dense, Dropout
from tensorflow.keras.callbacks import EarlyStopping
from sklearn.metrics import mean_squared_error
```

Step 2 - Describe and Read the Data

```
df = pd.read_csv('/content/drive/MyDrive/TimeSereisDatasets/EX-9/Copy of daily-website-visitors.csv')
df['Date'] = pd.to_datetime(df['Date'])
df.set_index('Date', inplace=True)
df['Unique.Visits'] = df['Unique.Visits'].str.replace(',', '').astype(int)
ts = df['Unique.Visits']

df['Page.Loads'] = df['Page.Loads'].str.replace(',', '').astype(int)
```

EX -9
17/04/2025

```
ts = df['Unique.Visits']
```

Step 3 - `Models`

```
model = Sequential()
model.add(LSTM(50, return_sequences=True, input_shape=(look_back, 1)))
model.add(Dropout(0.2))
model.add(LSTM(50, return_sequences=False))
model.add(Dropout(0.2))
model.add(Dense(25))
model.add(Dense(1))

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

# Early stopping to prevent overfitting
```

Step 4 - `Neural Network Model training and Evaluation`

```
history = model.fit(X_train, y_train,
                    epochs=50,
                    batch_size=32,
                    validation_data=(X_test, y_test),
                    callbacks=[early_stop],
                    verbose=1)
```

EX -9
17/04/2025

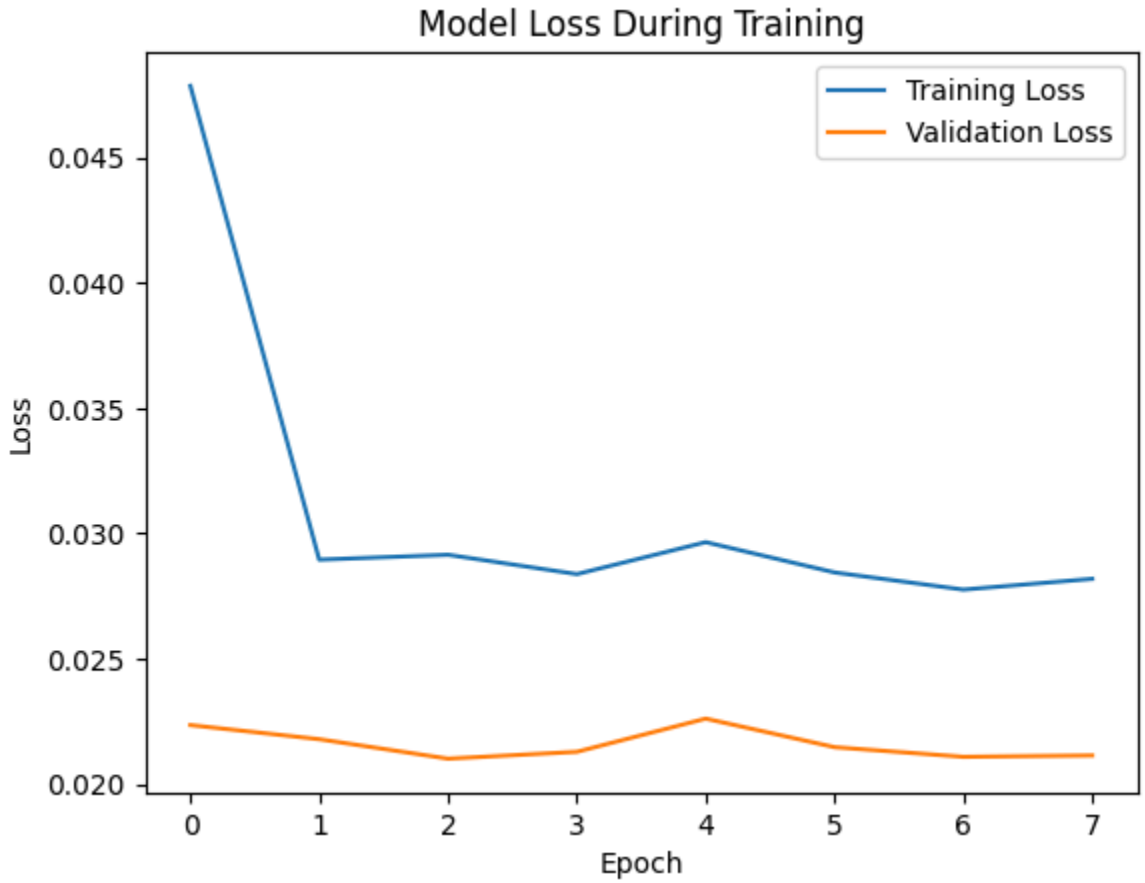
```
# Plot training history
plt.plot(history.history['loss'], label='Training Loss')
plt.plot(history.history['val_loss'], label='Validation Loss')
plt.title('Model Loss During Training')
plt.ylabel('Loss')
plt.xlabel('Epoch')
plt.legend()
plt.show()


# Make predictions
train_predict = model.predict(X_train)
test_predict = model.predict(X_test)

# Inverse transform predictions
train_predict = scaler.inverse_transform(train_predict)
y_train_actual = scaler.inverse_transform(y_train.reshape(-1, 1))
test_predict = scaler.inverse_transform(test_predict)
y_test_actual = scaler.inverse_transform(y_test.reshape(-1, 1))

# Calculate RMSE
train_rmse = np.sqrt(mean_squared_error(y_train_actual, train_predict))
test_rmse = np.sqrt(mean_squared_error(y_test_actual, test_predict))
print(f'Train RMSE: {train_rmse:.2f}')
print(f'Test RMSE: {test_rmse:.2f}')

# Plot predictions vs actual
plt.figure(figsize=(12,6))
plt.plot(ts.index[look_back:train_size+look_back], y_train_actual,
label='Training Actual')
plt.plot(ts.index[train_size+look_back:-1], y_test_actual, label='Test
Actual')
plt.plot(ts.index[look_back:train_size+look_back], train_predict,
label='Training Predictions')
plt.plot(ts.index[train_size+look_back:-1], test_predict, label='Test
Predictions')
plt.title('LSTM Model Predictions vs Actual')
plt.legend()
plt.show()
```

EX -9
17/04/2025

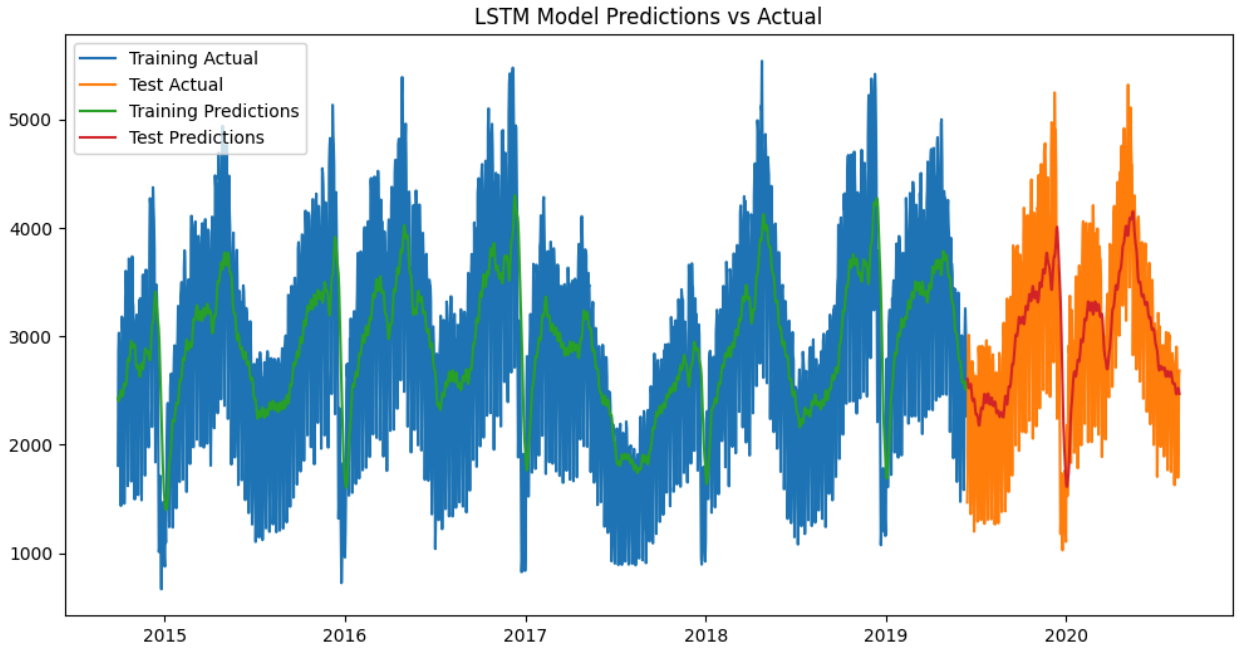
Step 5 - Multi Step Forecasting

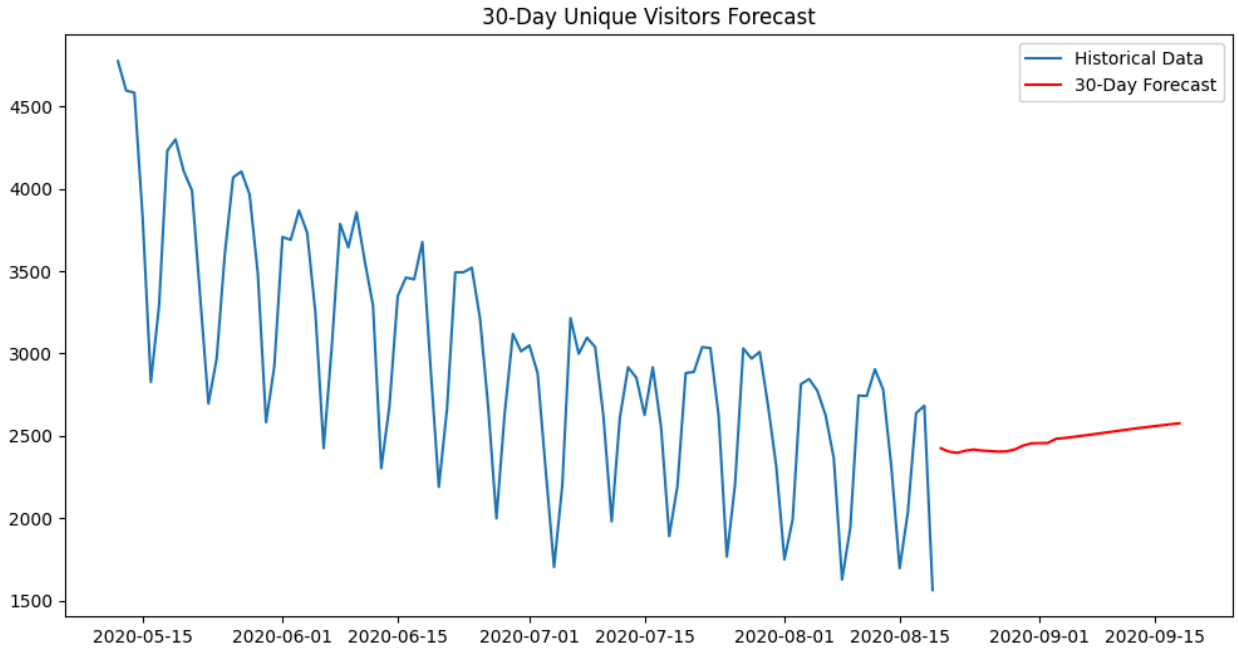
```
    def forecast_future(model, last_sequence, n_steps):
        forecast = []
        current_sequence = last_sequence.copy()

        for _ in range(n_steps):
            # Get prediction
            next_pred = model.predict(current_sequence.reshape(1, look_back, 1))
            forecast.append(next_pred[0,0])

            # Update sequence
            current_sequence = np.roll(current_sequence, -1)
            current_sequence[-1] = next_pred

        return forecast

    # Get last sequence from data
    last_sequence = scaled_data[-look_back:]

    # Forecast next 30 days
    forecast_steps = 30
    forecast_scaled = forecast_future(model, last_sequence, forecast_steps)
    forecast = scaler.inverse_transform(np.array(forecast_scaled).reshape(-1, 1))

    # Create date index for forecast
    forecast_dates = pd.date_range(start=ts.index[-1] + pd.Timedelta(days=1),
periods=forecast_steps)

    # Plot forecast
    plt.figure(figsize=(12,6))
    plt.plot(ts.index[-100:], ts.values[-100:], label='Historical Data')
    plt.plot(forecast_dates, forecast, label='30-Day Forecast', color='red')
    plt.title('30-Day Unique Visitors Forecast')
    plt.legend()
    plt.show()
```

EX -9
17/04/2025



EX -9
17/04/2025





Result:

Thus the Program has been Executed Successfully.

EX -9
17/04/2025