EX:No.9 DATE:12/04/25	**Develop neural network-based time series forecasting model.**

AIM:

To Develop neural network-based time series forecasting model.

ALGORITHM:

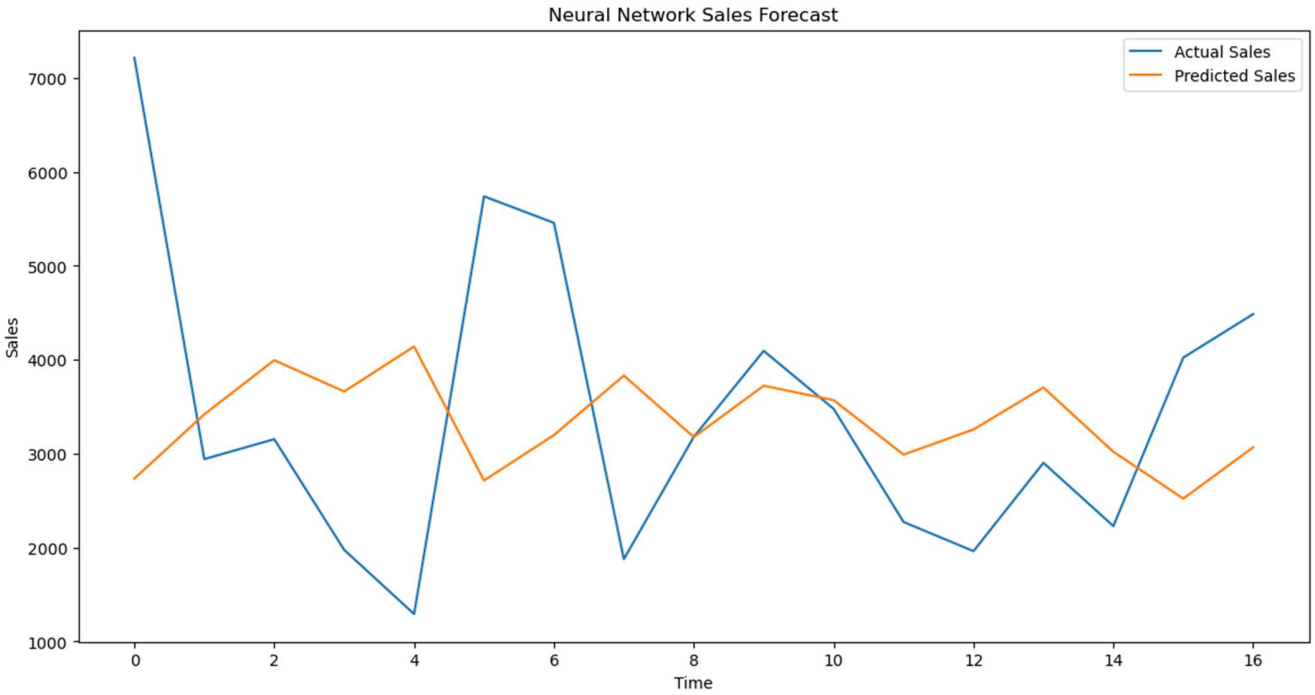
1. **Data Cleaning** – Loaded the dataset, parsed dates, fixed encoding issues, and selected only the PM2.5 column.
2. **Normalization** – Scaled PM2.5 values between 0 and 1 using MinMaxScaler to improve neural network performance.
3. **Sequence Creation** – Created supervised learning format by using the previous 10 timesteps to predict the next one.
4. **Train-Test Split** – Split the dataset into 80% training and 20% testing sets.
5. **Model Building** – Built an LSTM model with one LSTM layer (50 units) and one Dense output layer.
6. **Model Training** – Trained the model using training data for 20 epochs with a batch size of 32.
7. **Prediction & Inverse Scaling** – Predicted future values and converted them back to original scale.
8. **Visualization** – Plotted actual vs predicted PM2.5 values to evaluate model performance.

Code:

```
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 LSTM, Dense

# Load dataset
df = pd.read_csv('/content/us_air_pollution_2012_2021.csv', parse_dates=['Date'])
df.set_index('Date', inplace=True)

# Clean and select only PM2.5
df.columns = [col.replace("Â", "") for col in df.columns]
data = df[['PM2.5 (µg/m³)']].dropna()

# Normalize the data
scaler = MinMaxScaler()
scaled = scaler.fit_transform(data)
```

```
 (y) # Split into train and test
split = int(len(X) * 0.8)
X_train, X_test = X[:split], X[split:]
y_train, y_test = y[:split], y[split:]

# Build model
model = Sequential()
model.add(LSTM(50, activation='relu', input_shape=(10, 1)))
model.add(Dense(1))
model.compile(optimizer='adam',  loss='mse')

# Train model
model.fit(X_train, y_train, epochs=20, batch_size=32, validation_data=(X_test, y_test))

# Predict and inverse scale
pred = model.predict(X_test)
pred = scaler.inverse_transform(pred)
actual = scaler.inverse_transform(y_test)

# Plot
plt.plot(actual, label='Actual PM2.5')
plt.plot(pred, label='Predicted PM2.5')
plt.legend()
plt.title('PM2.5 Prediction using LSTM')
plt.show()
```

OUTPUT:



RESULT:

Thus, the program using the time series data implementation has been done successfully.