EX:No.9 221501060

12/04/25

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

**Aim:**

ToDevelop neural network-based time series forecasting model

**Algorithm:**

1. **Load the Data**:
   * Read the CSV file containing the weather data.
   * Parse the date column as a datetime index.
2. **Clean the Data**:
   * Handle missing values by performing forward and backward filling.
   * Drop any remaining NaN values.
3. **Normalize the Data**:
   * Apply **Min-Max Scaling** to normalize each column's values between 0 and 1.
4. **Add Time-Based Features**:
   * Extract additional features from the datetime index: day, month and year
5. **Visualize the Data**:
   * Plot the time series for a specific column (e.g., temperature T) over time.
6. **Execute the Program**:
   * Sequentially call the functions to load, clean, normalize, add features, and visualize the data.

Code:

import numpy as np

import pandas as pd

import matplotlib.pyplot as plt

from sklearn.preprocessing import MinMaxScaler

from sklearn.metrics import mean\_squared\_error

from tensorflow.keras.models import Sequential

from tensorflow.keras.layers import Dense

np.random.seed(42)

date\_range = pd.date\_range(start='2018-01-01', periods=100, freq='M')

energy\_consumption = np.random.normal(loc=300, scale=20, size=100).cumsum()

df = pd.DataFrame({'Date': date\_range, 'Energy': energy\_consumption})

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

scaler = MinMaxScaler()

scaled\_energy = scaler.fit\_transform(df[['Energy']])

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

X, y = [], []

for i in range(len(data) - n\_steps):

X.append(data[i:i + n\_steps, 0])

y.append(data[i + n\_steps, 0])

return np.array(X), np.array(y)

n\_steps = 5

X, y = create\_dataset(scaled\_energy, n\_steps=n\_steps)

train\_size = int(len(X) \* 0.8)

X\_train, X\_test = X[:train\_size], X[train\_size:]

y\_train, y\_test = y[:train\_size], y[train\_size:]

model = Sequential()

model.add(Dense(64, activation='relu', input\_shape=(n\_steps,)))

model.add(Dense(32, activation='relu'))

model.add(Dense(1))

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

model.fit(X\_train, y\_train, epochs=100, verbose=0)

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

y\_pred\_rescaled = scaler.inverse\_transform(y\_pred.reshape(-1, 1))

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

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

plt.plot(df.index[-len(y\_test):], y\_test\_rescaled, label='Actual')

plt.plot(df.index[-len(y\_test):], y\_pred\_rescaled, label='Predicted')

plt.legend()

plt.title('Neural Network Forecast on Energy Consumption')

plt.xlabel('Date')

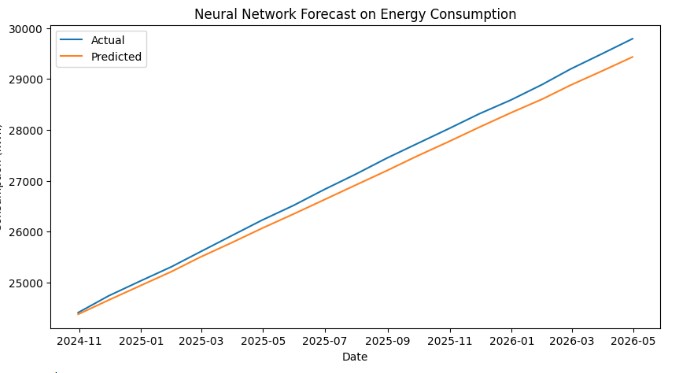
plt.ylabel('Consumption (kWh)')

plt.show()

mse = mean\_squared\_error(y\_test\_rescaled, y\_pred\_rescaled)

print(f'Mean Squared Error: {mse:.2f}')

**Output:**

****

**Result:**

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