**EX:No.9 221501034**

**15/04/25**

**DEVELOP NEURAL NETWORK-BASED TIME SERIES FORCASTING MODEL.**

**AIM:**

To implement program for Develop neural network-based time series forecasting model.

**ALGORITHM:**

**OBJECTIVE:**

Smooth the electric production data to reduce noise, highlight trends, and prepare for forecasting.

**BACKGROUND:**

1.Time series data has short-term fluctuations.

2.Moving average reduces noise and clarifies trends.

3.Smoothed data improves forecast accuracy and interpretability.

**SCOPE OF THE PROGRAM:**

1.Load and clean dataset

2.Convert date column to datetime

3.Aggregate data monthly and yearly

4.Apply 3-month and 12-month moving averages

5.Plot original vs smoothed data

**ALGORITHM:**

1.Import libraries

2.Load dataset

3.Preprocess and set datetime index

4.Resample data (monthly, yearly)

5.Apply 3-month & 12-month smoothing

6.Visualize results

**PROCESS:**

**# Install required packages (if not already available)**

**# !pip install pandas numpy matplotlib scikit-learn tensorflow**

**import pandas as pd**

**import numpy as np**

**import matplotlib.pyplot as plt**

**from sklearn.preprocessing import MinMaxScaler**

**from sklearn.model\_selection import train\_test\_split**

**from tensorflow.keras.models import Sequential**

**from tensorflow.keras.layers import Dense**

**from sklearn.metrics import mean\_squared\_error**

**# Load dataset**

**df = pd.read\_csv('/content/AAPL.csv', parse\_dates=['Date'], index\_col='Date')**

**# Use the 'Close' price column**

**data = df['Close'].values.reshape(-1, 1)**

**# Normalize the data**

**scaler = MinMaxScaler()**

**scaled\_data = scaler.fit\_transform(data)**

**# Function to prepare time series data for supervised learning**

**def create\_dataset(dataset, window\_size):**

**X, y = [], []**

**for i in range(len(dataset) - window\_size):**

**X.append(dataset[i:i+window\_size, 0])**

**y.append(dataset[i+window\_size, 0])**

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

**# Define time window**

**window\_size = 12**

**X, y = create\_dataset(scaled\_data, window\_size)**

**# Split into training and testing sets (80% training)**

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

**# Build Neural Network model**

**model = Sequential([**

**Dense(64, activation='relu', input\_shape=(window\_size,)),**

**Dense(32, activation='relu'),**

**Dense(1)**

**])**

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

**# Train the model**

**history = model.fit(X\_train, y\_train, epochs=100, batch\_size=16, validation\_data=(X\_test, y\_test), verbose=0)**

**# Predict on test set**

**predictions = model.predict(X\_test)**

**predictions = scaler.inverse\_transform(predictions.reshape(-1, 1))**

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

**# Plot results**

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

**plt.plot(actual, label='Actual')**

**plt.plot(predictions, label='Predicted', color='red')**

**plt.title('Neural Network Forecast vs Actual (AAPL Close Price)')**

**plt.xlabel('Time Steps')**

**plt.ylabel('Close Price (USD)')**

**plt.legend()**

**plt.grid(True)**

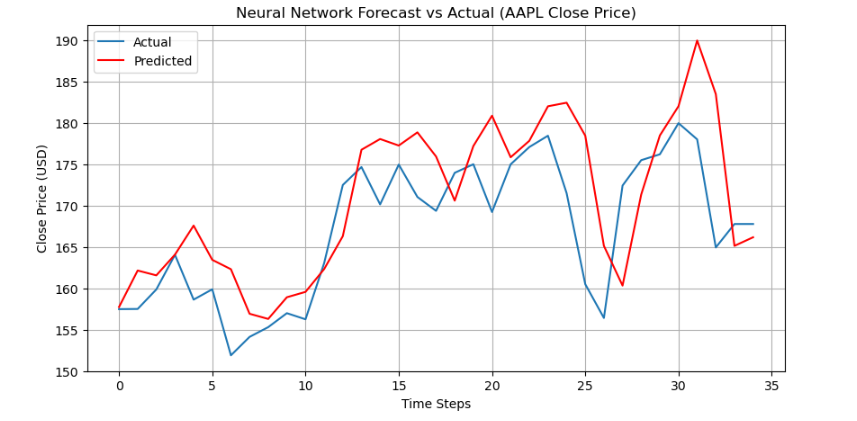
**plt.show()**

**# Show RMSE**

**rmse = np.sqrt(mean\_squared\_error(actual, predictions))**

**print(f'RMSE: {rmse:.4f}')**

**OUTPUT:**

****

**RESULT:**

The program to Develop neural network-based time series forecasting model created and executed successfully.