EX:No.9 221501046

Develop neural network-based time series forecasting model.

Aim:

Write a program to develop neural network-based time series forecasting model.

Algorithm:

1.**Import required libraries**:
Import numpy, pandas, matplotlib, MinMaxScaler from sklearn, and neural network layers from tensorflow.keras.

2.**Load the dataset**:
Read the CSV file containing art market data and parse the 'Date' column as datetime while setting it as the index.

3.**Select the target column**:
Extract the 'Price' column as the target for forecasting.

4.**Normalize the data**:
Scale the price values to a range between 0 and 1 using MinMaxScaler to improve neural network performance.

5.**Create sequences**:
Define a function that converts the scaled data into sequences of a specified number of time steps (TIME_STEPS = 10) and corresponding labels (next temperature value).

6.**Split the data**:
Divide the sequence data into training (80%) and testing (20%) sets.

7.**Build the LSTM model**:
Create a sequential model with an LSTM layer (50 units, ReLU activation) followed by a dense output layer with one neuron. Compile the model using the Adam optimizer and mean squared error loss.

8.**Train the model**:
Fit the model on the training data for 20 epochs.

9.**Make predictions**:
Use the trained model to predict temperature values on the test set.

10.**Inverse scale the predictions**:
Convert the scaled predictions and actual values back to the original scale using the inverse of MinMaxScaler.

11.**Visualize the results**:
Plot the actual vs. predicted price values to evaluate model performance visually.

Code:

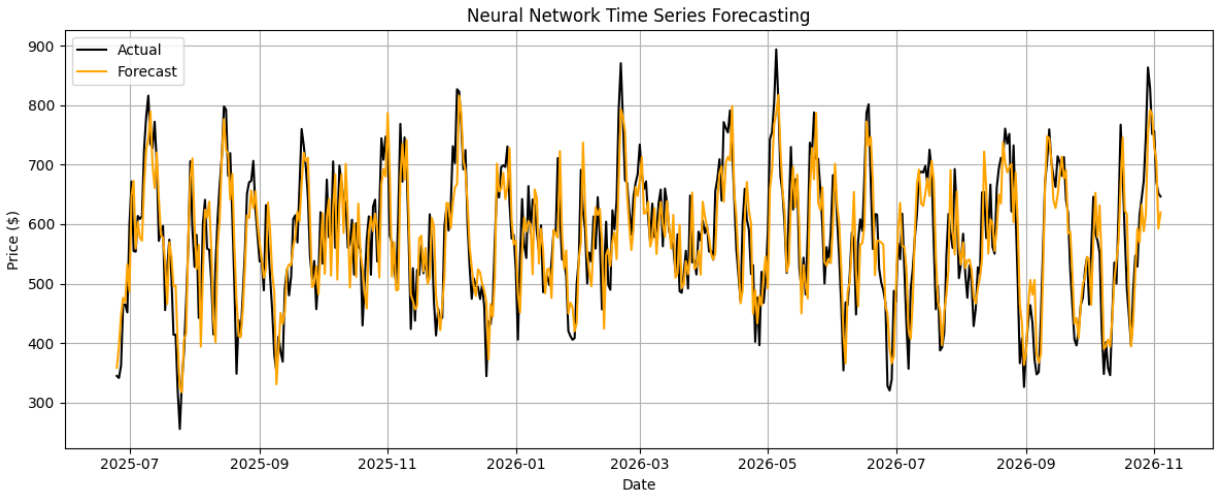
```
import numpy as np

import matplotlib.pyplot as plt
```

```
from sklearn.preprocessing import MinMaxScaler
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense
df = pd.read_csv("artmarket_with_dates.csv")
df['Date'] = pd.to_datetime(df['Date'])
df = df.sort_values('Date')
df.set_index('Date', inplace=True)
ts = df['Price ($)'].resample('D').mean().fillna(method='ffill')
ts_smooth = ts.rolling(window=5).mean().dropna()
scaler = MinMaxScaler()
ts_scaled = scaler.fit_transform(ts_smooth.values.reshape(-1, 1))
def create_dataset(series, window_size=10):
    X, y = [], []
    for i in range(len(series) - window_size):
        X.append(series[i:i + window_size])
        y.append(series[i + window_size])
    return np.array(X), np.array(y)
window_size = 10
X, y = create_dataset(ts_scaled, window_size)
split_index = int(len(X) * 0.8)
X_train, y_train = X[:split_index], y[:split_index]
X_test, y_test = X[split_index:], y[split_index:]
model = Sequential()
model.add(Dense(64, input_shape=(window_size,), activation='relu'))
model.add(Dense(32, activation='relu'))
model.add(Dense(1))
model.compile(optimizer='adam', loss='mse')
model.fit(X_train, y_train, epochs=50, batch_size=16, verbose=0)
y_pred = model.predict(X_test)
y_pred_inv = scaler.inverse_transform(y_pred)
y_test_inv = scaler.inverse_transform(y_test.reshape(-1, 1))
test_dates = ts_smooth.index[window_size + split_index:]
plt.figure(figsize=(12, 5))
plt.plot(test_dates, y_test_inv, label='Actual', color='black')
```

```
plt.plot(test_dates, y_pred_inv, label='Forecast', color='orange')
plt.title('Neural Network Time Series Forecasting')
plt.xlabel('Date')
plt.ylabel('Price ($)')
plt.legend()
plt.grid(True)
plt.tight_layout()
plt.show()
```

Output:



Result:

Thus, the program to develop neural network-based time series forecasting model was done.