EX:No.10 221501046

Develop vector auto regression model for multivariate time series data forecasting.

Aim:

Write a program to develop vector auto regression model for multivariate time series data forecasting.

Algorithm:

1.**Import necessary libraries**:
Import numpy, pandas, matplotlib.pyplot, VAR from statsmodels.tsa.api, and mean_squared_error from sklearn.metrics.

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

3.**Select multiple relevant columns**:
Choose important features for multivariate analysis like 'price'.

4.**Handle missing values**:
Use forward fill method (ffill) to fill in any missing values in the dataset.

5.**Split the data into training and testing sets**:
Reserve the last n observations for testing (e.g., n_obs = 10), and use the rest for training the model.

6.**Fit the VAR model**:
Initialize the VAR model with the training dataset and fit it to learn interdependencies among the variables.

7.**Forecast future values**:
Forecast the next n_obs steps using the trained VAR model.
Convert the forecasted values into a DataFrame with the same structure as the original data.

8.**Visualize actual vs. forecasted values**:
For each selected variable (e.g. price), plot the actual and forecasted values on the same graph.

9.**Evaluate model performance**:
Calculate the Root Mean Squared Error (RMSE) between actual and forecasted values for each variable.

10.**Display the RMSE values**.

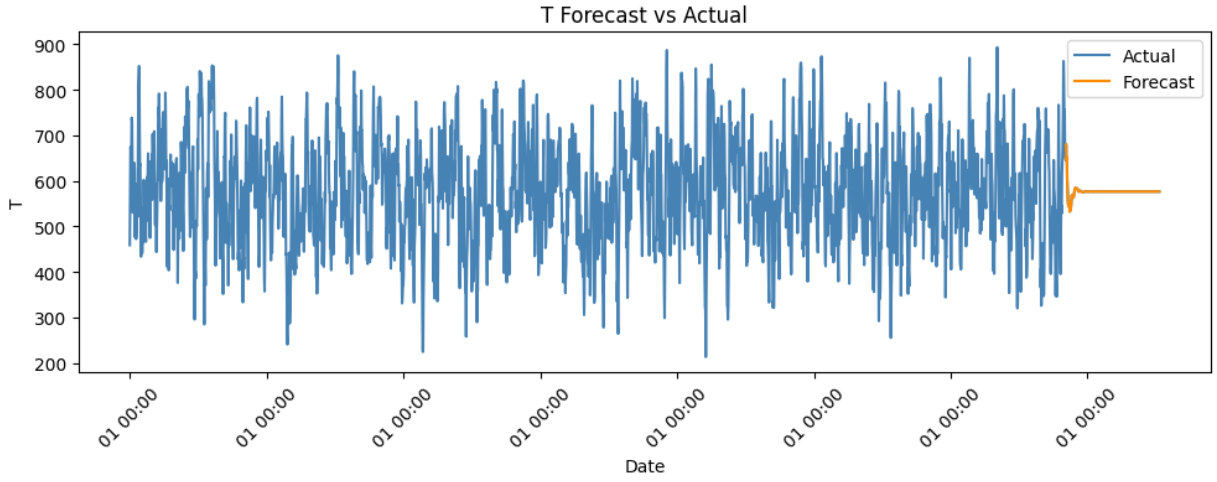
Code:

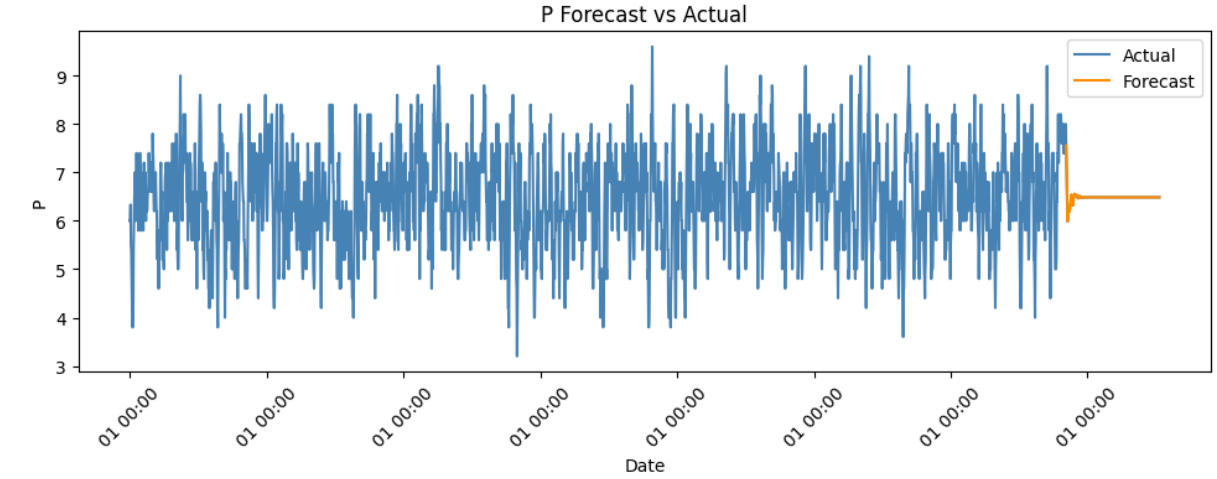
```
import pandas as pd

import matplotlib.pyplot as plt

from statsmodels.tsa.api import VAR

import matplotlib.dates as mdates


# Load data

df = pd.read_csv("artmarket_with_dates.csv")
```

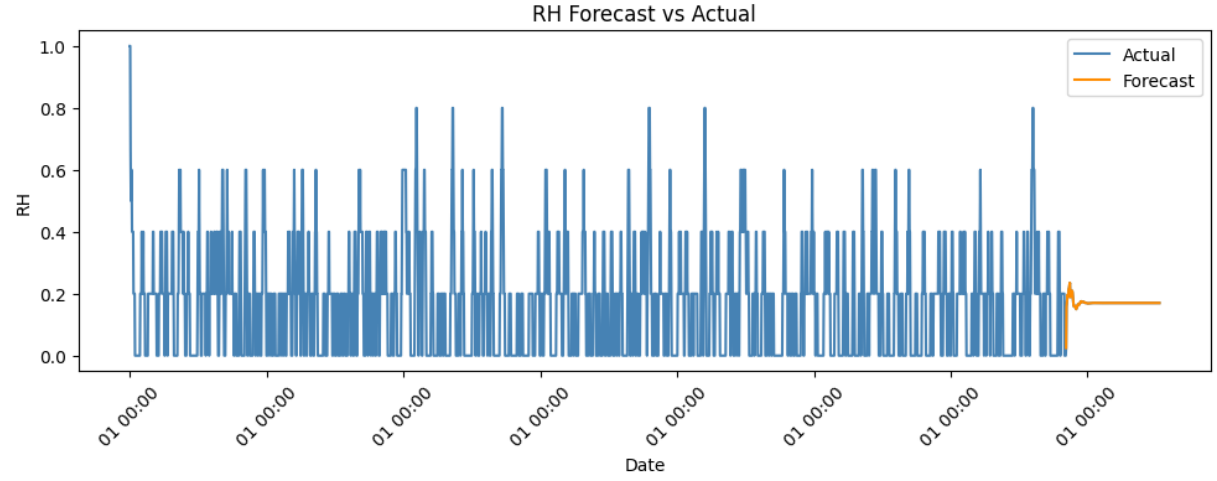
```
df["Date"] = pd.to_datetime(df["Date"])
df.sort_values("Date", inplace=True)

# Feature engineering
df["Style"] = df["Style"].astype(str)
df["Abstract Expressionism"] = df["Style"].apply(lambda x: 1 if x == "Abstract Expressionism" else 0)

# Group by day
df_daily = df.groupby("Date").agg({
    "Price ($)": "mean",
    "Delivery (days)": "mean",
    "Abstract Expressionism": "sum"
}).rename(columns={"Price ($)": "T", "Delivery (days)": "P", "Abstract Expressionism": "RH"})

# Smooth
df_smoothed = df_daily.rolling(window=5, min_periods=1).mean().dropna()

# Train VAR
model = VAR(df_smoothed)
results = model.fit(maxlags=15, ic='aic')

# Forecast
n_forecast = int(len(df_smoothed) * 0.1)
forecast_input = df_smoothed.values[-results.k_ar:]
forecast = results.forecast(forecast_input, steps=n_forecast)

forecast_index = pd.date_range(start=df_smoothed.index[-1] + pd.Timedelta(days=1),
periods=n_forecast, freq='D')
forecast_df = pd.DataFrame(forecast, index=forecast_index, columns=["T", "P", "RH"])
actual_vs_forecast = pd.concat([df_smoothed[["T", "P", "RH"]], forecast_df])

# Plot
fig, axes = plt.subplots(3, 1, figsize=(10, 12))
variables = ["T", "P", "RH"]
```

```
colors = {"Actual": "steelblue", "Forecast": "darkorange"}

for ax, var in zip(axes, variables):
    ax.plot(actual_vs_forecast.index, actual_vs_forecast[var], label="Actual", color=colors["Actual"])
    ax.plot(forecast_df.index, forecast_df[var], label="Forecast", color=colors["Forecast"])
    ax.set_title(f"{var} Forecast vs Actual")
    ax.set_xlabel("Date")
    ax.set_ylabel(var)
    ax.legend()
    ax.xaxis.set_major_formatter(mdates.DateFormatter('%d %H:%M'))
    ax.tick_params(axis='x', rotation=45)

plt.tight_layout()
plt.show()
```

Output:







Result:

Thus, the program to develop vector auto regression model for multivariate time series data forecasting was done.