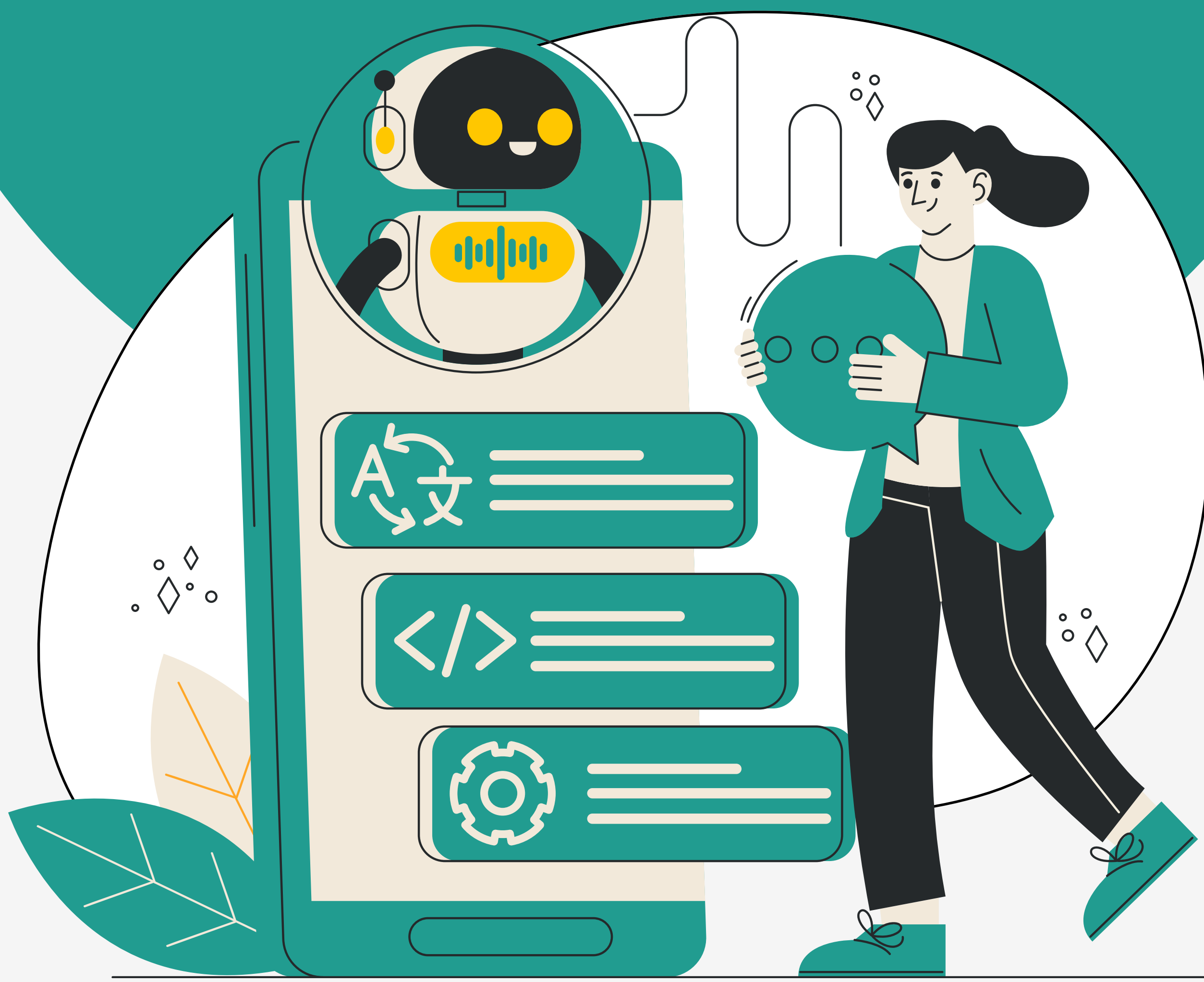
Time Series Interview Questions Scenario based and Practical

(Practice Project)



PW SKILLS

Easy Level

1.Load the uci repository data of individual household electric power consumption and convert the date to datetime format
https://archive.ics.uci.edu/dataset/235/individual+household+electric+power+consumption

Ans: Loading the above data

```
import pandas as pd

# Read the .txt file
df = pd.read_csv(r"E:\household_power_consumption.txt",
delimiter=';',header=0,low_memory=False)  # Use space (' ') or comma (',')
depending on delimiter in your .txt file
```

Converting date column to datetime format

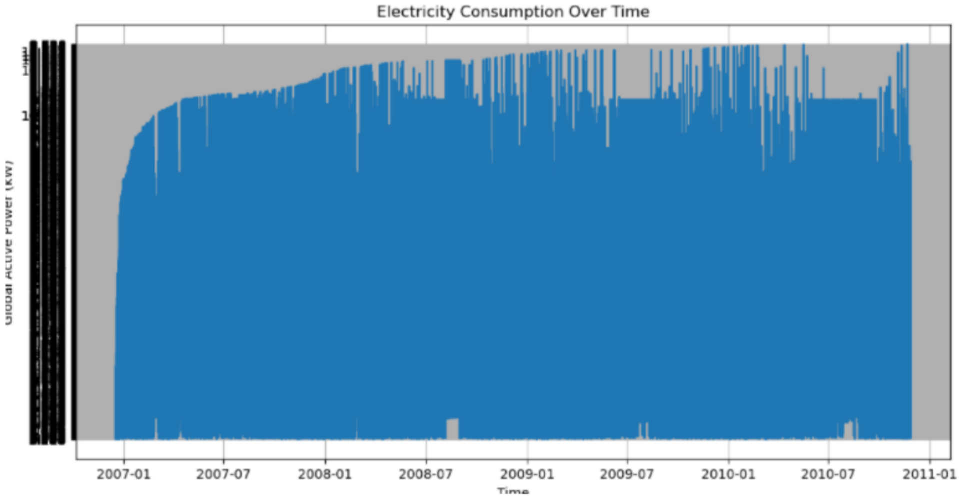
```
# Combine 'Date' and 'Time' columns into a single 'Datetime' column (if
separate)
df['Datetime'] = pd.to_datetime(df['Date'] + ' ' + df['Time'],
format='%d/%m/%Y %H:%M:%S')
```

2. How would you visualize the electricity consumption over time?
Ans: We can visualize electricity consumption over time using line plots. For example, using Matplotlib or Seaborn in Python:

```
import matplotlib.pyplot as plt
import pandas as pd

# Assuming df is already loaded and 'Datetime' column is created as shown
earlier
# Set the 'Datetime' column as the index
df.set_index('Datetime', inplace=True)

# Plot
plt.figure(figsize=(12, 6))
plt.plot(df.index, df['Global_active_power'])  # Use the relevant column
from your dataset
plt.title('Electricity Consumption Over Time')
plt.xlabel('Time')
plt.ylabel('Global Active Power (kW)')  # Adjust this label to the correct
unit
plt.grid(True)  # Optional: to make the plot clearer
plt.show()
```



3. What is the purpose of resampling in time series analysis, and how would you use it with the UCL electricity consumption dataset?

Ans: In order to aggregate or interpolate data at various temporal frequencies, resampling is utilized. To examine daily patterns, we may, for example, resample hourly data to daily data.

```
# Resample to daily frequency
daily_data = df.resample('D').sum()
```

4. How would you visually examine the electrical consumption data to look for patterns or seasonality?

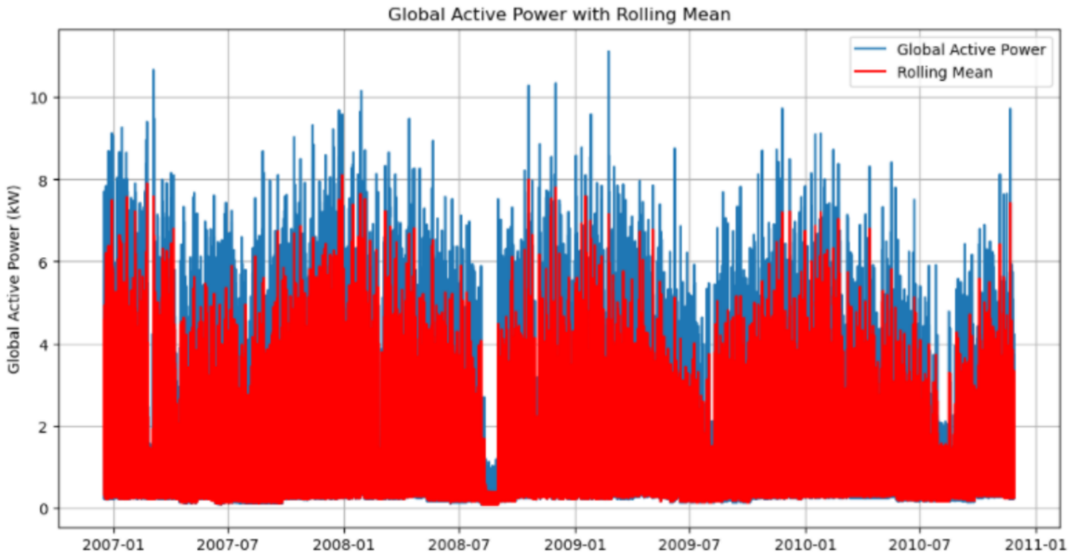
Ans: To check for trends or seasonality, you can plot the data and use rolling averages. For example:

```
import numpy as np
import matplotlib.pyplot as plt

# Convert the 'Global_active_power' column to numeric, setting
errors='coerce' to turn invalid values (like '?') into NaN
df['Global_active_power'] = pd.to_numeric(df['Global_active_power'],
errors='coerce')

# Calculate the rolling average (e.g., a 24-hour window)
df['rolling_mean'] =
df['Global_active_power'].rolling(window=24).mean()

# Plot the original data and the rolling mean
plt.figure(figsize=(12, 6))
plt.plot(df.index, df['Global_active_power'], label='Global Active
Power')
plt.plot(df.index, df['rolling_mean'], label='Rolling Mean',
color='red')
plt.title('Global Active Power with Rolling Mean')
plt.xlabel('Time')
plt.ylabel('Global Active Power (kW)')
plt.legend()
plt.grid(True)  # Optional: adds a grid to the plot for clarity
plt.show()
```



5.Before analysis, how would you prepare and clean the dataset?

Ans: The following actions are taken to clean and preprocess the data:

Handling Missing Values: Removing or imputing missing data points is one way to handle missing values.

Data type conversion: Convert time and date columns to the proper datetime forms.

Resampling: If necessary, aggregate the data into suitable time intervals (hourly, daily, etc.).

Feature engineering: Construct fresh features such as day-of-week indicators or rolling averages.

6. Which library you use to import Arima give code

Ans: For importing arima, sarima we use statsmodels library

```
from statsmodels.tsa.statespace.sarimax import ARIMA
```

Medium Questions:

1. How can you identify and handle missing values in the UCL electricity consumption dataset?

Answer: Missing values in time series data can be identified using the isna() or isnull() method in Pandas. To handle them, we can use interpolation or forward/backward filling. For example:

```
# Identify missing values
missing_values = df.isna().sum()

# Forward fill missing values
data_filled = df.fillna(method='ffill')

# Or interpolate missing values
data_interpolated = df.interpolate()
```

2. Write code to decompose the time series data into its components (trend, seasonality, and residuals).

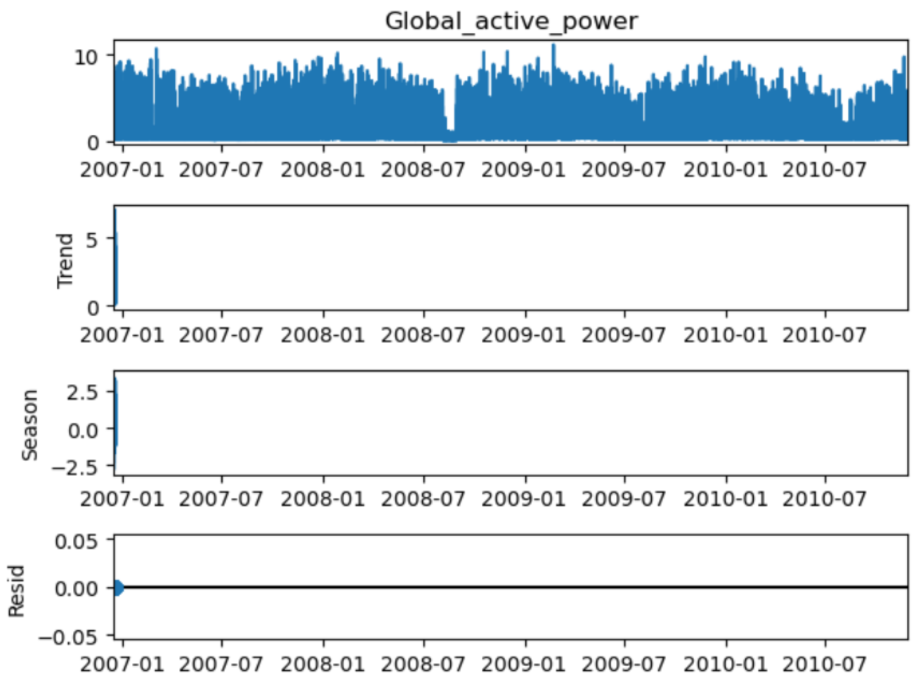
Answer: Time series decomposition can be done using methods like STL (Seasonal and Trend decomposition using Loess) in Python. For example:

```
from statsmodels.tsa.seasonal import STL
import matplotlib.pyplot as plt

# Ensure 'Global_active_power' is numeric
df['Global_active_power'] = pd.to_numeric(df['Global_active_power'],
errors='coerce')

# Decompose the time series using STL
stl = STL(df['Global_active_power'], period=24, robust=True)  # Assuming
daily seasonality (24-hour cycle)
result = stl.fit()

# Plot the STL components
result.plot()
plt.show()
```



3. How would you handle seasonality in the time series data when building a forecasting model?

Answer: To handle seasonality, we can use models that account for it, such as SARIMA (Seasonal ARIMA) or we can remove seasonality and use ARIMA. For example:

```
from statsmodels.tsa.statespace.sarimax import SARIMAX

# Fit SARIMA model
model = SARIMAX(df['consumption'], order=(1, 1, 1), seasonal_order=(1, 1,
1, 24))
model_fit = model.fit(disp=0)

# Forecast
forecast = model_fit.forecast(steps=10)
print(forecast)
```

4. How would you forecast future electricity consumption using a time series model?

Answer: We can use ARIMA (AutoRegressive Integrated Moving Average) for forecasting. Here's a basic example using statsmodels:

```
from statsmodels.tsa.arima_model import ARIMA

# Fit ARIMA model
model = ARIMA(df['consumption'], order=(5, 1, 0))  # Example order
model_fit = model.fit(disp=0)

# Forecast
forecast = model_fit.forecast(steps=10)  # Forecast next 10 periods
print(forecast)
```

5. Give code to evaluate performance of the model

Ans: We can evaluate the performance using metrics like Mean Absolute Error (MAE), Mean Squared Error (MSE), or Root Mean Squared Error (RMSE). For example:

```
from sklearn.metrics import mean_squared_error

# True values
true_values = df['consumption'][-10:]  # Actual values for the last 10
periods

# Forecasted values
forecasted_values = model_fit.forecast(steps=10)[0]

# Calculate RMSE
rmse = mean_squared_error(true_values, forecasted_values, squared=False)
print(f'RMSE: {rmse}')
```

Hard Questions;

1.What univariate analysis would you use to detect anomalies in the Voltage variable over time?

Answer: Steps for anomaly detection:

Visual Inspection: Plot the Voltage time series and check for abnormally high or low spikes.

Z-score method: For finding outliers-the values that are more than ±3 standard deviations away from the mean-a Z -score is calculated for each value.

Move Averages: Identify periods when Voltage significantly deviates from its normal range by using the rolling mean and standard deviation.

Isolation Forest or other anomaly detection models: Train a model specifically on Voltage data to find anomaly values.

2.How would you assess the impact of a holiday or special event on electricity consumption using univariate analysis?

Answer: Steps for Analysis:

Segmentation of data: Divide the data into three parts; before the event, at the time of the event taking place, and after the event.

Comparison of the distributions: Use visualization, such as histograms and box plots, to compare consumption across these periods. Statistical Testing: A hypothesis testing-a t-test, for instance-can be used to see if event-day consumption is statistically different from regular or typical days. Evaluate the impact by quantifying it using percentage changes in consumption during event days over and above normal days.

3.Consider an electricity company that wants to reduce peak demand. How will you do the analysis and identify the peak consumption times using this dataset? What would you suggest as a strategy for reducing peak demand?

Answer: Steps to analyze peak demand:

Univariate analysis: Plot Global_active_power against time to identify periods of highest consumption, for example evening hours.
Feature extraction: Consumption is profiled against the day of the week, holidays, and time of day to clearly show the recurring peak periods.
Strategies: Propose demand-side management approaches, including time-of-use pricing, incentive programs to utilize energy during off-peak hours, and encouraging the use of energy-efficient appliances.

4.You notice that there are some "NA" values in the Sub_metering_3 variable. How could you handle these missing values, and how may differânT imputation techniques affect the analysis?

Solution: Following is the step towards handling missing values:

Analyze the missingness: Determine if the missing values are random or follow a pattern.
Imputation techniques
Simple imputation: impute missing values with mean or median, which may dampen short-run fluctuations.
Forward/backward fill: Fill in gaps with previous or next observations. This might preserve trends but may lead to bias.
Model-based imputation: Use regression or k-NN imputation for higher quality estimates.
Impact on analysis: Imputation can affect distribution and dispersion, which may, in turn, impact subsequent analyses and models.

5.You are tasked with forecasting electricity consumption for the next month. How would you prepare the data by using a univariate analysis in order to choose an appropriate forecasting model?

Answer: Forecasting steps :

Univariate analysis: Explore the Global_active_power time series properties: separating trends, seasonality, and stationarity.
Data preparation: Use transformations-such as a logarithm-thereafter take first or higher-order differences to make the data stationary.
Model Selection: Based on the univariate analysis, select one of the univariate time series forecasting models: ARIMA, SARIMA, or Exponential Smoothing.
Evaluation: Use RMSE and other metrics in model validation, revising based on its performance.