EDA-Time Series Exploration

Time Series Exploration is an essential step in the Exploratory Data Analysis (EDA) process when dealing with time-stamped data. It involves understanding and visualizing the characteristics of a time series to gain insights into its behavior, patterns, and trends. Here's a summary of Time Series Exploration for EDA, along with examples:

Key Steps:

1. **Visualize the Time Series**: Plot the time series over different periods (e.g., daily, weekly, monthly) using line plots or area charts to understand the overall trend.
2. **Check for Seasonality**: Identify any regular periodic patterns (seasonal fluctuations) that repeat over fixed intervals (e.g., daily, monthly, yearly).
3. **Detect Trend and Stationarity**: Determine if the time series has a clear upward or downward trend, and whether it is stationary (constant variance) or non-stationary.
4. **Identify Outliers and Anomalies**: Look for unusual patterns or values that deviate significantly from the rest of the data.
5. **Examine Autocorrelation**: Analyze how values at one time point relate to values at previous or subsequent time points.

Example:

Suppose we have a dataset containing daily closing stock prices for Apple Inc. over 2010-2022. We'll use Python with libraries like pandas and matplotlib to perform the exploration.

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

# Load data into a Pandas dataframe
df = pd.read_csv('apple_stock_prices.csv')

# Visualize time series (daily closing prices)
plt.figure(figsize=(10,6))
plt.plot(df['date'], df['price'])
plt.title('Apple Stock Prices over Time')
plt.xlabel('Date')
plt.ylabel('Price ($)')
plt.show()

# Check for seasonality (e.g., daily fluctuations)
df_monthly = df.resample('M').mean()
plt.figure(figsize=(10,6))
plt.plot(df_monthly['date'], df_monthly['price'])
plt.title('Monthly Averages of Apple Stock Prices')
plt.xlabel('Date')
plt.ylabel('Price ($)')
plt.show()

# Detect trend and stationarity
from statsmodels.tsa.seasonal import seasonal_decompose
decomposition = seasonal_decompose(df['price'], model='additive')
trend = decomposition.trend
seasonality = decomposition.seasonal

# Plot the trend and seasonality components
plt.figure(figsize=(10,6))
plt.subplot(411)
plt.plot(trend, label='Trend')
plt.legend(loc='best')
plt.subplot(412)
plt.plot(seasonality, label='Seasonality')
plt.legend(loc='best')
plt.show()

# Identify outliers and anomalies (e.g., using IQR method)
Q1 = df['price'].quantile(0.25)
Q3 = df['price'].quantile(0.75)
IQR = Q3 - Q1
outliers = df[(df['price'] < Q1 - 1.5 * IQR) | (df['price'] > Q3 + 1.5 * IQR)]

# Examine autocorrelation (e.g., using ACF plot)
from statsmodels.graphics.tsaplots import plot_acf
```

```
plot_acf(df['price'], lags=20)
plt.show()
```

By following these steps and using the example above, you'll be able to gain insights into the characteristics of your time series data, including trends, seasonality, outliers, and autocorrelation. This will help inform further analysis and modeling efforts.

Curated by Brajesh Kumar