Lab 3

September 29, 2024

0.0.1 Time series Data

Time series data visualization is essential for analyzing trends, patterns, and fluctuations over time. Various techniques can be used to visualize time series data, depending on the nature of the data and the insights you wish to extract.

Techniques for Time Series Data Visualization

1. Line Chart: Ideal for continuous data over time.
2. Area Chart: Emphasizes the magnitude of change over time.
3. Bar Chart: Useful for discrete time intervals.
4. Heatmap: Visualizes data intensity over time.
5. Box Plot: Shows the distribution of data over time.
6. Calendar Heatmap: Visualizes data over days in a calendar format.
7. Lag Plot: Detects autocorrelation in time series data.

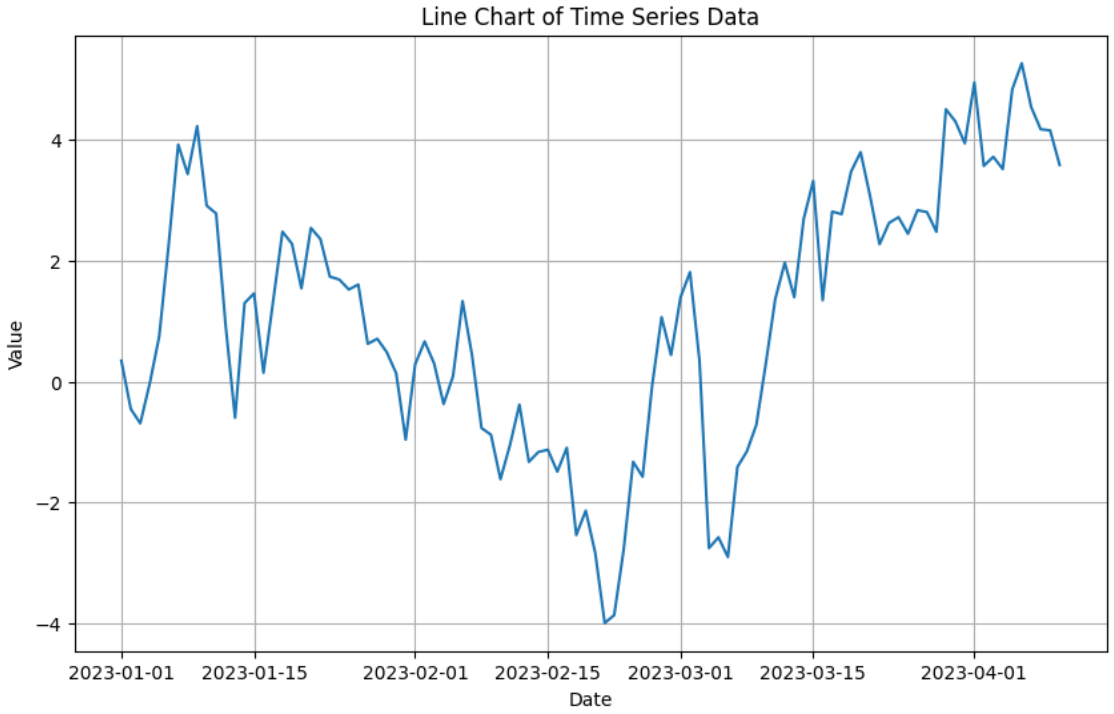
Example Code

1. Line Chart using matplotlib

[44]:
```
import matplotlib.pyplot as plt
import pandas as pd
import numpy as np

# Create a sample time series data
dates = pd.date_range(start="2023-01-01", periods=100, freq="D")
data = np.random.randn(100).cumsum()  # Cumulative sum to simulate a trend
time_series = pd.Series(data, index=dates)

# Plot the line chart
plt.figure(figsize=(10, 6))
plt.plot(time_series)
plt.title("Line Chart of Time Series Data")
plt.xlabel("Date")
plt.ylabel("Value")
plt.grid(True)
plt.show()
```

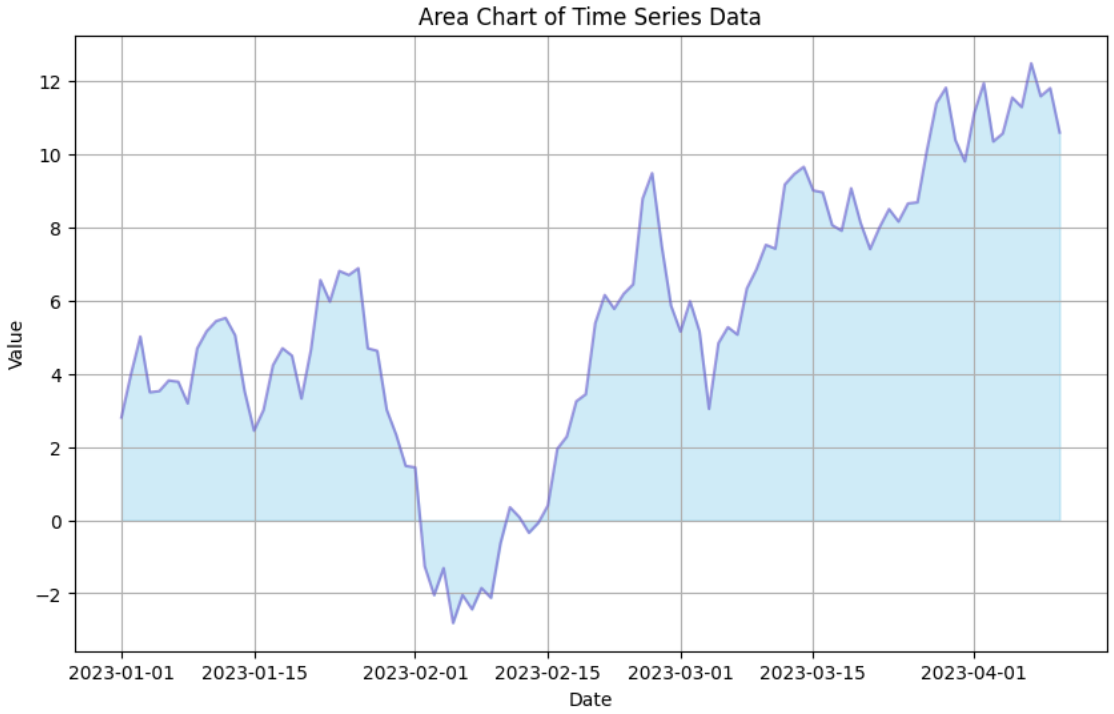


2. Area Chart using matplotlib

[45]:
```
import matplotlib.pyplot as plt
import pandas as pd
import numpy as np

# Create a sample time series data
dates = pd.date_range(start="2023-01-01", periods=100, freq="D")
data = np.random.randn(100).cumsum()
time_series = pd.Series(data, index=dates)

# Plot the area chart
plt.figure(figsize=(10, 6))
plt.fill_between(time_series.index, time_series.values, color="skyblue",␣
 ↪alpha=0.4)
plt.plot(time_series, color="Slateblue", alpha=0.6)
plt.title("Area Chart of Time Series Data")
plt.xlabel("Date")
plt.ylabel("Value")
plt.grid(True)
plt.show()
```

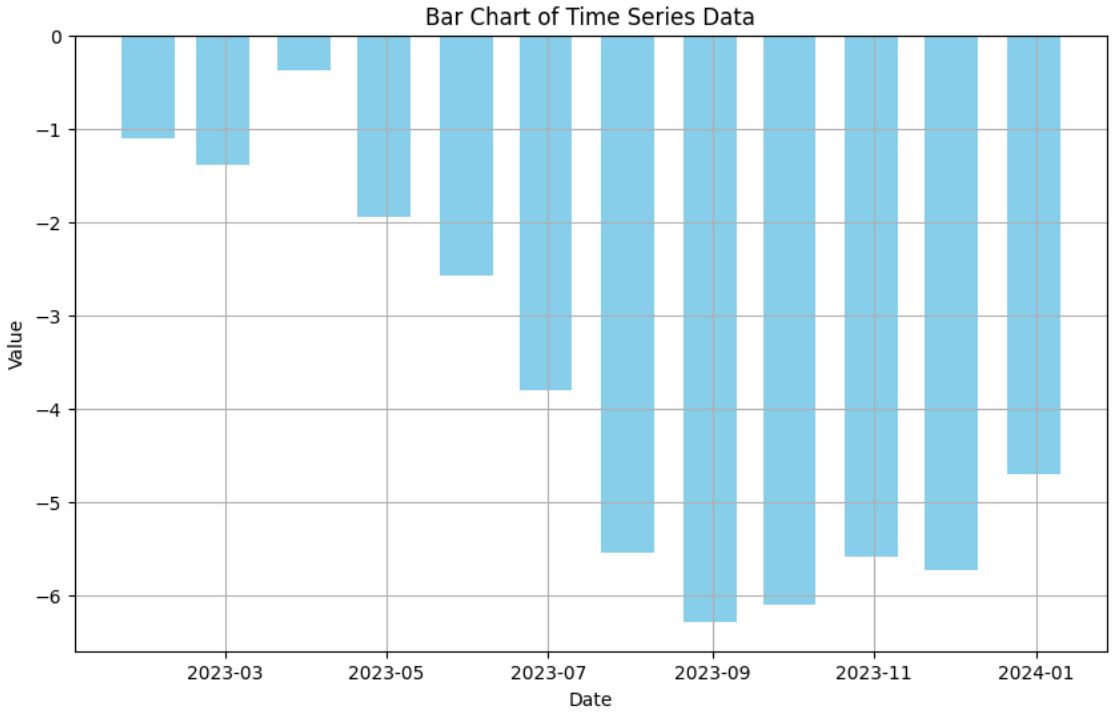


3. Bar chart

[46]:
```
import matplotlib.pyplot as plt
import pandas as pd
import numpy as np

# Create a sample time series data
dates = pd.date_range(start="2023-01-01", periods=12, freq="ME")  # Monthly data
data = np.random.randn(12).cumsum()
time_series = pd.Series(data, index=dates)

# Plot the bar chart
plt.figure(figsize=(10, 6))
plt.bar(time_series.index, time_series.values, width=20, color="skyblue")
plt.title("Bar Chart of Time Series Data")
plt.xlabel("Date")
plt.ylabel("Value")
plt.grid(True)
plt.show()
```



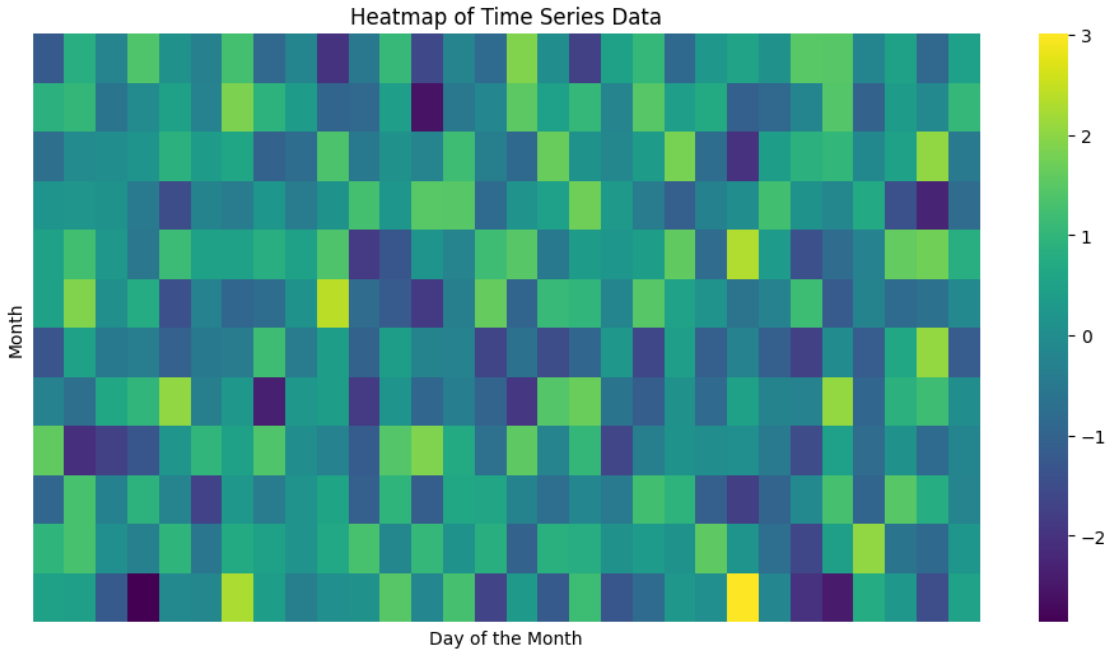
4. Heatmap using seaborn

[47]:
```
import seaborn as sns
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt

# Create a sample time series data
dates = pd.date_range(start="2023-01-01", periods=360, freq="D")
data = np.random.randn(360)
time_series = pd.Series(data, index=dates)

# Reshape the data for a heatmap (assuming daily data for one year)
data_matrix = time_series.values.reshape((12, 30))  # Simplified for␣
 ↪illustration

# Plot the heatmap
plt.figure(figsize=(12, 6))
sns.heatmap(data_matrix, cmap="viridis", xticklabels=False, yticklabels=False)
plt.title("Heatmap of Time Series Data")
plt.xlabel("Day of the Month")
plt.ylabel("Month")
plt.show()
```



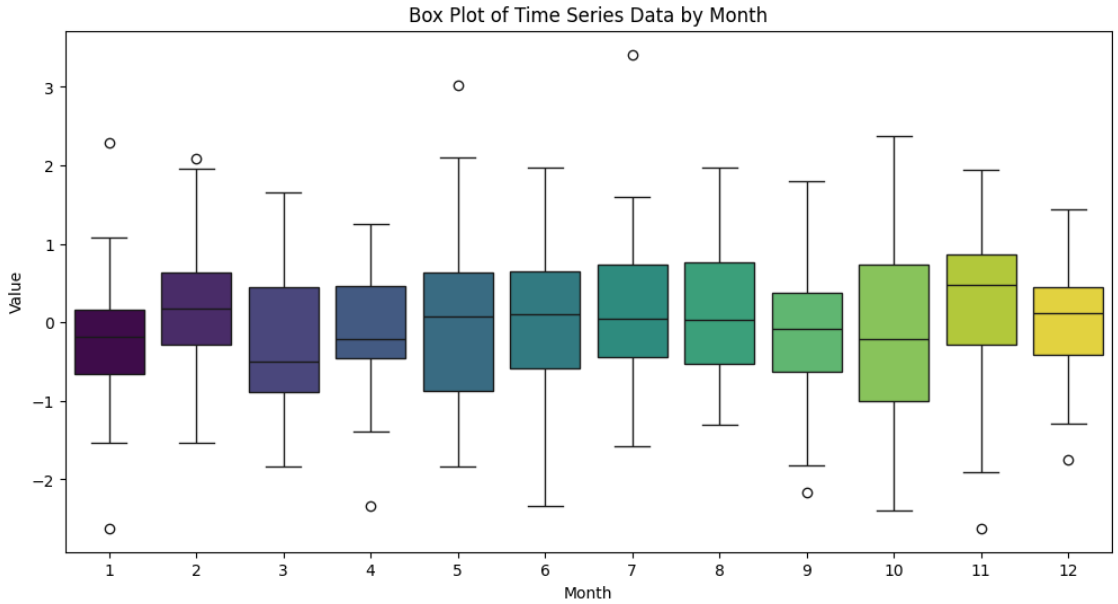
5. Box Plot using seaborn

[48]:
```
import seaborn as sns
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt

# Create a sample time series data
dates = pd.date_range(start="2023-01-01", periods=365, freq="D")
data = np.random.randn(365)
time_series = pd.Series(data, index=dates)

# Create a DataFrame for seaborn
df = pd.DataFrame({"Date": dates, "Value": data})
df["Month"] = df["Date"].dt.month

# Plot the box plot
plt.figure(figsize=(12, 6))
sns.boxplot(x="Month", y="Value", hue="Month", data=df, palette="viridis",␣
 ↪legend=False)
plt.title("Box Plot of Time Series Data by Month")
plt.xlabel("Month")
plt.ylabel("Value")
plt.show()
```



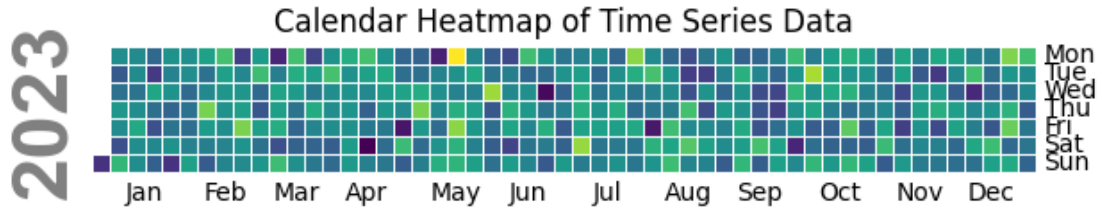
6. Calendar Heatmap using calmap

[49]:
```
import calmap
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt

# Create a sample time series data
dates = pd.date_range(start="2023-01-01", periods=365, freq="D")
data = np.random.randn(365)
time_series = pd.Series(data, index=dates)

# Plot the calendar heatmap
plt.figure(figsize=(12, 6))
calmap.calendarplot(time_series, cmap="viridis", fillcolor="grey", linewidth=0.
↪5)
plt.title("Calendar Heatmap of Time Series Data")
plt.show()
```

```
<Figure size 1200x600 with 0 Axes>
```

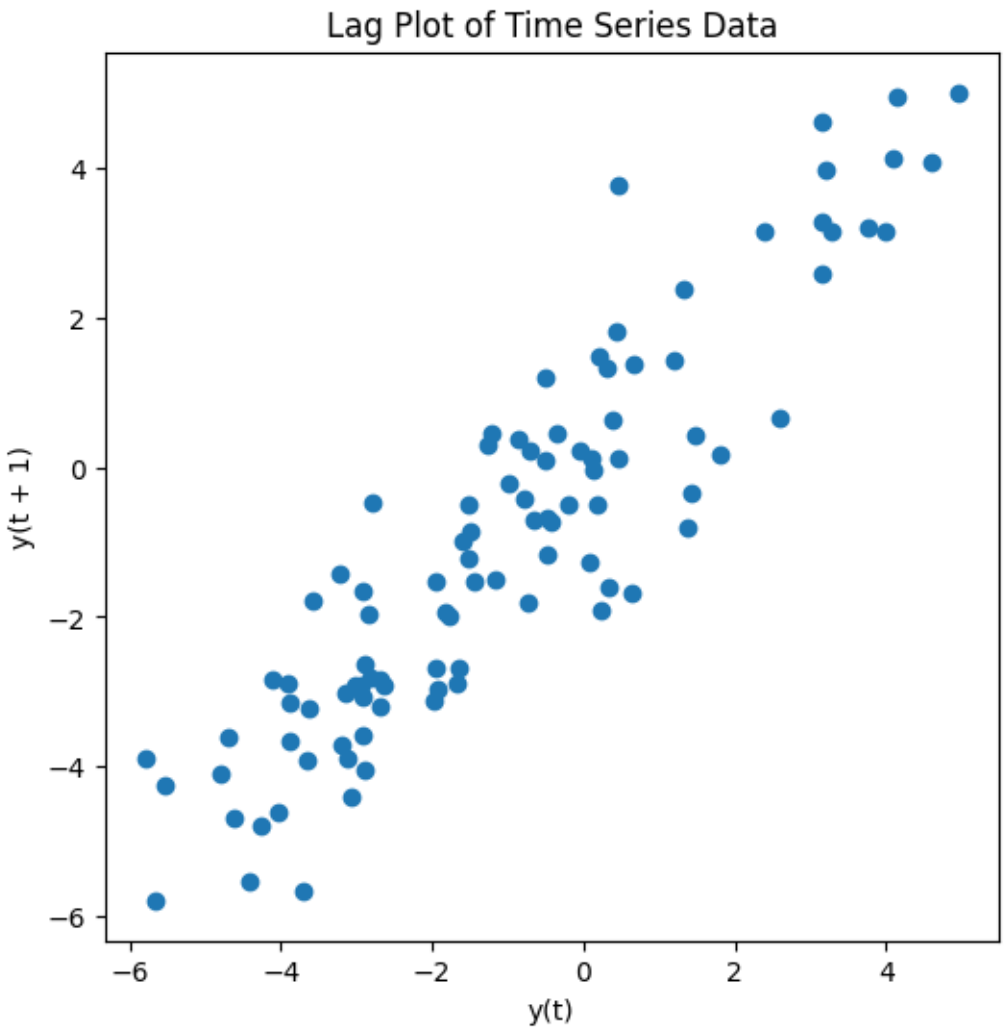


7. Lag Plot using pandas.plotting

[50]:
```
import pandas as pd
import numpy as np
from pandas.plotting import lag_plot

# Create a sample time series data
dates = pd.date_range(start="2023-01-01", periods=100, freq="D")
data = np.random.randn(100).cumsum()
time_series = pd.Series(data, index=dates)

# Plot the lag plot
plt.figure(figsize=(6, 6))
lag_plot(time_series)
plt.title("Lag Plot of Time Series Data")
plt.show()
```



Subword-Level Representations Subword units like Byte Pair Encoding (BPE) or Unigram Language Model are used to represent text at the subword level. This helps in handling rare words and capturing meaningful subword units.

[51]:
```
import sentencepiece as spm

# Train a SentencePiece model (this is just an example, you need a text corpus␣
 ↪for actual training)
spm.SentencePieceTrainer.train(
    input=r"C:
 ↪\Users\kshitiz\AppData\Roaming\nltk_data\corpora\stopwords\english",
    model_prefix="spm_model",
    vocab_size=140,
)
```

```
# Load the trained model
sp = spm.SentencePieceProcessor(model_file="spm_model.model")

# Encode text to subword units
text = "hello world"
subword_level = sp.encode_as_pieces(text)
print(subword_level)  # Output example: ['he', 'llo', 'world']
```

```
['he', 'l', 'l', 'o', ' ', 'w', 'or', 'l', 'd']
```