Lektion 5: Visualisering af (støjfuld) data

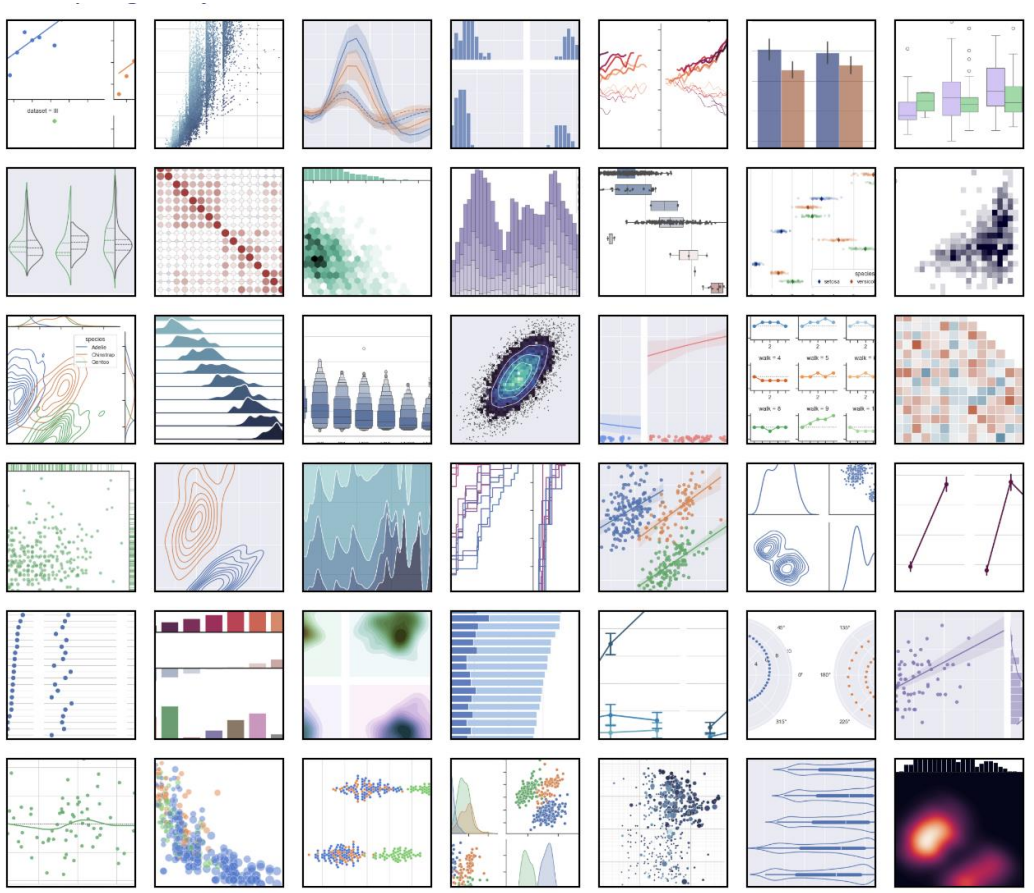
Assoc. Prof. Jesper Rindom Jensen

Agenda

- Vigtighed af visualisering
- Grundlæggende visualisering
  - Matplotlib https://matplotlib.org/
  - Seaborn https://seaborn.pydata.org/
- Visualisering af støjfuld data

Hvad er datavisualisering?

- Grafisk repræsentation af data.
- Afsløring af mønstre, trends, og indsigter.
- Essentiel for eksplorativ dataanalyse.



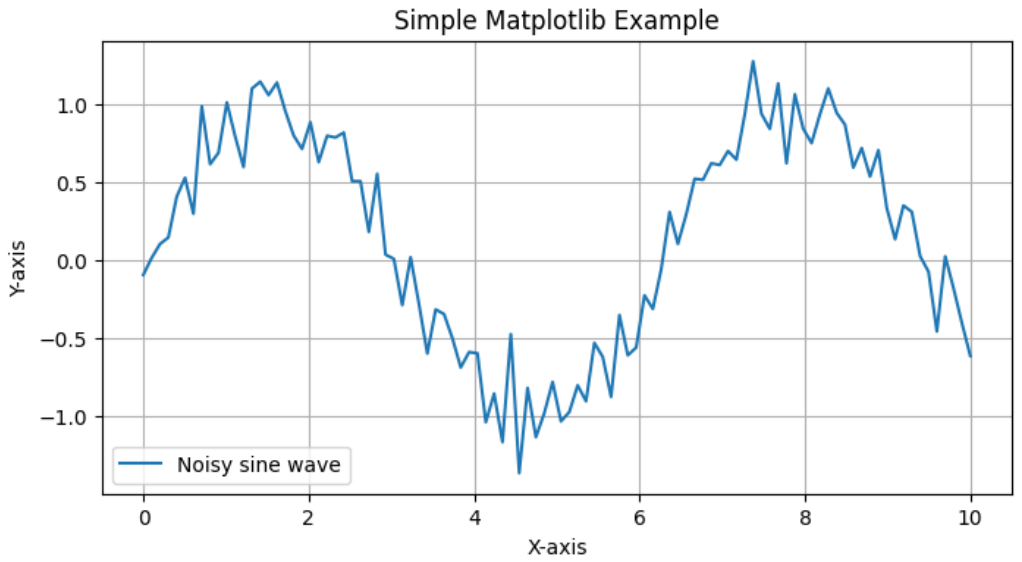
Hvorfor visualisere støjfuld data?

- Hjælper med at forstå datakvalitet:
  - Outliers?
  - Støj?
- Giver forståelse af statistisk fordeling af data.
- Understøtter modelevaluering og fejlfinding ved at visualisere problematiske features.
- Effektiv kommunikation af observationer til stakeholders.

Matplotlib Basics

- Standard plotbibliotekt i Python.
- Høj grad af customisering.
- Bruges ofte med NumPy og Pandas
- https://matplotlib.org/
- Terminologi:
  - **Figure**: Container til plots
  - **Axes**: Individuelle plots i figur
  - **Labels, Legends, Grid**: Forklaring og øget læsbarhed.

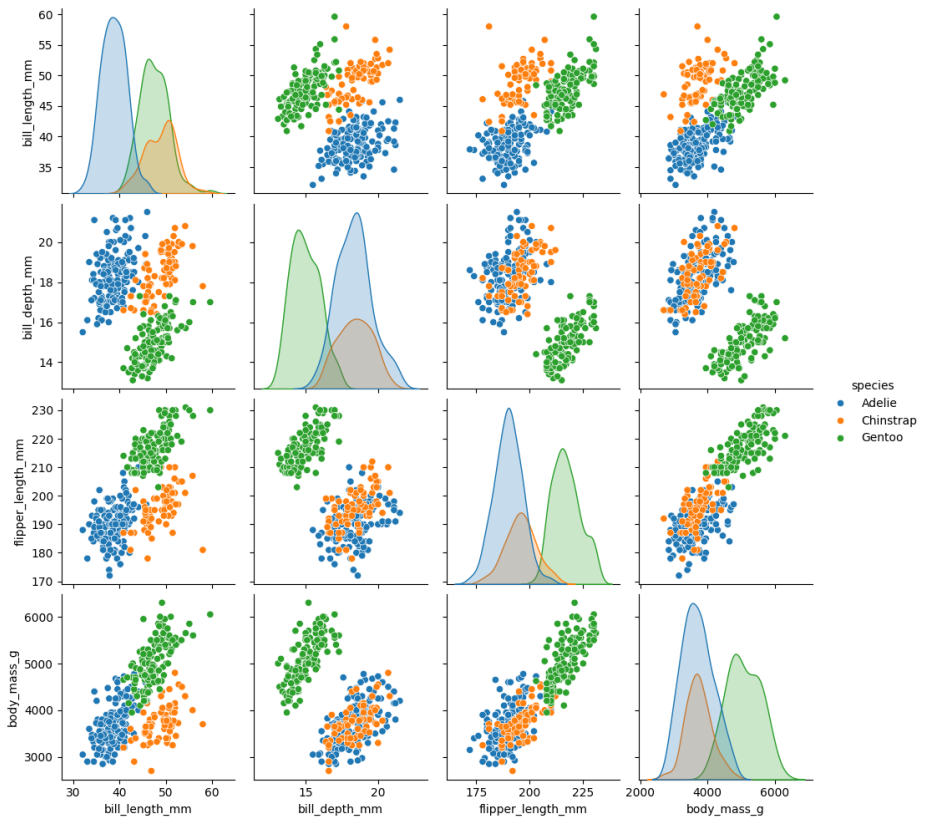
```python
import matplotlib.pyplot as plt
import numpy as np

x = np.linspace(0, 10, 100)
y = np.sin(x) + np.random.randn(100) * 0.2
plt.figure(figsize=(8,4))
plt.plot(x, y, label='Noisy sine wave')
plt.title('Simple Matplotlib Example')
plt.xlabel('X-axis')
plt.ylabel('Y-axis')
plt.legend()
plt.grid(True)
plt.show()
```



Seaborn basics

- Bygget ovenpå matplotlib
- Tilbyder mere komplicerede og polerede statistiske visualiseringer.
- Integreret med Pandas DataFrames.
- **Eksempler**
  - Parplot
  - Distributionsplots
  - Korrelationsplot

```
import seaborn as sns
import matplotlib.pyplot as plt

# Load the built-in penguins dataset
penguins = sns.load_dataset('penguins')

# Create a pairplot colored by penguin species
sns.pairplot(penguins, hue='species')
plt.show()
```

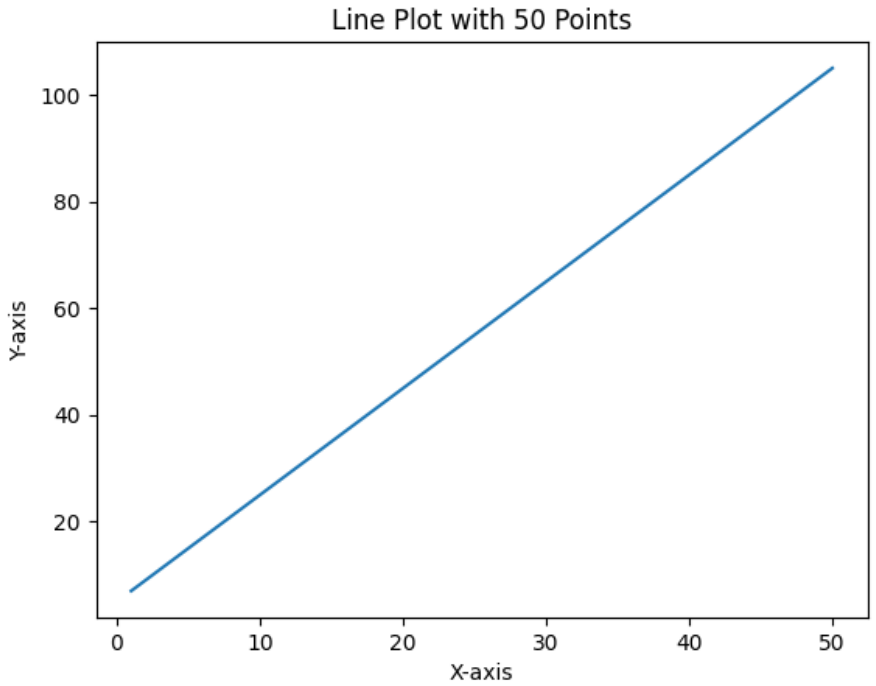


Typer af dataplot

- Linjediagram (line plot)
- Punktdiagram (scatter plot)
- Histogram
- Boksplot (box plot) / Violinplots
- Varmekort (heatmap)
- Plots med usikkerheder (error bars)

Linjediagram

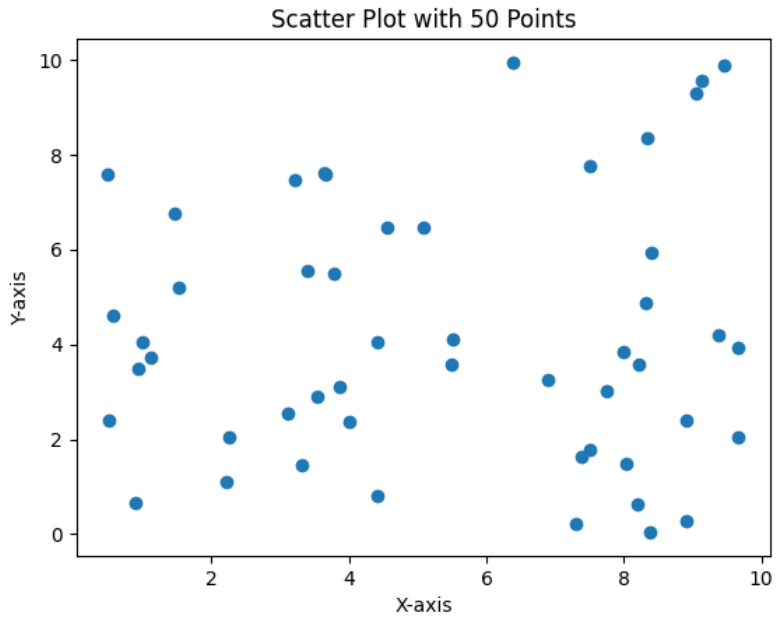
- Anvendelig til tidsseriedata.
- Afslører trends, periodicitet, skift osv.




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

# Generate 50 data points using NumPy
x = np.linspace(1, 50, 50)  # Creates an array of 50 evenly spaced numbers between 1 and 50
y = 2 * x + 5  # Define a relationship between x and y (example: y = 2x + 5)

# Create the plot
plt.plot(x, y)

# Add labels and title
plt.xlabel("X-axis")
plt.ylabel("Y-axis")
plt.title("Line Plot with 50 Points")

# Display the plot
plt.show()
```

Punktdiagram

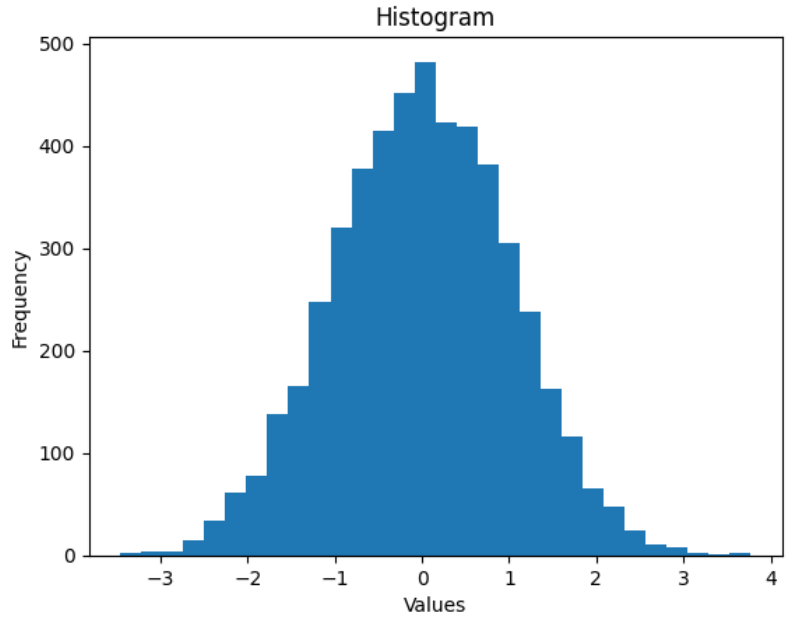
- Viser enkelte datapunkter.
- Visualisering af distributioner/korrelationer




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

# Generate 50 random data points using NumPy
x = np.random.rand(50) * 10  # Creates an array of 50 random numbers between 0 and 10
y = np.random.rand(50) * 10  # Creates another array of 50 random numbers between 0 and 10

# Create the scatter plot
plt.scatter(x, y)

# Add labels and title
plt.xlabel("X-axis")
plt.ylabel("Y-axis")
plt.title("Scatter Plot with 50 Points")

# Display the plot
plt.show()
```

Histogrammer

- Viser frekvensdistribuering af data.
- Forståelse af dataspredning og støjkarakteristikker.



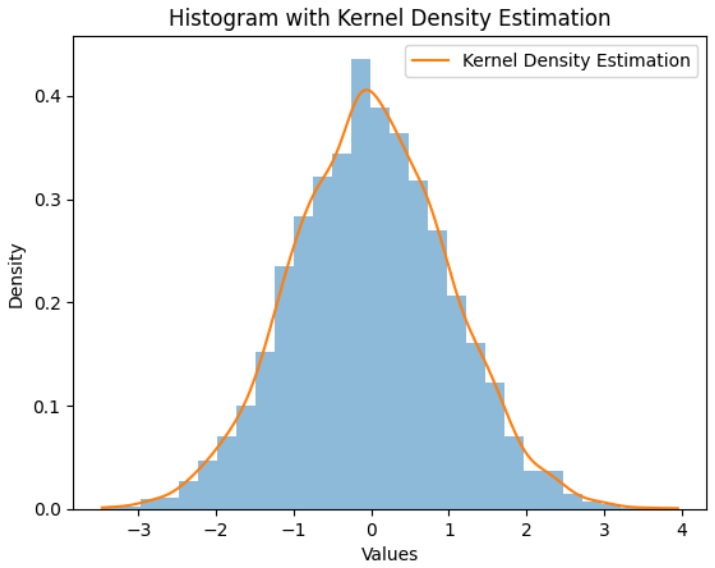

```python
import matplotlib.pyplot as plt
import numpy as np

# Generate random data points using NumPy
data = np.random.randn(5000)  # sample from
a standard normal distribution

# Create the histogram
plt.hist(data, bins=30)  # 'bins' specifies
the number of bins in the histogram

# Add labels and title
plt.xlabel("Values")
plt.ylabel("Frequency")
plt.title("Histogram")

# Display the plot
plt.show()
```

Hvordan vælger vi ”bins”?

Estimering af kernel densitet (KDE)

- Alternativ, kan sandsynlighedstætheden estimeres, fx med Gaussiske kernels:

$$p(x) = \frac{1}{nh\sigma}\frac{1}{\sqrt{2\pi}}\sum_{i=1}^{n}\exp\left(\frac{-(x-x_i)^2}{2h^2\sigma^2}\right)$$




```python
import matplotlib.pyplot as plt
import numpy as np
from scipy.stats import gaussian_kde

# Generate random data points using NumPy
data = np.random.randn(5000)  # sample from a standard normal distribution

# Create the histogram with density normalization
hist, bins, _ = plt.hist(data, bins=30, density=True, alpha=0.5)

# Perform Kernel Density Estimation
density = gaussian_kde(data)
xs = np.linspace(data.min(), data.max(), 200)
density_values = density(xs)

# Plot the estimated kernel density
plt.plot(xs, density_values, label='Kernel Density Estimation')

# Add labels and title
plt.xlabel("Values")
plt.ylabel("Density")
plt.title("Histogram with Kernel Density Estimation")
plt.legend()

# Display the plot
plt.show()
```

https://docs.scipy.org/doc/scipy/reference/generated/scipy.stats.gaussian_kde.html

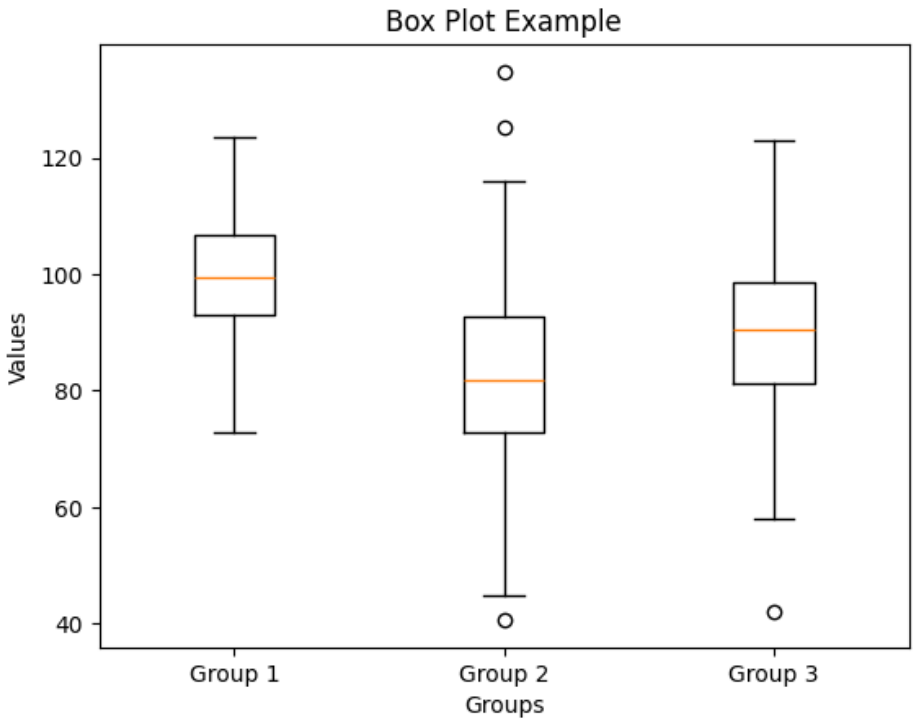
Boksplots

- Viser opsummerende statistikker og outliers.
- Baseret på kvartiler:
  - Q_0: Minimum - laveste datapunkt
  - Q_4: Maksimum – højeste datapunkt
  - Q_2: Median – midterste datapunkt
  - Q_1: Første kvartil – median af nederste halvdel af datasæt (25 % af data under)
  - Q_4: Tredje kvartil – median af øverste halvdel af datasæt (25 % af data over)
- Whiskers: typisk baseret på interkvartil range (IQR),
  - Fx. Q_1-1.5 IQR og Q_3+1.5 IQR. **(OBS: kan variere!)**

$$\mathrm{IQR} = Q_3 - Q_1$$

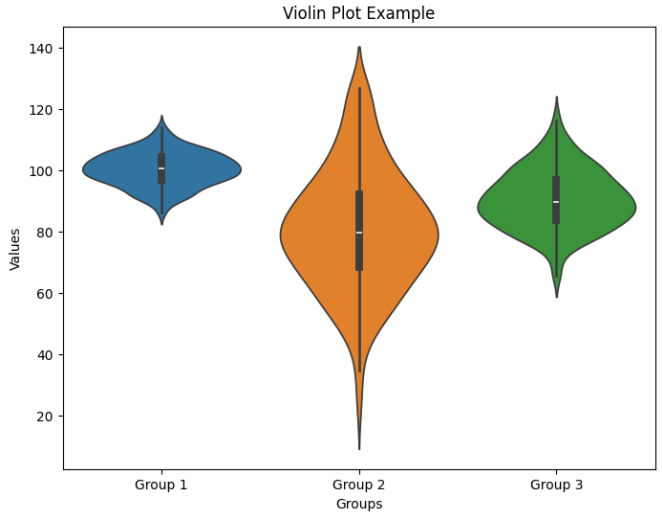
Boksplots

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

# Generate sample data for three groups
group1 = np.random.normal(100, 10, 200)  # Mean 100,
standard deviation 10,
group2 = np.random.normal(80, 15, 200)   # Mean 80,
standard deviation 15,
group3 = np.random.normal(90, 12, 200)   # Mean 90,
standard deviation 12,

# Combine the data into a list
data = [group1, group2, group3]

# Create the box plot
plt.boxplot(data, labels=['Group 1', 'Group 2', 'Group
3'])

# Add labels and title
plt.xlabel("Groups")
plt.ylabel("Values")
plt.title("Box Plot Example")

# Display the plot
plt.show()
```



Violinplots

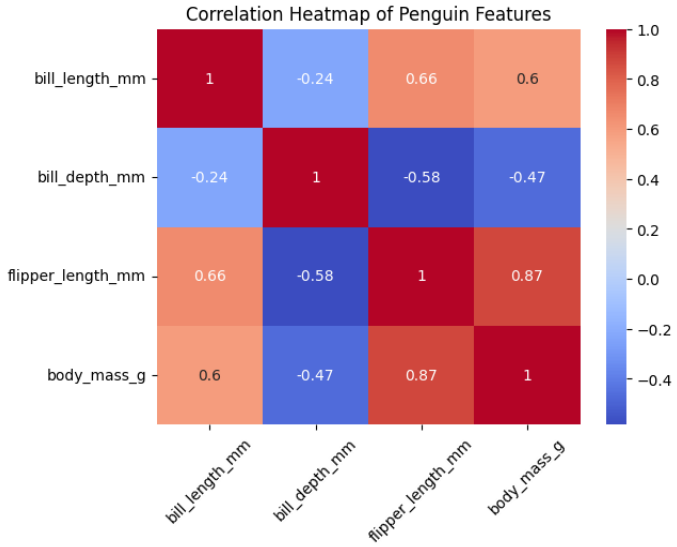
- KDE kan bruges i kombination med boxplots.
- Kaldes også for violinplots.




```
import matplotlib.pyplot as plt
import numpy as np
import seaborn as sns  # Import seaborn for violin plot

# Generate sample data for three groups (same as before)
group1 = np.random.normal(100, 5, 200)
group2 = np.random.normal(80, 20, 200)
group3 = np.random.normal(90, 10, 200)

# Combine the data into a list (same as before)
data = [group1, group2, group3]

# Create the violin plot using seaborn
plt.figure(figsize=(8, 6))  # Adjust figure size if needed
sns.violinplot(data=data)  # data is a list of data for each group

# Customize the plot
plt.xticks(ticks=[0, 1, 2], labels=['Group 1', 'Group 2', 'Group 3'])  # Set x-axis labels
plt.xlabel("Groups")
plt.ylabel("Values")
plt.title("Violin Plot Example")

# Display the plot
plt.show()
```

Varmeplot

- Hurtigt overblik over max/min værdier (fx til evaluering).
- Forståelse for featurekorrelation.

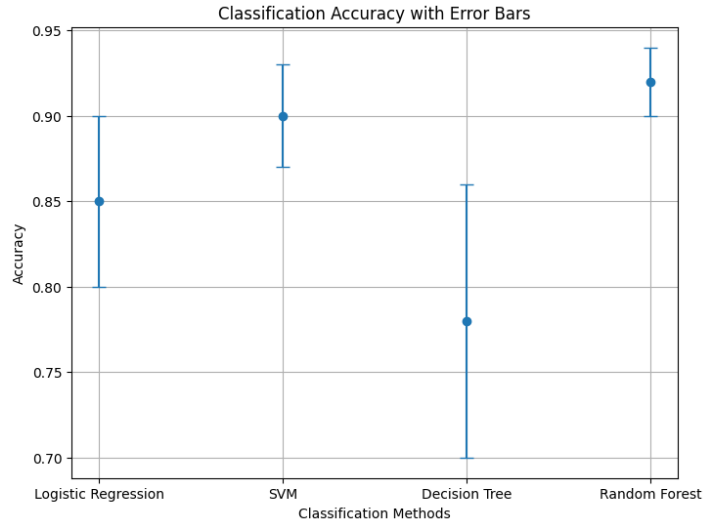


- Er range for colorbar valgt fornuftigt her?

```python
import matplotlib.pyplot as plt
import seaborn as sns
import pandas as pd

# Load the Palmer Penguins dataset
penguins = sns.load_dataset('penguins')

# Select features for correlation analysis
features = ['bill_length_mm',
'bill_depth_mm', 'flipper_length_mm',
'body_mass_g']
selected_data = penguins[features]

# Calculate correlation matrix
correlation_matrix = selected_data.corr()

# Create heatmap using seaborn
sns.heatmap(correlation_matrix, annot=True,
cmap='coolwarm')

# Add labels/title
plt.title("Correlation Heatmap of Penguin
Features")
plt.xticks(rotation=45)

# Display the plot
plt.show()
```

Plot med usikkerheder

- Visning af nøjagtigheder (fx klassifikationsrate) bør altid indeholde usikkerheder.
- Eksempelvis standardafvigelser eller konfidensintervaller.




```python
import matplotlib.pyplot as plt
import numpy as np
import pandas as pd

# Sample data for classification methods
(replace with your actual data)
methods = ['Logistic Regression', 'SVM',
'Decision Tree', 'Random Forest']
accuracies = [0.85, 0.90, 0.78, 0.92]
errors = [0.05, 0.03, 0.08, 0.02]

# Example 1: Error bars
plt.figure(figsize=(8, 6))
plt.errorbar(methods, accuracies, yerr=errors,
fmt='o', capsize=5)
plt.xlabel('Classification Methods')
plt.ylabel('Accuracy')
plt.title('Classification Accuracy with Error
Bars')
plt.grid(True)
plt.show()
```

Konfidensinterval

- Range hvori en målt statistisk parameter (fx middelværdi) forventes at lægge med en hvis konfidens (fx 95 % sikkerhed).
- Ved antagelse om normalfordelt data er konfidensintervallet:

$$\mathrm{CI}_{95\%} = \bar{x} \pm 1.96 \frac{\sigma}{\sqrt{n}}$$

https://amsi.org.au/ESA_Senior_Years/SeniorTopic4/4h/4h_2content_11.html

- Kan udledes for andre fordelinger og konfidenser også.

Examples

- https://colab.research.google.com/drive/1gllvsstDQj7UEVNEpzo_d1raUzGUaibe?usp=sharing

Opgave 1

- Genbesøg og færdiggør filtreringsøvelsen fra lektion 4.
- Lav tydelige visualiseringer, der viser signalerne før og efter filtrering.
- Brug Matplotlib line plots med tydelige farver eller stilarter (Husk akselabels og en forklarende legend)

Opgave 2

- Genbesøg dit/dine datasæt fra Workshop 1. Analyser og visualiser datakvalitet og distribution (pair plots, histogrammer).
- Undersøg for outliers ved hjælp af boks- eller violinplots.
- Træn en classifyer og visualisér klassifikationsrate med usikkerheder.
- Brug Seaborn til pairplots/histogrammer og Matplotlib til error bars.
- Error bars kan vise standardafvigelsen eller 95% konfidensintervaller.