IIT Guwahati

Advanced Certification Programme in Data Science Business Analytics

Week 16
Data Science Using Python Packages (Part 2)

Topics Covered

- Pandas Advanced Use Cases
- Q & A

Overview of Seaborn Visualisation

Techniques for Effective Data Visualisation

What is seaborn?

- Provides a Python library for creating statistical visualisations

Why seaborn?

- Simplifies complex visualisations
- Handles aesthetics and colours automatically
- Works seamlessly with pandas DataFrames

Built on matplotlib

- Provides a high-level interface for drawing attractive and informative graphics

Loading the Titanic Dataset

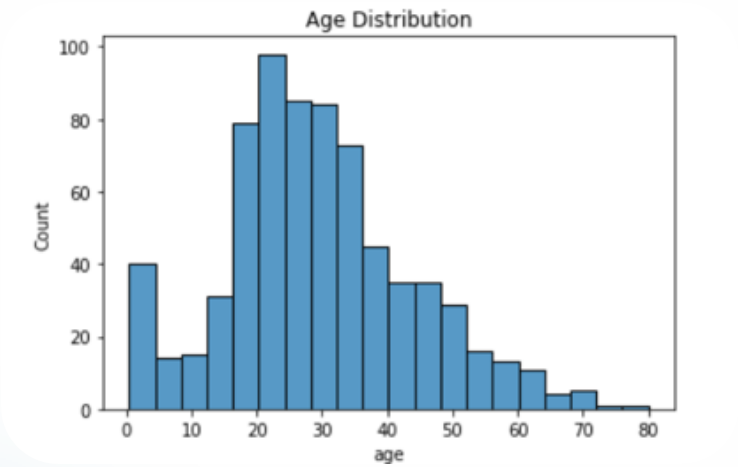
An Introduction to Data Exploration

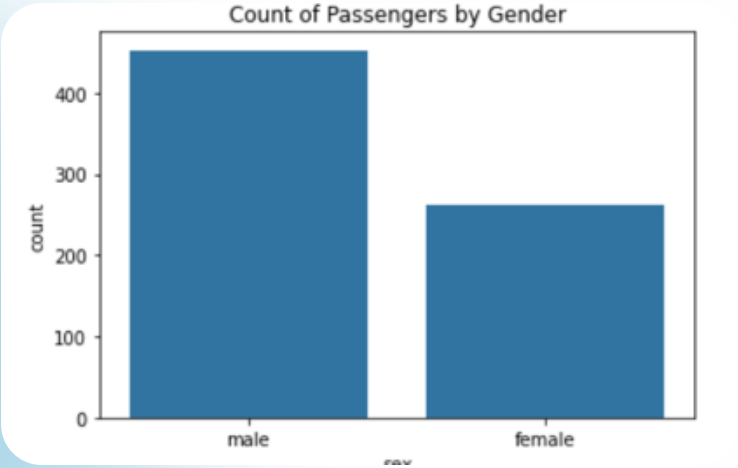
Code example:

```
import seaborn as sns
import pandas as pd

# Load Titanic dataset
titanic = sns.load_dataset("titanic")

# Display first few rows
print(titanic.head())
```

Understanding Univariate Analysis

Exploring Single Variables in Data





- Analyses one variable at a time
- **Histogram**: Shows distribution of numeric data
- **Count plot**: Displays count of categorical data

Understanding Univariate Analysis

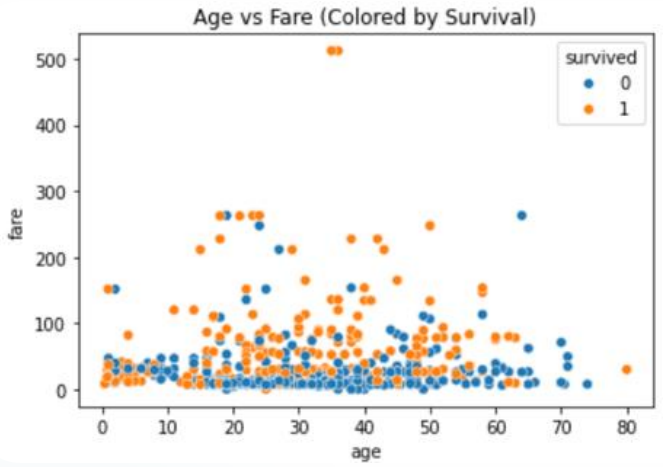
Understanding the Syntax

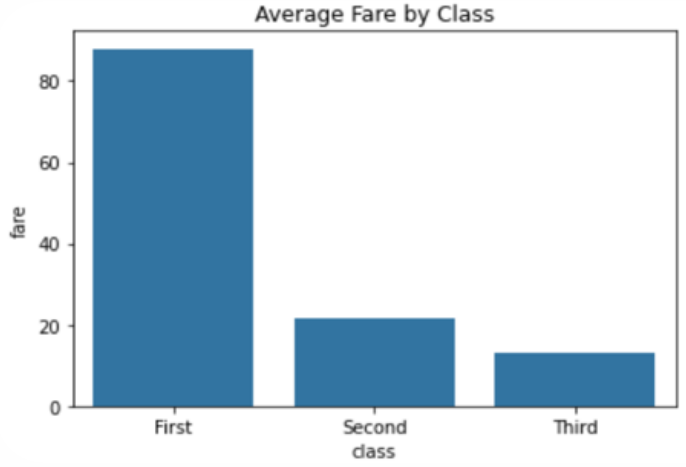
Code example:

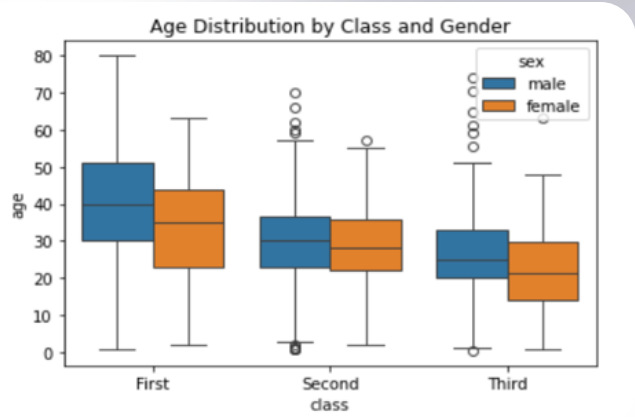
```
import matplotlib.pyplot as plt


# Histogram for age distribution
sns.histplot(data=titanic, x="age", kde=True)
plt.title("Age Distribution")
plt.show()


# Count plot for gender
sns.countplot(data=titanic, x="sex")
plt.title("Count of Pas
plt.show()
```

Understanding Bivariate Analysis

Exploring Relationships Between Two Variables







- Analyses the relationship between two variables
- **Scatter plot:** Examines relationship between two variables
- **Bar plot:** Explores relationship between two numeric variables
- **Box plot:** Shows distribution across categories

Understanding Bivariate Analysis

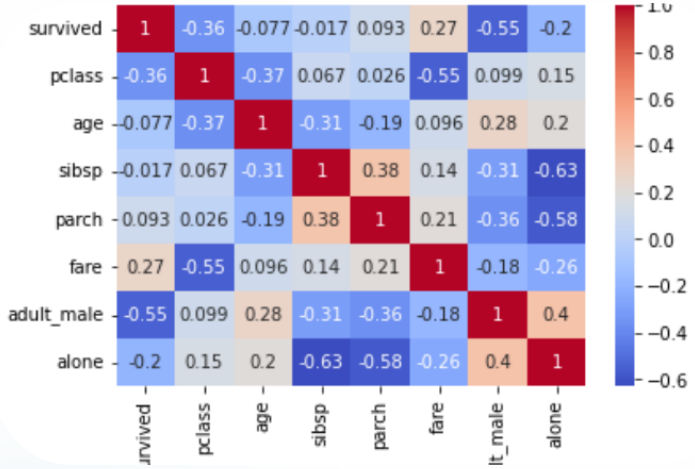
Syntax for Better Understanding

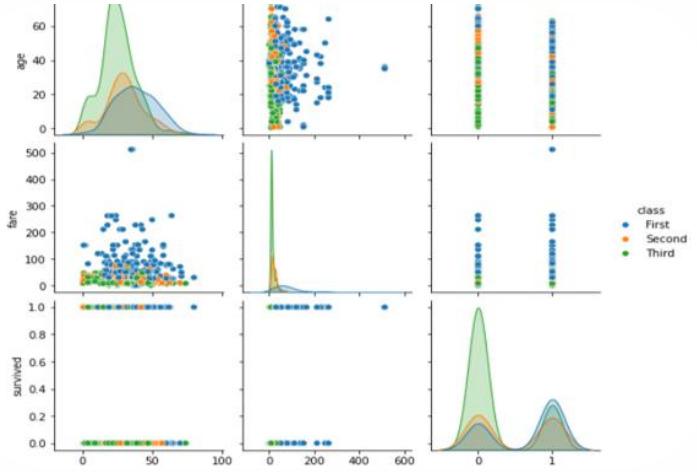
Code example:

```
# Scatter plot: Age vs Fare
sns.scatterplot(data=titanic, x="age", y="fare", hue="survived")
plt.title("Age vs Fare (Colored by Survival)")
plt.show()

# Bar plot: Average fare by class
sns.barplot(data=titanic, x="class", y="fare", ci=None)
plt.title("Average Fare by Class")
plt.show()

# Box plot: Age distribution by class
sns.boxplot(data=titanic, x="class", y="age")
plt.title("Age Distribution by Class")
plt.show()
```

Understanding Multivariate Analysis

Analysing Multiple Variables Simultaneously





- Analyses relationships among more than two variables
- **Heatmap**: Displays correlation matrix
- **Pair plot**: Shows pairwise relationships in a dataset

Understanding Multivariate Analysis

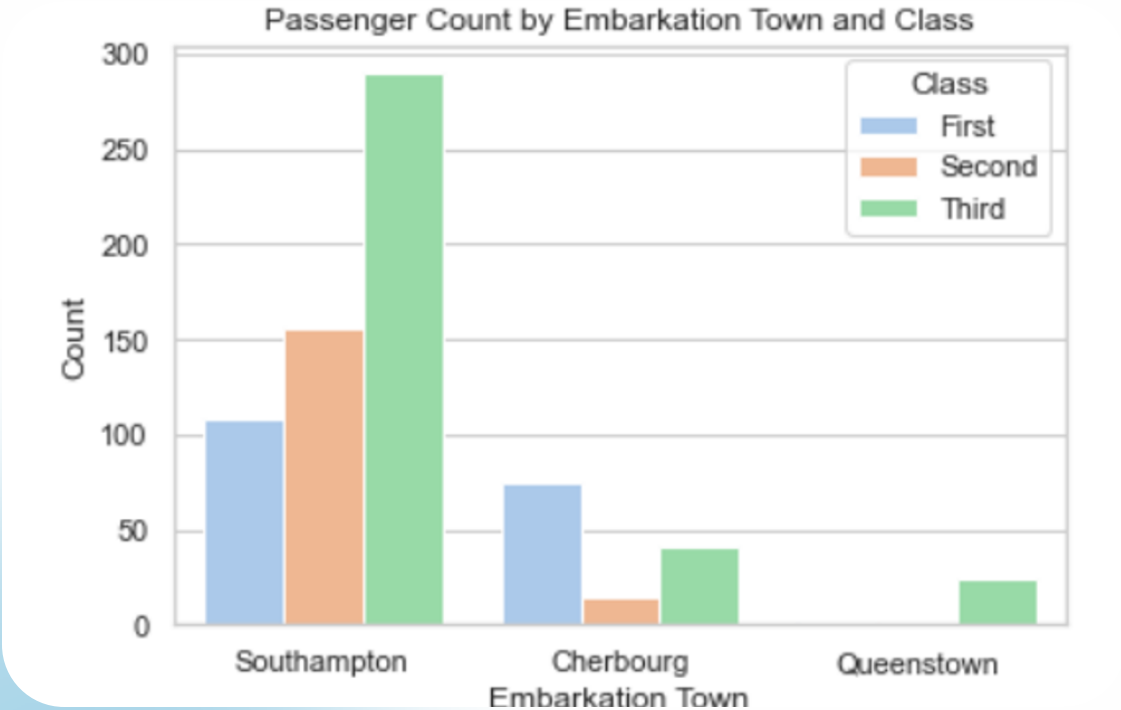
Analysing Multiple Variables Simultaneously

Code example:

```
# Heatmap: Correlation between numerical variables
corr = titanic.corr(numeric_only=True)
sns.heatmap(corr, annot=True, cmap="coolwarm")
plt.title("Correlation Heatmap")
plt.show()

# Pair plot for selected variables
sns.pairplot(titanic, vars=["age", "fare", "survived"], hue="class")
plt.show()
```

Mastering Advanced Customisations

Tailoring Data Visualisations to Your Needs



Enhancing visualisations

- Adds annotations
- Customises colours
- Changes styles

Mastering Advanced Customisations

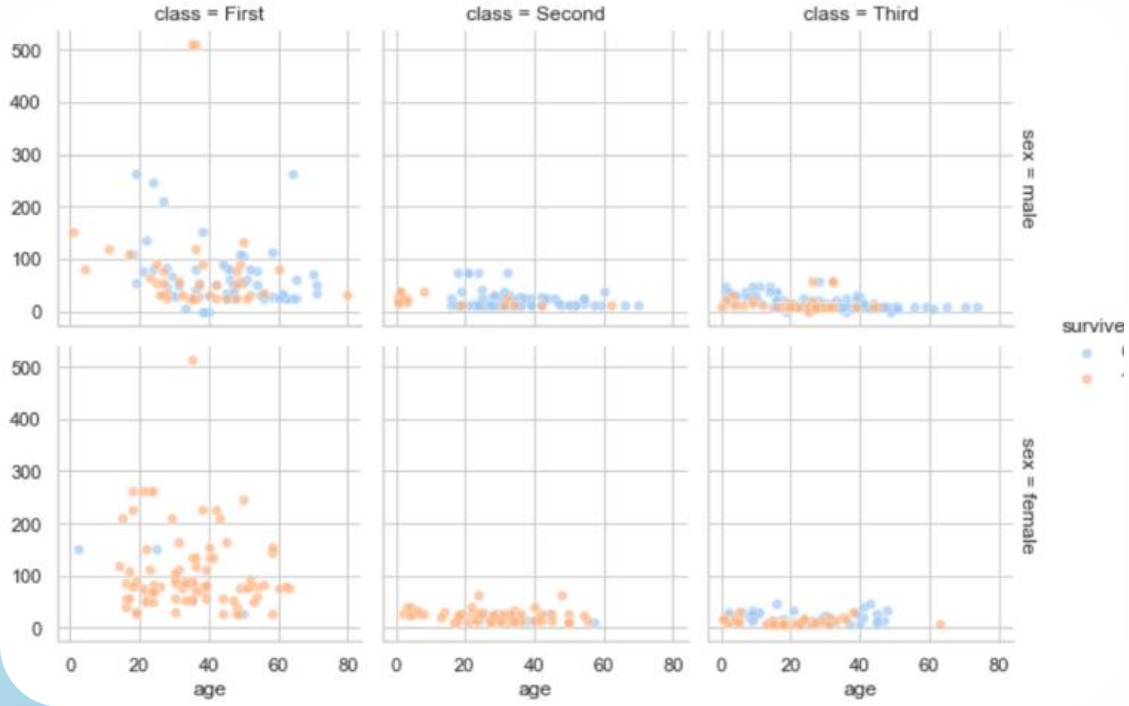
Tailoring Data Visualisations to Your Needs

Code example:

```
# Changing the style and palette
sns.set_theme(style="whitegrid", palette="pastel")

# Customized count plot
sns.countplot(data=titanic, x="embark_town", hue="class")
plt.title("Passenger Count by Embarkation Town and Class")
plt.xlabel("Embarkation Town")
plt.ylabel("Count")
plt.legend(title="Class")
plt.show()
```

Understanding Faceted Visualisations

Breaking Data into Clear, Comparative Views



- **Definition:** Creates multiple subplots for different categories

- **Key method:** FacetGrid

Understanding Faceted Visualisations

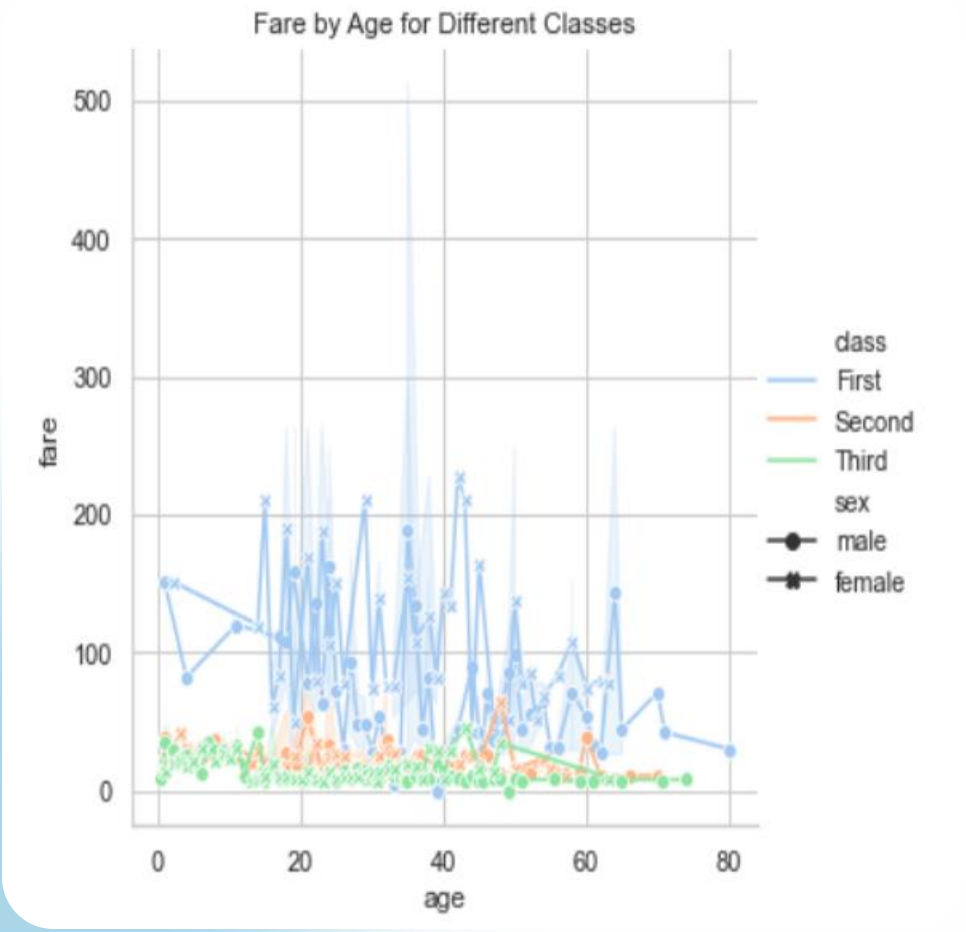
Breaking Data into Clear, Comparative Views

Code example:

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

# Load Titanic dataset
titanic = sns.load_dataset("titanic")

# Faceted grid: Survival by age and sex
g = sns.FacetGrid(titanic, col="sex", row="class", hue="survived", margin_titles=True)
g.map(sns.scatterplot, "age", "fare", alpha=0.7)
g.add_legend()
plt.show()
```

Exploring Advanced Relational Plots

Visualising Complex Relationships in Data



- **Segments in relational plots:** Visualises trends or connections between data points
- **Key method:** sns.relplot
- **Example:** Uses line segments to connect data

Exploring Advanced Relational Plots

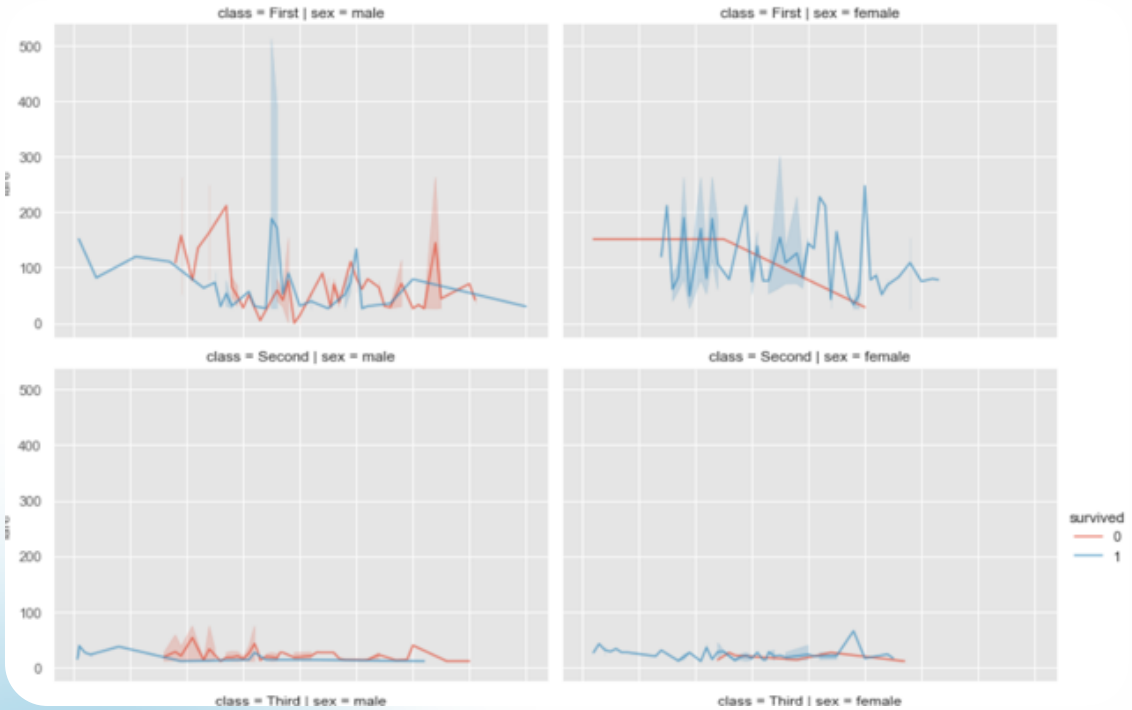
Visualising Complex Relationships in Data

Code example:

```
# Relational plot with line segments for fare by age
sns.relplot(
    data=titanic, x="age", y="fare", kind="line",
    hue="class", style="sex", markers=True, dashes=False
)
plt.title("Fare by Age for Different Classes")
plt.show()
```

Visualising with Panels and Segments

Techniques for Detailed and Comparative Views

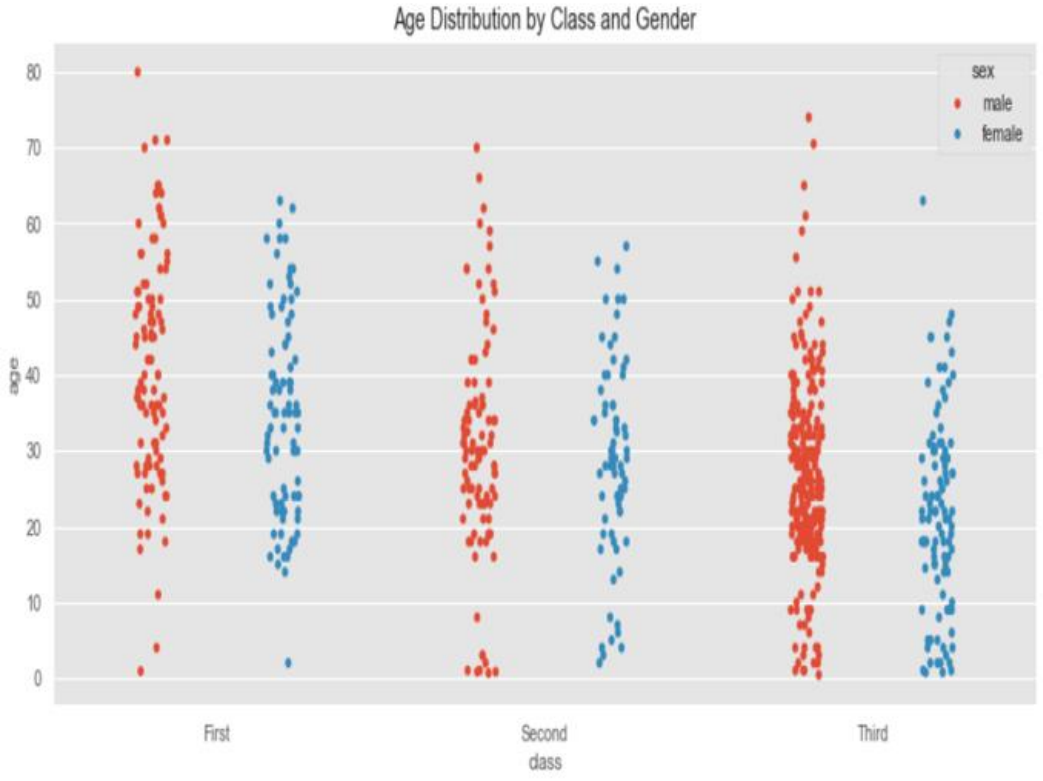



```
# FacetGrid with segments for survival and fare
g = sns.FacetGrid(titanic, col="sex", row="class", hue="survived", height=4, aspect=1.5)
g.map(sns.lineplot, "age", "fare", alpha=0.7)
g.add_legend()
plt.show()
```

- **Complex grids:** Combines multiple dimensions using relplot or FacetGrid

Exploring Advanced Categorical Plots

Enhancing Data Insights with Categorical Visualisations



Strip plot with jitter

- Improves clarity in distributions

Swarm plot

- Avoids overlapping points

Exploring Advanced Categorical Plots

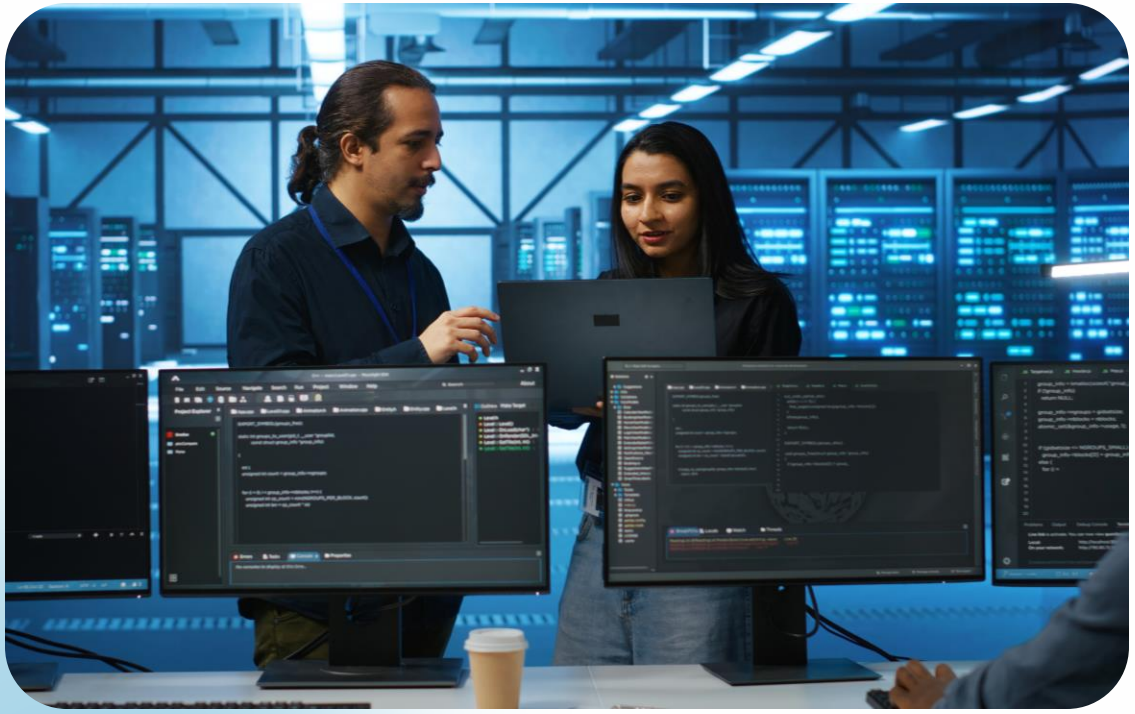
Enhancing Data Insights with Categorical Visualisations

Code example:

```
# Strip plot with jitter
sns.stripplot(data=titanic, x="class", y="age", jitter=True, hue="sex", dodge=True)
plt.title("Age Distribution by Class and Gender")
plt.show()


# Swarm plot
sns.swarmplot(data=titanic, x="class", y="fare", hue="sex", dodge=True)
plt.title("Fare Distribution by Class and Gender")
plt.show()
```

Implementing Best Practices

Guidelines for Effective Data Analysis



- Explore the dataset before visualising
- Use appropriate plots for the data type
- Keep visuals simple and clear
- Apply meaningful titles and labels
- Choose colour palettes carefully for accessibility

Pandas Advanced Use Cases

Accessing the Titanic Dataset

Starting Your Data Exploration Journey

Dataset source

- Uses the Titanic dataset from Stanford University for examples
- https://web.stanford.edu/class/archive/cs/cs109/cs109.1166/stuff/titanic.csv

Loading code

```
import pandas as pd


# Load Titanic dataset
titanic = pd.read_csv("https://web.stanford.edu/class/archive/cs/cs109/cs109.1166/stuff/titanic.csv")
```

Use Case 1: Querying with query

Efficient Ways to Filter and Analyse Data

Code example:

```
# Query: Find all female passengers over 30

who paid more than 50

filtered = titanic.query("Age > 30 and Sex ==

'female' and Fare > 50")

print(filtered.head())
```

- Simplifies filtering logic using string-based queries
- Provides cleaner alternative to chaining multiple conditions

Use Case 2: Transforming Data with applymap

Applying Functions Element-wise Across DataFrames

Code example

```
# Apply a function to
the entire DataFrame
(if numeric) transformed =
titanic[["Age",
"Fare"]].applymap(lam
bda x: x * 1.1 if
pd.notnull(x) else x)
print(transformed.head
())
```

- Applies a function elementwise across numeric columns
- Supports percentage-based transformations

Use Case 3: Reshaping with stack and unstack

Transforming Data Structures for Better Analysis

Code example

```
# Pivot and stack/unstack operations
pivoted = titanic.pivot_table(index="Sex", columns="Pclass", values="Fare", aggfunc="mean")
stacked = pivoted.stack()
unstacked = stacked.unstack()

print("Pivot Table:\n", pivoted)
print("\nStacked:\n", stacked)
print("\nUnstacked:\n", unstacked)
```

- Reshapes the dataset for group-level analysis
- Facilitates switching between hierarchical indexing using stack or unstack

Use Case 4: Efficient String Operations

Optimising Text Data Processing in Pandas

Code example

```
# Clean and transform string data
titanic["Name_Length"] =
titanic["Name"].str.len()
titanic["Title"] =
titanic["Name"].str.extract(r'([A-Za-z]+)\.')


print(titanic[["Name", "Name_Length",
"Title"]].head())
```

- **str.len():** Computes the length of string values
- **str.extract():** Extracts patterns using regex, such as titles from names

Use Case 5: Advanced Time-Series Handling

Techniques for Complex Temporal Data Analysis

Code example

```
# Create artificial date column for demonstration
import numpy as np
titanic["Travel_Date"] = pd.date_range("1912-04-01", periods=len(titanic), freq="D")

# Analyze passenger counts over time
daily_counts = titanic.groupby(titanic["Travel_Date"].dt.weekday)["PassengerId"].count()

print("Daily Passenger Counts:\n", daily_counts)
```

- Serves demonstration purposes
- Aggregates data by day of the week
- Analyses passenger counts over time
- Demonstrates advanced datetime operations and aggregates data by day of the week

Use Case 6: Generating Interactive Reports with Styler

Enhancing Data Presentation and Visual Appeal

Code example

```
# Style DataFrame
styled =
titanic.head(10).style.background_gradient(
cmap="viridis").set_precision(2)
styled
```

- Uses Styler to create visually appealing tables
- Highlights patterns for better interpretation

Hands-on session

- Provides learners with the Titanic dataset
- Encourages replication and modification of examples for better understanding

Async Recap

1. **Linear Algebra**
Use vectors and matrices to represent and manage high-dimensional data structures

2. **Matrix Operations**
Perform addition, transpose, and rank to manipulate and analyse data effectively

3. **Descriptive Statistics**
Apply mean, median, and range to summarise and interpret data distributions

4. **Data Visualisation**
Use bar, pie, and box plots to make data trends and patterns easier to understand

5. **Probability**
Measure uncertainty and support decision-making through probabilistic analysis

Q & A

Thank you