
Seaborn
created by :
The easylearn academy

What is Seaborn

- **Seaborn** is a Python data visualization library based on **Matplotlib**.
- It provides a high-level interface for drawing **attractive and informative statistical graphics**.
- Seaborn makes it easier to create complex visualizations with fewer lines of code.

- **Purpose in AI/ML:**
  - Visualizations help understand data distributions, identify patterns, detect outliers, evaluate model performance, and communicate insights effectively.

- **Core Module:**
  - seaborn is the high-level interface for creating plots.

Key Features of seaborn

- **High-Level Interface**:
  - Seaborn provides simple functions for complex visualizations, reducing the need for extensive Matplotlib customization.
- **Statistical Visualizations**:
  - Specialized support for plots like heatmaps, violin plots, box plots, and regression plots, which are common in statistical analysis.
- **Themes and Aesthetics**:
  - Built-in themes (e.g., darkgrid, whitegrid) and color palettes for visually appealing plots.
- **Integration with Pandas**:
  - Works seamlessly with DataFrames, allowing direct plotting from columns and handling missing data gracefully.
- **Faceting**:
  - Functions like FacetGrid and PairGrid enable multi-panel plots to explore relationships across subsets of data.

Advantages of Seaborn

- Simplifies complex Matplotlib tasks.
- Produces publication-quality visuals with minimal code.
- Excellent for exploratory data analysis (EDA).
- Customizable via Matplotlib for advanced tweaks.

Limitations of Seaborn

- Less flexible than raw Matplotlib for highly customized plots.
- Performance can be slower for very large datasets.
- Primarily focused on statistical visualizations, not general-purpose plotting.

Seaborn Vs. Matplotlib

Aspect	Seaborn	Matplotlib
Focus	Statistical visualizations with high-level functions	Low-level library for highly customizable, general-purpose plots.
Ease of Use	Concise syntax, less code for complex statistical plots	Requires more boilerplate code for complex plots
Aesthetics	Modern, attractive default styles and color palettes	Older, less visually appealing defaults (improved post-version 2.0)
Pandas Integration	Optimized for pandas DataFrames, seamless column-based plotting	Manual data extraction from DataFrames required
Customization	Abstracts customization, relies on Matplotlib for fine-tuning	Granular control over every plot element

How to Get Started

- Install Seaborn
  - pip install seaborn

- **Basic Workflow:**
  - Import matplotlib.pyplot as plt.
  - Import seaborn as sns
  - Create data (e.g., using NumPy or Pandas).
  - Use plotting functions like sns.plot(), sns.scatterplot(), etc.
  - Customize with labels, titles, etc.
  - Display with plt.show() or save with plt.savefig().

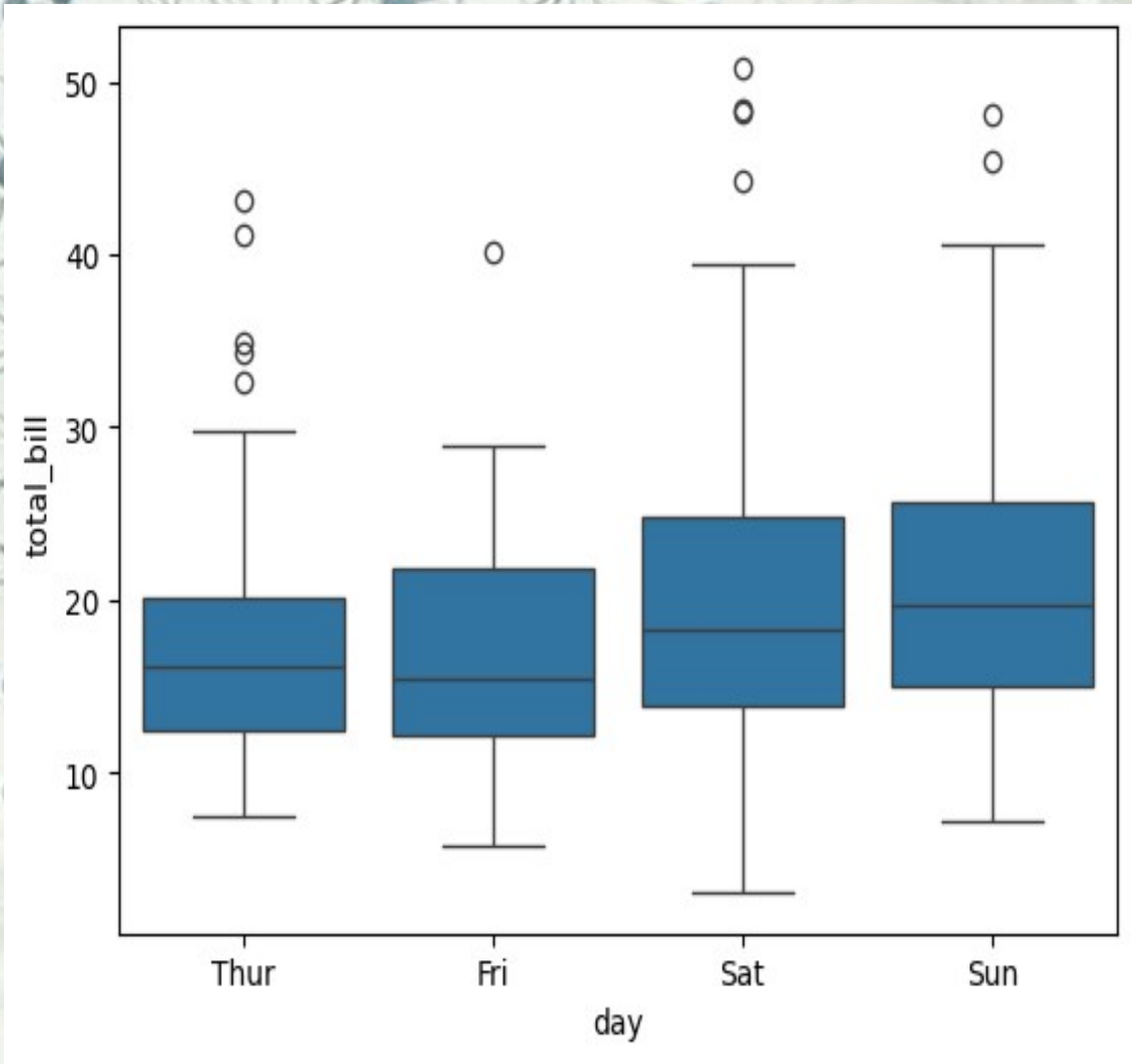
- **How to get default dataset**
  - sns.get_dataset_names()

Example

- import numpy as np
- import pandas as pd
- import matplotlib.pyplot as plt
- %matplotlib inline
- import seaborn as sns

- **# Load a sample dataset of default seaborn**
- **# Create a boxplot**

- tips = sns.load_dataset("tips")
- sns.boxplot(x="day",y="total_bill",data=tips)
- plt.show()



Type of Plot

Scatter Plot (sns.scatterplot)	Line Plot (sns.lineplot)	Histogram (sns.histplot)	KDE Plot (sns.kdeplot)	Box Plot (sns.boxplot)
ECDE Plot ()	Rug Plot ()	Bar Plot (sns.barplot)	Count Plot (sns.countplot)	Box Plot ()
Violin Plot (sns.violinplot)	Pair Plot (sns.pairplot)	Cat Plot ()	Heatmap (sns.heatmap)	Strip Plot ()
	Swarm Plot ()	Point Plot ()	Joint plot (sns.jointplot)	

How to Select

relplot (relational)	DisPlot	CatPlot
• Scatter plot • Line Plot	• Histogram • KDE Plot • ECDF Plot • Rug Plot	• Bar plot • Count Plot • Box plot • Violin plot • Pair Plot • Cat Plot • Heatmap • Strip Plot • Swarm Plot • Point Plot • Joint Plot

Scatter Plot

- **Purpose:**
  - Visualize the relationship between two continuous variables by plotting individual data points.
- **Use Case:**
  - Exploring correlations between features or between features and target variables.
  - Visualizing results of dimensionality reduction (e.g., PCA, t-SNE) for clustering tasks.
  - Comparing predicted vs. actual values in regression or classification tasks.
- **Key Features:**
  - Customizable markers, colors, and sizes to encode additional variables (e.g., hue, size).
  - Supports transparency (alpha) for overlapping points.
  - Integrates with hue and style for categorical differentiation.
  - Can be combined with sns.regplot for regression line overlays.

Scatter Plot

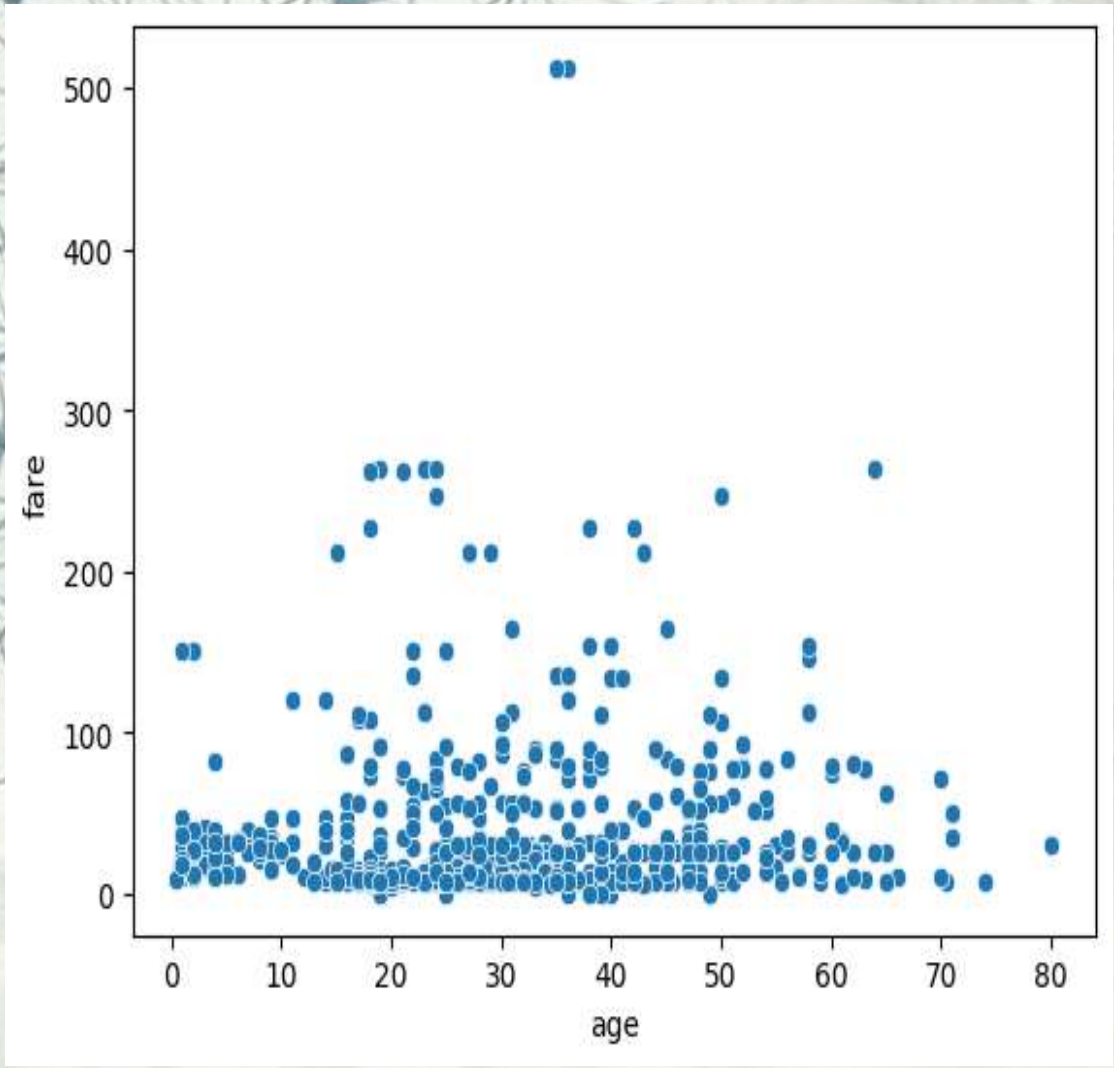
- **Syntax:**
  - sns.scatterplot(x="col1", y="col2", data=df, hue="category", size="size_col", style="style_col", alpha=0.5)
- **Key Attribute:**
  - **x, y:** Column names (or data) for x- and y-axes (continuous variables, e.g., features or predictions).
  - **data:** DataFrame containing the data. hue: Column name for coloring points by category (e.g., class labels).
  - **size:** Column name to vary point sizes (e.g., sample weights or feature importance).
  - **style:** Column name to vary point markers (e.g., different models or clusters).
  - **alpha:** Transparency of points (0 to 1, useful for overlapping points).

Example ***# Fare over age***

- import numpy as np
- import pandas as pd
- import matplotlib.pyplot as plt
- %matplotlib inline
- import seaborn as sns

- # use of titanic dataset
- tit = sns.load_dataset('titanic')

- sns.scatterplot(x='age', y='fare', data=tit)
- plt.show()



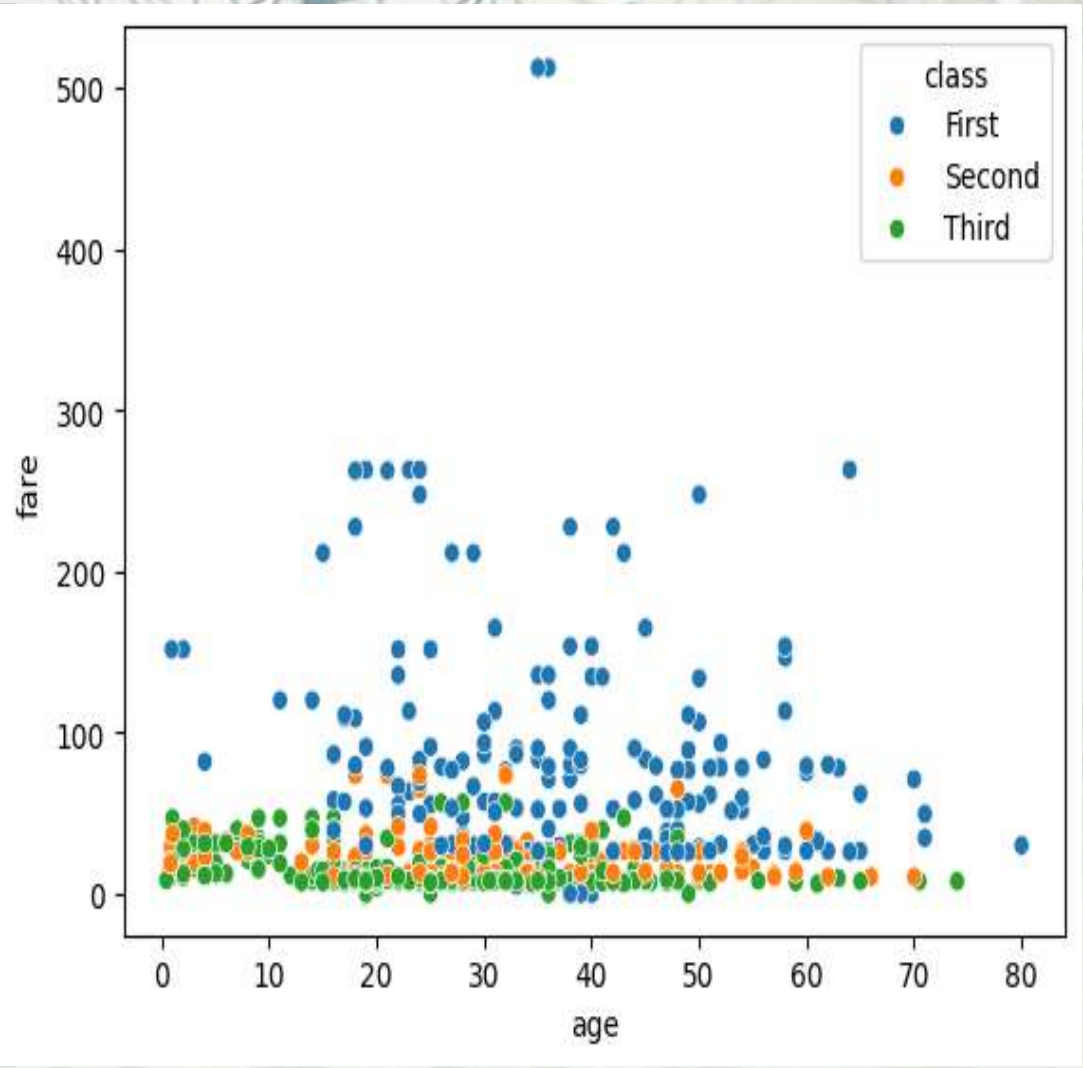
Example

 # Fare over age by class

- import numpy as np
- import pandas as pd
- import matplotlib.pyplot as plt
- %matplotlib inline
- import seaborn as sns

- # use of titanic dataset
- tit = sns.load_dataset('titanic')

- sns.scatterplot(x='age', y='fare', data=tit, hue='class')
- plt.show()



Line Plot

- **Purpose:**
  - Display trends or relationships over a continuous variable, such as timeJon time plottime (typically x-axis) or time (y-axis).
- **Use Case:**
  - Plotting training and validation loss/accuracy curves over epochs during model training.
  - Visualizing time-series data or sequential model performance metrics.
  - Comparing model performance across iterations or experiments.
- **Key Features:**
  - Automatically computes confidence intervals for aggregated data.
  - Supports hue for plotting multiple lines (e.g., training vs. validation metrics).
  - Customizable line styles and markers for clarity.
  - Handles missing data gracefully with optional interpolation.

Line Plot

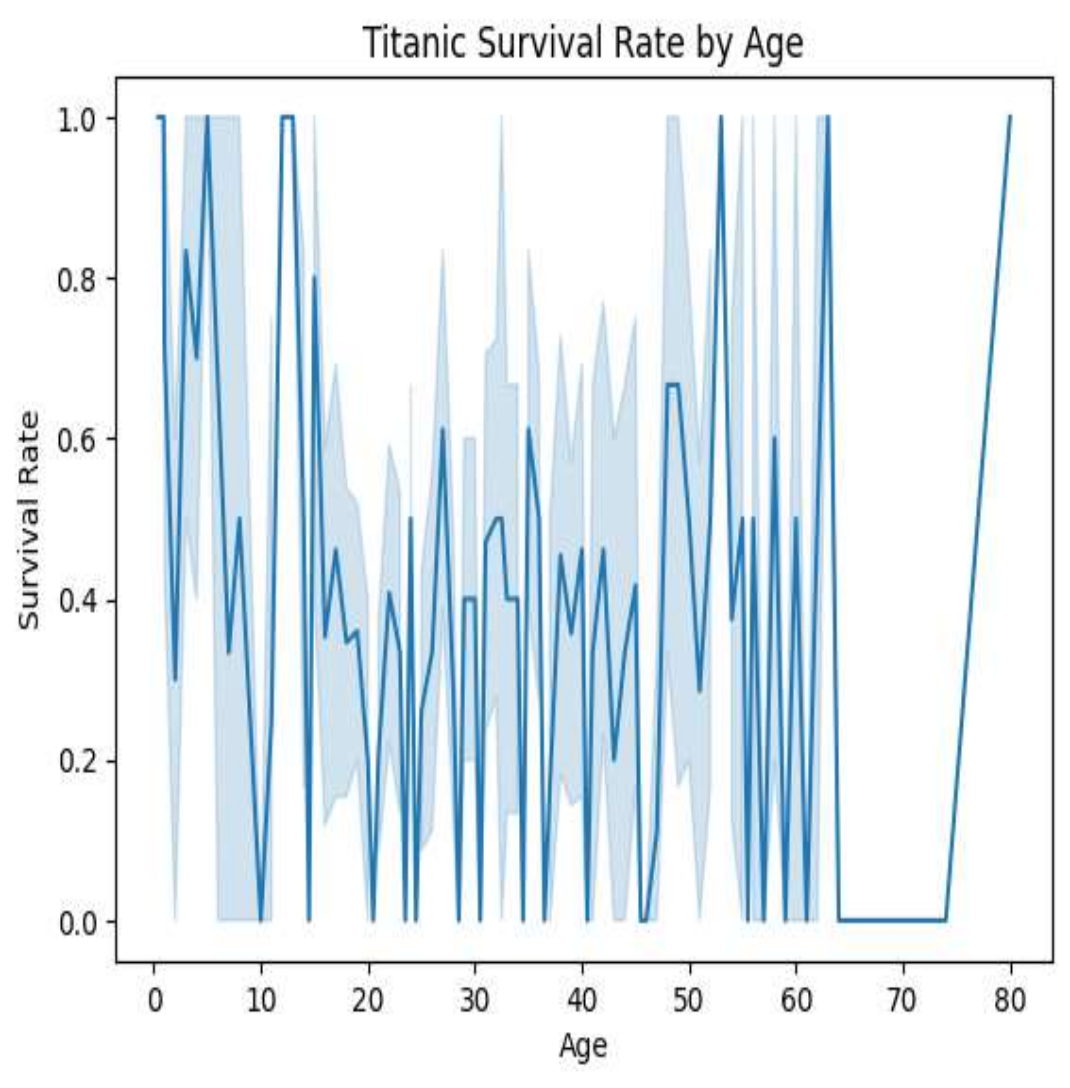
- **Syntax:**
  - sns.lineplot(x="time", y="value", data=df, hue="group", style="group", markers=True)
- **Key Attribute:**
  - **x, y:** Column names for x- and y-axes (e.g., epochs, loss/accuracy).
  - **data:** DataFrame containing the data.
  - **hue**: Column name for coloring lines by category (e.g., training vs. validation).
  - **style:**Column name for varying line styles/markers (e.g., different models).
  - **markers:** Boolean to show markers at data points (improves readability).

Example #*average survival by age*

- import seaborn as sns
- import matplotlib.pyplot as plt
- %matplotlib inline
- import pandas as pd
- **# Load dataset**
- titanic = sns.load_dataset("titanic")
- **# Drop rows where age or survived is missing**
- tit = titanic.dropna(subset=['age', 'survived'])

- **# Create lineplot: average survival by age**
- sns.lineplot(x='age', y='survived', data=tit)

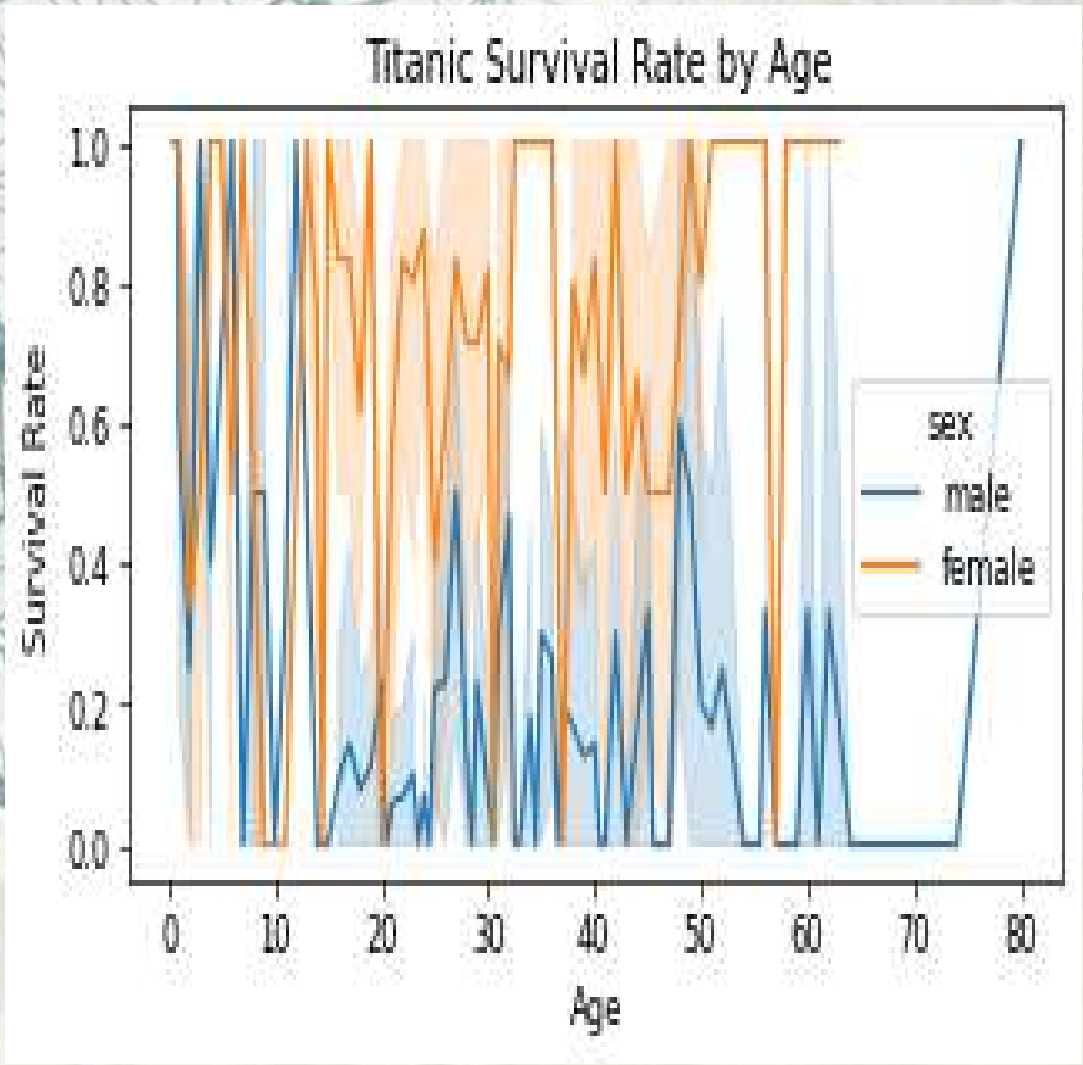
- **# Add labels and title**
- plt.title('Titanic Survival Rate by Age')
- plt.xlabel('Age')
- plt.ylabel('Survival Rate')
- plt.show()



Example

#average survival by age over gender

```
import seaborn as sns
import matplotlib.pyplot as plt
%matplotlib inline
import pandas as pd
# Load dataset
titanic = sns.load_dataset("titanic")
tit = titanic.dropna(subset=['age', 'survived'])
plt.figure(figsize=(6,3),dpi=60)
sns.lineplot(x='age', y='survived', data=tit, hue='sex')
plt.title('Titanic Survival Rate by Age')
plt.xlabel('Age')
plt.ylabel('Survival Rate')
plt.show()
```



Histogram

- **Purpose:**
  - Show the distribution of a single continuous variable by grouping data into bins.
- **Use Case:**
  - Analyzing feature distributions to identify skewness, outliers, or multimodality.
  - Visualizing the distribution of model predictions or residuals.
  - Checking data normality for statistical assumptions in ML models.
- **Key Features:**
  - Flexible binning options (e.g., number of bins, bin width).
  - Supports kernel density estimation (kde=True) for smoothed distributions.
  - Customizable hue for comparing distributions across categories
  - Stacked or overlaid histograms for multi-group comparisons.

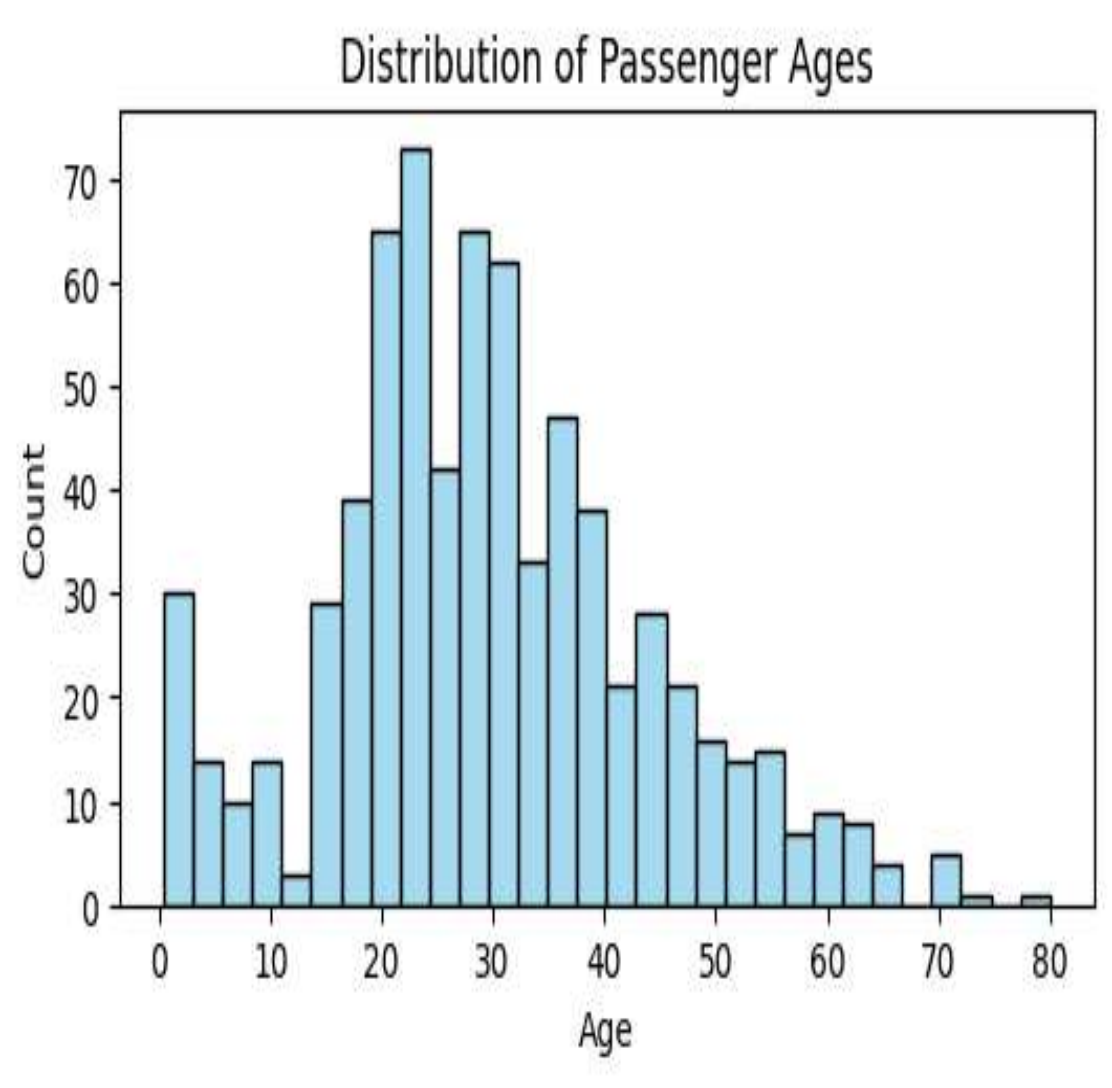
Histogram

- **Syntax:**
  - sns.histplot(x="col", data=df, bins=30, kde=True, hue="category", stat="density")
- **Key Attribute:**
  - **x:** Column name for the variable to plot (continuous, e.g., feature or prediction).
  - **data:** DataFrame containing the data.
  - **bins:** Number of bins or binning method (controls granularity).
  - **kde:** Boolean to overlay kernel density estimate (shows smooth distribution).
  - **hue:** Column name for comparing distributions by category (e.g., classes).
  - **stat:** Output type (e.g., "count", "density", "probability").

Example #Distribution of Passenger Ages

- import seaborn as sns
- import matplotlib.pyplot as plt
- %matplotlib inline
- **# Load dataset**
- titanic = sns.load_dataset("titanic")
- plt.figure(figsize=(6,3),dpi=100)
- sns.histplot(data=titanic, x='age', bins=30, kde=False, color='skyblue')

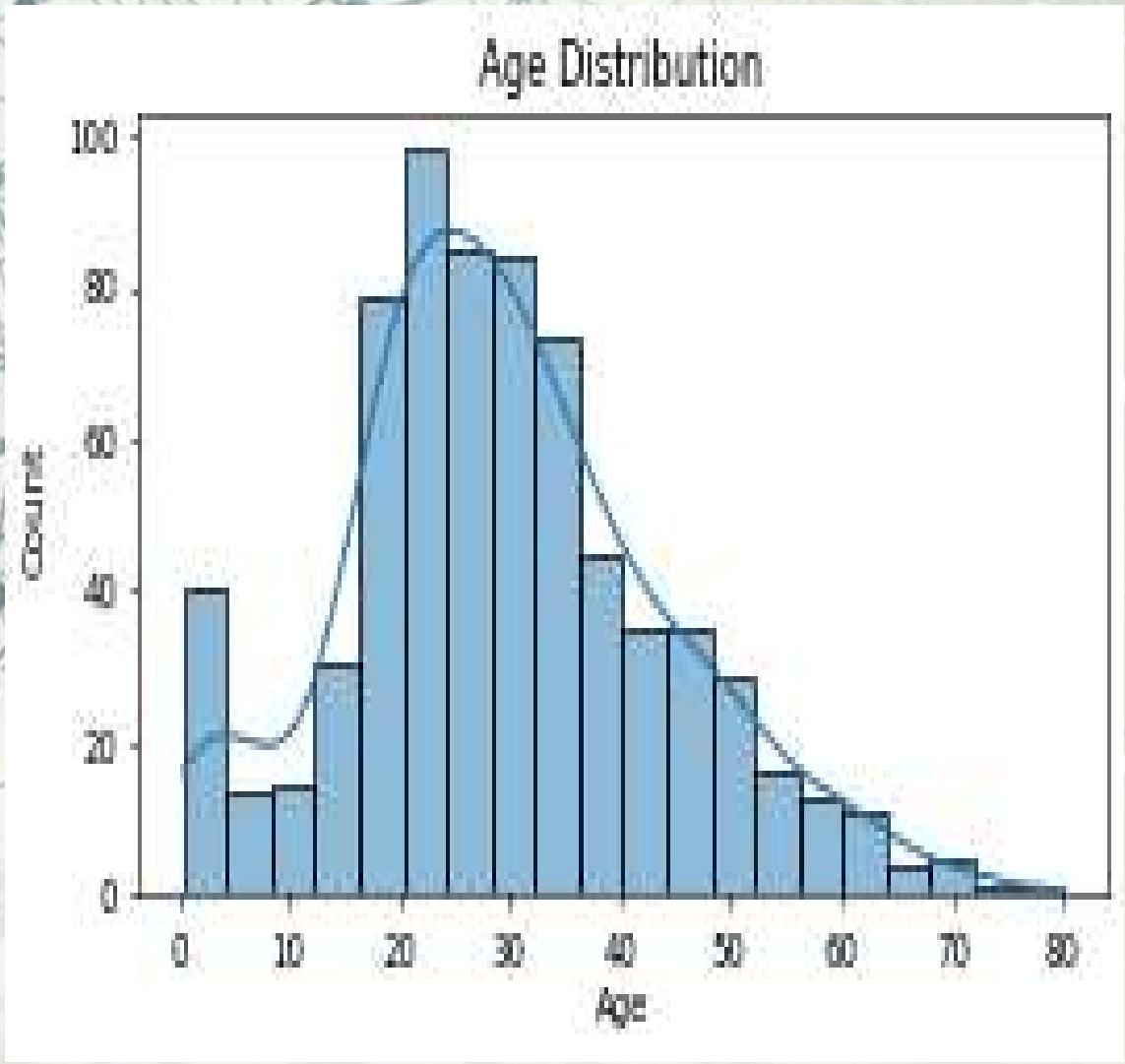
- plt.title('Distribution of Passenger Ages')
- plt.xlabel('Age')
- plt.ylabel('Count')
- plt.show()



Example

- import numpy as np
- import pandas as pd
- import matplotlib.pyplot as plt
- %matplotlib inline
- import seaborn as sns

- # use of titanic dataset
- tit = sns.load_dataset('titanic')
- plt.figure(figsize=(6,3),dpi=50)
- sns.histplot(tit['age'].dropna(), kde=True)
- plt.title('Age Distribution')
- plt.xlabel('Age')
- plt.show()



KDE(Kernel Density Estimate) Plot

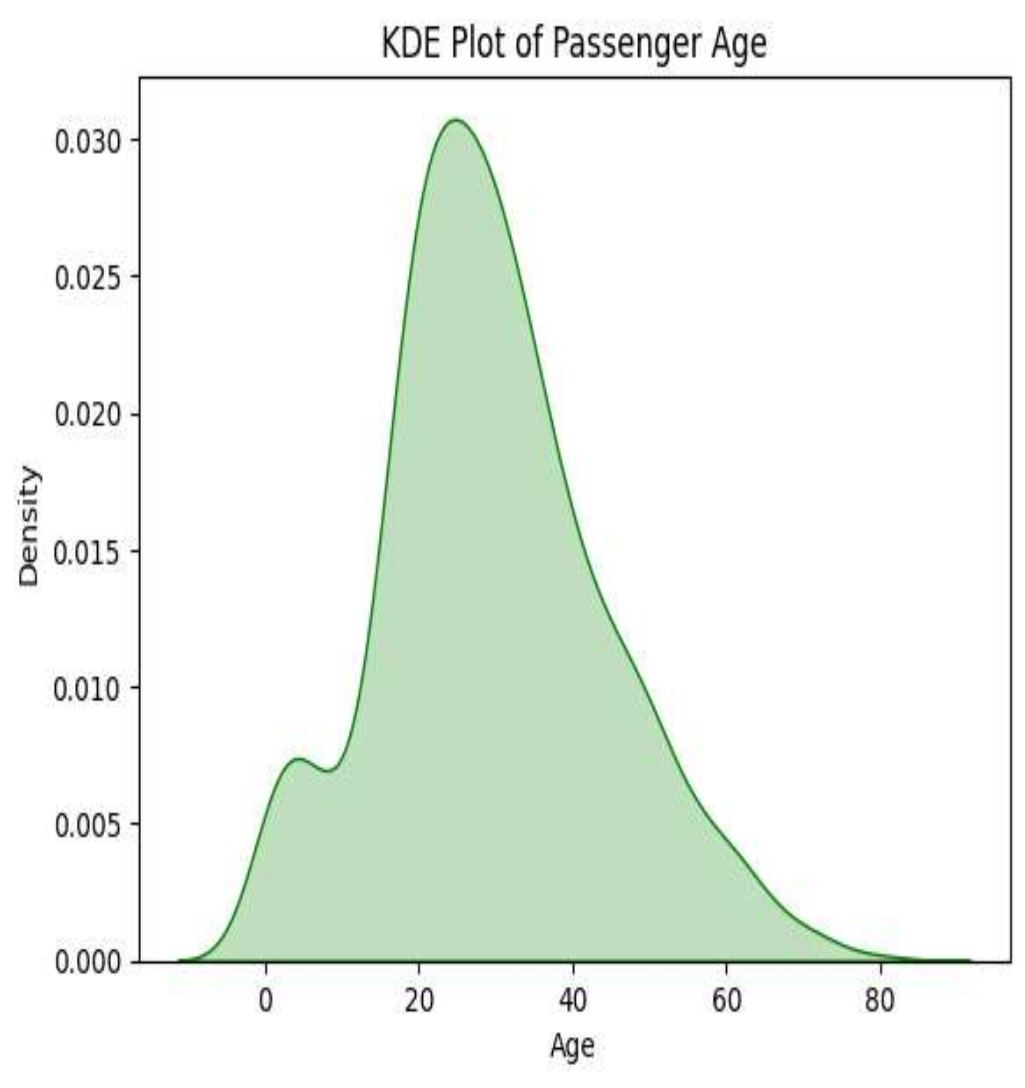
- **Purpose:**
  - Estimate and visualize the probability density of a continuous variable.
- **Use Case:**
  - Visualizing smooth feature distributions without binning.
  - Comparing distributions of predicted vs. actual values.
  - Analyzing cluster density in unsupervised learning tasks.
- **Key Features:**
  - Smooth density estimation with customizable bandwidth (bw_method).
  - Supports hue for comparing multiple distributions.
  - Options for filled areas (fill=True) or contour plots.
  - Handles multivariate data with 2D KDE plots.

KDE(Kernel Density Estimate) Plot

- **Syntax:**
  - sns.kdeplot(x="col", data=df, hue="category", fill=True, bw_adjust=0.5)
- **Key Attribute:**
  - **x (or y):** Column name for continuous variable (e.g., feature).
  - **data:** DataFrame containing the data.
  - **hue:** Column name for comparing distributions by category.
  - **fill:** Boolean to fill area under the curve (enhances visibility).
  - **bw_adjust:** Bandwidth adjustment for density smoothness.

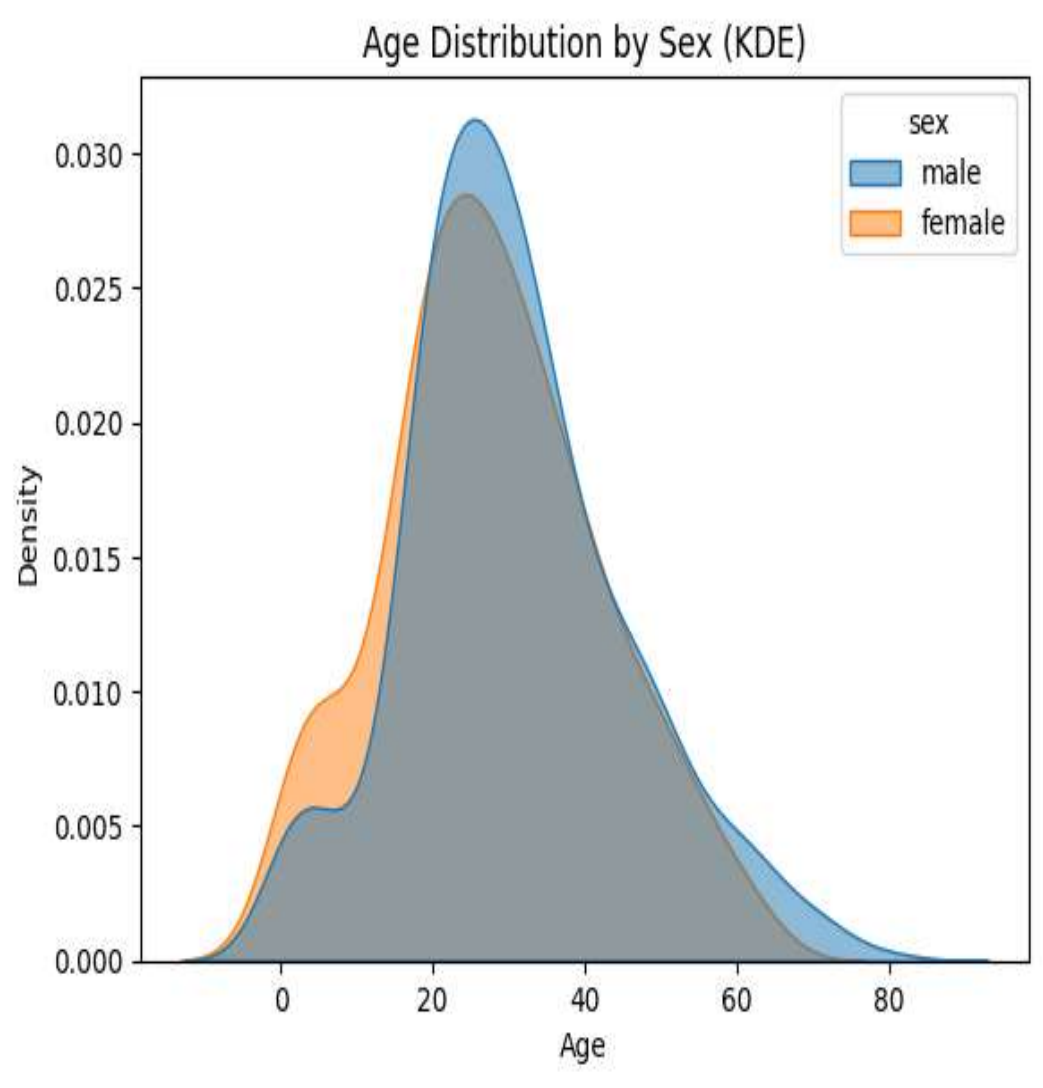
Example

- import seaborn as sns
- import matplotlib.pyplot as plt
- %matplotlib inline
- import pandas as pd
- **# Load dataset**
- titanic = sns.load_dataset("titanic")
- titanic_age = titanic.dropna(subset=['age'])
- sns.kdeplot(data=titanic_age, x='age', fill=True, color='green')
- plt.title('KDE Plot of Passenger Age')
- plt.xlabel('Age')
- plt.ylabel('Density')
- plt.show()



Example

- import seaborn as sns
- import matplotlib.pyplot as plt
- %matplotlib inline
- import pandas as pd
- **# Load dataset**
- titanic = sns.load_dataset("titanic")
- titanic_age = titanic.dropna(subset=['age'])
- sns.kdeplot(data=titanic, x='age', hue='sex', fill=True, common_norm=False, alpha=0.5)
- plt.title('Age Distribution by Sex (KDE)')
- plt.xlabel('Age')
- plt.ylabel('Density')
- plt.show()



ECDF Plot (Empirical Cumulative Distribution Function)

- **Purpose:**
  - Visualizes the cumulative distribution of a dataset, showing the proportion (or count) of observations below each value.
- **Use Case:**
  - Comparing distributions across multiple groups (e.g., comparing sales across regions).
  - Assessing the spread, percentiles, or goodness-of-fit of a dataset without binning.
  - Visualizing continuous data where histograms or KDEs might obscure details.
- **Key Features:**
  - Directly represents each data point, avoiding binning or smoothing parameters.
  - Monotonically increasing curve, ideal for comparing multiple distributions.
  - Supports hue for coloring by categorical variables, stat for proportion/count, and complementary for plotting 1-CDF.
  - Can be created using sns.ecdfplot() or sns.displot(kind="ecdf").

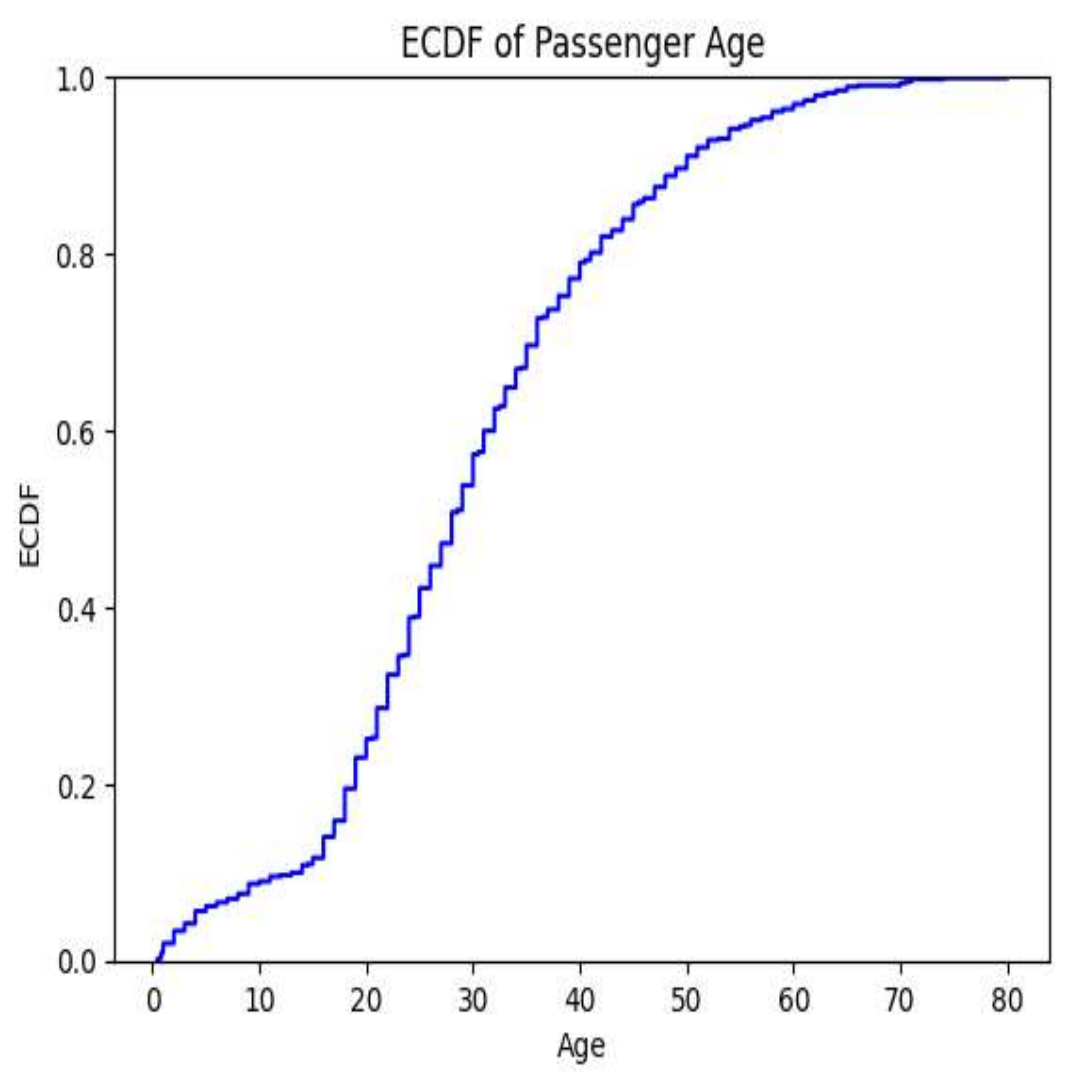
ECDF Plot (Empirical Cumulative Distribution Function)

- **Syntax:**
  - sns.ecdfplot(data=None, x=None, y=None, hue=None, weights=None, stat="proportion“ , complementary=False, palette=None, ax=None, **kwargs)
  - sns.displot(data=None, x=None, hue=None, kind="ecdf", stat="proportion", complementary=False, palette=None, col=None, row=None, **kwargs)
- **Key Attribute:**
  - x, y: Variable to plot (one is required; y for vertical ECDF, x for horizontal).
  - hue: Categorical variable for color grouping (e.g., "sex").
  - weights: Array-like weights for weighted ECDF.
  - stat: Output statistic ("proportion" for 0-1 scale, "count" for raw counts).
  - complementary: If True, plots 1-CDF (complementary CDF).
  - palette: Color palette for hue (e.g., "Set2").
  - col, row (for displot): Facet variables for subplots.

Example

- import seaborn as sns
- import matplotlib.pyplot as plt
- %matplotlib inline
- import pandas as pd
- **# Load dataset**
- titanic = sns.load_dataset("titanic")
- titanic_clean = titanic.dropna(subset=['age'])
- sns.ecdfplot(data=titanic_clean, x='age', color='blue')
- plt.title('ECDF of Passenger Age')
- plt.xlabel('Age')
- plt.ylabel('ECDF')
- plt.show()

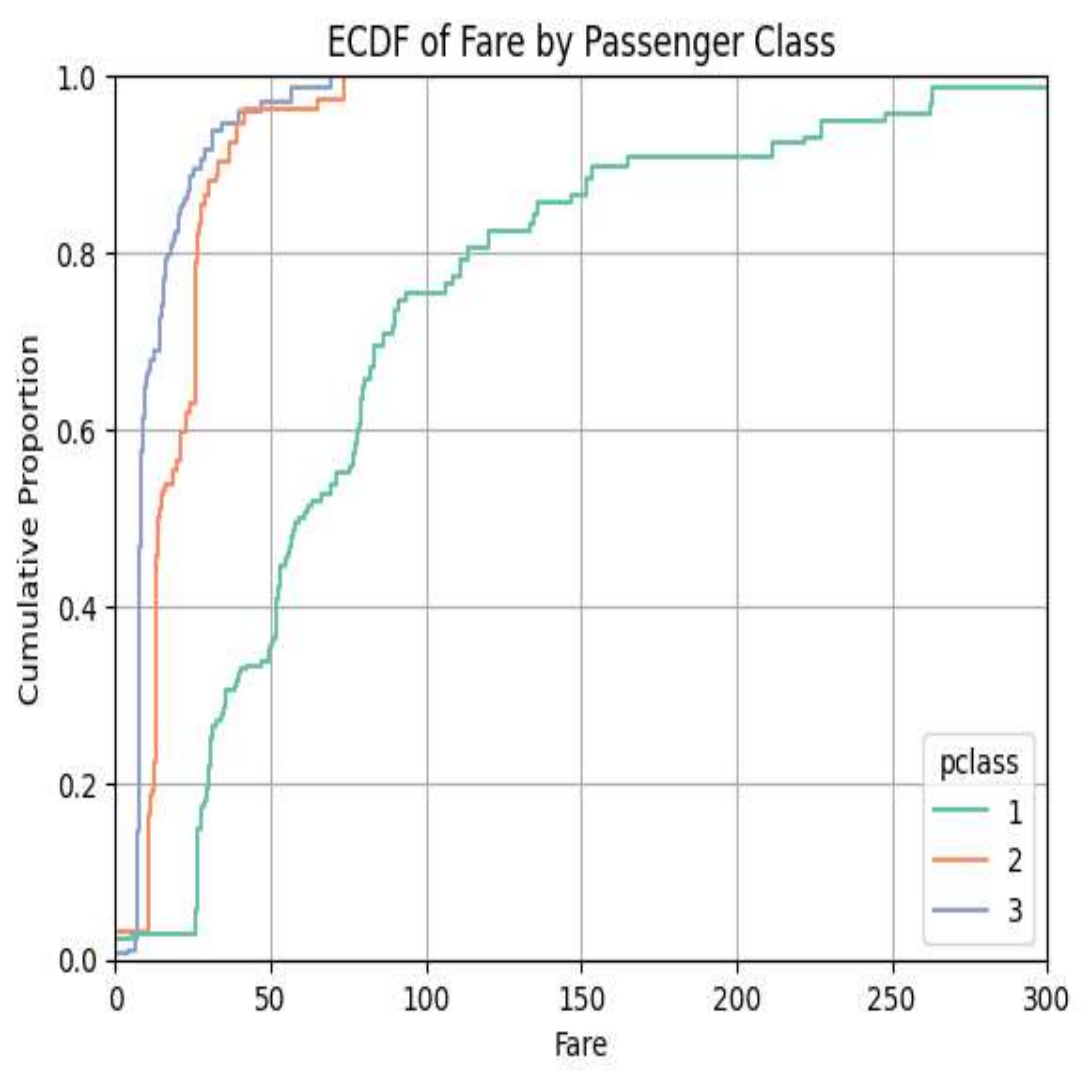
=



Example

- import seaborn as sns
- import matplotlib.pyplot as plt
- %matplotlib inline
- import pandas as pd
- **# Load dataset**
- titanic = sns.load_dataset("titanic")

- titanic_clean = titanic.dropna(subset=['fare'])
- sns.ecdfplot(data=titanic_clean, x='fare', hue='pclass', palette='Set2')
- plt.title('ECDF of Fare by Passenger Class')
- plt.xlabel('Fare')
- plt.ylabel('Cumulative Proportion')
- plt.xlim(0, 300) # Focus on reasonable fare range
- plt.grid(True)
- plt.show()



Rug Plot

- **Purpose:**
  - Displays individual data points as small ticks along an axis to show the marginal distribution.
- **Use Case:**
  - Complementing other plots (e.g., KDE or scatter plots) to show exact data point locations.
  - Visualizing the density of observations in univariate or bivariate distributions.
  - Useful for small to medium datasets where individual points are meaningful.
- **Key Features:**
  - Unobtrusive, adding ticks without cluttering the main plot.
  - Supports hue for coloring by categories and height to adjust tick size.
  - Often paired with sns.kdeplot() or sns.scatterplot() for context.

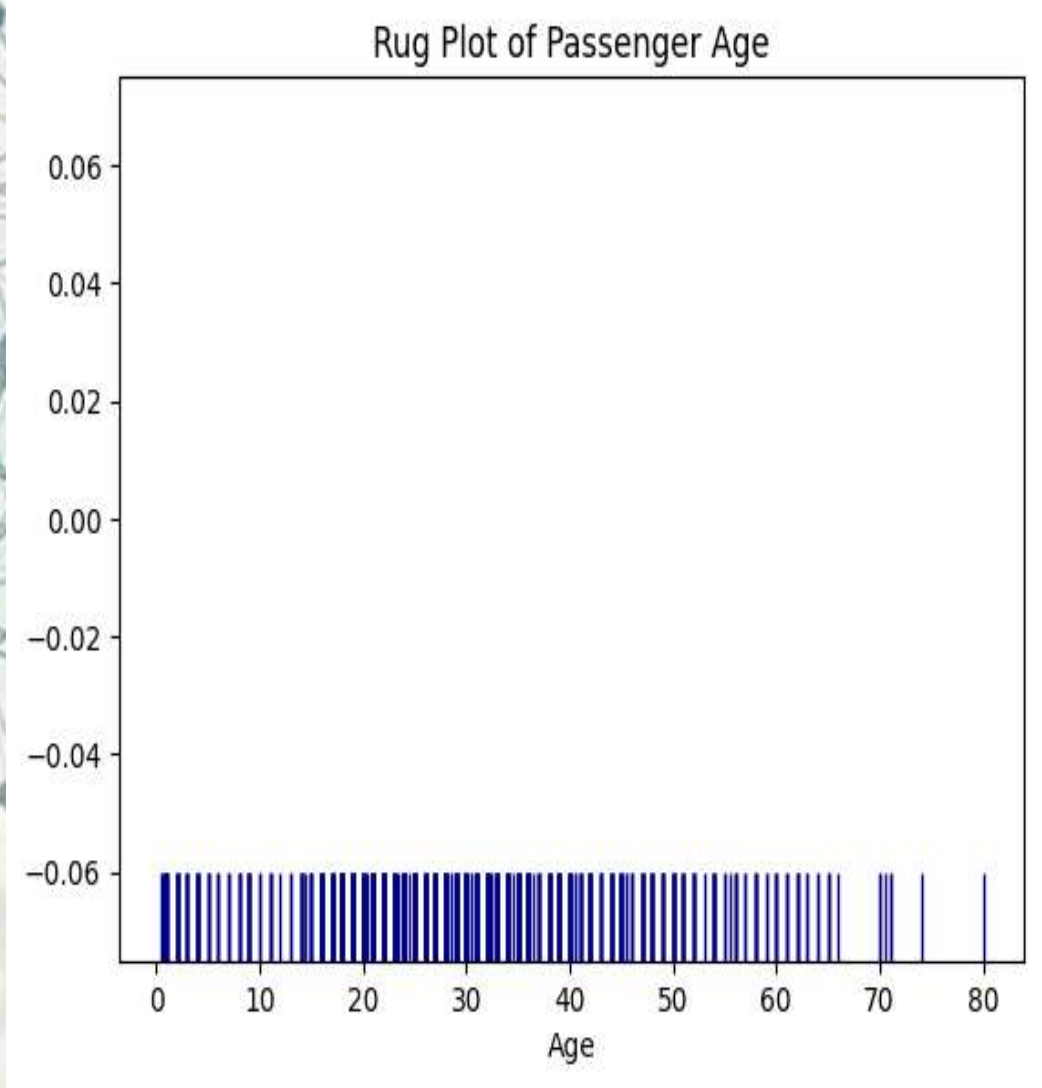
Rug Plot

- **Syntax:**
  - sns.rugplot(data=None, x=None, y=None, hue=None, height=0.05, expand_margins=True, palette=None, ax=None, **kwargs)
- **Key Attribute:**
  - **x, y:** Variable for ticks (one is required; x for horizontal, y for vertical).
  - **hue:** Categorical variable for coloring ticks.
  - **height:** Proportion of axis for tick length (e.g., 0.05 for 5% of axis).
  - **expand_margins:** If True, adjusts plot margins to fit rug ticks.
  - **palette:** Color palette for hue. axis: Specify axis to rug ("x" or "y") if both x and y are provided.

Example

- import seaborn as sns
- import matplotlib.pyplot as plt
- %matplotlib inline
- import pandas as pd
- **# Load dataset**
- titanic = sns.load_dataset("titanic")

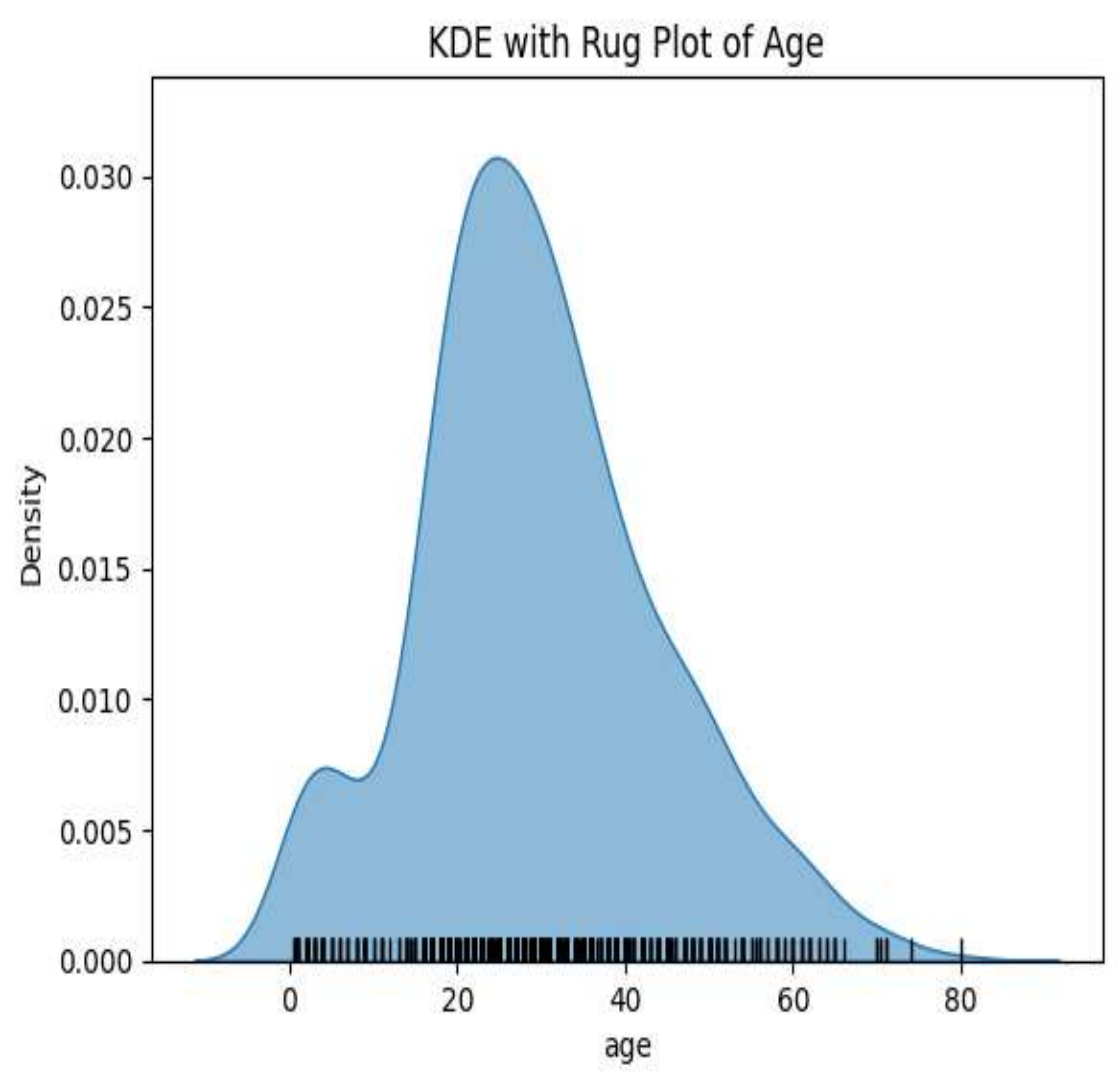
- titanic_clean = titanic.dropna(subset=['age'])
- sns.rugplot(data=titanic_clean, x='age', height=0.1, color='darkblue')
- plt.title('Rug Plot of Passenger Age')
- plt.xlabel('Age')
- plt.show()



Example

- import seaborn as sns
- import matplotlib.pyplot as plt
- %matplotlib inline
- import pandas as pd
- **# Load dataset**
- titanic = sns.load_dataset("titanic")
- titanic_clean = titanic.dropna(subset=['age'])
- sns.kdeplot(data=titanic_clean, x='age', fill=True, alpha=0.5)
- sns.rugplot(data=titanic_clean, x='age', color='black')
- plt.title('KDE with Rug Plot of Age')
- plt.show()

=



Bar Plot

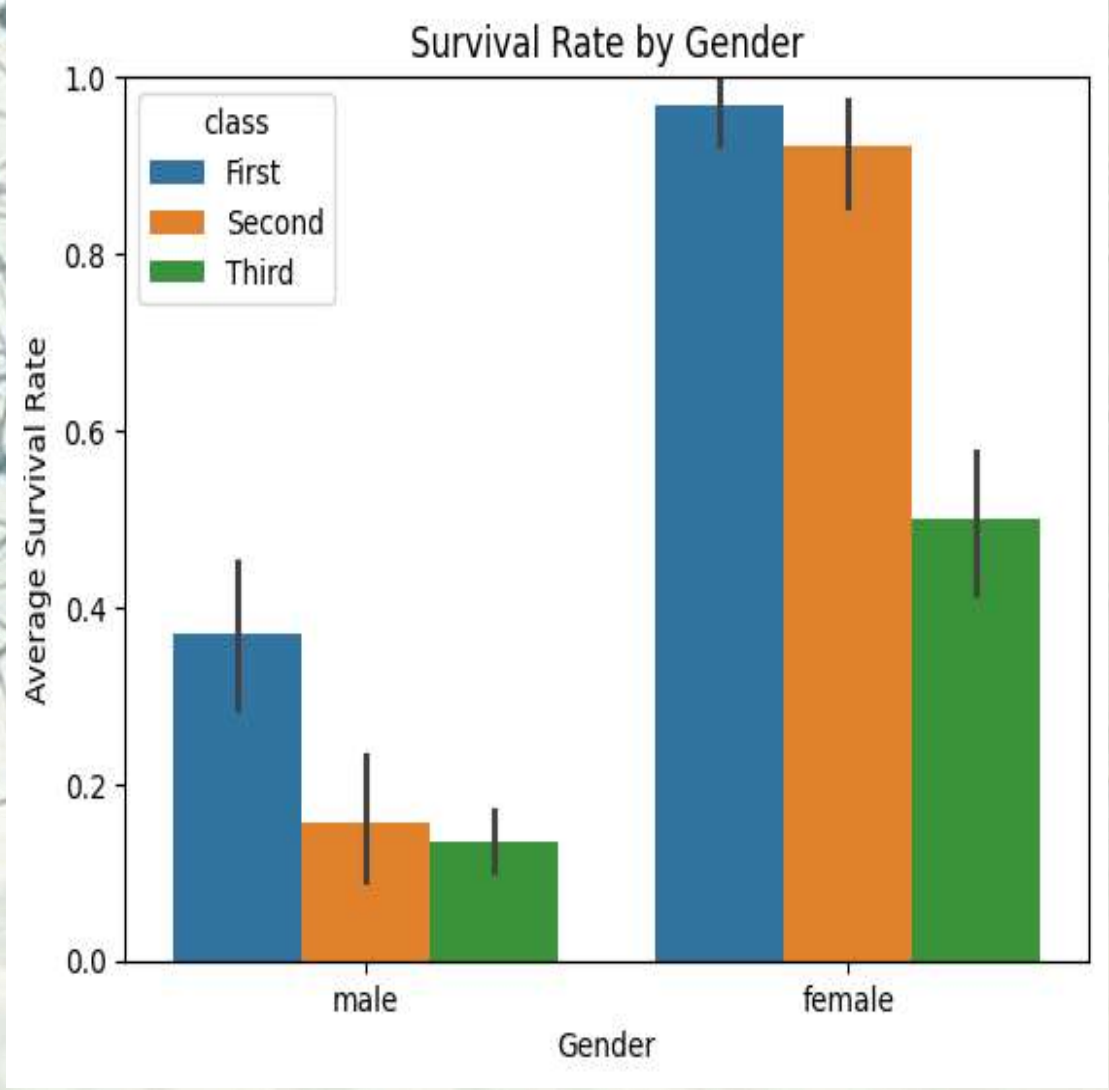
- **Purpose:**
  - Compare a continuous metric (e.g., mean, sum) across categories.
- **Use Case:**
  - Comparing model performance metrics (e.g., accuracy, precision) across models or hyperparameters.
  - Visualizing feature importance scores from models like Random Forest or XGBoost.
  - Summarizing aggregated metrics by group or class.
- **Key Features:**
  - Automatically computes summary statistics (default: mean) with confidence intervals.
  - Supports hue for nested categorical comparisons.
  - Customizable error bars (e.g., standard deviation, confidence intervals).
  - Flexible aggregation functions via estimator parameter.

Bar Plot

- **Syntax:**
  - sns.barplot(x="category", y="value", data=df, hue="group", estimator="mean")
- **Key Attribute:**
  - **x, y:** Column names for categorical (x) and continuous (y) variables.
  - **data:** DataFrame containing the data.
  - **hue:** Column name for nested categories (e.g., models).
  - **estimator:** Function for aggregation (e.g., mean, sum, median).

Example

- import seaborn as sns
- import matplotlib.pyplot as plt
- %matplotlib inline
- import pandas as pd
- **# Load dataset**
- titanic = sns.load_dataset("titanic")
- sns.barplot(data=titanic, x='sex', y='survived', hue='class')
- plt.title('Survival Rate by Gender')
- plt.xlabel('Gender')
- plt.ylabel('Average Survival Rate')
- plt.ylim(0, 1) # Since survival is 0 or 1
- plt.show()



Count Plot

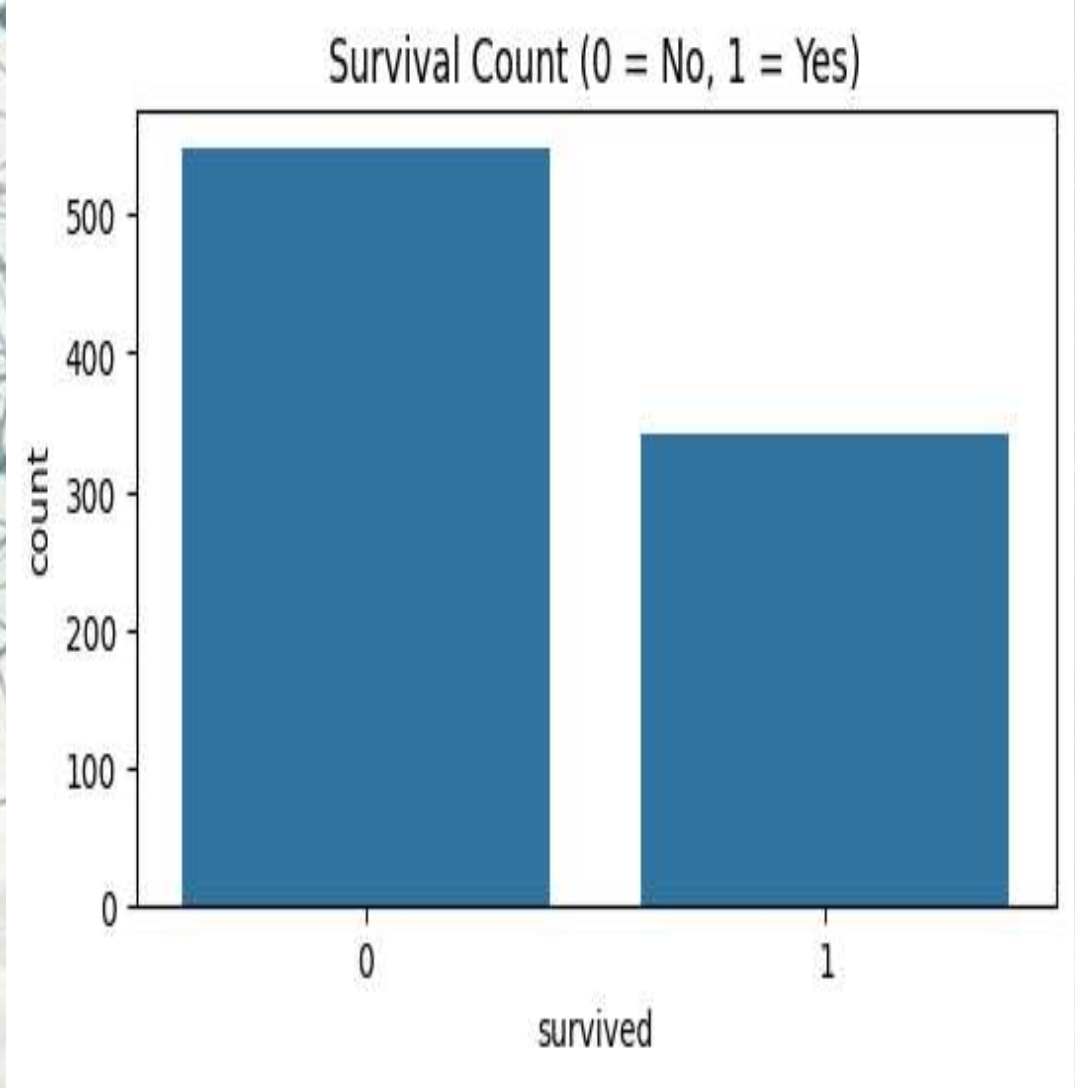
- **Purpose:**
  - Show the count of occurrences for each category in a categorical variable.
- **Use Case:**
  - Visualizing class imbalances in classification datasets.
  - Analyzing the frequency of categorical features.
  - Comparing group sizes in stratified sampling or cross-validation.
- **Key Features:**
  - Simple and intuitive for categorical data.
  - Supports hue for nested categorical comparisons.
  - Customizable bar order and orientation (horizontal/vertical).
  - Stacks or dodges bars for multi-group comparisons.

Count Plot

- **Syntax:**
  - sns.countplot(x="category", data=df, hue="group", order=["cat1", "cat2"])
- **Key Attribute:**
  - **x (or y):** Column name for categorical variable (e.g., class labels).
  - **data:** DataFrame containing the data.
  - **hue:** Column name for nested categorical comparison (e.g., subgroups).
  - **order:** List to specify category order (controls bar arrangement).

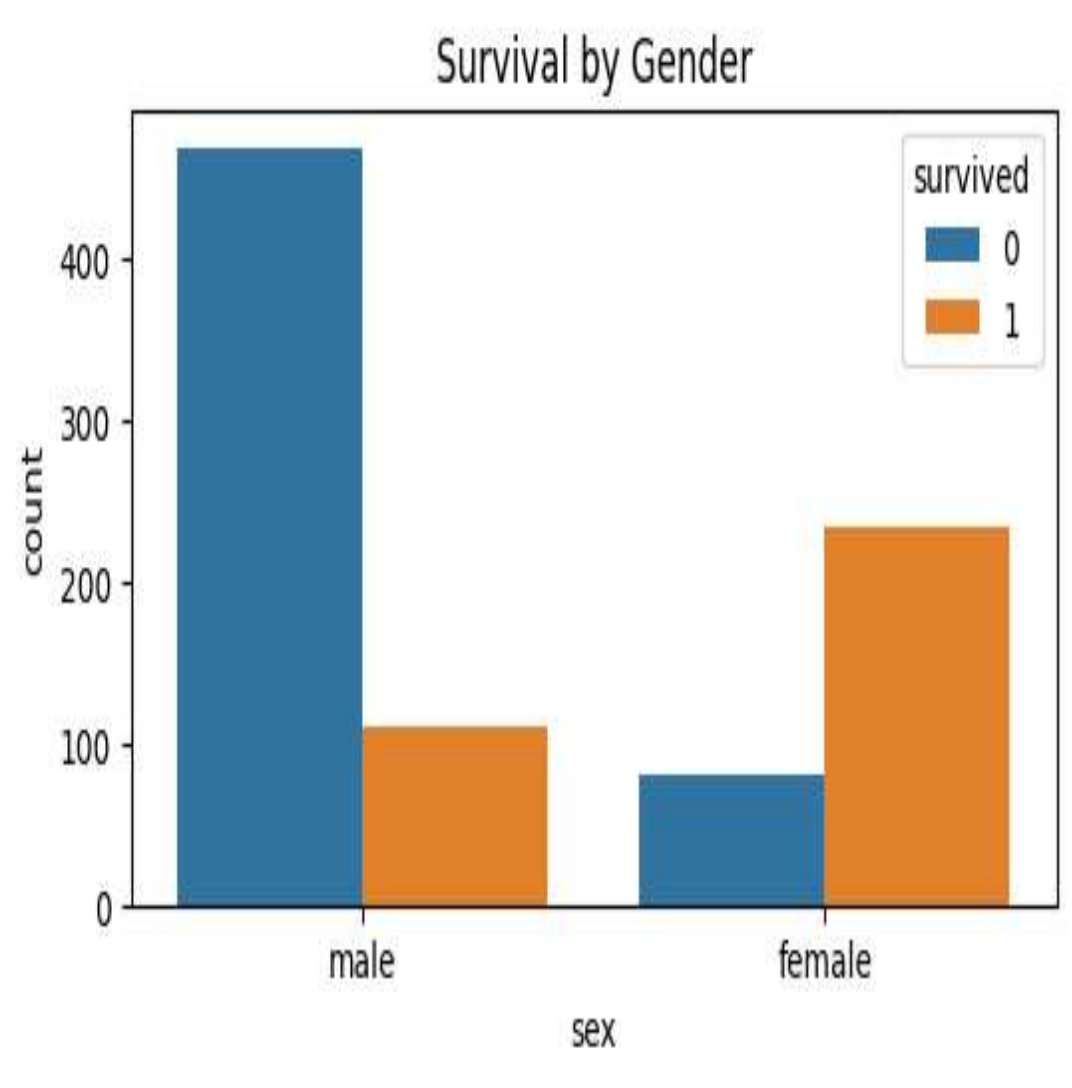
Example

- import seaborn as sns
- import matplotlib.pyplot as plt
- %matplotlib inline
- import pandas as pd
- **# Load dataset**
- titanic = sns.load_dataset("titanic")
- plt.figure(figsize=(6,3),dpi=100)
- sns.countplot(x='survived', data=tit)
- plt.title('Survival Count (0 = No, 1 = Yes)')
- plt.show()



Example

- import seaborn as sns
- import matplotlib.pyplot as plt
- %matplotlib inline
- import pandas as pd
- **# Load dataset**
- titanic = sns.load_dataset("titanic")
- plt.figure(figsize=(6,3),dpi=100)
- sns.countplot(x='sex', hue='survived', data=tit)
- plt.title('Survival by Gender')
- plt.show()



Box Plot

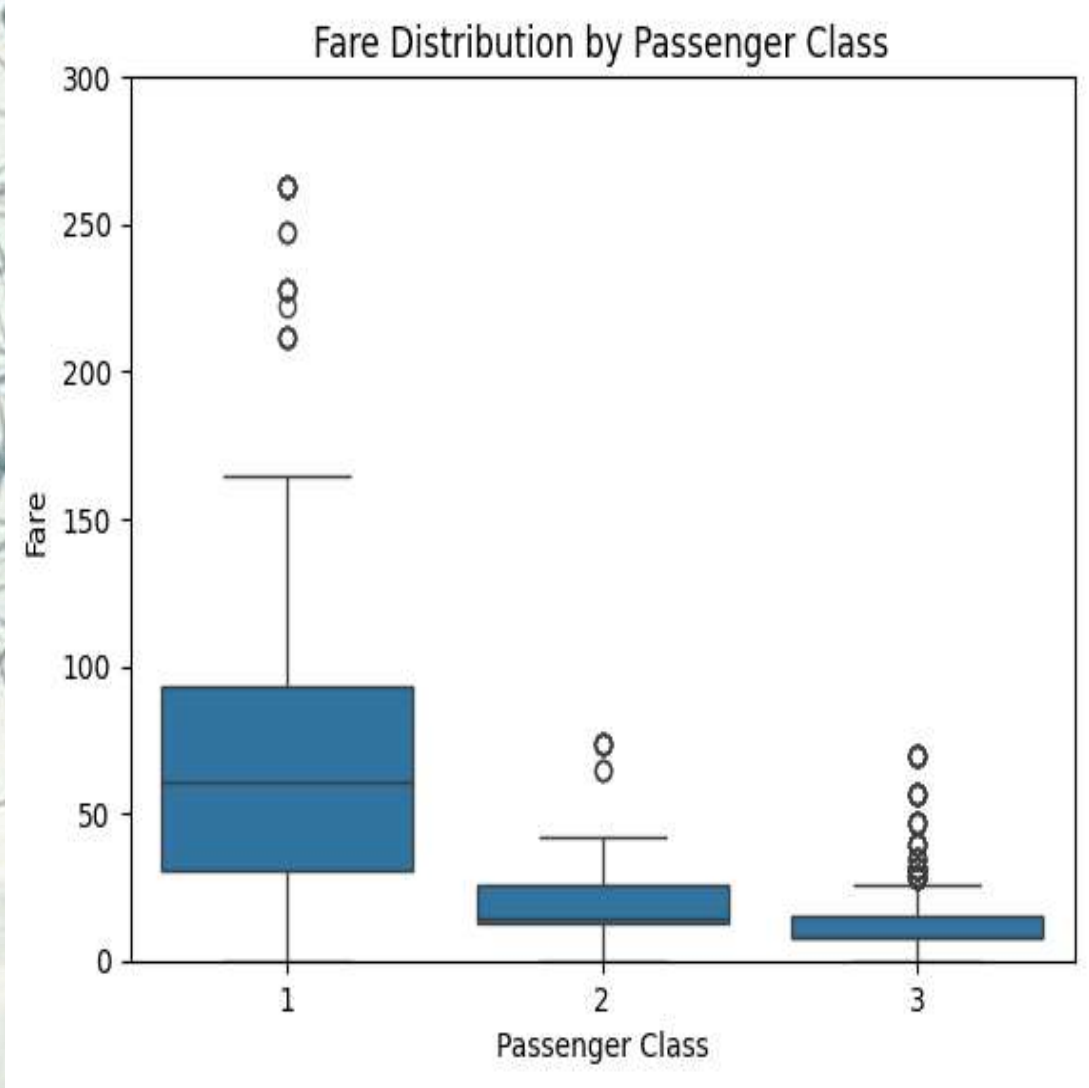
- **Purpose**
  - Summarize the distribution of a continuous variable, showing quartiles, median, and outliers.
- **Use Case:**
  - Identifying outliers in features or model predictions. Comparing feature distributions across classes or groups.
  - Visualizing model performance metrics (e.g., accuracy, F1-score) across folds or models.
- **Key Features:**
  - Automatically detects outliers based on interquartile range (IQR).
  - Supports hue for side-by-side comparisons across categories.
  - Customizable whisker length and box width.
  - Can display individual data points with showfliers=True.

Box Plot

- **Syntax:**
  - sns.boxplot(x="category", y="value", data=df, hue="group", showfliers=True)
- **Key Attribute:**
  - **x, y:** Column names for categorical (x) and continuous (y) variables (e.g., class, feature value).
  - **data:** DataFrame containing the data.
  - **hue:** Column name for side-by-side boxes by category (e.g., models or subgroups).
  - **showfliers:** Boolean to display outliers (useful for anomaly detection).

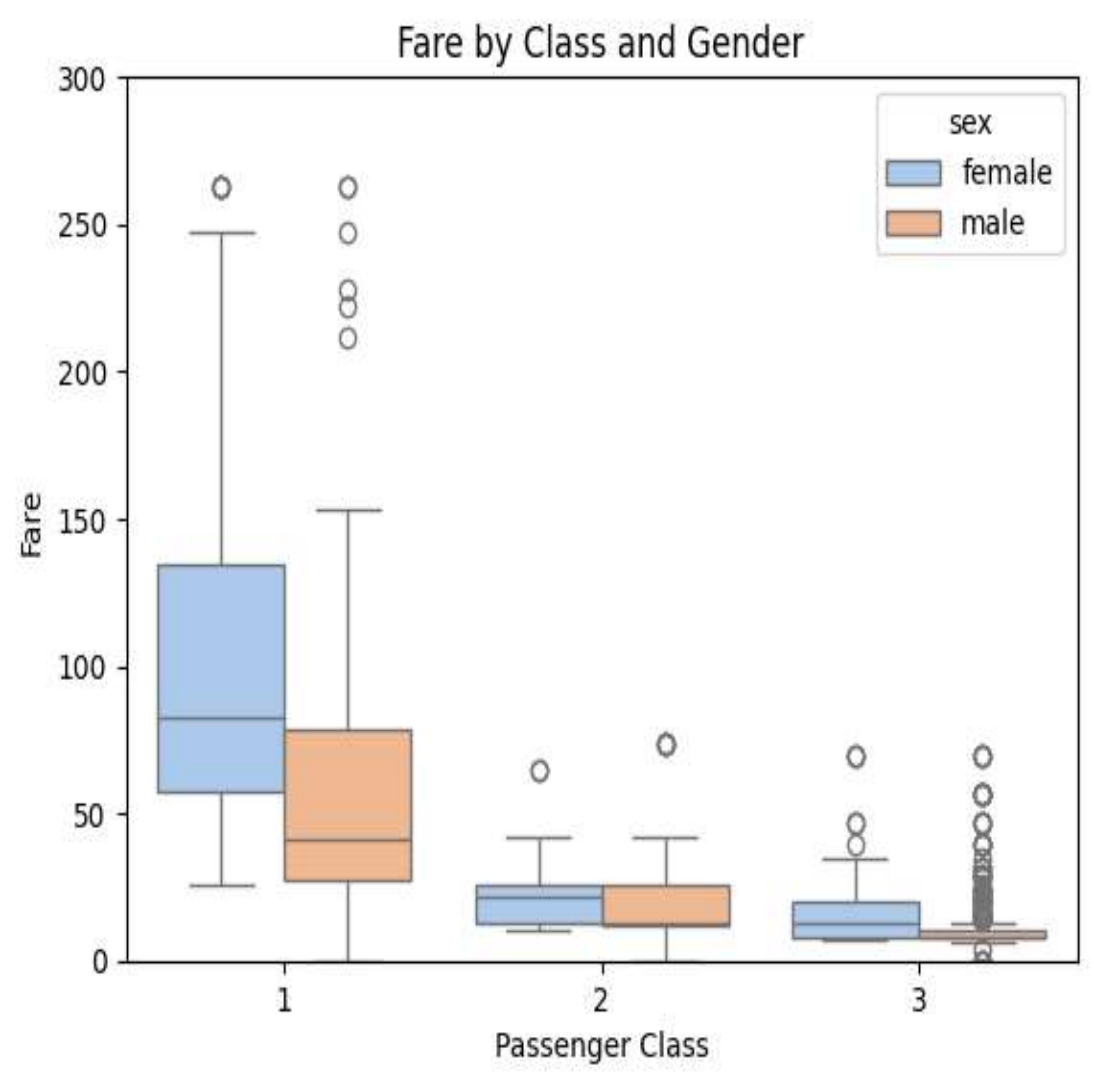
Example

- import seaborn as sns
- import matplotlib.pyplot as plt
- %matplotlib inline
- import pandas as pd
- **# Load dataset**
- titanic = sns.load_dataset("titanic")
- plt.figure(figsize=(6,3),dpi=50)
- sns.countplot(x='pclass', hue='survived', data=tit)
- plt.title('Survival Rate across Passenger Classes')
- plt.show()



Example

- import seaborn as sns
- import matplotlib.pyplot as plt
- %matplotlib inline
- import pandas as pd
- **# Load dataset**
- titanic = sns.load_dataset("titanic")
- sns.boxplot(data=titanic, x='pclass', y='fare', hue='sex', palette='pastel')
- plt.title('Fare by Class and Gender')
- plt.xlabel('Passenger Class')
- plt.ylabel('Fare')
- plt.ylim(0, 300)
- plt.show()



Violin Plot

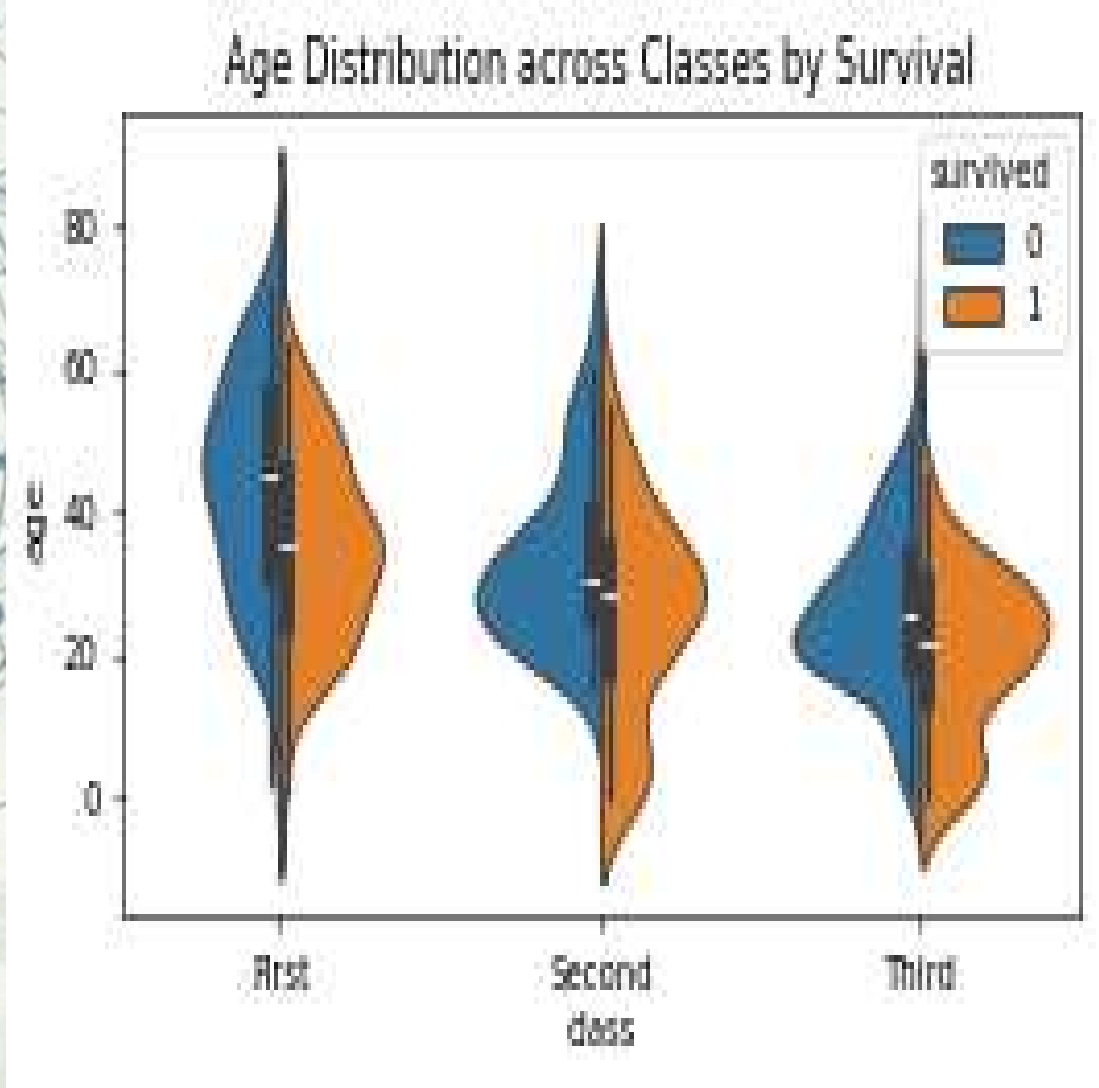
- **Purpose:**
  - Combine a box plot with a kernel density plot to show the full distribution of data.
- **Use Case:**
  - Visualizing feature distributions across classes or groups with more detail than box plots.
  - Comparing model performance metrics across categories.
  - Exploring data density for classification or clustering tasks.
- **Key Features:**
  - Shows both distribution shape (via KDE) and summary statistics (via box plot).
  - Supports hue for split or grouped violins.
  - Customizable bandwidth for density estimation.
  - Option to show individual data points with inner="points".

Violin Plot

- **Syntax:**
  - sns.violinplot(x="category", y="value", data=df, hue="group", split=True, inner="quartiles")
- **Key Attribute:**
  - **x, y:** Column names for categorical (x) and continuous (y) variables.
  - **data:** DataFrame containing the data. hue: Column name for split or grouped violins (e.g., classes).
  - **split:** Boolean to split violins for two hue levels (compact comparison).
  - **inner:** Controls inner display (e.g., "box", "quartiles", "points").

Example

- import seaborn as sns
- import matplotlib.pyplot as plt
- %matplotlib inline
- import pandas as pd
- **# Load dataset**
- titanic = sns.load_dataset("titanic")
- plt.figure(figsize=(6,3),dpi=50)
- sns.violinplot(x='class', y='age', hue='survived', data=tit, split=True)
- plt.title('Age Distribution across Classes by Survival')
- plt.show()



Pair Plot

- **Purpose:**
  - Create a grid of scatter plots and histograms for pairwise relationships among multiple variables.
- **Use Case:**
  - Exploring correlations and relationships between all features in a dataset during EDA.
  - Identifying potential collinearity or redundant features.
  - Visualizing high-dimensional data for feature selection.
- **Key Features:**
  - 1 Automatically generates scatter plots for all variable pairs and histograms/KDE for diagonals.
  - Supports hue for coloring by categorical variables.
  - Customizable plot types on diagonal (e.g., histogram, KDE).
  - Option to focus on specific variables with vars parameter.

Pair Plot

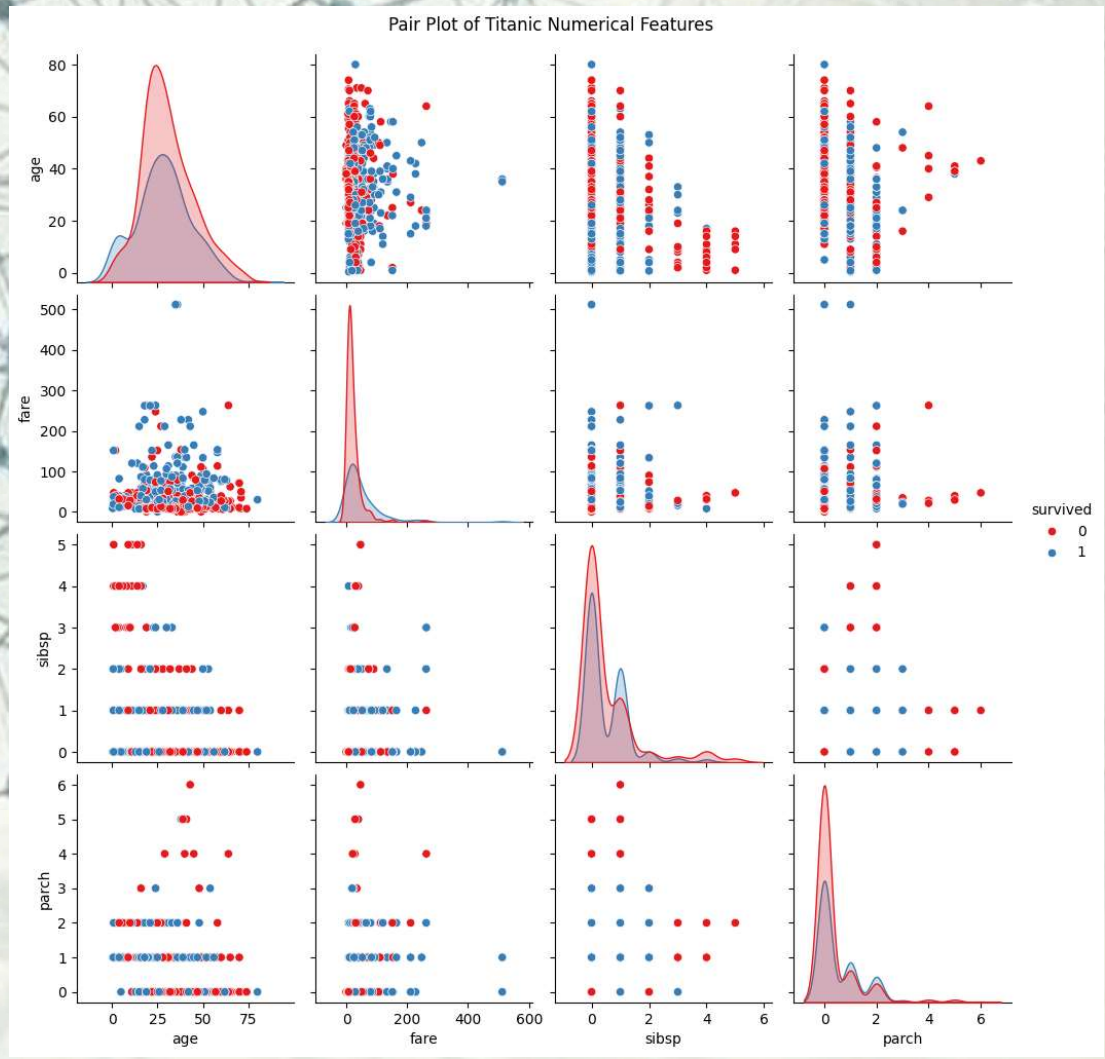
- **Syntax:**
  - sns.pairplot(data=df, hue="category", vars=["col1", "col2"], diag_kind="kde")
- **Key Attribute:**
  - **data:** DataFrame containing the data.
  - **hue:** Column name for coloring by category (e.g., class labels).
  - **vars:** List of column names to plot (limits variables for large datasets).
  - **diag_kind:** Diagonal plot type ("hist" or "kde").

Example

- import seaborn as sns
- import matplotlib.pyplot as plt
- %matplotlib inline
- import pandas as pd
- **# Load dataset**
- titanic = sns.load_dataset("titanic")
- plt.figure(figsize=(6,3),dpi=50)
- sns.pairplot(tit, hue='survived', diag_kind='kde')
- plt.suptitle('Pairplot of Titanic Dataset', y=1.02)
- plt.show()

Example

- import seaborn as sns
- import matplotlib.pyplot as plt
- %matplotlib inline
- import pandas as pd
- **# Load dataset**
- titanic = sns.load_dataset("titanic")
- plt.figure(figsize=(6,3),dpi=50)
- titanic_num = titanic[['age', 'fare', 'sibsp', 'parch', 'survived']].dropna()
- sns.pairplot(titanic_num, hue='survived', palette='Set1')
- plt.suptitle('Pair Plot of Titanic Numerical Features', y=1.02)
- plt.show()



Cat Plot

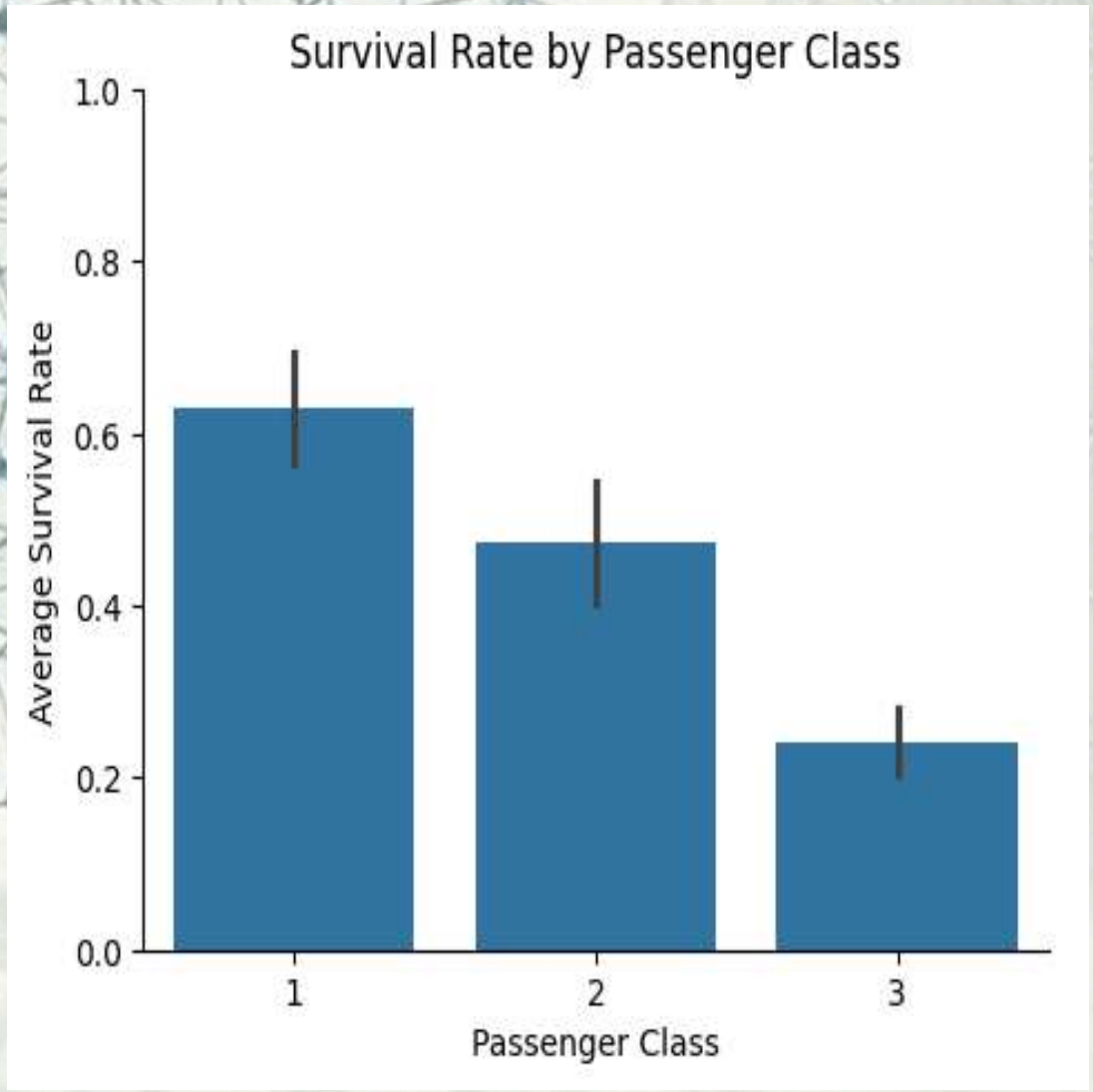
- **Purpose:**
  - Provide a flexible interface for visualizing categorical data with various plot types (e.g., box, violin, bar).
- **Use Case:**
  - Comparing feature distributions or model metrics across categories or groups.
  - Visualizing relationships in categorical data with faceting.
  - General-purpose plotting for categorical comparisons in EDA or evaluation..
- **Key Features:**
  - Supports multiple plot kinds (e.g., kind="box", "violin", "bar").
  - Integrates with hue, col, and row for multi-dimensional comparisons.
  - Faceting for creating subplots based on categorical variables.
  - Highly customizable with Seaborn’s styling options.

Cat Plot

- **Syntax:**
  - sns.catplot(x="category", y="value", hue="group", col="facet", kind="box", data=df)
- **Key Attribute:**
  - **x, y:** Column names for categorical (x) and continuous (y) variables.
  - **data:** DataFrame containing the data.
  - **hue:** Column name for nested categories.
  - **col, row**: Columns for faceting (creates subplots).
  - **kind:** Plot type ("box", "violin", "bar", etc.).

Example

- import seaborn as sns
- import matplotlib.pyplot as plt
- %matplotlib inline
- import pandas as pd
- **# Load dataset**
- titanic = sns.load_dataset("titanic")
- sns.catplot(data=titanic, x='pclass', y='survived', kind='bar', height=4, aspect=1.2)
- plt.title('Survival Rate by Passenger Class')
- plt.xlabel('Passenger Class')
- plt.ylabel('Average Survival Rate')
- plt.ylim(0, 1)
- plt.show()



Heatmap

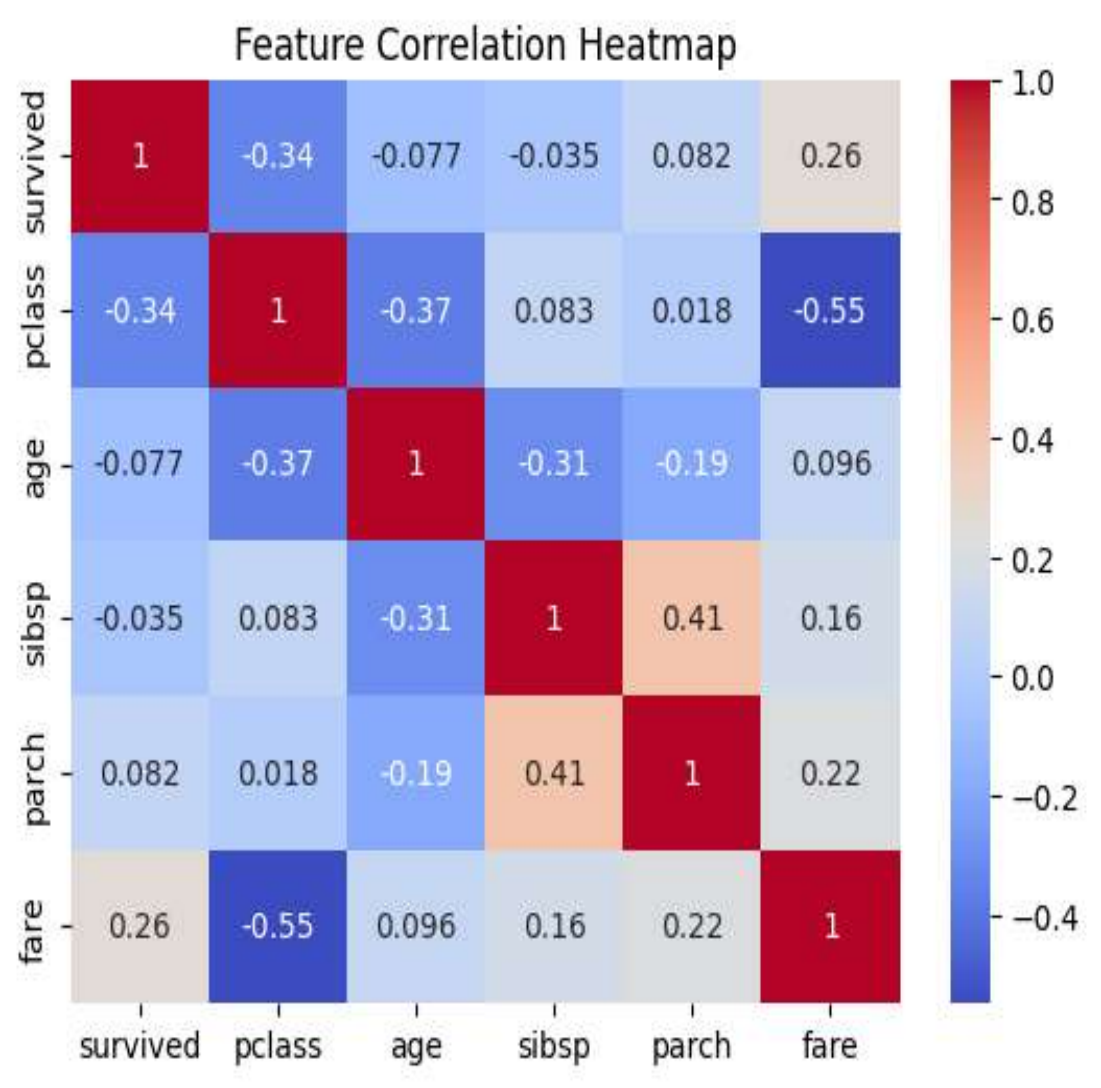
- **Purpose:**
  - Visualize a matrix of values using color intensity.
- **Use Case:**
  - Displaying feature correlation matrices to identify multicollinearity.
  - Visualizing confusion matrices for classification model evaluation.
  - Showing attention matrices in transformer models or similarity matrices in clustering.
- **Key Features:**
  - Customizable color maps (cmap) for better visualization.
  - Supports annotations (annot=True) to display matrix values.
  - Options for scaling (e.g., vmin, vmax) and clustering.
  - Handles missing data with optional masking.

Heatmap

- **Syntax:**
  - sns.heatmap(data=corr_matrix, annot=True, cmap="coolwarm", vmin=-1, vmax=1)
- **Key Attribute:**
  - **data:** 2D array or DataFrame (e.g., correlation matrix, confusion matrix).
  - **annot:** Boolean to display values in cells (enhances readability).
  - **cmap:** Color map for visualization (e.g., "coolwarm", "viridis").
  - **vmin, vmax:** Minimum/maximum values for color scaling.

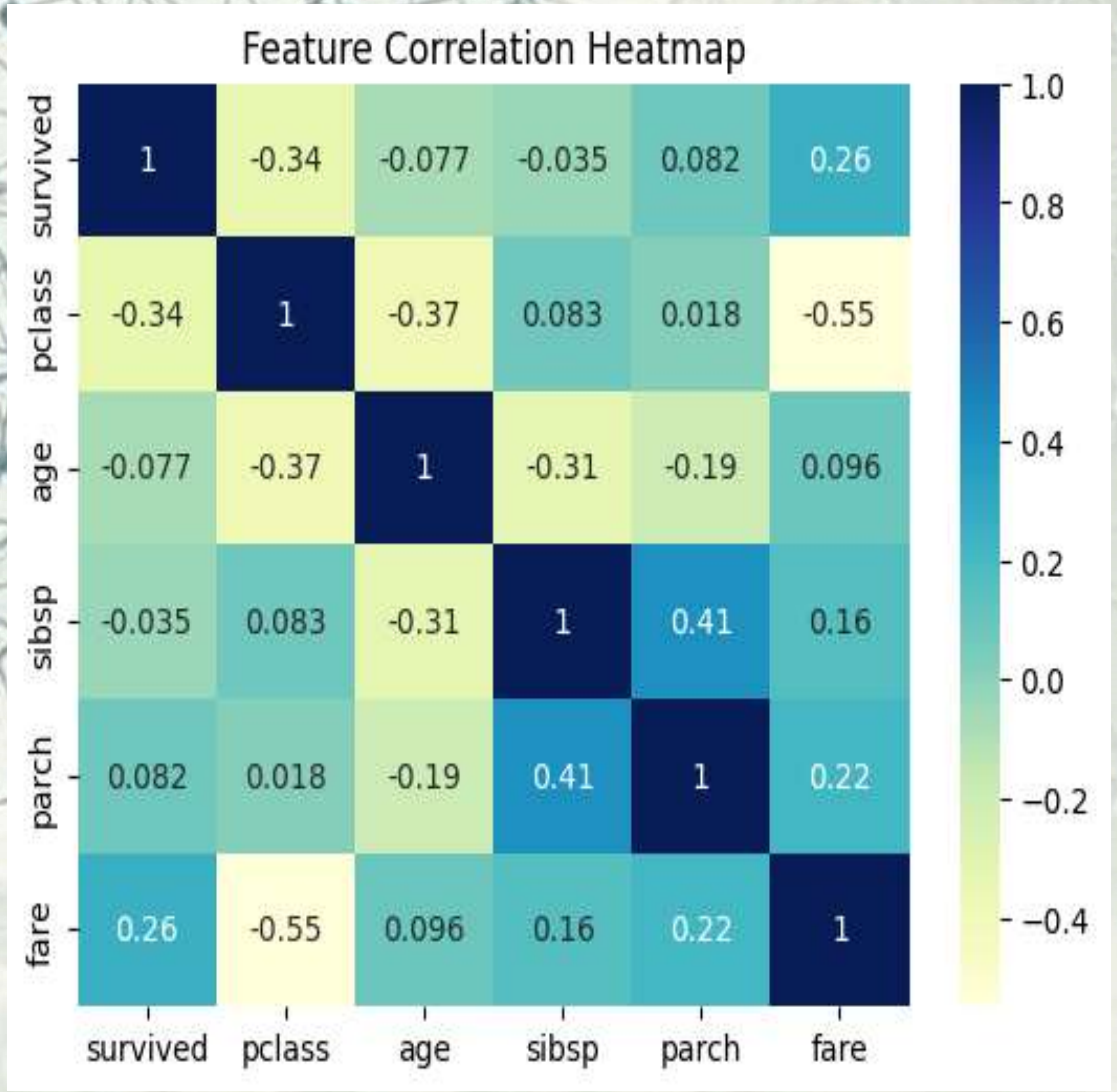
Example

```
import seaborn as sns
import matplotlib.pyplot as plt
%matplotlib inline
import pandas as pd
# Load dataset
titanic = sns.load_dataset("titanic")
numerical_titanic = tit.select_dtypes(include=['float64', 'int64'])
corr = numerical_titanic.corr()
sns.heatmap(corr, annot=True, cmap='coolwarm')
plt.title('Feature Correlation Heatmap')
plt.show()
```



Example

- import seaborn as sns
- import matplotlib.pyplot as plt
- %matplotlib inline
- import pandas as pd
- **# Load dataset**
- titanic = sns.load_dataset("titanic")
- numerical_titanic = tit.select_dtypes(include=['float64', 'int64'])
- corr = numerical_titanic.corr()
- sns.heatmap(corr, annot=True, cmap=YlGnBu')
- plt.title('Feature Correlation Heatmap')
- plt.show()



Strip Plot

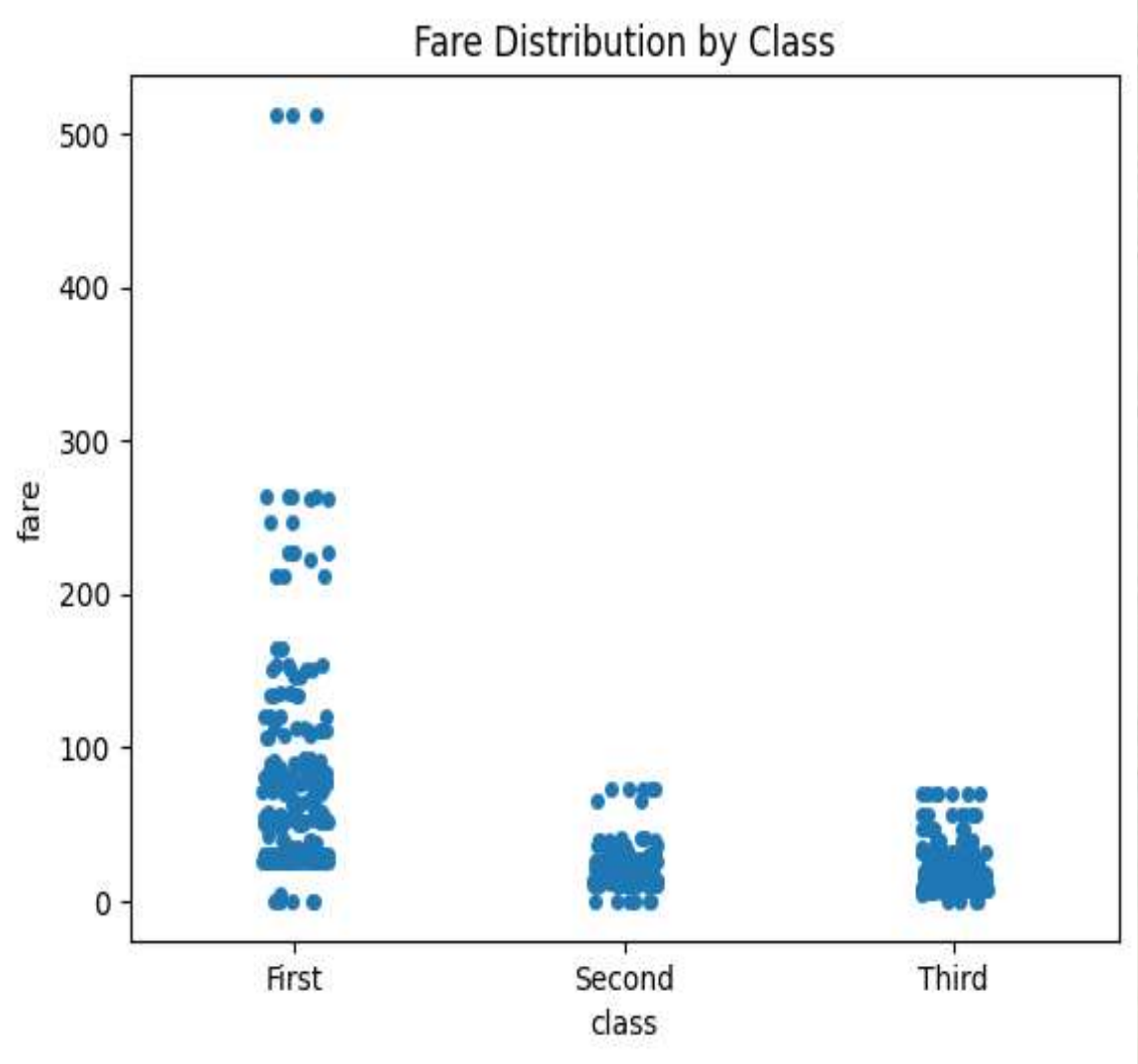
- **Purpose:**
  - Creates a scatter plot where one axis is categorical, using jitter to reduce overplotting of points.
- **Use Case:**
  - Visualizing the distribution of a numerical variable across categorical groups (e.g., test scores by class).
  - Showing individual observations alongside summary plots like box or violin plots.
  - Suitable for small to medium datasets where individual points are interpretable.
- **Key Features:**
  - Uses jitter to randomly offset points along the categorical axis, reducing overlap.
  - Supports hue, size, and dodge for multidimensional relationships (e.g., color by group, split by another variable).
  - Can be combined with sns.boxplot() or sns.violinplot() for richer visualizations.

Strip Plot

- **Syntax:**
  - sns.stripplot(data=None, x=None, y=None, hue=None, order=None, hue_order=None, jitter=True, dodge=False, size=5, palette=None, ax=None, **kwargs)
- **Key Attribute:**
  - **x, y:** One is categorical (e.g., x="day"), the other numerical (e.g., y="total_bill").
  - **hue:** Categorical variable for coloring points.
  - **jitter:** True or float (e.g., 0.1) to add random offset to categorical axis; False for no jitter.
  - **dodge:** If True, separates hue categories along the categorical axis. size: Point size (default: 5).
  - **palette:** Color palette for hue.
  - **order:** List to set categorical axis order (e.g., ["Thur", "Fri", "Sat", "Sun"]).
  - **col, row (for catplot):** Facet variables for subplots.

Example

- import seaborn as sns
- import matplotlib.pyplot as plt
- %matplotlib inline
- import pandas as pd
- **# Load dataset**
- titanic = sns.load_dataset("titanic")
- sns.stripplot(data=titanic, x="class", y="fare")
- plt.title("Fare Distribution by Class")
- plt.show()



Example

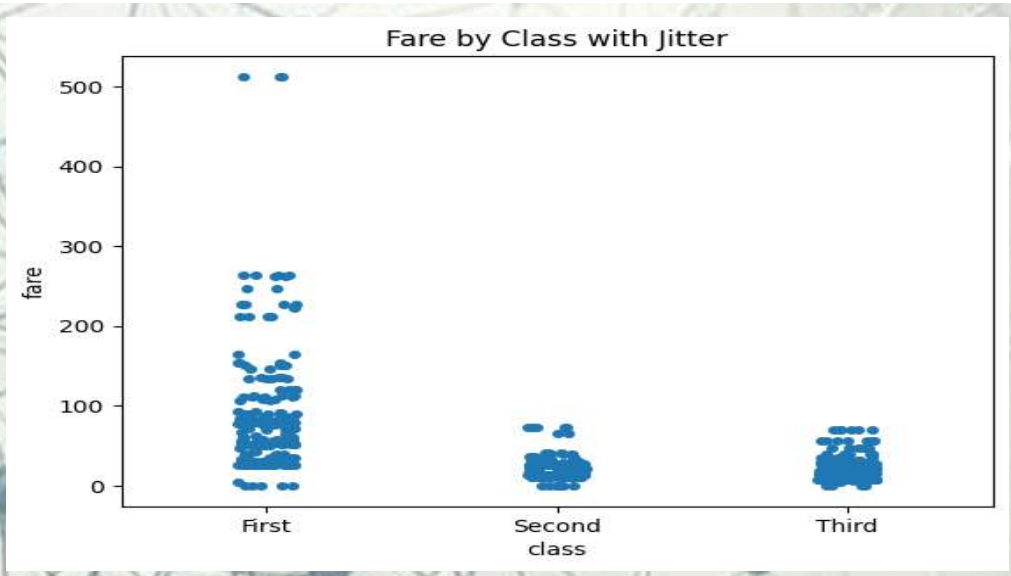
- import seaborn as sns
- import matplotlib.pyplot as plt
- %matplotlib inline
- import pandas as pd
- **# Load dataset**
- titanic = sns.load_dataset("titanic")
- **#Add Hue for Gender**
- sns.stripplot(data=titanic, x="class", y="fare", hue="sex")
- plt.title("Fare by Class and Sex")
- plt.show()

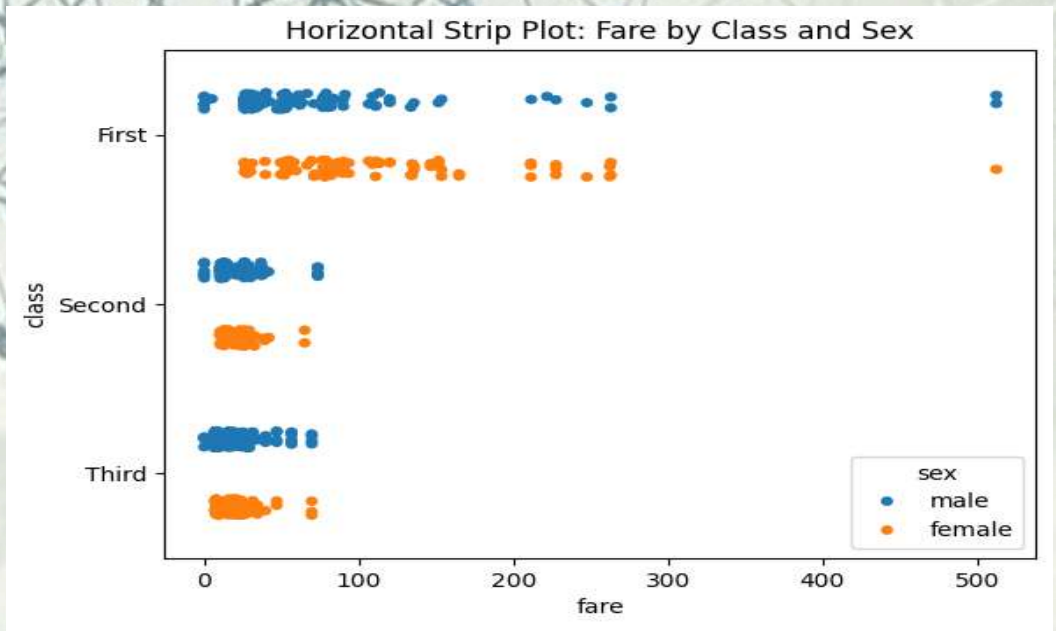
Example

- import seaborn as sns
- import matplotlib.pyplot as plt
- %matplotlib inline
- import pandas as pd
- **# Load dataset**
- titanic = sns.load_dataset("titanic")
- **#Add Jitter to Avoid Overlapping Points**
- sns.stripplot(data=titanic, x="class", y="fare", jitter=True)
- plt.title("Fare by Class with Jitter")
- plt.show()

OR

```
sns.stripplot(data=titanic, y="class", x="fare", hue="sex", jitter=True, dodge=True)
plt.title("Horizontal Strip Plot: Fare by Class and Sex")
plt.show()
```





Swarm Plot

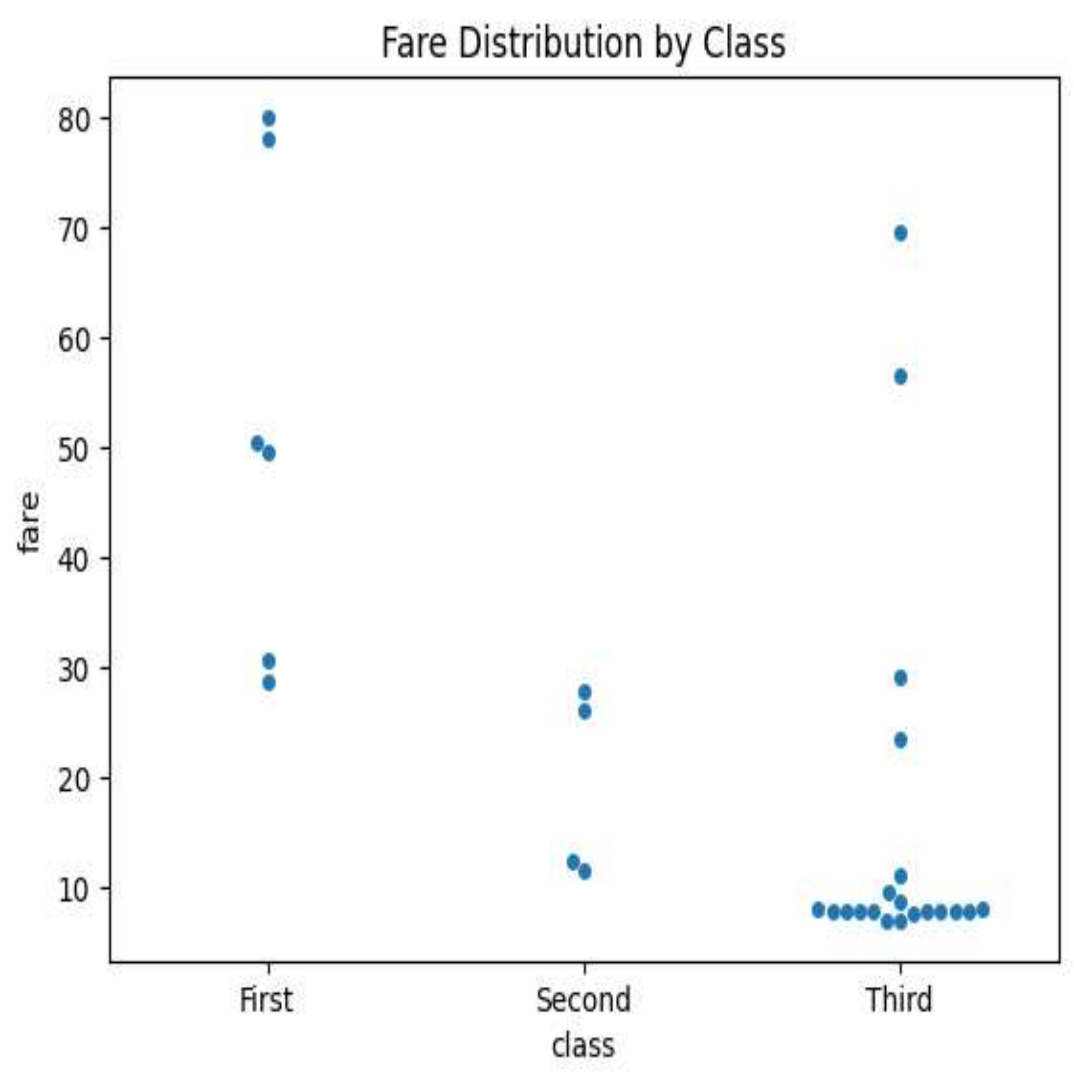
- **Purpose:**
  - Similar to a strip plot but arranges points along the categorical axis to avoid overlap, creating a "beeswarm" effect.
- **Use Case:**
  - Visualizing the distribution and density of numerical data across categories (e.g., customer spend by store location).
  - Highlighting individual data points in small to medium datasets where overlap is a concern.
  - Enhancing box or violin plots to show both distribution and individual observations.
- **Key Features:**
  - Points are adjusted non-randomly to prevent overlap, improving clarity over strip plots.
  - Scales poorly with large datasets due to space constraints; warnings issued if points cannot be placed.
  - Supports hue, dodge, and size for customization.

Swarm Plot

- **Syntax:**
  - sns.swarmplot(data=None, x=None, y=None, hue=None, order=None, hue_order=None, dodge=False, size=5, palette=None, ax=None, warn_thresh=0.05, **kwargs)
  - sns.catplot(data=None, x=None, y=None, hue=None, kind="swarm", dodge=False, size=5, col=None, row=None, palette=None, **kwargs)
- **Key Attribute:**
  - x, y: One is categorical (e.g., x="day"), the other numerical (e.g., y="total_bill").
  - hue: Categorical variable for coloring points.
  - dodge: If True, separates hue categories along the categorical axis.
  - size: Point size (default: 5).
  - palette: Color palette for hue.
  - order: List to set categorical axis order.
  - warn_thresh: Threshold for warning about too many points (default: 0.05).
  - col, row (for catplot): Facet variables for subplots.

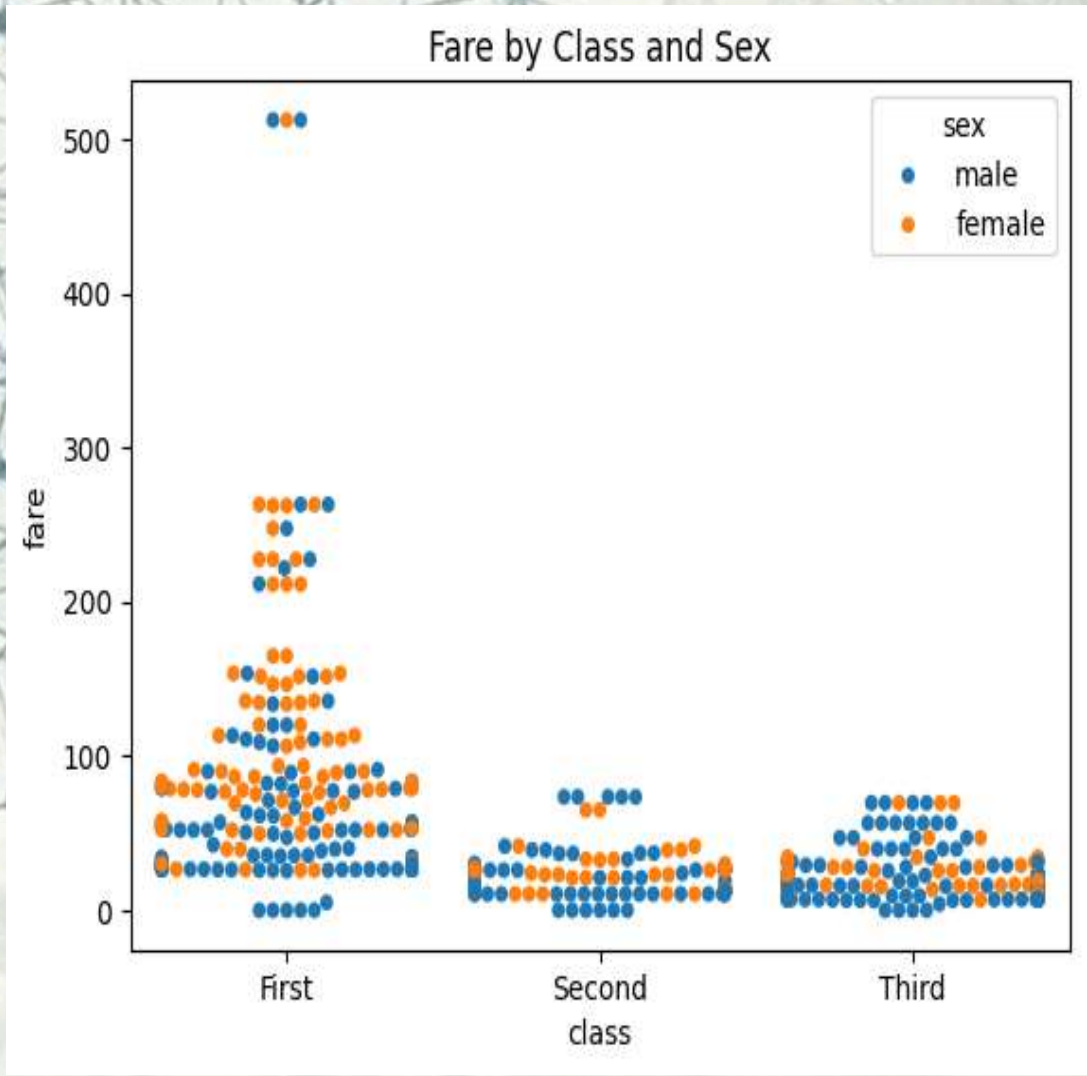
Example

- import seaborn as sns
- import matplotlib.pyplot as plt
- %matplotlib inline
- import pandas as pd
- **# Load dataset**
- titanic = sns.load_dataset("titanic")
- sns.swarmplot(data=titanic.sample(30) , x="class", y="fare")
- plt.title("Fare Distribution by Class")
- plt.show()



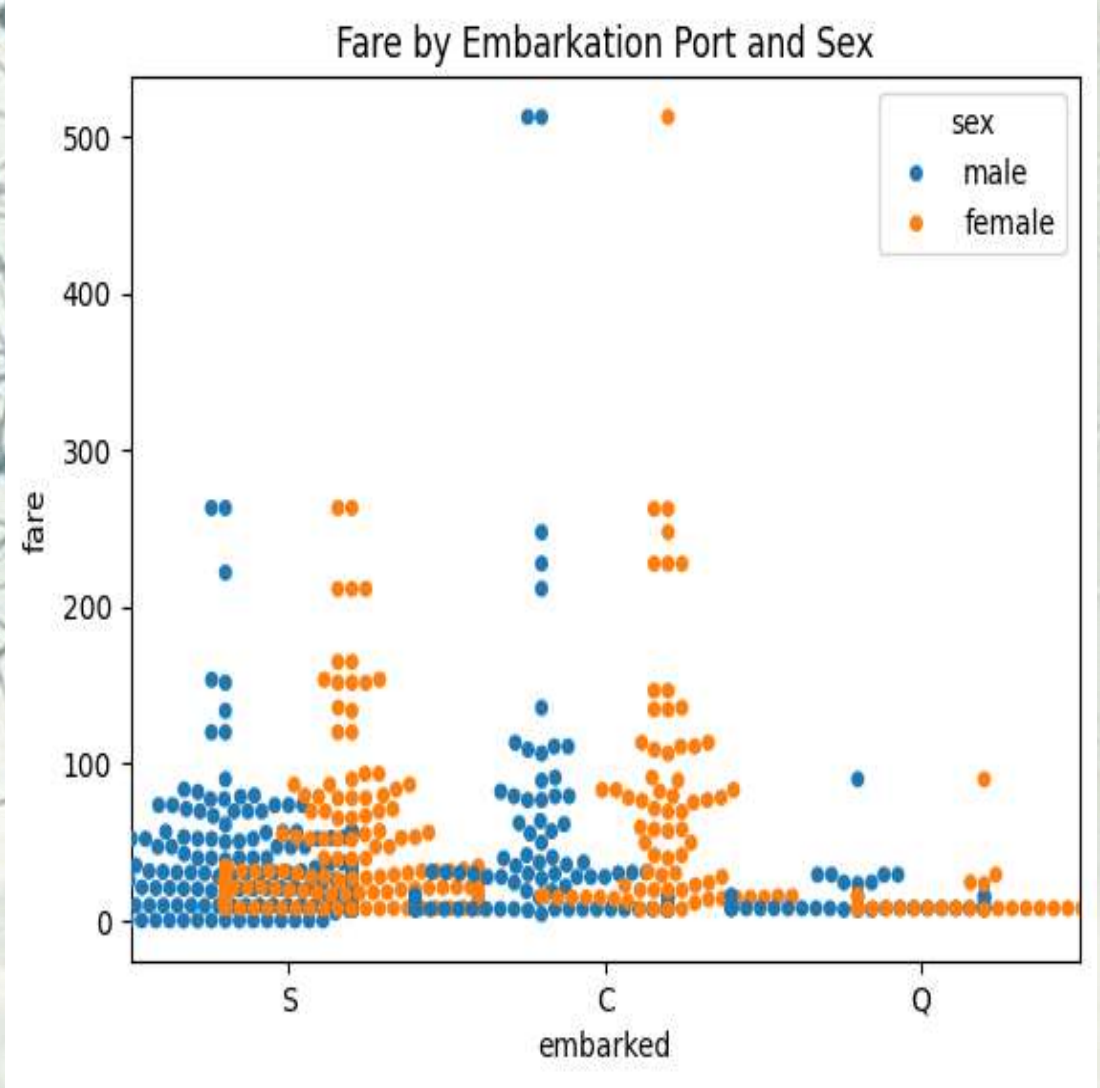
Example

- import seaborn as sns
- import matplotlib.pyplot as plt
- %matplotlib inline
- import pandas as pd
- **# Load dataset**
- titanic = sns.load_dataset("titanic")
- sns.swarmplot(data=titanic, x="class", y="fare", hue="sex")
- plt.title("Fare by Class and Sex")
- plt.show()



Example

- import seaborn as sns
- import matplotlib.pyplot as plt
- %matplotlib inline
- import pandas as pd
- **# Load dataset**
- titanic = sns.load_dataset("titanic")
- sns.swarmplot(data=titanic, x="embarked", y="fare", hue="sex", dodge=True)
- plt.title("Fare by Embarkation Port and Sex")
- plt.show()



Point Plot

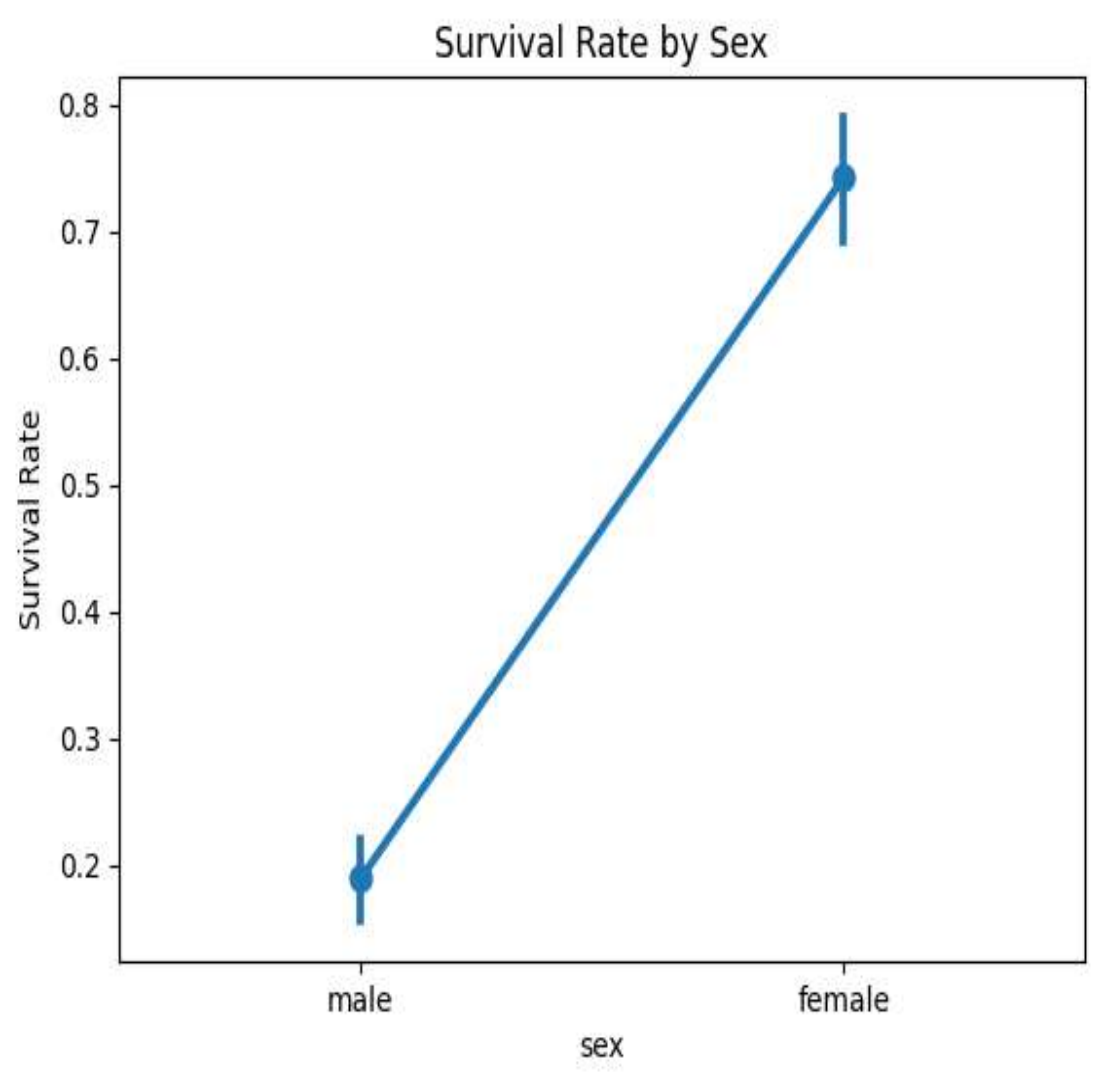
- **Purpose:**
  - Shows point estimates (e.g., mean) and confidence intervals for a numerical variable across categorical groups.
- **Use Case:**
  - Comparing central tendencies (e.g., average sales by month) across categories.
  - Summarizing data when individual points are less important than group-level statistics.
  - Visualizing trends or differences in aggregated data (e.g., mean test scores by school).
- **Key Features:**
  - Plots point estimates (default: mean) with error bars (default: 95% confidence intervals).
  - Supports estimator (e.g., median), errorbar (e.g., standard deviation), and hue for grouping.
  - Accessible via sns.catplot(kind="point") or sns.pointplot().

Point Plot

- **Syntax:**
  - sns.pointplot(data=None, x=None, y=None, hue=None, order=None, hue_order=None, estimator="mean", errorbar="ci", n_boot=1000, units=None, dodge=False, palette=None, ax=None, **kwargs)
  - sns.catplot(data=None, x=None, y=None, hue=None, kind="point", estimator="mean", errorbar="ci", dodge=False, col=None, row=None, palette=None, **kwargs)
- **Key Attribute:**
  - **x, y:** One is categorical (e.g., x="day"), the other numerical (e.g., y="total_bill").
  - **hue:** Categorical variable for coloring points/lines.
  - **estimator:** Function for point estimate (e.g., "mean", "median", or custom function).
  - **errorbar:** Error bar type (e.g., "ci" for confidence interval, "sd" for standard deviation, None for no bars).
  - **n_boot:** Number of bootstrap iterations for confidence intervals (default: 1000).
  - **dodge:** If True, separates hue categories along the categorical axis.
  - **palette:** Color palette for hue.
  - **order:** List to set categorical axis order.
  - **col, row (for catplot):** Facet variables for subplots.

Example

- import seaborn as sns
- import matplotlib.pyplot as plt
- %matplotlib inline
- import pandas as pd
- **# Load dataset**
- titanic = sns.load_dataset("titanic")
- sns.pointplot(data=titanic, x="sex", y="survived")
- plt.title("Survival Rate by Sex")
- plt.ylabel("Survival Rate")
- plt.show()



Example

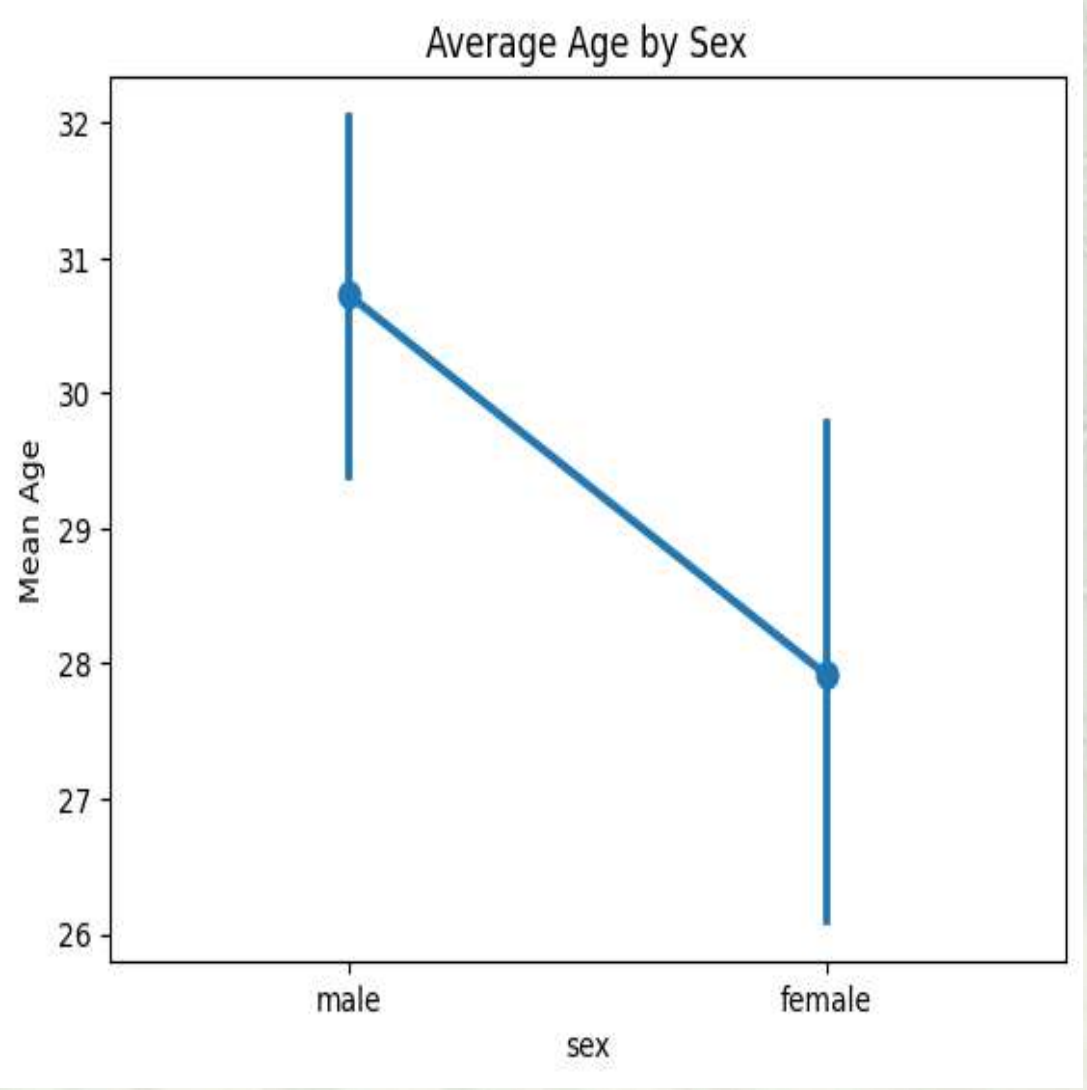
- import seaborn as sns
- import matplotlib.pyplot as plt
- %matplotlib inline
- import pandas as pd
- **# Load dataset**
- titanic = sns.load_dataset("titanic")
- sns.pointplot(data=titanic, x="class", y="survived")
- plt.title("Survival Rate by Class")
- plt.ylabel("Survival Rate")
- plt.show()

Example

- import seaborn as sns
- import matplotlib.pyplot as plt
- %matplotlib inline
- import pandas as pd
- **# Load dataset**
- titanic = sns.load_dataset("titanic")
- sns.pointplot(data=titanic, x="class", y="survived", hue="sex", dodge=True)
- plt.title("Survival Rate by Class and Sex")
- plt.ylabel("Survival Rate")
- plt.show()

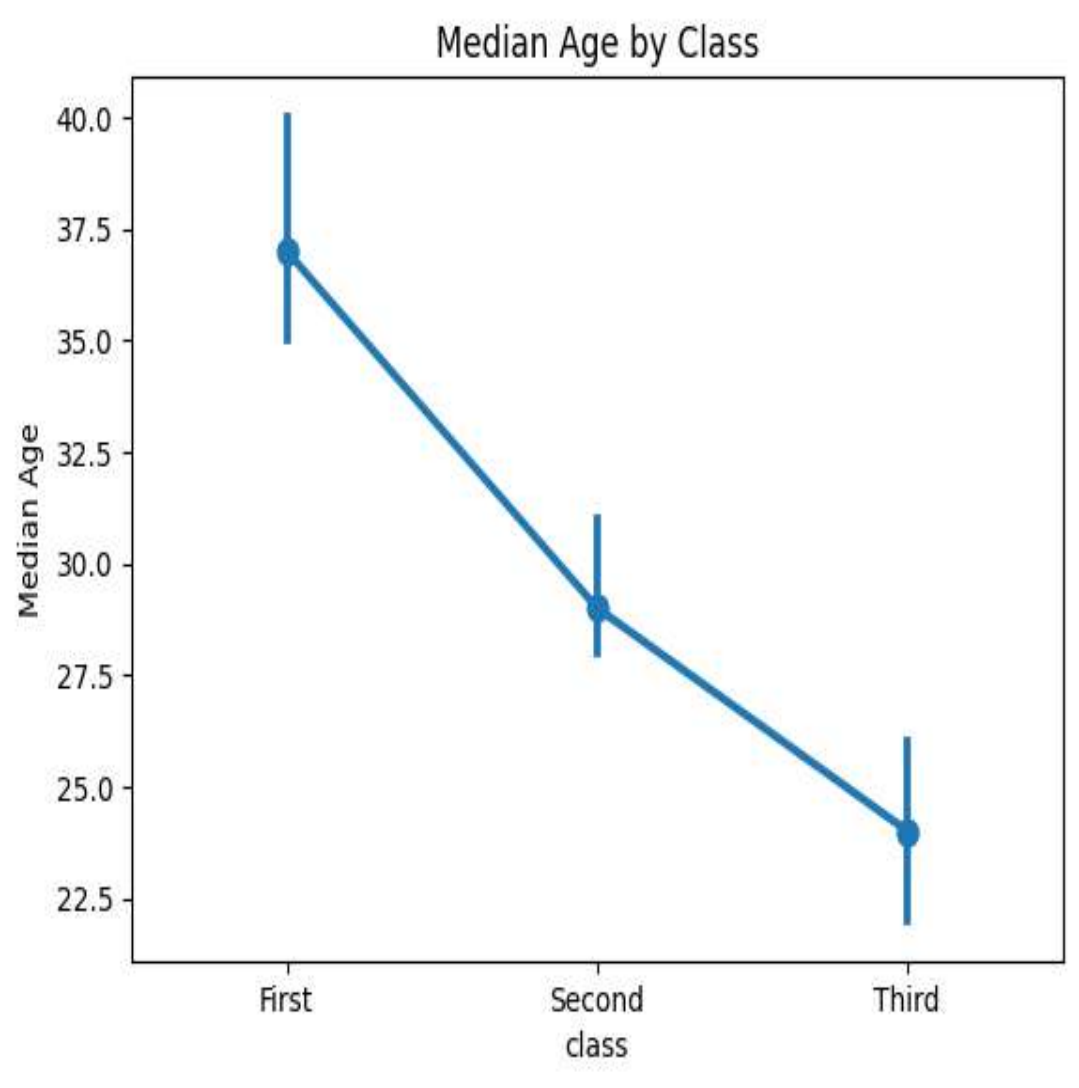
Example

- import seaborn as sns
- import matplotlib.pyplot as plt
- %matplotlib inline
- import pandas as pd
- **# Load dataset**
- titanic = sns.load_dataset("titanic")
- sns.pointplot(data=titanic, x="sex", y="age")
- plt.title("Average Age by Sex")
- plt.ylabel("Mean Age")
- plt.show()



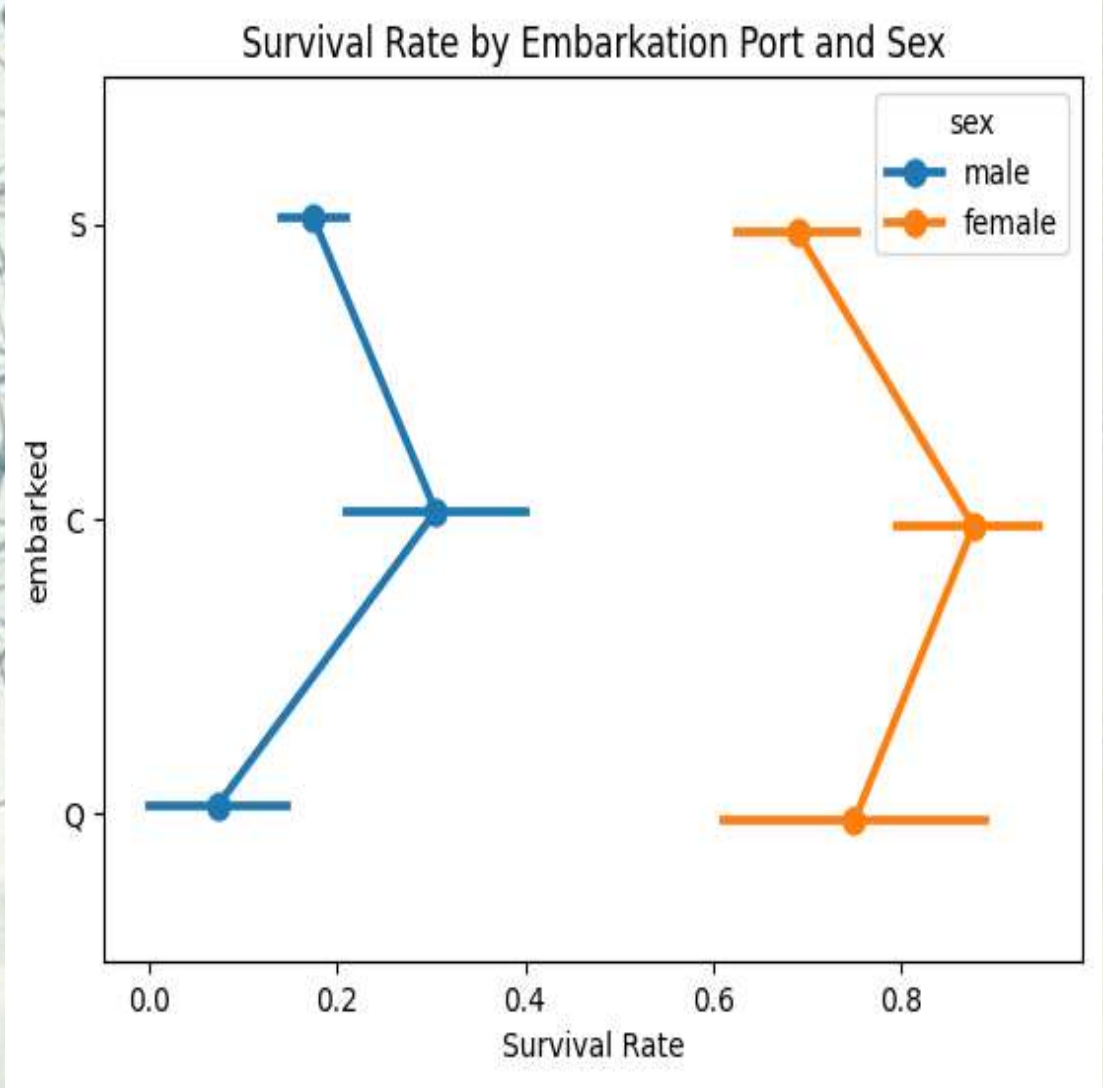
Example

- import seaborn as sns
- import matplotlib.pyplot as plt
- %matplotlib inline
- import pandas as pd
- **# Load dataset**
- titanic = sns.load_dataset("titanic")
- sns.pointplot(data=titanic, x="class", y="age", estimator=np.median)
- plt.title("Median Age by Class")
- plt.ylabel("Median Age")
- plt.show()



Example

- import seaborn as sns
- import matplotlib.pyplot as plt
- %matplotlib inline
- import pandas as pd
- **# Load dataset**
- titanic = sns.load_dataset("titanic")
- sns.pointplot(data=titanic, y="embarked", x="survived", hue="sex", dodge=True)
- plt.title("Survival Rate by Embarkation Port and Sex")
- plt.xlabel("Survival Rate")
- plt.show()



Joint Plot

- **Purpose:**
  - Combine a scatter plot with marginal histograms or KDE plots for two continuous variables.
- **Use Case:**
  - Analyzing bivariate relationships with marginal distributions.
  - Visualizing feature vs. target relationships for regression or classification.
  - Comparing predicted vs. actual values with distribution context.
- **Key Features:**
  - Supports multiple plot types (e.g., scatter, hexbin, KDE, regression).
  - Customizable marginal plots (histogram, KDE, rug).
  - Option to add regression lines with kind="reg".
  - Supports hue for categorical differentiation.

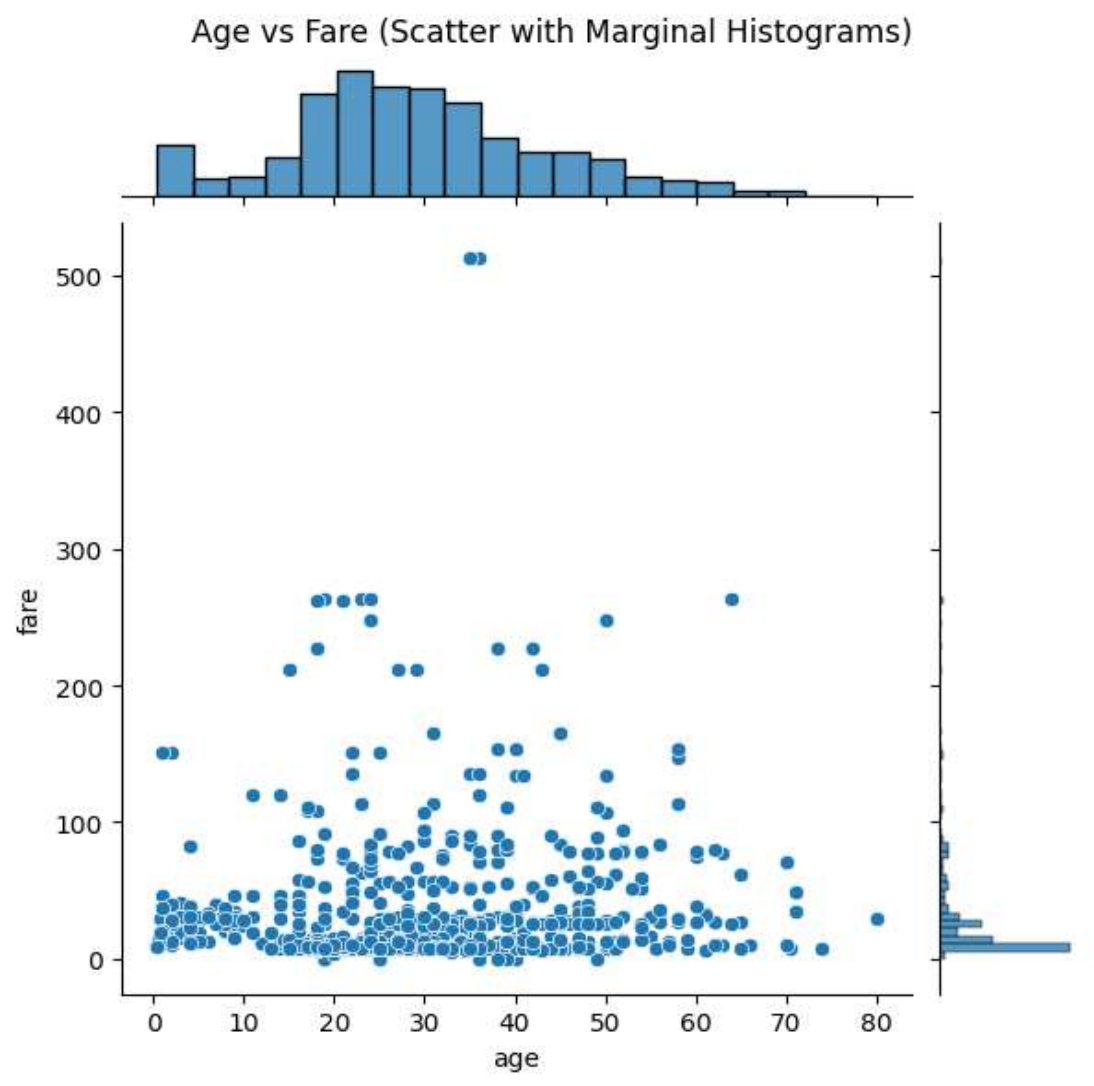
Joint Plot

- **Syntax:**
  - sns.jointplot(x="col1", y="col2", data=df, kind="scatter", hue="category")
- **Key Attribute:**
  - **x, y:** Column names for continuous variables (e.g., feature, target).
  - **data:** DataFrame containing the data.
  - **kind:** Plot type ("scatter", "kde", "hex", "reg").
  - **hue:** Column name for coloring by category (e.g., classes).

Example

- import seaborn as sns
- import matplotlib.pyplot as plt
- %matplotlib inline
- import pandas as pd
- **# Load dataset**
- titanic = sns.load_dataset("titanic")
- titanic_clean = titanic.dropna(subset=["age", "fare"])

- sns.jointplot(data=titanic_clean, x="age", y="fare")
- plt.suptitle("Age vs Fare (Scatter with Marginal Histograms)", y=1.02)
- plt.show()



Example

- import seaborn as sns
- import matplotlib.pyplot as plt
- %matplotlib inline
- import pandas as pd
- **# Load dataset**
- titanic = sns.load_dataset("titanic")
- sns.jointplot(data=titanic_clean, x="age", y="fare", kind="kde", fill=True, cmap="Blues")
- plt.suptitle("Age vs Fare (KDE with Marginals)", y=1.02)
- plt.show()