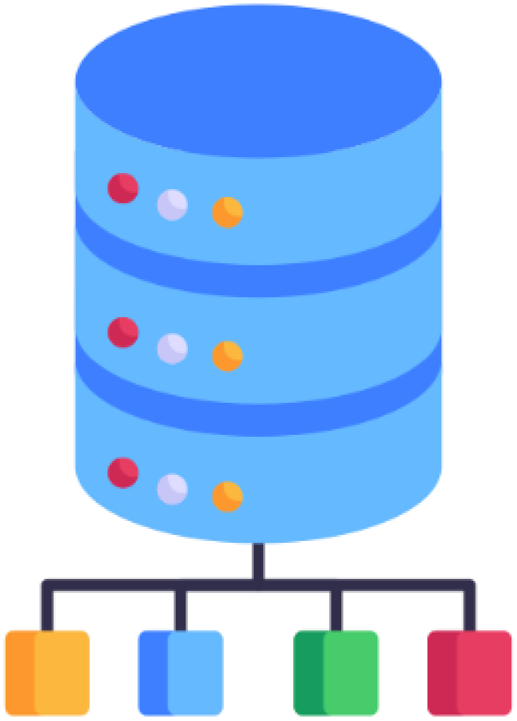
Exploratory Data Analysis (EDA)

Outline

- Features and Feature Types
- Dataset Types, Properties and Sources
- Data Exploration – Univariate
- Data Exploration - Bivariate
- Data Exploration - Multivariate

Features and Feature Types

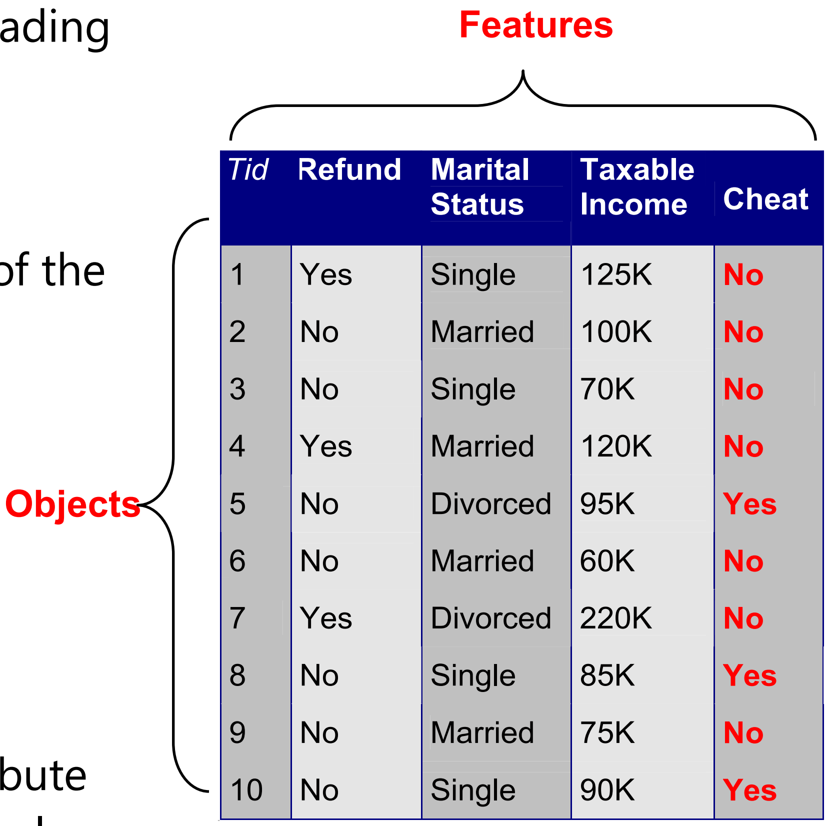


Steps for Doing Machine Learning

1. **Define Problem & Success Metrics**: Frame it as an ML task
2. **Acquire & Explore Data (EDA)**: Gather data and uncover initial insights
3. **Prepare Data & Engineer Features**: Clean, transform, and create predictive features
4. **Select Model & Establish Baseline**: Choose algorithms and a benchmark for comparison
5. **Train & Tune Model**: Train the model and optimize its hyperparameters
6. **Evaluate Final Model**: Test performance on unseen data against success metrics
7. **Deploy for Inference**: Integrate the model to make live predictions
8. **Monitor & Maintain**: Continuously track performance and retrain as needed

Anatomy of Data

- **Data object:** An individual sample or entity
  - e.g., a customer, a transaction, or a sensor reading
  - Also known as a record, instance, sample, or entity
- **Features:** A specific property or characteristic of the data object
  - e.g., Age, Income, Temperature
  - Also known as attribute, variable, field, or characteristic
- **Raw vs. Derived Features:**
  - **Raw:** collected or measured value of an attribute according to an appropriate measurement scale
  - **Derived:** constructed from data in one or more raw features (e.g., calculating "Debt-to-Income Ratio" from "Total Debt" and "Annual Income")


Features
Objects

Tid	Refund	Marital Status	Taxable Income	Cheat
1	Yes	Single	125K	No
2	No	Married	100K	No
3	No	Single	70K	No
4	Yes	Married	120K	No
5	No	Divorced	95K	Yes
6	No	Married	60K	No
7	Yes	Divorced	220K	No
8	No	Single	85K	Yes
9	No	Married	75K	No
10	No	Single	90K	Yes

Derived Features

- **Aggregates:** defined over a group or period, e.g., count, sum, average, minimum, or maximum of the values
- **Flags:** indicate presence or absence of some characteristic within a dataset, e.g., a flag indicating whether or not a bank account has been overdrawn
- **Ratios:** capture relationship between two or more raw data values, e.g., a ratio between a loan applicant's salary and the amount for which they are requesting
- **Mappings:** convert continuous features into categorical features, e.g., map the salary values to low, medium, and high
- **Others:** no restrictions to the ways in which we can combine data to make derived features, e.g., use satellite photos to count the number of cars in the parking lots and use this as a proxy measure of activity within a competitor's stores!

Goals for Derived Features

- To **improve** the accuracy and performance of ML models by transforming the raw data into a more meaningful representation that can better capture the underlying relationships in the data

- To help to **reduce** the dimensionality of a dataset and make it easier to visualize and understand the relationships between variables

Feature Types

Type	Subtype	Examples
Categorical (Qualitative)	**Nominal**	Product type, name
	Ordinal	Size measured as small<medium<large
	Binary	Spam email (yes/no, true/false, 0/1)
	Date / Time	Job start date
Numerical (Quantitative)	**Discrete** Countable values (integers)	Number of students in a class
	Continuous Measurable values (infinite decimals).	Height, weight, salary

Understanding the type of variables is crucial for selecting appropriate statistical methods, visualization techniques, and ML algorithms

Categorical Features

- **Categorical data are strings that represent qualitative data**
  - **Often selected from a group of categories, also called labels**
- **Nominal**, e.g., country of birth, gender, eye color, etc.
  - No inherent order or ranking
  - Operators applicable: **=, ≠**
  - 1:1 transformation permissible, e.g. ID: 974 ⇒ Qatar

- **Ordinal**, e.g. grade (A, B, C, D, F), degree (bachelor, master, PhD), height (tall, medium, short), etc.

  - Represent categories that can be meaningfully ordered
  - Operator applicable: **=, ≠, <, >, ≥, ≤**
  - Order-preserving transformation permitted,
  - e.g. height (tall, medium, short) to (1, 2, 3)

Nominal vs. Ordinal

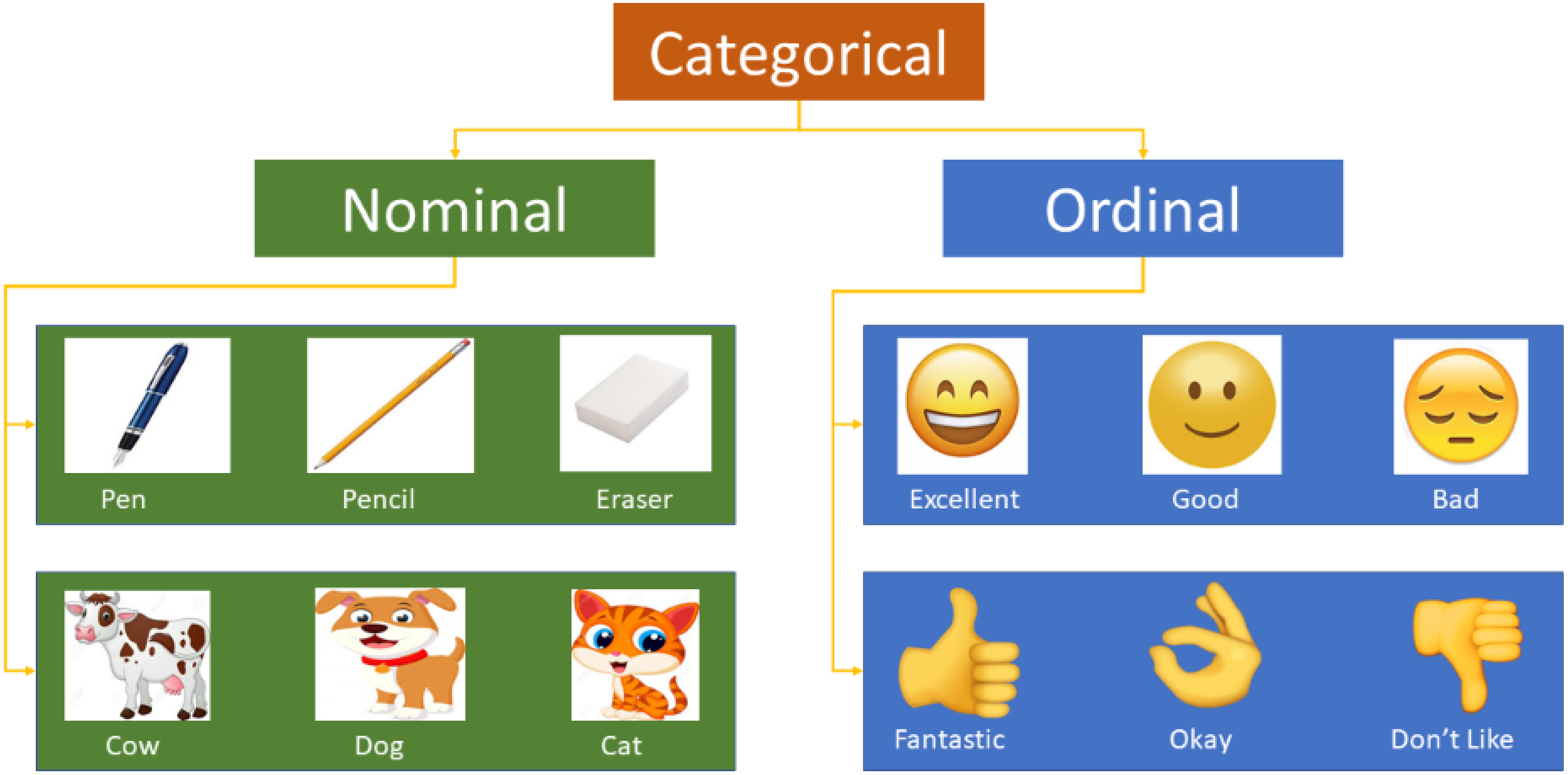
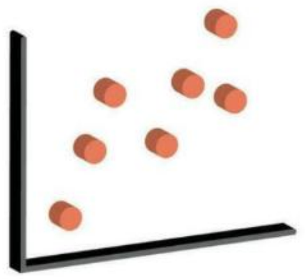


Image source

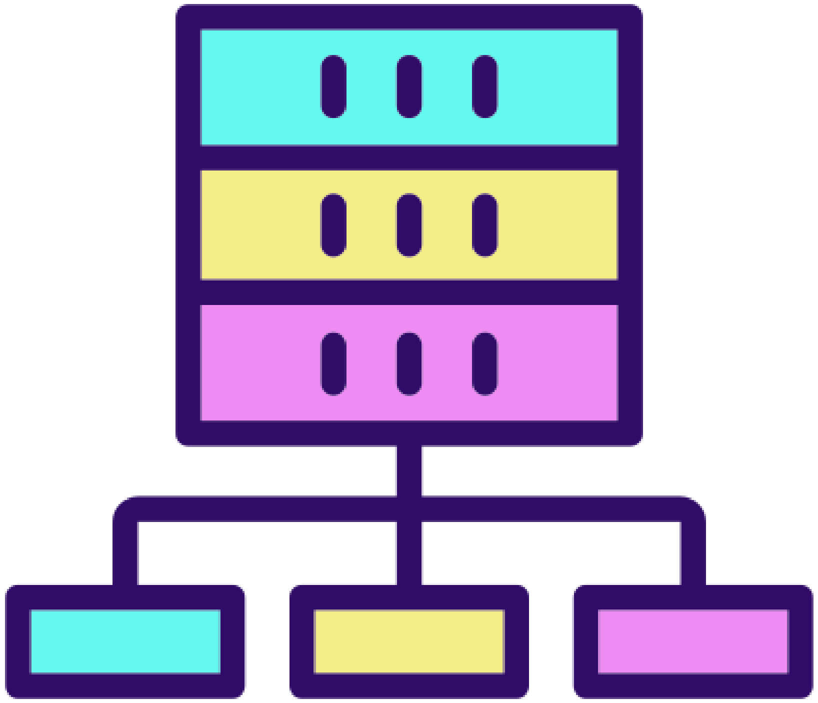
Numerical Features



- **Discrete**
  - Whole numbers (counts) typically integers
  - E.g., The number of cars in a parking lot, the number of students in a class, or the count of items in a basket.

- **Continuous**
  - Measurable numeric variable that may contain any value within a range
  - Typically represented decimal numbers and fractions
  - E.g., Height, weight, temperature, or distance

Dataset Types, Properties and Sources



Dataset Types (1 of 4)

Structured (Tabular) Data: Clearly defined rows and columns with a fixed schema

- E.g., Customer records in a SQL database, sales transactions in CSV files, student grades in a relational table

Semi-Structured Data: Flexible schema with self-describing structure (tags, keys)

- E.g., event logs stored as JSON files

Unstructured Text Data: Free-form text without a predefined schema

- E.g., Customer support tickets, emails, social media posts, open-ended survey responses

Dataset Types (2 of 4)

🖼️ **Image Data:** Pixel-based visual data

- E.g., Medical X-rays, satellite imagery, product photos for visual inspection or classification

🔊 **Audio Data:** Sound signals typically represented as waveforms or spectrograms

- E.g., Call center recordings, voice commands for virtual assistants, meeting transcripts input

🎥 **Video Data:** Sequences of images with temporal context

- E.g., CCTV footage for security analytics, traffic monitoring videos, lecture recordings

Dataset Types (3 of 4)

📈 **Time-Series Data:** Observations indexed by time, often with temporal dependencies

- E.g., Stock prices by minute, server CPU usage over time, IoT sensor readings collected every second

⚡ **Transactional / Event Data:** Discrete events generated by user or system actions

- E.g., E-commerce purchases, clickstream logs, login/logout events

Graph / Network Data: Entities and relationships represented as nodes and edges

- E.g., Social networks (users and friendships), citation networks, fraud detection graphs linking accounts and transactions

Dataset Types (4 of 4)

🛰️ **Geospatial Data:** Data associated with geographic locations and spatial relationships

- E.g., GPS traces from delivery vehicles, map coordinates of infrastructure, satellite-based land-use data.

🌊 **Streaming / Real-Time Data:** Continuously generated data requiring near-real-time processing

- E.g., Live sensor feeds, financial market ticks, real-time user activity events.

🧩 **Multimodal Data:** Combination of multiple data types in a single problem

- E.g., Product listings with text descriptions, images, and prices; medical records combining notes, images, and heart rate measurements time series

Data Matrix

- Data can often be represented or abstracted as an $n \times d$ *data matrix*, with *n* rows and *d* columns, given as

$$D = \left(\begin{array}{c|cccc} & X_1 & X_2 & \cdots & X_d \\ \hline \mathbf{x}_1 & x_{11} & x_{12} & \cdots & x_{1d} \\ \mathbf{x}_2 & x_{21} & x_{22} & \cdots & x_{2d} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ \mathbf{x}_n & x_{n1} & x_{n2} & \cdots & x_{nd} \end{array}\right)$$

- **Rows:** Also called *instances*, *examples*, *records*, *transactions*, *objects*, *points*, *feature-vectors*, etc. Given as a *d*-tuple

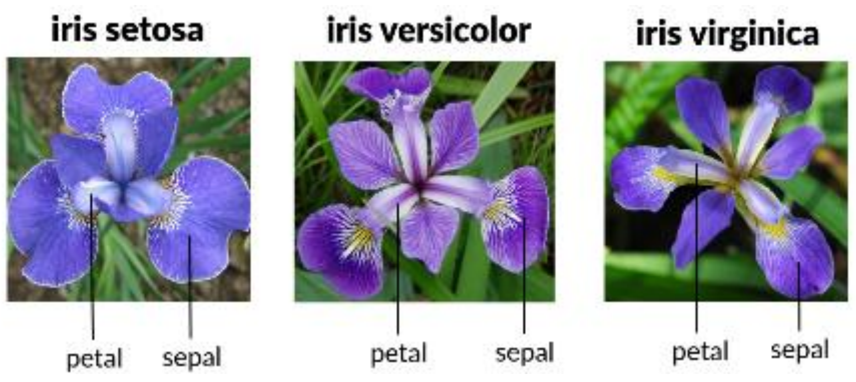
$$\mathbf{x}_i = (x_{i1}, x_{i2}, \ldots, x_{id})$$

- **Columns:** Also called *attributes*, *properties*, *features*, *dimensions*, *variables*, *fields*, etc. Given as an *n*-tuple

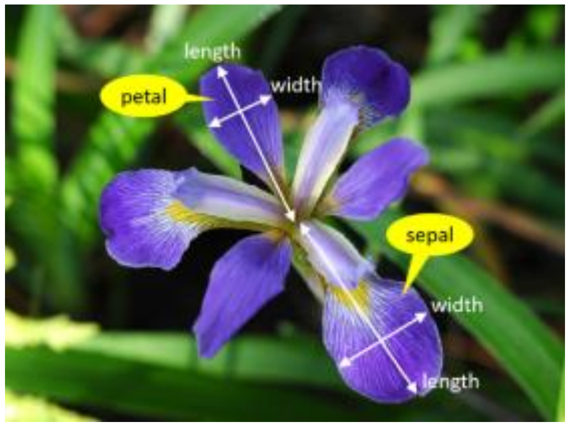
$$X_j = (x_{1j}, x_{2j}, \ldots, x_{nj})$$

Iris Dataset Extract

Data to quantify
the morphologic variation
of *Iris* flowers
Wikipedia



	Sepal length X_1	Sepal width X_2	Petal length X_3	Petal width X_4	Class X_5
$\mathbf{x}_1$	5.9	3.0	4.2	1.5	Iris-versicolor
$\mathbf{x}_2$	6.9	3.1	4.9	1.5	Iris-versicolor
$\mathbf{x}_3$	6.6	2.9	4.6	1.3	Iris-versicolor
$\mathbf{x}_4$	4.6	3.2	1.4	0.2	Iris-setosa
$\mathbf{x}_5$	6.0	2.2	4.0	1.0	Iris-versicolor
$\mathbf{x}_6$	4.7	3.2	1.3	0.2	Iris-setosa
$\mathbf{x}_7$	6.5	3.0	5.8	2.2	Iris-virginica
$\mathbf{x}_8$	5.8	2.7	5.1	1.9	Iris-virginica
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$
$\mathbf{x}_{149}$	7.7	3.8	6.7	2.2	Iris-virginica
$\mathbf{x}_{150}$	5.1	3.4	1.5	0.2	Iris-setosa



Dataset Properties

Size:	Measured in terms of the total number of records or total number of bytes, e.g. Small (MB), medium (GB) and large (TB)
Dimensionality:	Number of features
Density:	• How many of the data points in a dataset are filled with meaningful, non-zero values • Sparse: The vast majority of values are zero/null • Dense: Most data points are filled with meaningful, non-zero values.
Resolution:	• The level of detail at which data is recorded • Related to the intended purpose

Data Density & Resolution

Property	Explanation	Real-World Example
Data Density	Describes how many of the data points in a dataset are filled with meaningful, non-zero values • **Sparse**: Vast majority of values are zero/null. Non-zero entries are rare but often highly informative • **Dense**: The opposite; most data points are filled with meaningful, non-zero values	• E-commerce: A **matrix of all customers and all products**. Most cells will be '0' because a customer only buys a tiny fraction of the total items available, making the data sparse • Demographics: A table of **census respondents** with columns for age, gender, and income level. Most cells would be filled, making the data dense
Resolution (or Granularity)	The level of detail at which data is recorded. The ideal resolution is dictated entirely by the analytical objective; too fine can be noisy, while too coarse can hide crucial patterns.	Sales Data: Transaction data can be aggregated at different resolutions. • Objective: Staffing for the lunch rush → **High-resolution** (per-minute sales) is needed • Objective: Analyzing annual growth → **Low-resolution** (monthly or quarterly sales) is sufficient and more efficient

Data Sources

- **Public data**
  - Data hubs https://www.kaggle.com/datasets https://www.openml.org , GitHub
  - Open data such as https://data.gov/ & https://www.data.gov.qa/
  - Data conferences
  - Many others…
- **Enterprise/Organisational data warehouse**
  - An organisational database for decision making
  - A central data repository separate from operational systems
  - Equipped with data analysis and reporting tools
- **Your own collected/generated data**

Data Exploration - Univariate

02.eda\5-univariate-eda.ipynb

Exploratory Data Analysis (EDA)

- EDA is the critical process of performing **initial investigations** on data with the help of summary statistics and graphical representations to:
  - **Discover Patterns:** Identify trends and relationships within the data
  - **Spot Anomalies:** Detect outliers or errors that could skew your analysis
  - **Test Hypotheses:** Validate initial ideas about the data
  - **Check Assumptions:** Ensure the data meets the requirements for statistical models
- Examples:
  - Customer churn: EDA helps you identify which demographics are leaving and why
  - Factors influencing a disease: EDA might reveal correlations between age, lifestyle, and diagnosis

EDA Types

- **Univariate EDA** explore a single feature at a time to understand the data distribution and identify any outliers
- **Bivariate EDA** looking at two features at a time to understand the relationship and identify any patterns that might exist
- **Multivariate EDA** looking at three or more features at a time to understand the relationships and identify any patterns

Python EDA Toolkit

🛠️ To perform effective EDA, we rely on a powerful trio of Python libraries:

1. **Pandas:** The workhorse for data manipulation and analysis (`pd`)
2. **Matplotlib:** The foundational library for creating static, animated, and interactive visualizations (`plt`)
3. **Seaborn:** Built on top of Matplotlib, it provides a high-level interface for drawing informative statistical graphics (`sns`)

```python
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
# Load the classic Titanic dataset
df = sns.load_dataset('titanic')
# Set the aesthetic style of the plots
sns.set_theme(style="whitegrid")
print("Libraries imported and data loaded
successfully!")
```

🔍 First Look: Initial Data Inspection

Before diving into complex visualizations, we must first understand the basic structure of our dataset.

Key Techniques:

- **View Data:** `.head()` and `.tail()` let you peek at the actual data values
- **Data Structure:** `.info()` provides a concise summary of the DataFrame, including data types and non-null counts.
- **Summary Statistics:** `.describe()` generates descriptive statistics (count, mean, std, min, max, etc.) for numerical columns
- **Dimensions:** `.shape` tells you exactly how many rows (observations) and columns (features) you have

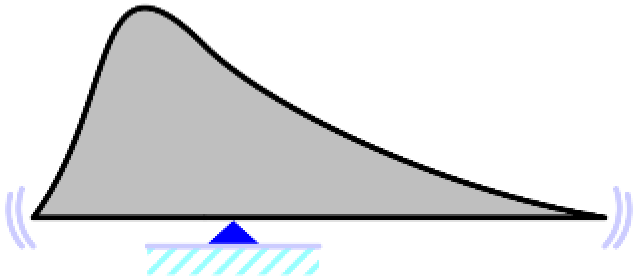
```python
# View the first 5 rows
print("First and last 5 rows:")
display(df.head())
display(df.tail())

# Get a concise summary of the dataframe
print("\nDataFrame Info:")
print(df.info())

# Check dimensions
print(f"\nShape of the dataset: {df.shape}")

# Statistical summary of numerical columns
print("\nStatistical Summary:")
display(df.describe())
```

Summary Statistics - Central Tendency

- Mean and Median for continuous attributes:

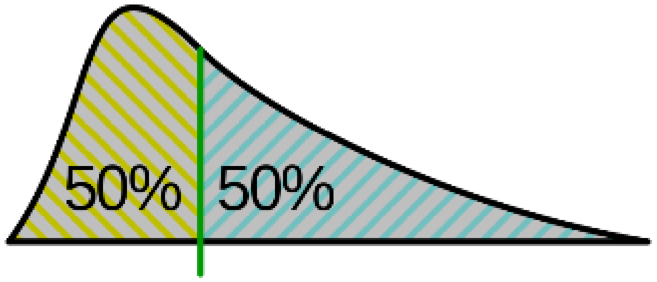
  - **Mean**

$$\bar{x} = \frac{1}{n}\sum_{i=1}^{n} x_i$$



  - **Median** (Middle value if odd number of values, or average of the middle two values otherwise)

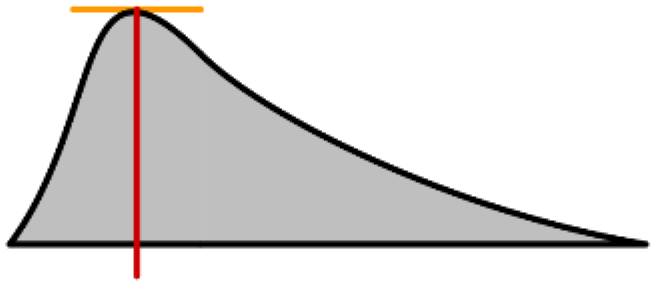
    ➢ Median is a better indication of "average" when data distribution is skewed, or outliers are present



  - Trimmed Mean and Median (after trimming top and bottom p%)

Summary Statistics - Central Tendency

- **Mode** for categorical attributes:



  - Frequency counts of values that a feature takes
  - Proportion: Frequency count for a value divided by the total sample size
  - **Mode**: the most frequently occurred value

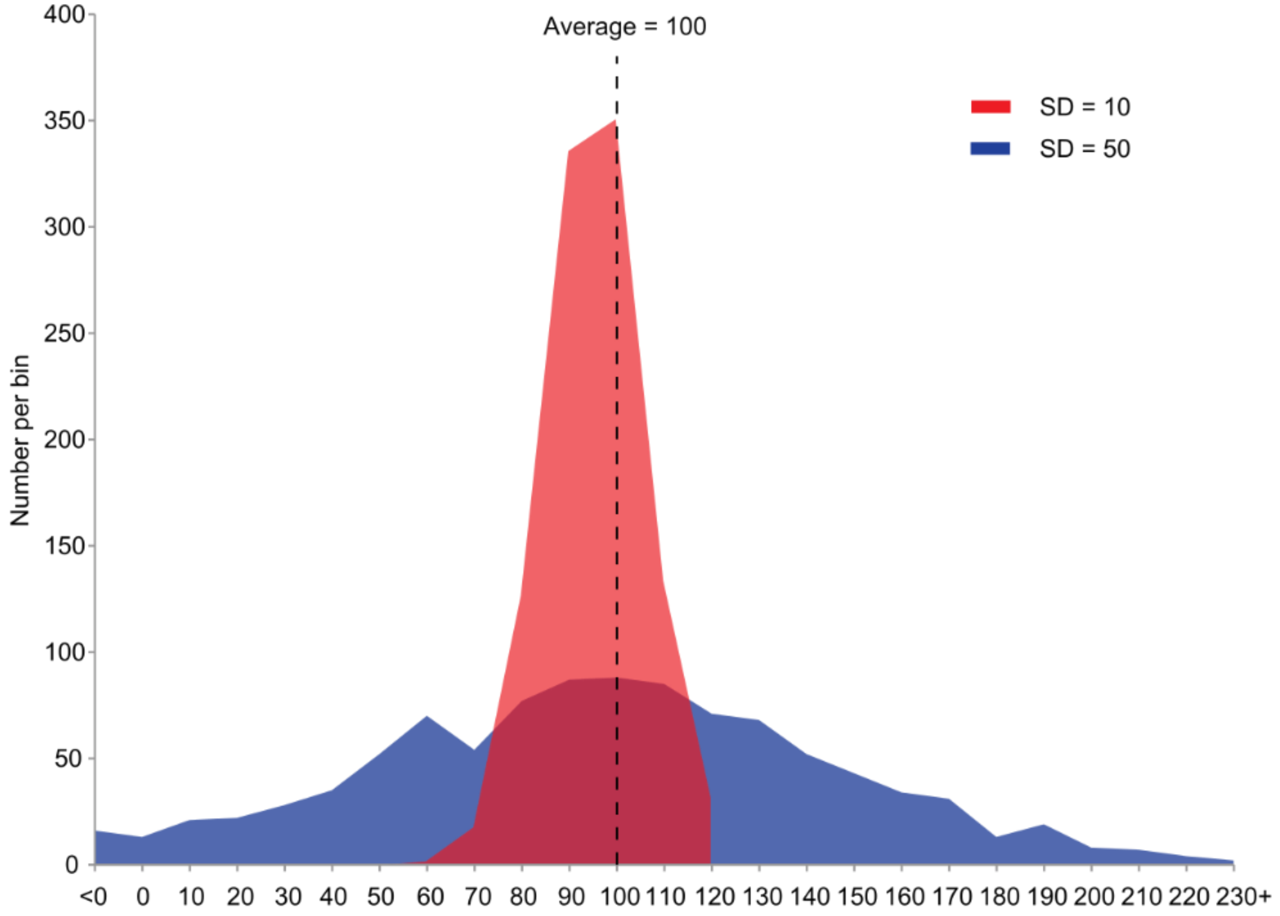
Summary Statistics - Measures of Spread

- Measure how "spread out" the values are
- Range $$range(x) = \max(x) - \min(x)$$
- Variance (σ^2) $$\sigma^2 = \frac{1}{m-1}\sum_{i=1}^{m}(x_i - \bar{x})^2$$
- Standard Deviation (σ) $$\sigma = \sqrt{\frac{1}{m-1}\sum_{i=1}^{m}(x_i - \bar{x})^2}$$
- Percentiles of continuous attributes:
  - Given an attribute *x* and an integer *p* ($0 \le p \le 100$), the percentile x_p is a value of x such that *p*% observed values of *x* are less than x_p
  - Q_1 (25th percentile), Q_3 (75th percentile). Q_3 means 75% of the data values are less than Q_3
  - Inter-quartile range: IQR = $Q_3 - Q_1$

Measures of Spread, *cont'd*

- Measure how the values are stretched or squeezed
  - aka **Measures of dispersion**

Two datasets with the <u>same mean</u> but **different dispersion**. The blue dataset is much more **dispersed** than the red dataset.



https://en.wikipedia.org/wiki/Statistical_dispersion

Summary Statistics using Pandas

```
df.describe()

df[['DepTime', 'DepDelay', 'ArrTime',
'ArrDelay']].agg(['mean', 'min', 'max'])


price_mean = df['price'].mean()

price_median = df['price'].median()

price_std = df['price'].std()

price_var = df['price'].var()

price_quantiles =

     df['price'].quantile([0.25,0.5,0.75])
```

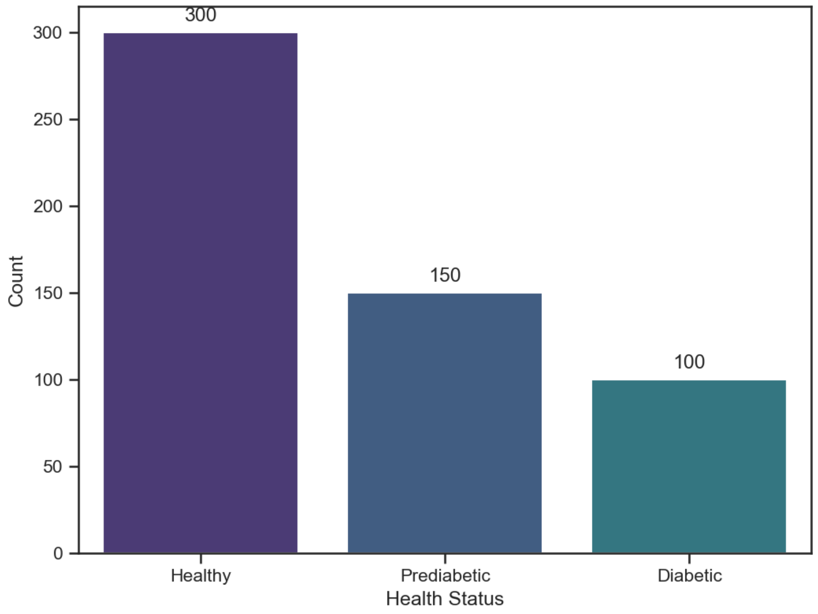
Motto: Visualize Before Analyzing!

- Data visualization gives us a more holistic sense
- Allows understanding patterns, distributions, and relationships among different features
- *Anscombe's quartet* datasets having the same mean, standard deviation, and regression line, but which are qualitatively different.
  - It illustrates the importance of looking at a set of data graphically and not only relying on basic statistic properties.

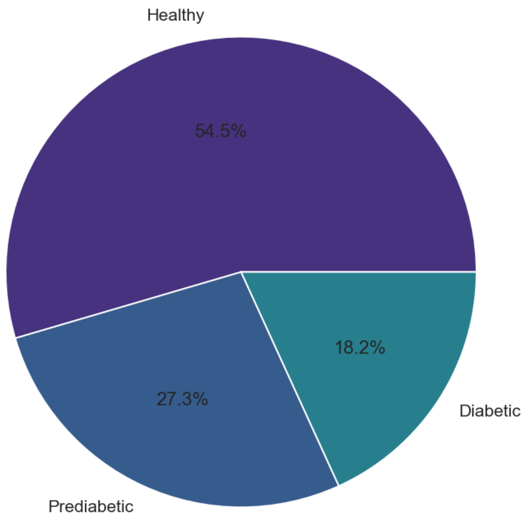
02.eda\**4.why-visualization-anscombe**.ipynb

Data visualization for categorical data

- **Bar Chart**: categories on one axis and the corresponding frequencies or proportions on the other axis



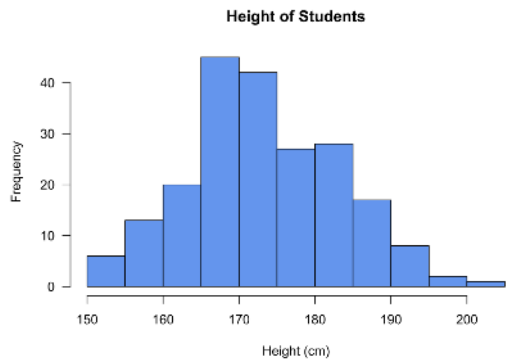
- **Pie Chart**: shows relative size of each category within the whole



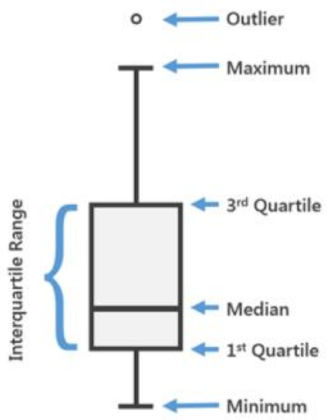
02.eda**2-categorical-variables**.ipynb

Data visualization for numerical data

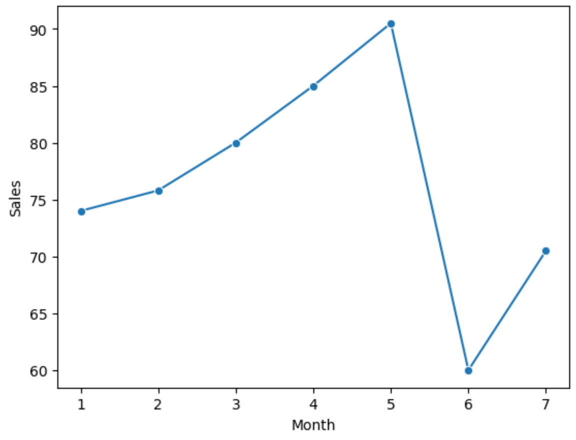
- **Histogram**: represent the distribution of a continuous numerical variable



- **Boxplot**: Depicts the spread and central tendency of the data, including median, quartiles, and potential outliers



- **Line plot**: visualize trends in data over time

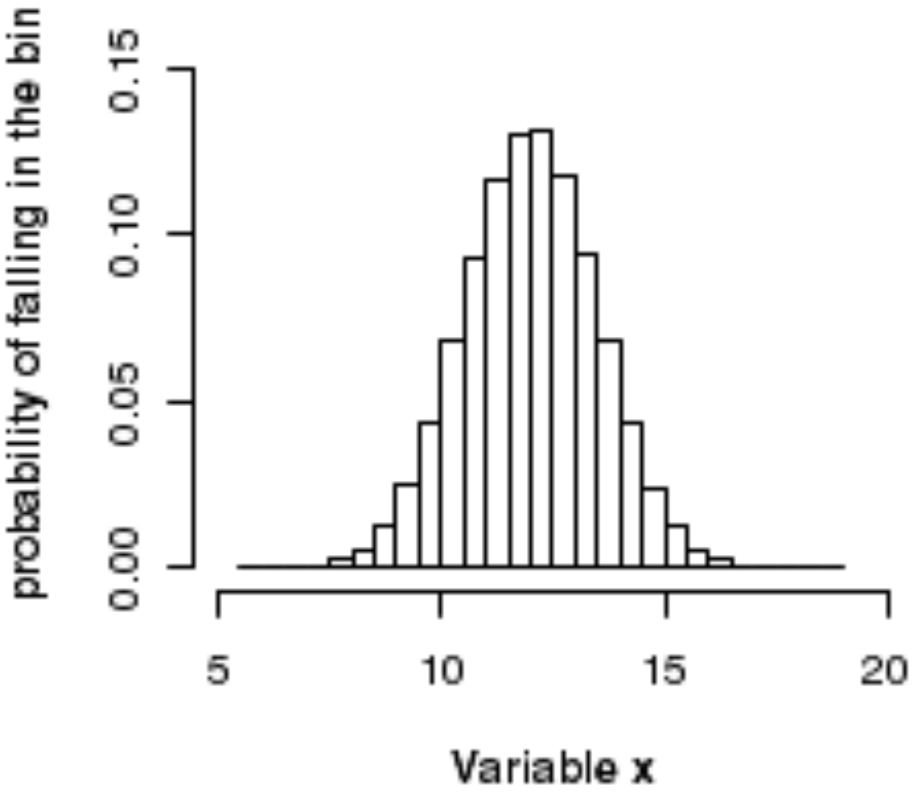


02.eda\3-numerical-variables.ipynb

Histogram to probability distribution

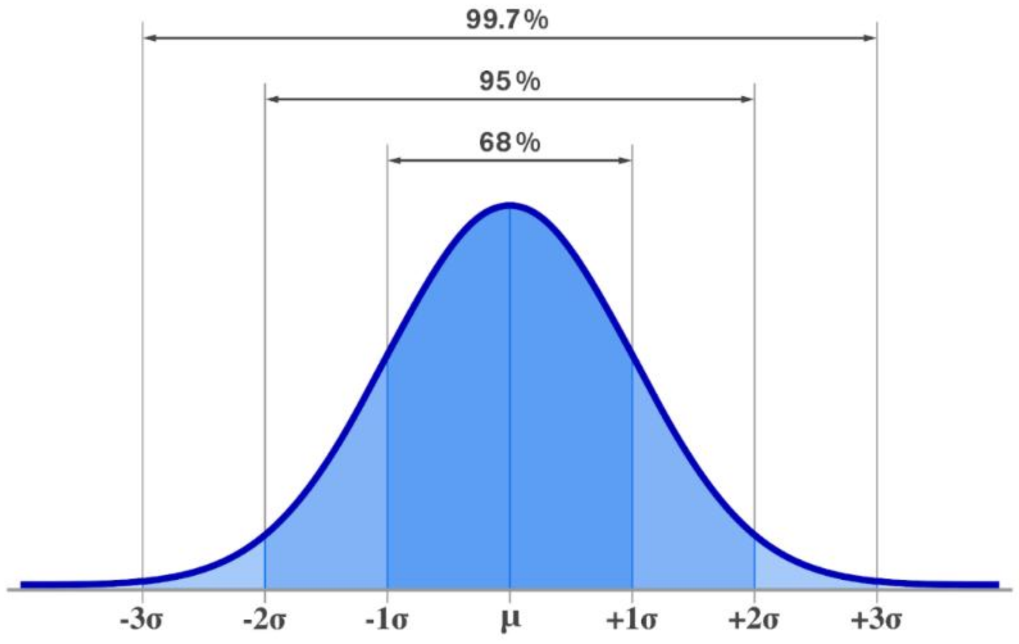
- Divide the count for each interval by the total number of observations in the dataset multiplied by the width of the interval

$$\text{Probability distribution} = \frac{\text{Count for each interval}}{\text{Count of observations in the dataset} \times \text{Width of the interval}}$$



Normal Distribution

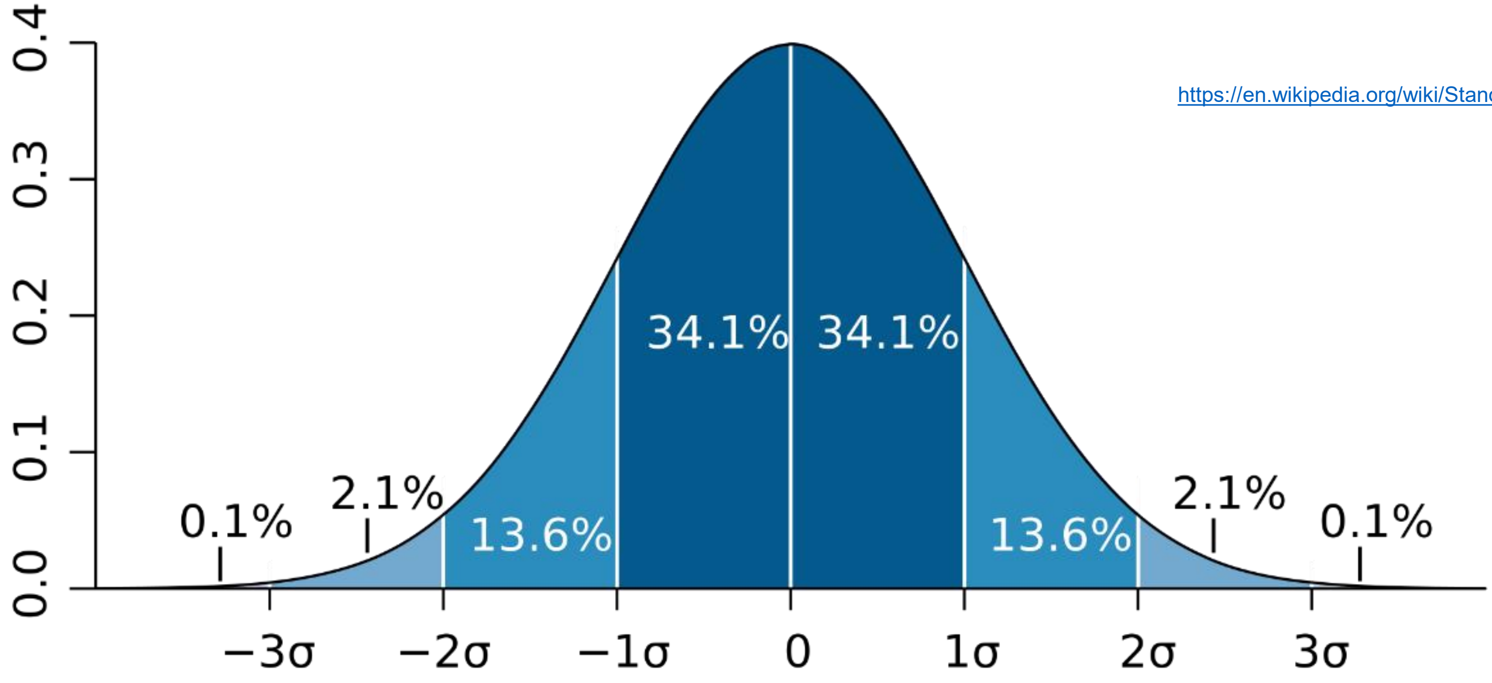
- Many phenomena follow a normal distribution
  - Height, blood pressure, exam scores, etc.



- Symmetric:
  - Most of the observations occur around the central peak
  - Probabilities for values further away from the center decrease equally in both directions
  - Extreme values in both tails of the distribution are similarly unlikely

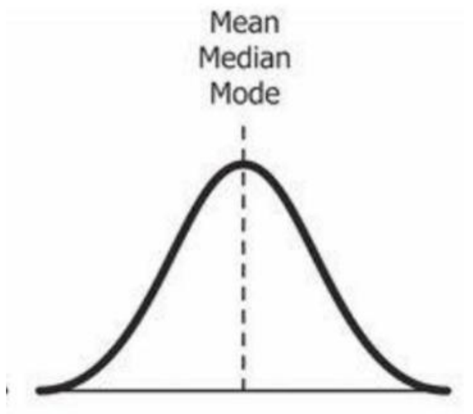
Spread Properties of Normal Distribution Curve

- The 68−95−99.7 rule:
  - From μ–σ to μ+σ: contains about 68% of the values
  - From μ–2σ to μ+2σ: contains about 95% of it
  - From μ–3σ to μ+3σ: contains about 99.7% of it

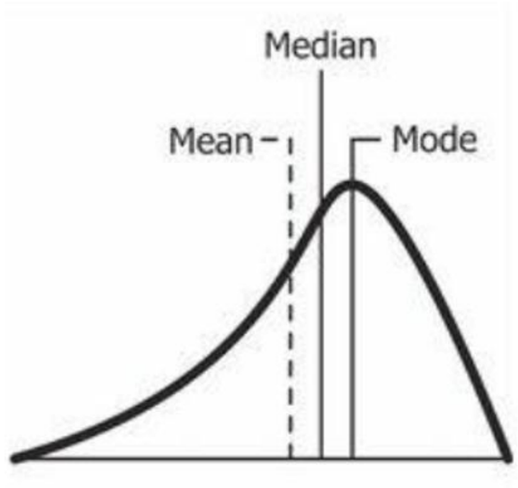


https://en.wikipedia.org/wiki/Standard_deviation

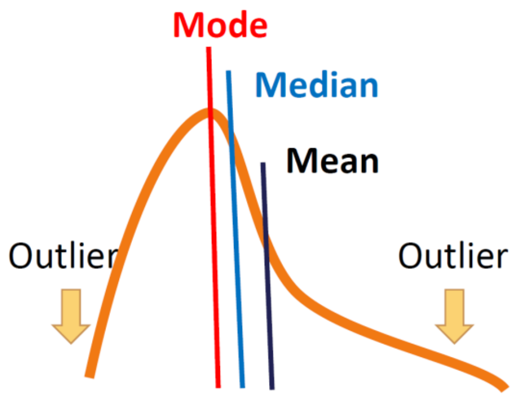
Dark blue is one standard deviation on either side of the mean, or 68% of the values



- In normal **symmetric distribution**, the mean, median and mode are the same



- A distribution is skewed if one of its tails is longer than the other
- A **left-skewed** (negative-skewed) distribution has a long left tail



- A **right-skewed** (positive-skewed) distribution has a long right tail

Coefficient of Skewness

- The coefficient of skewness quantifies the asymmetry of a probability distribution around its mean, indicating how much the data distribution diverges from symmetry
  - Skewness > 0 denotes a right-skewed distribution, where the tail extends towards higher values
  - Skewness < 0 signifies a left-skewed distribution, where the tail extends towards lower values
  - Skewness = 0 signals a perfectly symmetrical distribution
- The formula for skewness is often given as:

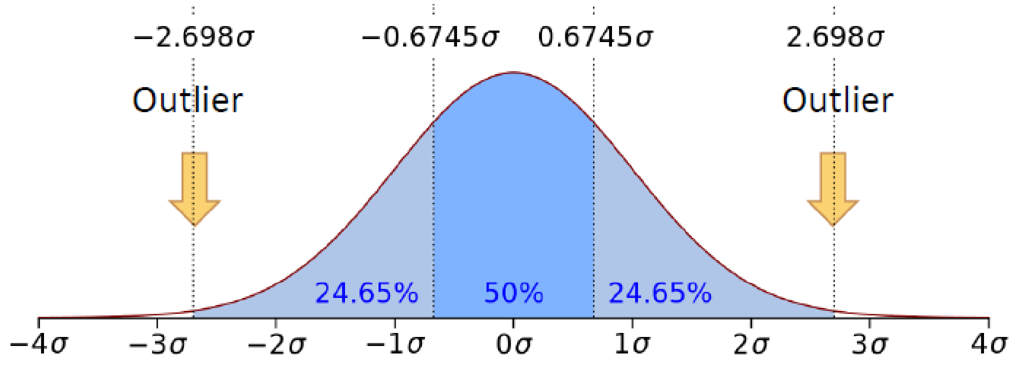
$$\text{Skewness} = \frac{3(\text{Mean} - \text{Median})}{\text{Standard Deviation}}$$

pandas

```
skewness = df.price.skew()
```

Detecting Outliers

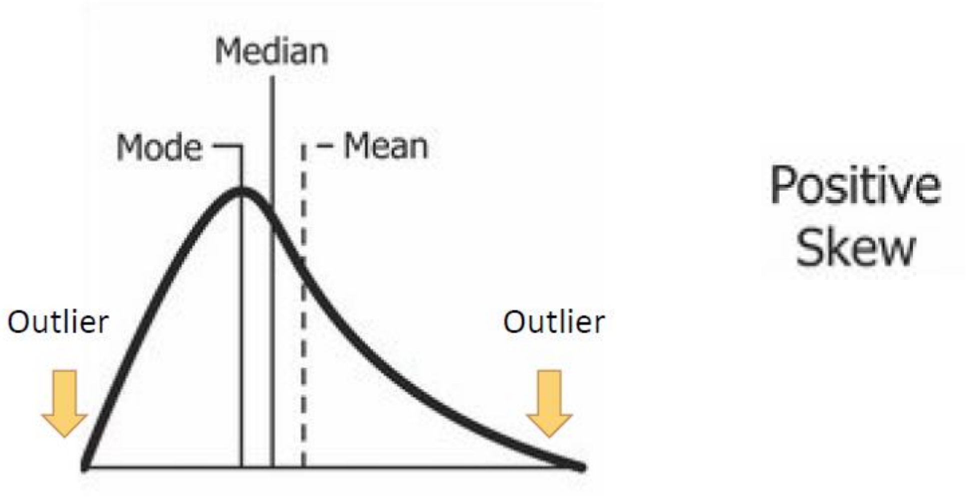
- An outlier is a data point which is **significantly different** from the remaining data



- ≈99% of the observations of a normally distributed variable lie within the mean ± 3 x standard deviations
- Values outside mean **± 3 x standard deviations** are considered outliers

`02.eda\6-univariate-eda.ipynb`

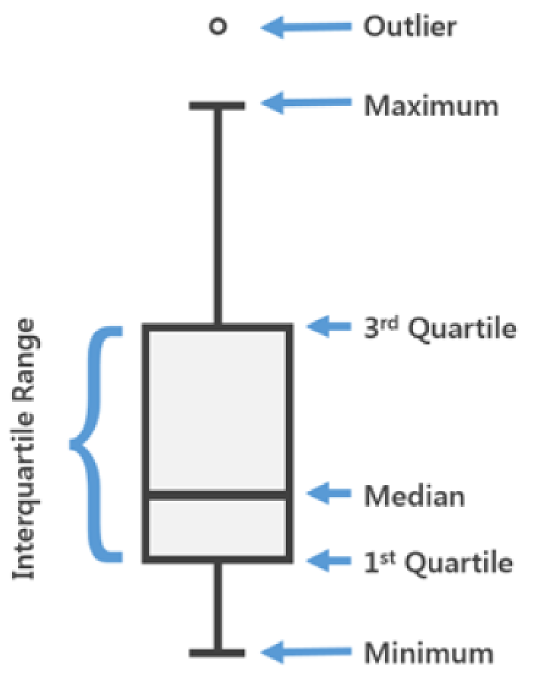
Outliers for skewed distributions

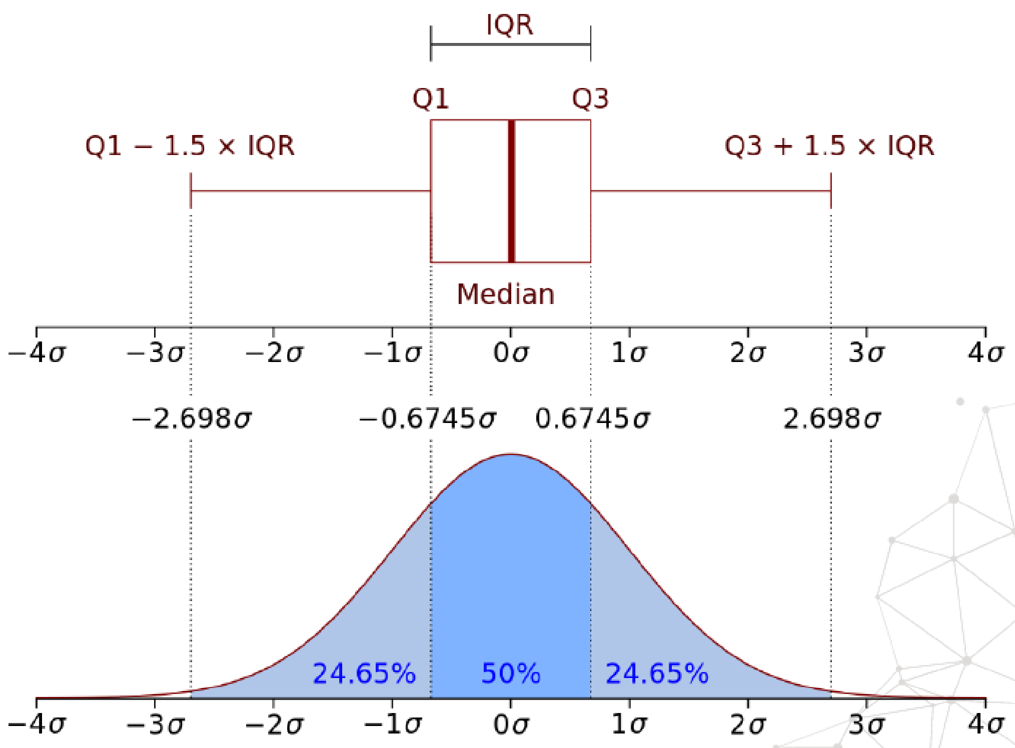


Calculate the quantiles, and then the inter-quantile range (IQR), as follows:

- `IQR = 75th Quantile - 25th Quantile`
- `Upper limit = 75th Quantile + IQR × 1.5`
- `Lower limit = 25th Quantile - IQR × 1.5`
- Values outside the limits are considered **outliers** that can be dropped or replaced by the mean or median

Visualizing outliers using Boxplots





Images taken from pro.arcgis.com and wiki.commons

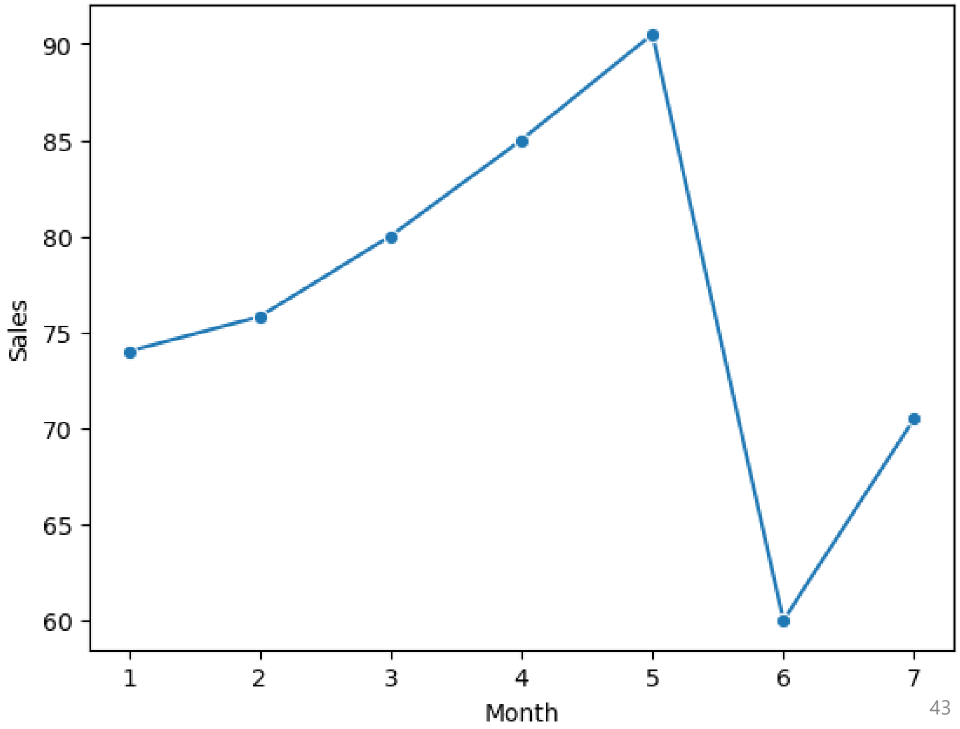
Boxplot: data is represented with a box

- The ends of the box are at the 1st and 3rd quartiles, i.e., the height of the box is IQR
- The median is marked by a line within the box
- Whiskers: two lines outside the box extended to `± IQR × 1.5`

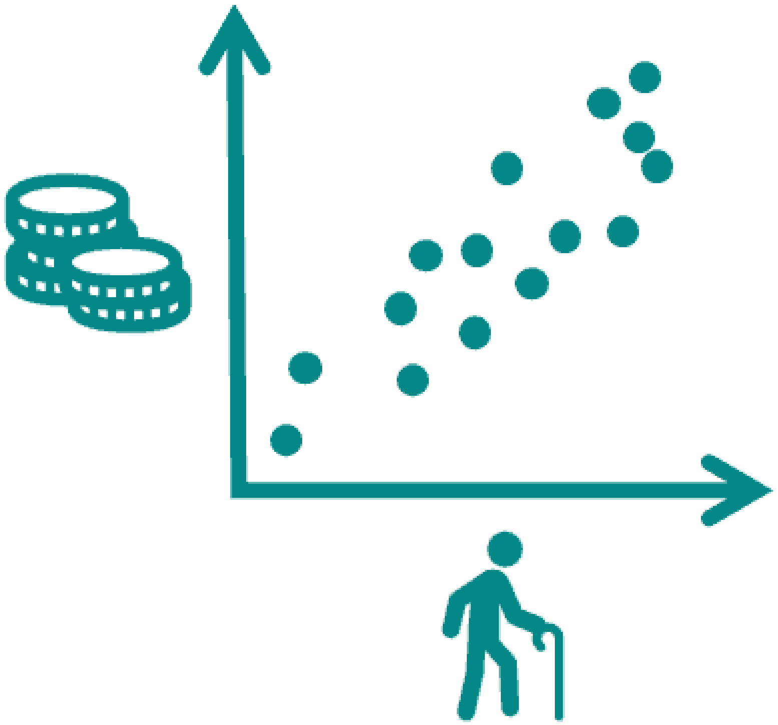
Line Plot - analyzing a Single Variable over Time

- Line Plot display data points connected by straight lines
- It is particularly effective for visualizing trends in data over time
  - E.g., Tracking stock prices over months to identify trends or patterns
- They help identify seasonal patterns such as increasing, decreasing, or cyclical trends
  - E.g., Peak air travel around June-August

e.g., LinePlot visualizes the monthly sales.
We can observe any fluctuations or any seasonal patterns in sales over the course of the year
e.g. upward trend with growth in sales then a sharp drop in the month of June



Data Exploration - Bivariate



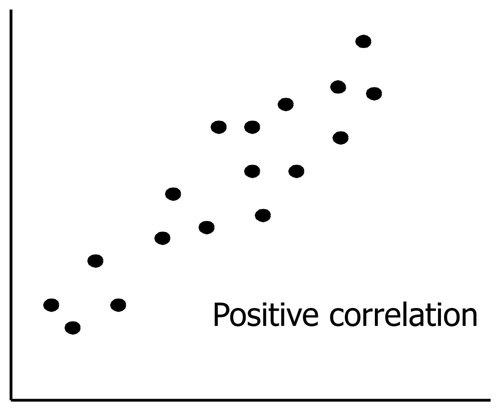
02.eda\7-bivariate-eda.ipynb

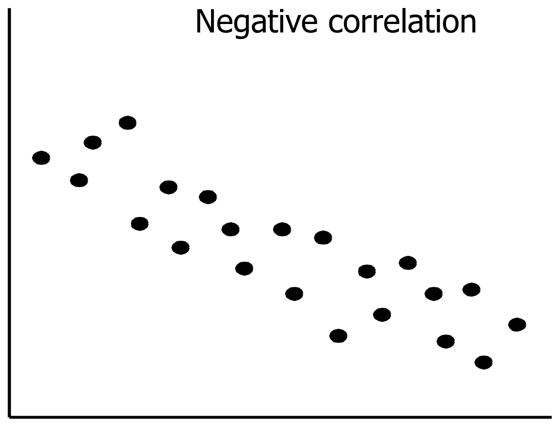
Types of Bivariate Analysis

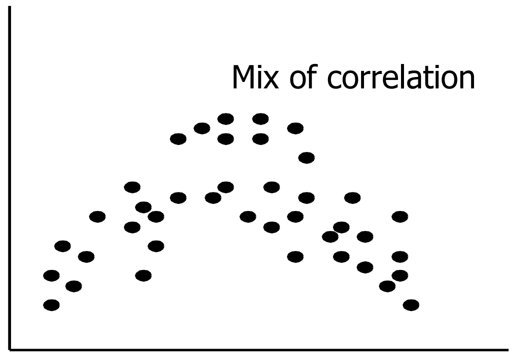
- 📈 **Numerical vs. Numerical**: examines the relationship between two numerical variables.
  - E.g.,: In a real estate dataset, we analyze the correlation between the house size in m^2 of a house and its price. Are larger houses more expensive?
- 📊 **Categorical vs. Numerical**: explore how a categorical variable affects a numerical one.
  - E.g.,: Studying how the type of car (SUV, sedan, etc.) impacts fuel efficiency (miles per gallon) in an automotive dataset.
  - E.g., Connection between the level of education and income in demographic studies
- 📚 **Categorical vs. Categorical**: focuses on the association between two categorical variables.
  - E.g.,: In an e-commerce dataset, we assess if there’s a connection between a customer’s gender and their preferred payment method

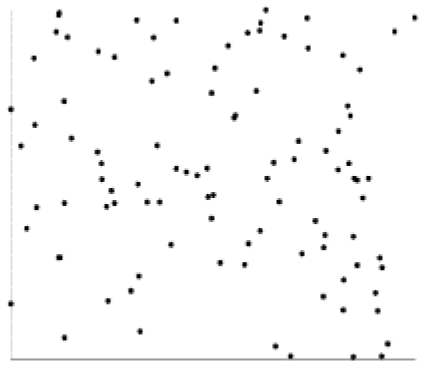
Scatter Plot

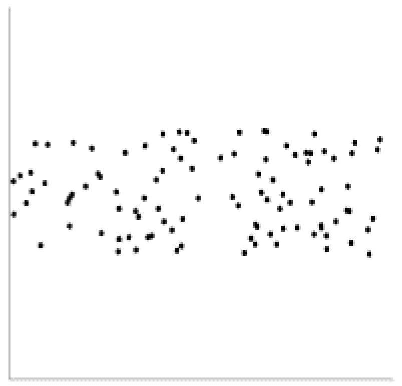
- **Scatter Plot** visualize the relationship between two continuous numerical variables by plotting one variable along the x-axis and the other variable along the y-axis
- Useful to determine whether there is a correlation between the variables (positive, negative, or none), the strength of the correlation, and the presence of any outliers

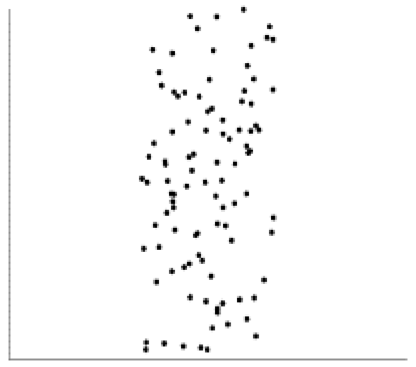










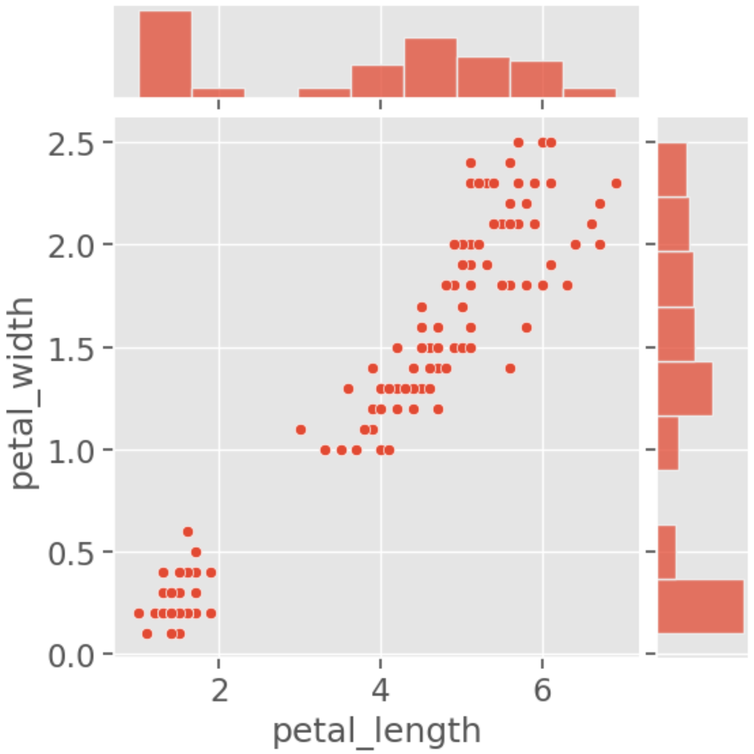


Not Correlated Data

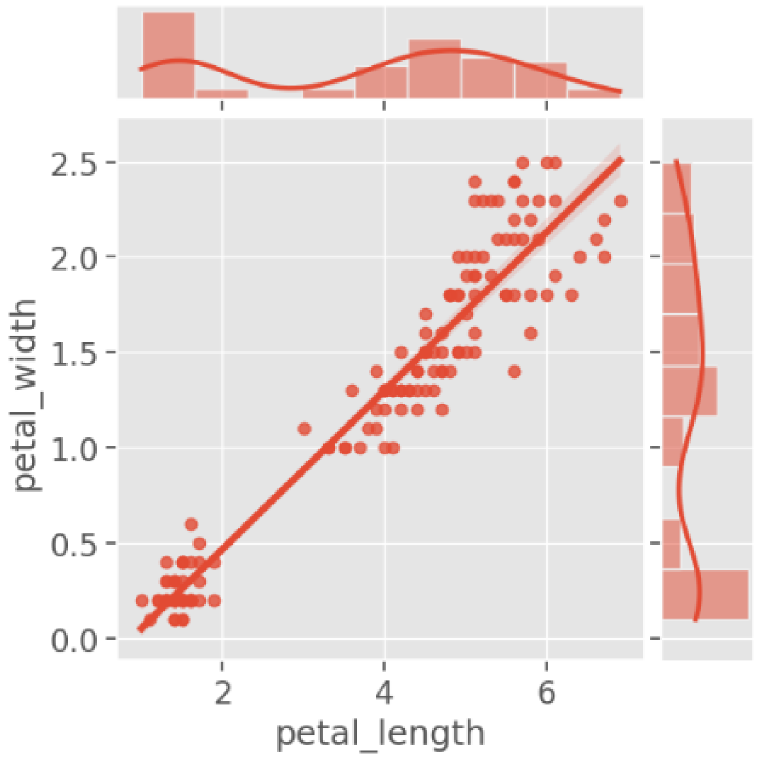
Joint Plot - Visualizing Pairs of Continuous Features

- **Joint Plot** combines multiple plots, such as scatter plot and histogram, to visualize the correlation between the variables, including their individual distributions

```
sns.jointplot(x='petal_length', y='petal_width', data=iris, kind='scatter')
pass
```



```
sns.jointplot(x='petal_length', y='petal_width', data=iris, kind='reg')
pass
```



Iris Characteristics: Strong linear relationship between petal length and width

Pearson's correlation

- Pearson correlation is a statistical measure that **quantifies the linear relationship between two continuous variables**. It provides insights into how closely related two variables are and the direction of their relationship (positive or negative)
  - The Pearson correlation coefficient, denoted by r, ranges from -1 to 1
  - r=1 indicates a perfect positive linear relationship
  - r=−1 indicates a perfect negative linear relationship
  - r=0 indicates no linear relationship
  - A positive r value suggests that as one variable increases, the other tends to increase as well
  - A negative r value indicates that as one variable increases, the other tends to decrease
- In Python, you can calculate the Pearson correlation coefficient using the corr() function from pandas or pearsonr() function from the scipy.stats module

Pearson correlation coefficient

$$r = \frac{\sum(X_i - \bar{X})(Y_i - \bar{Y})}{\sqrt{\sum(X_i - \bar{X})^2 \sum(Y_i - \bar{Y})^2}}$$

Where:

- X_i and Y_i are individual data points.
- $\bar{X}$ and $\bar{Y}$ are the means of X and Y respectively.

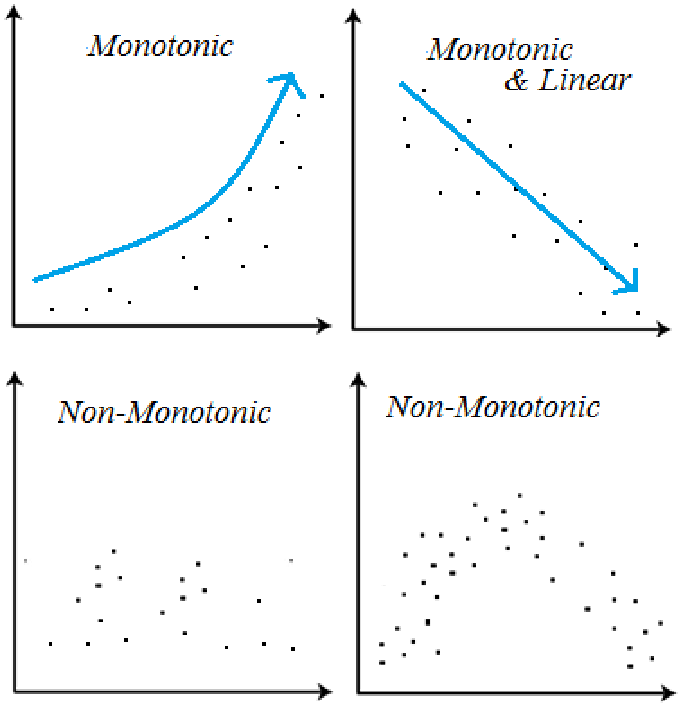
- The numerator calculates the covariance between X and Y, which measures how they vary together from their means
- The denominator normalizes the covariance by the standard deviations of X and Y, ensuring that the correlation coefficient is scaled appropriately

Pearson's correlation assumptions

- Pearson's correlation coefficient is a parametric measure of the linear relationship between two continuous variables. It makes certain assumptions about the data, including:
  - **Linearity**: there is a linear relationship between the two variables. If the relationship between the variables is not linear, Pearson's correlation may not accurately reflect the relationship.
  - **Normality**: the data is normally distributed. This means that the distribution of the residuals (the difference between the values) should follow a normal distribution.
  - **Independence**: the observations are independent of one another. This means that the value of one observation does not influence the value of another observation.
- If these assumptions are not met, Pearson's correlation may not accurately reflect the relationship between the variables. In these cases, non-parametric methods, such as Spearman's rank correlation, may be more appropriate

Spearman's Rank Correlation

- Spearman's Rank Correlation tells us **how consistently two variables move together**, based on their ranked positions rather than their actual values.
  - **Monotonic:** The two variables move in the *same general direction* — when one increases, the other tends to increase consistently.
  - **Linear Monotonic:** The variables move in the same direction at an approximately constant rate (straight-line trend).
  - **Non-Monotonic:** The variables *do not follow any consistent direction* — the relationship goes up and down unpredictably
  - **Non-parametric:** does not assume a specific distribution of the data



Students	Maths	Science
A	35	24
B	20	35
C	49	39
D	44	48
E	30	45

Students	Maths	Rank	Science	Rank	d	d square
A	35	3	24	5	2	4
B	20	5	35	4	1	1
C	49	1	39	3	2	4
D	44	2	48	1	1	1
E	30	4	45	2	2	4
						14

$$\rho = 1 - \frac{6\sum d_i^2}{n(n^2-1)}$$

ρ = Spearman's rank correlation coefficient

d_i = difference between the two ranks of each observation

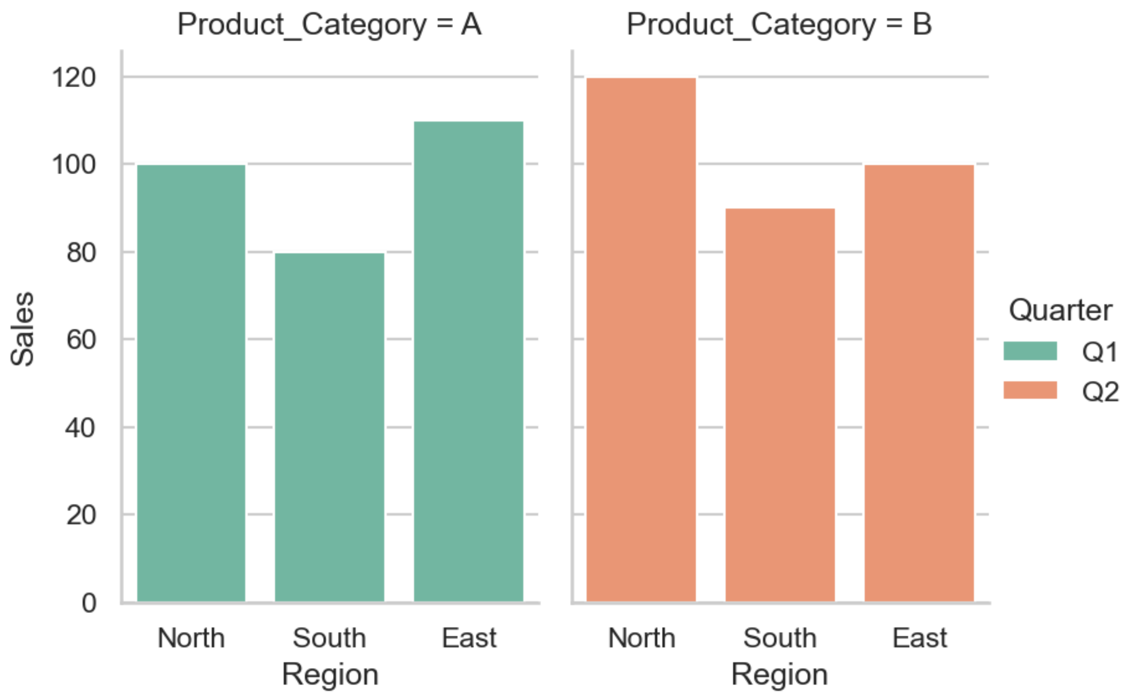
n = number of observations

1 - (6 * 14) / 5(25 - 1) = 0.3

The Spearman's Rank Correlation for the given data is 0.3. The value is near 0, which means that there is a weak correlation between the two ranks.

Grouped Bar Plots - Numerical by Categories

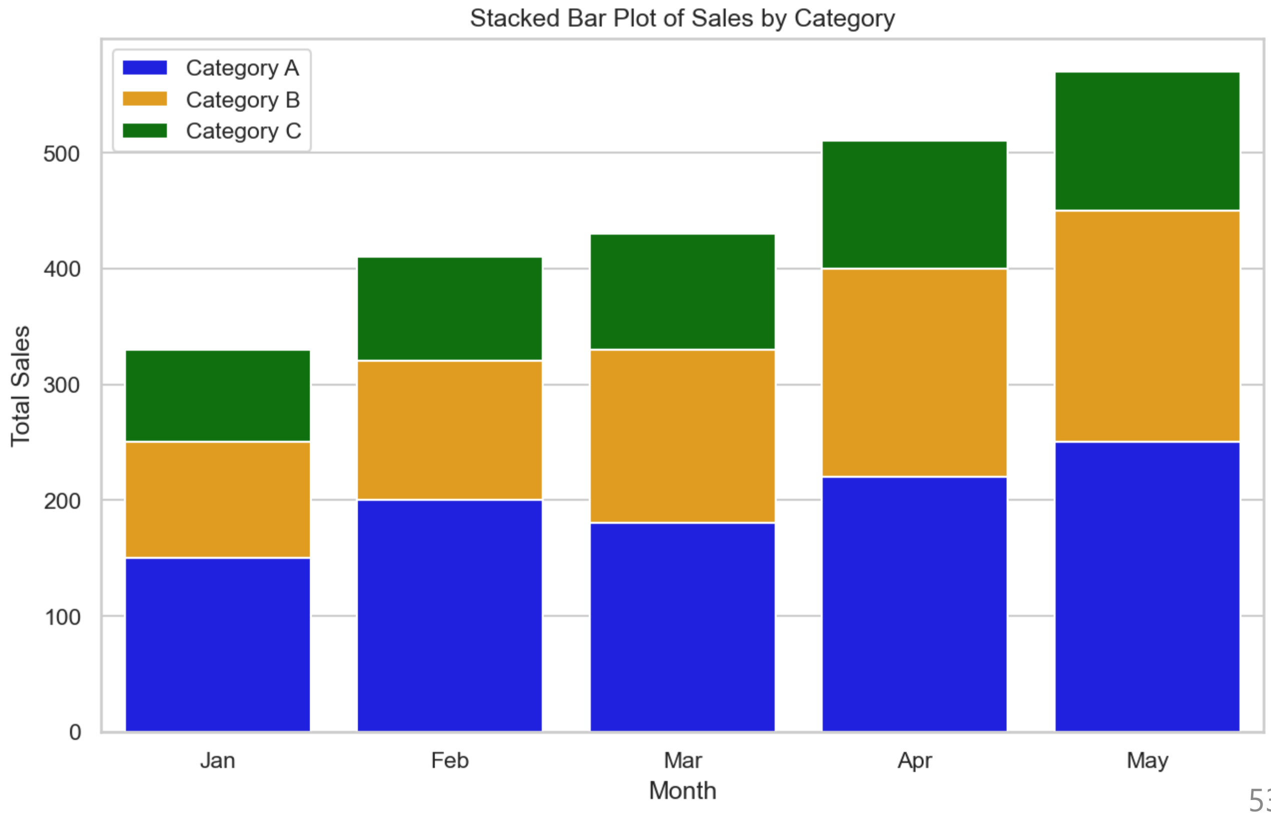
- **Grouped Bar Plots** visualize and compare how a numerical feature varies across different categories of a single categorical feature. This allows identifying patterns and trends
  - E.g., visualize how the sales of each product category vary across different regions and quarters (using searborn `sns.catplot`)



Stacked Bar Plot - Numerical by Categories

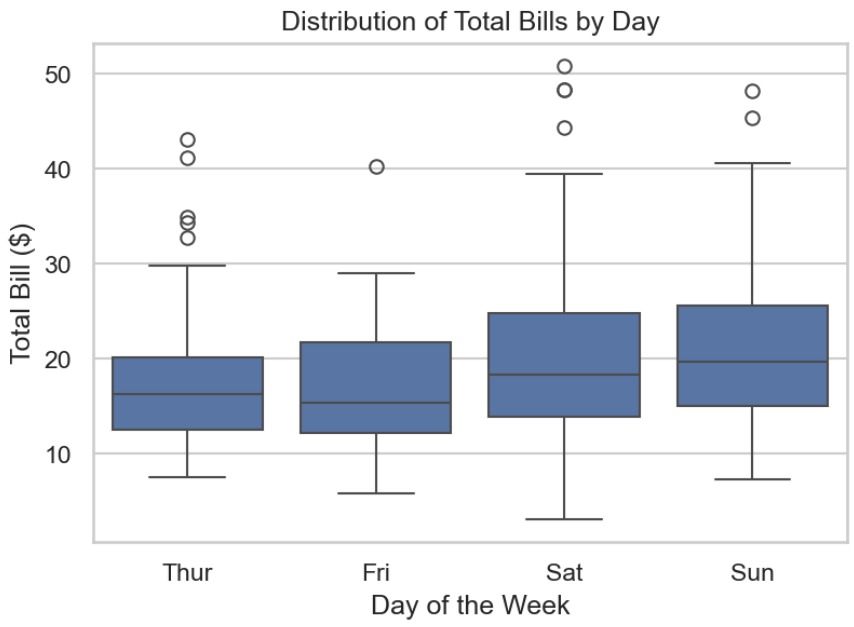
- **Stacked Bar Plot** visualize how a numerical feature varies across a categorical feature while showcasing the contribution of each subcategory to the total (e.g., `df.set_index('Month').plot(kind='bar', stacked=True)`)
  - Each bar in the plot represents the **total value**, and the segments within the bar correspond to contribution of each subcategory to the total

This chart allows for a quick comparison of the sales **composition** across different months and categories

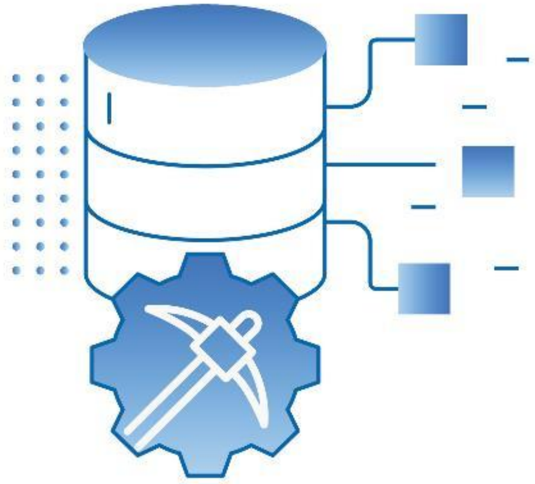


Grouped Box Plots

- **Grouped Box Plots**, allows comparing the distributions of a continuous variable across **multiple categories**
  - This visualization is particularly useful for identifying patterns, differences, and relationships between the categorical and continuous variables



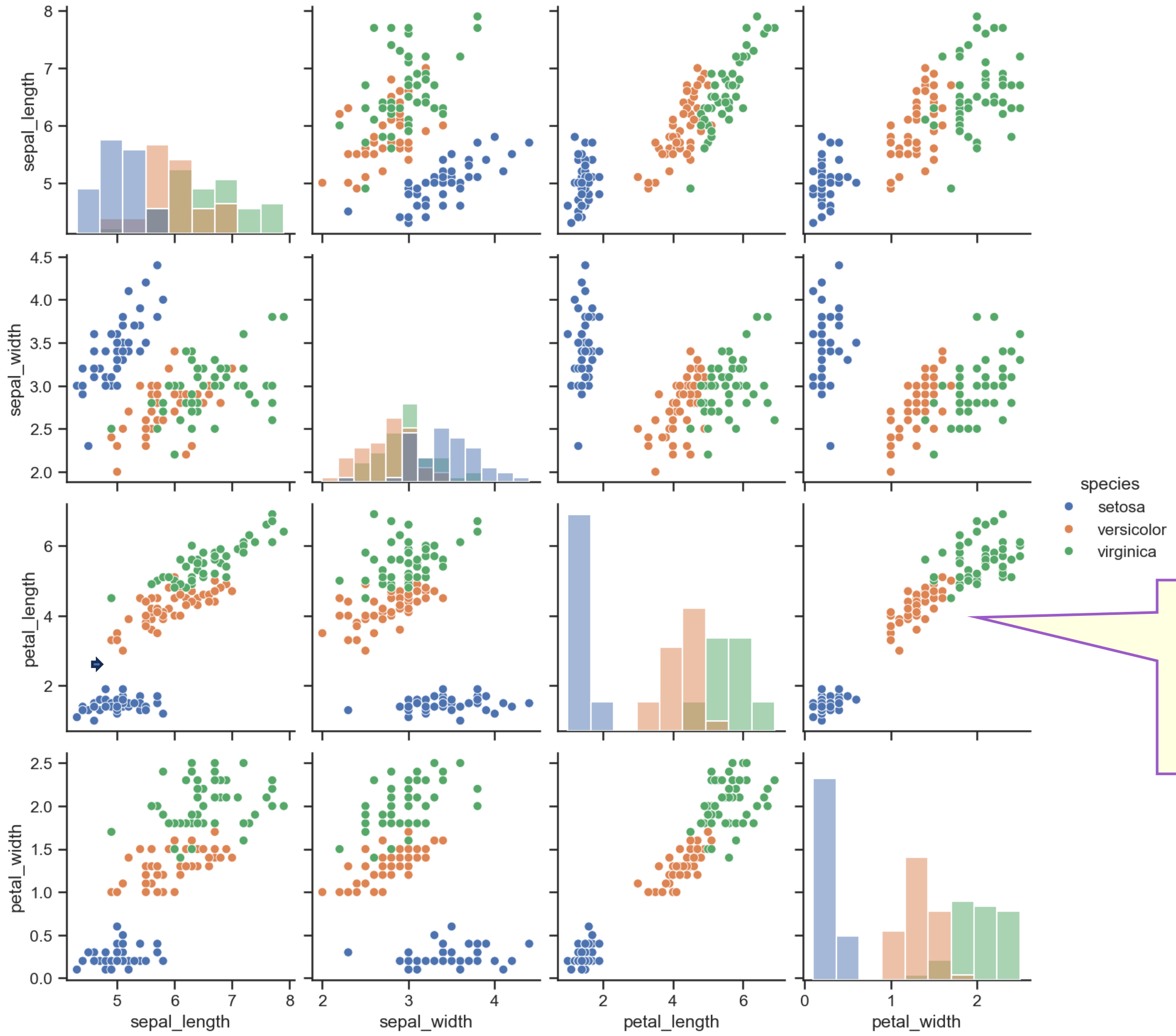
Data Exploration - Multivariate



02.eda\7-multivariate-eda.ipynb

Scatter Plot matrix

- **A Scatter Plot matrix** (used for Numerical vs. Numerical), also known as a **Pairs Plot**, is used to examine the relationships between multiple variables simultaneously.
  - It consists of a grid of scatter plots where each variable is plotted against every other variable in the dataset



Petal dimensions discriminate species more strongly than sepal dimensions

Strong linear relationship between petal length and width

Correlation Matrix & Heatmap

- A correlation matrix (used for Numerical vs. Numerical) is a table that shows the correlation coefficients between many variables
- A heatmap plots data as a color-encoded matrix

```
cormat = iris.corr(method="pearson")
round(cormat,2)
```

]:

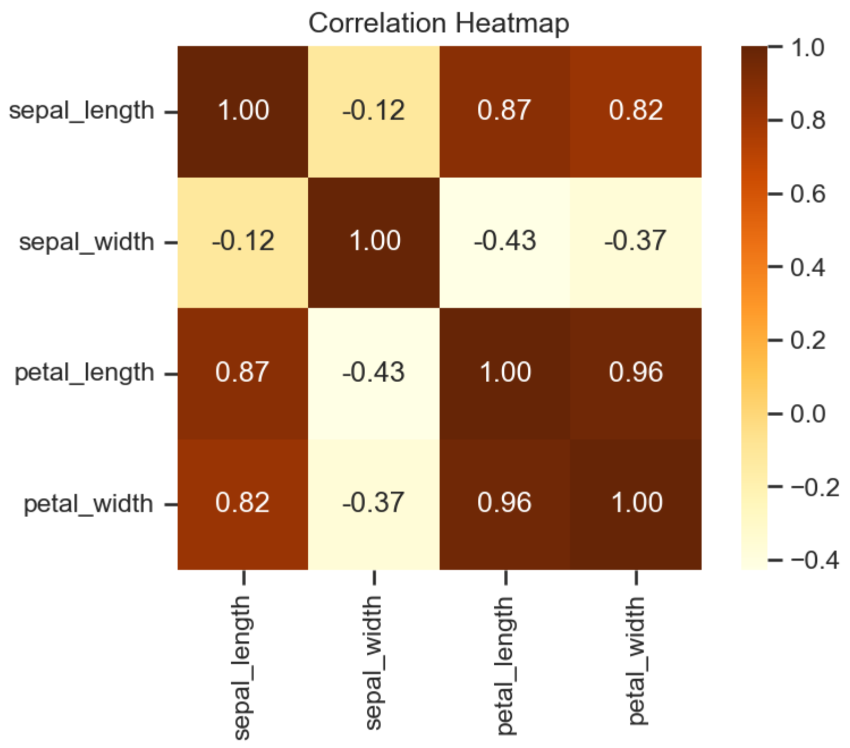
	sepal_length	sepal_width	petal_length	petal_width
sepal_length	1.00	-0.12	0.87	0.82
sepal_width	-0.12	1.00	-0.43	-0.37
petal_length	0.87	-0.43	1.00	0.96
petal_width	0.82	-0.37	0.96	1.00

```
In [65]:  cormat = iris.corr(method="spearman")
          round(cormat,2)
```

Out[65]:

	sepal_length	sepal_width	petal_length	petal_width
sepal_length	1.00	-0.17	0.88	0.83
sepal_width	-0.17	1.00	-0.31	-0.29
petal_length	0.88	-0.31	1.00	0.94
petal_width	0.83	-0.29	0.94	1.00

```
sns.heatmap(cormat)
```



We can observe a strong positive correlation between petal length and width

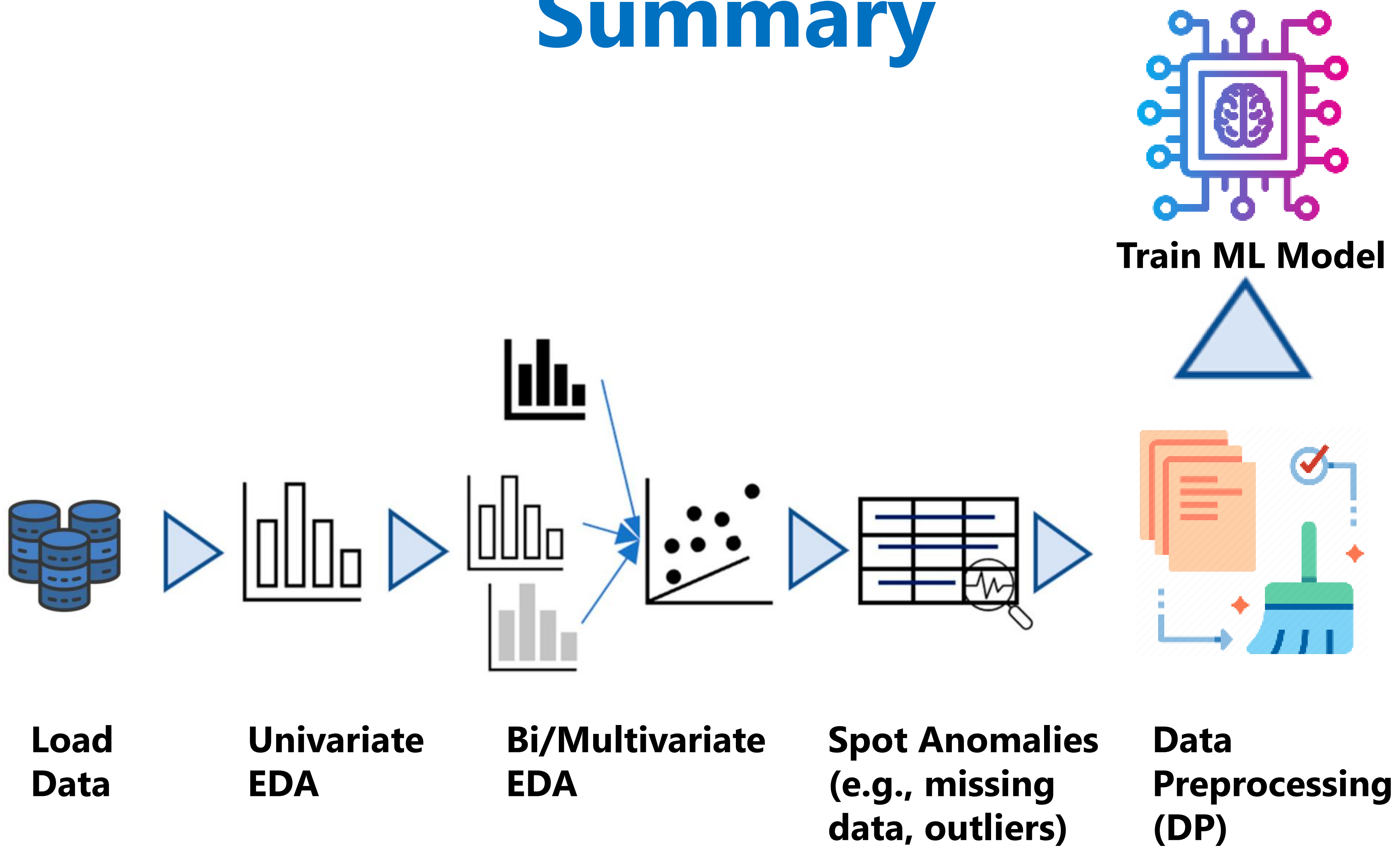
📝 Summary & EDA Checklist

- EDA is the **foundation** of any data project; skip it at your own risk.
- Start simple (**Univariate**), then explore relationships (**Bivariate**), and finally the big picture (**Multivariate**)
- **Visualizations** are friendlier and often more insightful than raw tables
- Always **document** your process to ensure reproducibility.

EDA Quick Reference Checklist:

1. **Import** libraries and load data
2. **Inspect** the data (`.info()`, `.describe()`, `.head()`)
3. **Clean** basics (handle obvious missing values or duplicates)
4. **Univariate:** Plot histograms/boxplots for numbers, countplots for categories
5. **Bivariate:** check correlations (scatter, heatmap, boxplots)
6. **Multivariate:** Pairplots and correlation matrices
7. **Summarize** findings and plan feature engineering

Summary



Train ML Model
Load Data
Univariate EDA
Bi/Multivariate EDA
Spot Anomalies (e.g., missing data, outliers)
Data Preprocessing (DP)