



## Welcome to this session: Tutorial - Data Cleaning, Preprocessing and EDA

The session will start shortly...

Questions? Drop them in the chat.  
We'll have dedicated moderators  
answering questions.



# Safeguarding & Welfare

We are committed to all our students and staff feeling safe and happy; we want to make sure there is always someone you can turn to if you are worried about anything.

If you are feeling upset or unsafe, are worried about a friend, student or family member, or you feel like something isn't right, speak to our safeguarding team:



Ian Wyles  
Designated Safeguarding  
Lead



Simone Botes



Nurhaan Snyman



Rafiq Manan



Ronald Munodawafa



Tevin Pitts

Scan to report a  
safeguarding concern



or email the Designated  
Safeguarding Lead:  
Ian Wyles  
[safeguarding@hyperiondev.com](mailto:safeguarding@hyperiondev.com)

# Skills Bootcamp Data Science

---

- The use of disrespectful language is prohibited in the questions, this is a supportive, learning environment for all - please engage accordingly. **(Fundamental British Values: Mutual Respect and Tolerance)**
- No question is daft or silly - **ask them!**
- There are **Q&A sessions** midway and at the end of the session, should you wish to ask any follow-up questions. Moderators are going to be answering questions as the session progresses as well.
- If you have any questions outside of this lecture, or that are not answered during this lecture, please do submit these for upcoming Academic Sessions. You can submit these questions here: [Questions](#)

# Skills Bootcamp Data Science

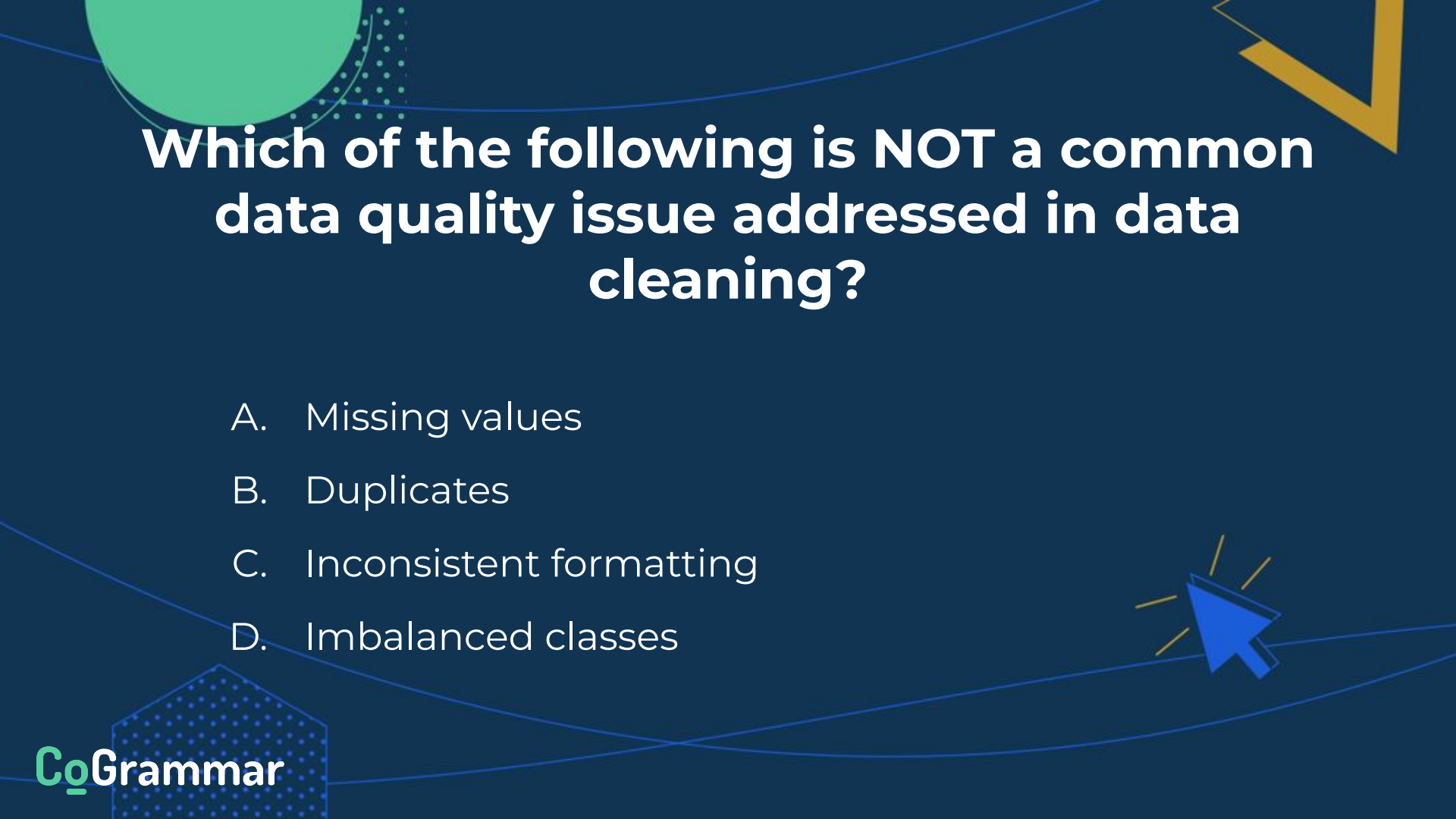
---

- For all **non-academic questions**, please submit a query:  
**[www.hyperiondev.com/support](http://www.hyperiondev.com/support)**
- **Report a safeguarding incident:** **[www.hyperiondev.com/safeguardreporting](http://www.hyperiondev.com/safeguardreporting)**
- We would love your feedback on lectures: **[Feedback on Lectures](#)**
- If you are hearing impaired, please kindly use your computer's function through Google chrome to enable captions.

## Learning Outcomes

---

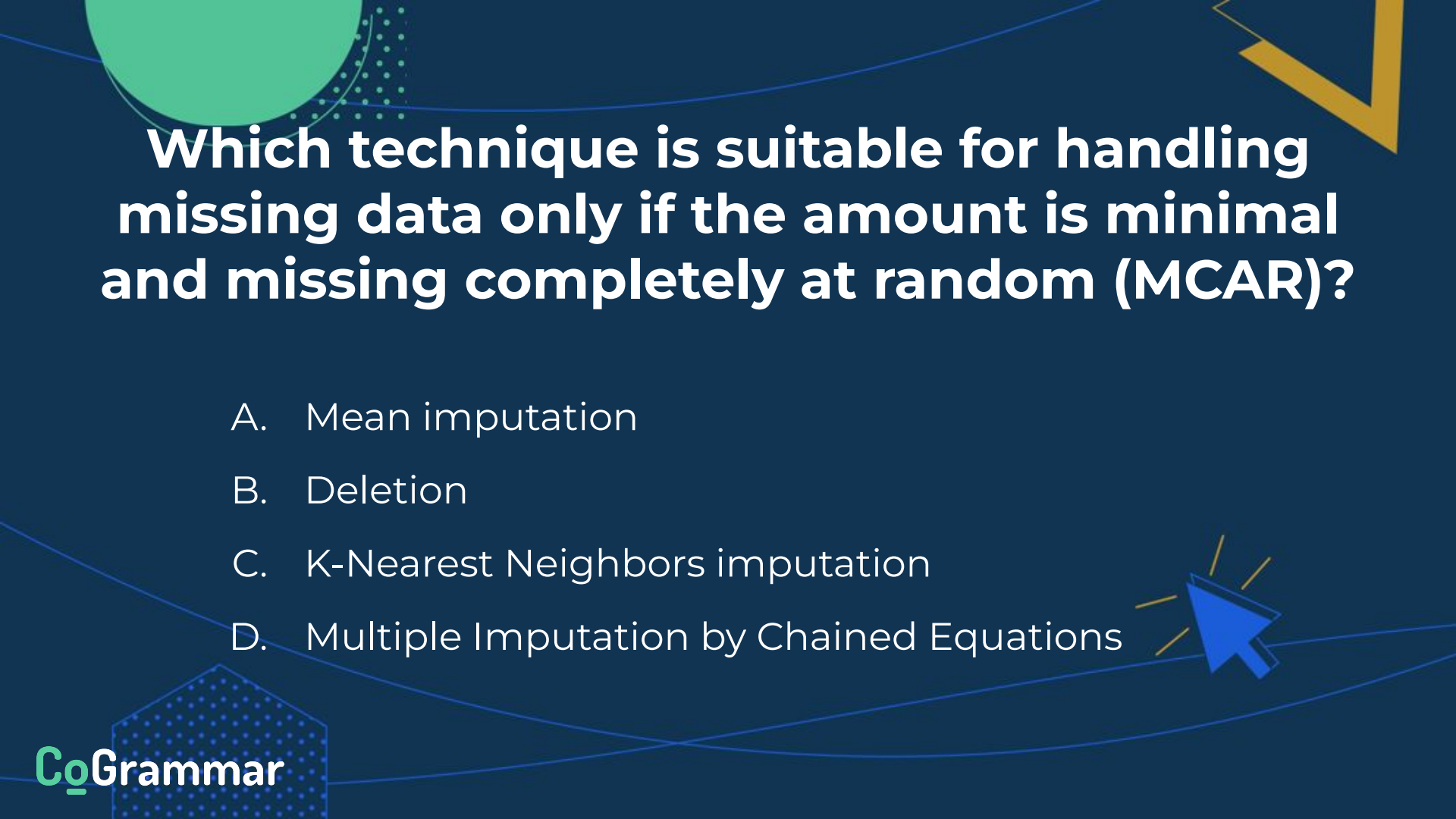
- ❖ **Clean and preprocess data** by handling whitespace, standardizing text, and formatting categorical variables.
- ❖ **Differentiate between standardization and normalization** and apply them appropriately.
- ❖ **Perform exploratory data analysis (EDA)** to detect patterns, outliers, and correlations in data.



# Which of the following is NOT a common data quality issue addressed in data cleaning?

- A. Missing values
- B. Duplicates
- C. Inconsistent formatting
- D. Imbalanced classes





**Which technique is suitable for handling missing data only if the amount is minimal and missing completely at random (MCAR)?**

- A. Mean imputation
- B. Deletion
- C. K-Nearest Neighbors imputation
- D. Multiple Imputation by Chained Equations

# Data Cleaning

- ❖ Data cleaning is a crucial step in the data science pipeline
- ❖ Ensures **data quality and reliability** for analysis and modeling
- ❖ Common data quality issues include **missing data, duplicates, inconsistent formatting, and outliers**



# Handling Missing Data

- ❖ Missing data refers to the **absence of values in one or more variables in a dataset.**
- ❖ Identifying missing values:
  - Look for **null, NaN, or empty cells in the dataset.**
  - Use functions like **isnull()** or **isna()** in Pandas

```
# Identify missing values
```

```
df_missing.isnull().sum()
```

```
✓ 0.0s
```

```
total_bill    10
```

```
tip           0
```

```
sex          10
```

```
smoker       0
```

```
day          0
```

```
time         0
```

```
size         0
```

```
dtype: int64
```

# Understand Missing Data Mechanisms

- ❖ **MCAR:** Missing Completely at Random (missingness unrelated to any variables)
  - Smoking status is not recorded in a random sample of patients
- ❖ **MAR:** Missing at Random (missingness depends on observed variables)
  - Smoking status is not documented in female patients because the doctor was too shy to ask 😊
- ❖ **MNAR:** Missing Not at Random (missingness depends on missing values themselves)
  - Smoking status is not recorded in patients admitted as an emergency, who are also more likely to have worse outcomes from surgery

# Techniques for Handling Missing Data

- ❖ **Deletion:** Remove records with missing values (only suitable if missing data is minimal and random).
  - Suitable for random missingness
  - Not the first resort, dropping data means losing some important context or skewing the dataset in some cases

```
df.shape
```

✓ 0.0s

```
(244, 7)
```

```
# Deletion: Remove records with missing values
df_deleted = df_missing.dropna()
df_deleted.shape
```

✓ 0.0s

```
(225, 7)
```

# Techniques for Handling Missing Data

❖ **Imputation:** Fill in missing values with estimated or calculated values.

➤ Simple imputation: Mean, median, or mode imputation

```
# Simple Imputation: Fill missing values with mean for numeric columns and mode for categorical  
# columns  
df_imputed = df_missing.copy()  
df_imputed['total_bill'] = df_imputed['total_bill'].fillna(df_imputed['total_bill'].mean())  
df_imputed['sex'] = df_imputed['sex'].fillna(df_imputed['sex'].mode()[0])
```

➤ Advanced imputation: K-Nearest Neighbors (KNN), Multiple Imputation by Chained Equations (MICE)

■ We'll get to KNN in another lecture 😊

# Dealing with Duplicates

- ❖ Identify duplicates using functions like `duplicated()` in Pandas

```
# Show all duplicated rows  
df_duplicates[df_duplicates.duplicated(keep=False)]
```

`keep = False` just marks all duplicates.

	total_bill	tip	sex	smoker	day	time	size
46	22.23	5.00	Male	No	Sun	Dinner	2
92	5.75	1.00	Female	Yes	Fri	Dinner	2
123	15.95	2.00	Male	No	Thur	Lunch	2
158	13.39	2.61	Female	No	Sun	Dinner	2
198	13.00	2.00	Female	Yes	Thur	Lunch	2
202	13.00	2.00	Female	Yes	Thur	Lunch	2
234	15.53	3.00	Male	Yes	Sat	Dinner	2
244	22.23	5.00	Male	No	Sun	Dinner	2
245	15.53	3.00	Male	Yes	Sat	Dinner	2
246	13.39	2.61	Female	No	Sun	Dinner	2
247	5.75	1.00	Female	Yes	Fri	Dinner	2
248	15.95	2.00	Male	No	Thur	Lunch	2

# Dealing with Duplicates

- ❖ Dropping duplicates is fine and encouraged, it does not cause the data to lost necessary context

```
# Remove duplicate records  
df_deduplicated = df_duplicates.drop_duplicates()
```

# Data Formatting and Standardization

- ❖ Consistent data formatting is essential for accurate analysis and compatibility
- ❖ Common formatting issues:
  - **Date and time formats:** Ensure consistent representation (e.g., YYYY-MM-DD, HH:MM:SS)
  - **Text case inconsistencies:** Convert text to a consistent case (lowercase or uppercase)
  - **Inconsistent value representations:** Standardize values (e.g., "Yes"/"No" vs. "Y"/"N")

# Data Formatting and Standardization

- ❖ Techniques for standardizing data:
  - Convert date/time columns using `to_datetime()`
  - Convert text case using `str.lower()` or `str.upper()`
  - Map inconsistent values to standardized representations

```
# Standardize text case for 'sex' and 'smoker' columns
df['sex'] = df['sex'].str.upper()
df['smoker'] = df['smoker'].str.title()
```

```
df.head()
```

✓ 0.0s

	total_bill	tip	sex	smoker	day	time	size
0	16.99	1.01	FEMALE	No	Sun	Dinner	2
1	10.34	1.66	MALE	No	Sun	Dinner	3
2	21.01	3.50	MALE	No	Sun	Dinner	3
3	23.68	3.31	MALE	No	Sun	Dinner	2
4	24.59	3.61	FEMALE	No	Sun	Dinner	4



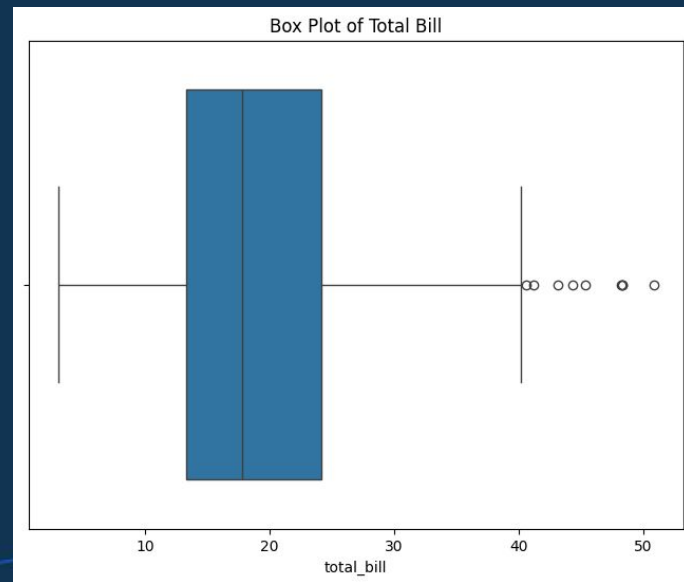
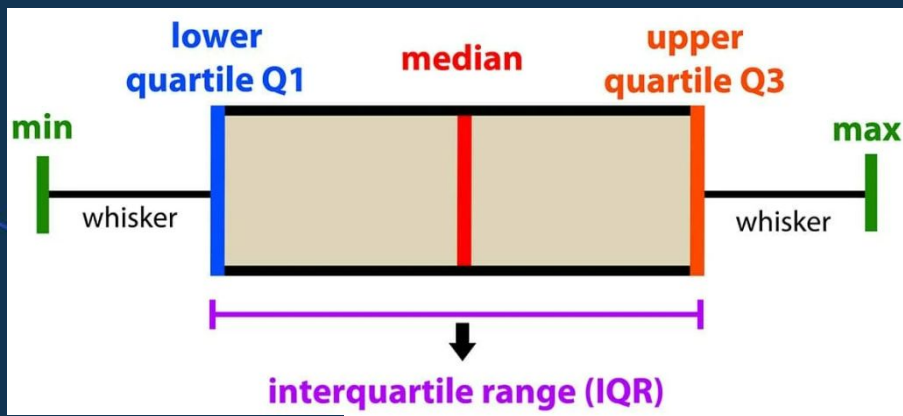
# Outliers

- ❖ Outliers are data points that significantly deviate from the rest of the data distribution



# Identifying Outliers

- ❖ Visual inspection using plots like box plots, scatter plots, or histograms



# Identifying Outliers

- ❖ Statistical methods like z-score or interquartile range (IQR)
  - Much less common given how good box plot already show outliers

```
# Identify outliers using z-score
from scipy import stats

z_scores = np.abs(stats.zscore(df['total_bill']))
threshold = 2.5
outliers_zscore = np.where(z_scores > threshold)

outliers_zscore
```

✓ 0.0s

```
(array([ 59, 102, 156, 170, 182, 197, 212]),)
```

```
# Identify outliers using Interquartile Range (IQR)
Q1 = df['total_bill'].quantile(0.25)
Q3 = df['total_bill'].quantile(0.75)
IQR = Q3 - Q1

outliers_iqr = df[(df['total_bill'] < (Q1 - 1.5 * IQR)) | (df['total_bill'] > (Q3 + 1.5 * IQR))]
len(outliers_iqr)
```

✓ 0.0s

9

MagicPython

# Handling Outliers

- ❖ **Removal:** Remove outliers if they are erroneous or irrelevant to the analysis
  - Use when outliers are clearly erroneous or irrelevant to the analysis
  - Be cautious, as removing outliers may result in loss of information

```
# Removal: Remove outliers
```

```
df_removed = df[~((df['total_bill'] < (Q1 - 1.5 * IQR)) | (df['total_bill'] > (Q3 + 1.5 * IQR)))]
```

```
df_removed.shape
```

```
✓ 0.0s
```

```
MagicPython
```

```
(235, 9)
```

# Handling Outliers

- ❖ **Transformation:** Apply mathematical transformations (e.g., logarithmic, square root) to reduce the impact of outliers
  - Use when outliers are valid but have a skewed distribution
  - Helps to reduce the impact of outliers while preserving the data

# Handling Outliers

- ◆ **Winsorization:** Replace extreme values with the nearest non-outlier values
  - Use when outliers are valid but need to be treated to reduce their influence
  - Maintains the overall distribution shape while limiting the extreme values

# Iterating

- ❖ Data cleaning is an iterative process that may require multiple rounds
- ❖ Continuously assess and refine the cleaned data based on analysis results and feedback
- ❖ Integrate data cleaning with data analysis and modeling for optimal results



# fuzzywuzzy

- ❖ Provides string matching and similarity scoring functions
- ❖ Key features:
  - **Ratio:** Calculates the similarity ratio between two strings
  - **Partial Ratio:** Calculates the similarity ratio considering substrings
  - **Token Set Ratio:** Calculates the similarity ratio considering common tokens



# fuzzywuzzy

```
!pip3 install fuzzywuzzy python-Levenshtein
```

```
from fuzzywuzzy import fuzz
```

```
fuzz.ratio("apple", "appel")
```

✓ 0.0s

80

```
fuzz.partial_ratio("apple", "app")
```

✓ 0.0s

100

*# Gives a 100 if every token in the first string is in the second string*  

```
fuzz.token_set_ratio("apple orange", "orange apple")
```

✓ 0.0s

100

# chardet

- ❖ Detects the encoding of a given byte string
- ❖ Key features:
  - Supports various encodings (e.g., UTF-8, ISO-8859-1, etc.)
  - Provides confidence scores for detected encodings

```
!pip3 install chardet
```

```
import chardet
```

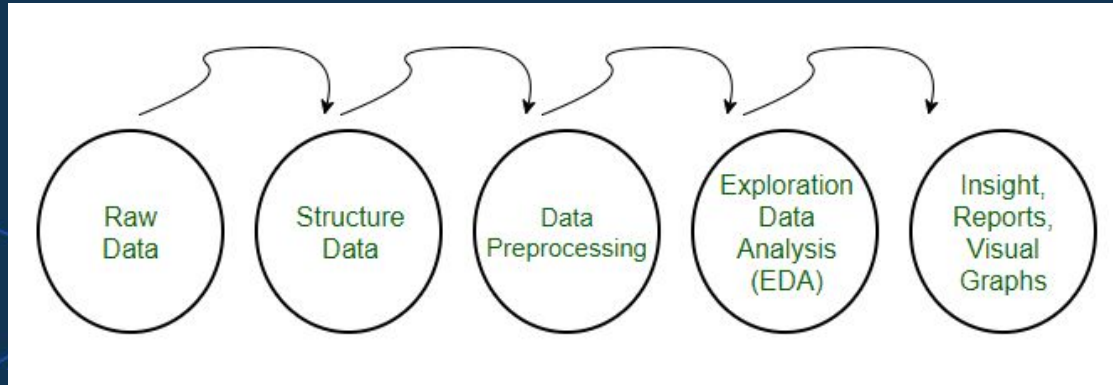
```
byte_string = b"Hello, World!"  
encoding = chardet.detect(byte_string)  
encoding
```

✓ 0.0s

```
{'encoding': 'ascii', 'confidence': 1.0, 'language': ''}
```

# Data Preprocessing

- ❖ Data preprocessing is a crucial step in the data science pipeline, going beyond basic cleaning to ensure data quality and suitability for machine learning.



Source: [GeeksForGeeks](https://www.geeksforgeeks.org/)



# Importance

- ❖ **Improved data quality:** Addresses complex issues beyond basic cleaning
- ❖ **Enhanced model performance:** Optimizes data for learning algorithms
- ❖ **Reduced computational complexity:** Reduces dimensionality and creates efficient representations

# Feature Scaling

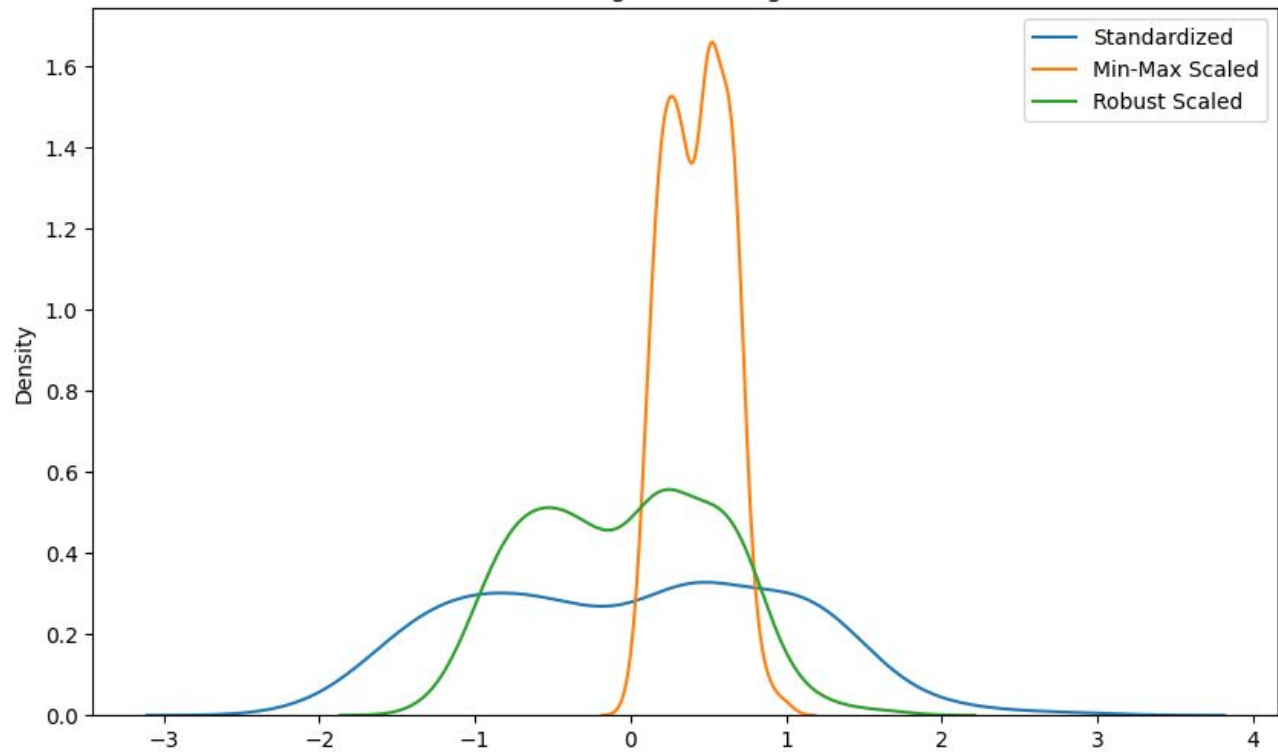
- ❖ Purpose: Ensure fair comparison and contribution of features
- ❖ Techniques:
  - **Standardization (Z-score normalization):** Transforms features to have zero mean and unit variance
$$X' = \frac{X - \mu}{\sigma}$$
  - **Min-max scaling:** Scales features to a specific range, typically 0 to 1
$$X' = \frac{X - X_{min}}{X_{max} - X_{min}}$$
  - **Robust scaling:** Uses robust statistics (median and interquartile range) to scale features
    - $(X - \text{median}) / \text{IQR}$



# Considerations

- ❖ **Standardization:** Good default, assumes Gaussian distribution
- ❖ **Min-max scaling:** Suitable for bounded features or non-Gaussian data
- ❖ **Robust scaling:** Recommended when outliers are present

Effects of Scaling on Bill Length Distribution





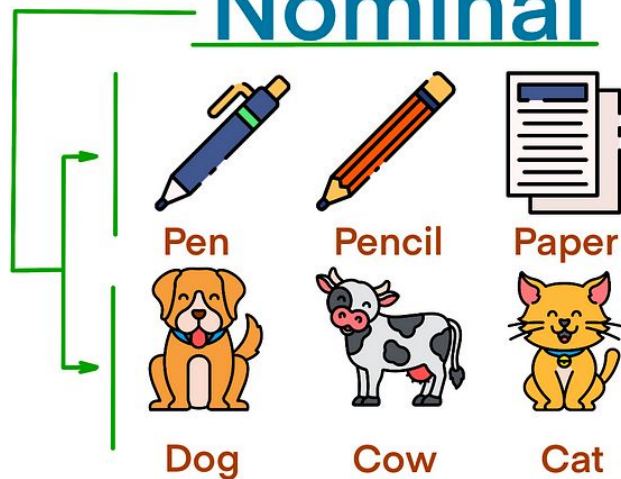
# Nominal vs. Ordinal

- ❖ **Nominal:** Categories without inherent order (e.g., color)
- ❖ **Ordinal:** Categories with meaningful order (e.g., size)



# Categorical

## Nominal



## Ordinal



# Encoding Nominal variables

- ❖ One-hot encoding: Creates binary dummy variables for each category
  - Increases dimensionality, which may impact model performance

Index	Animal	One-Hot code	Index	Dog	Cat	Sheep	Lion	Horse
0	Dog		0	1	0	0	0	0
1	Cat	➔	1	0	1	0	0	0
2	Sheep		2	0	0	1	0	0
3	Horse		3	0	0	0	0	1
4	Lion		4	0	0	0	1	0

Source: [AnalyticsVidhya](#)

# Encoding Nominal variables

- ❖ Binary encoding: Assigns unique binary codes to categories
  - Useful when the number of categories is large, and one-hot encoding leads to high dimensionality

	City	City_0	City_1	City_2	City_3
0	Delhi	0	0	0	1
1	Mumbai	0	0	1	0
2	Hyderabad	0	0	1	1
3	Chennai	0	1	0	0
4	Bangalore	0	1	0	1
5	Delhi	0	0	0	1
6	Hyderabad	0	0	1	1
7	Mumbai	0	0	1	0
8	Agra	0	1	1	0

Source: [AnalyticsVidhya](https://www.analyticsvidhya.com)

# Encoding Ordinal variables

- ❖ Label encoding: Assigns numerical labels based on order
  - Maintains ordinal information but implies linear relationships between categories
  - May not be appropriate if the ordinal relationship is not linear

Degree	
0	1
1	4
2	2
3	3
4	3
5	4
6	5
7	1
8	1

Source: [AnalyticsVidhya](#)

# Encoding Ordinal variables

- ❖ Ordinal encoding: Assigns numerical labels based on order
  - Preserves ordinal information without implying linear relationships
  - Suitable when the ordinal relationship between categories is meaningful

Degree	
0	1
1	4
2	2
3	3
4	3
5	4
6	5
7	1
8	1

Source: [AnalyticsVidhya](#)

# Handling High-Cardinality

## ❖ The Curse of Dimensionality:

As the number of features grows, the amount of data we need to accurately be able to distinguish between these features (in order to give us a prediction) and generalize our model (learned function) grows EXPONENTIALLY.

Source: [AnalyticsVidhya](#)

# Frequency-based Encoding

- ❖ Replaces categories with occurrence count
- ❖ Useful when the frequency of categories is informative

# Target Encoding

- ❖ Replaces categories with mean/median of target variable
- ❖ Captures the relationship between categories and the target variable

	class	Marks
0	A,	50
1	B	30
2	C	70
3	B	80
4	C	45
5	A	97
6	A	80
7	A	68

	class
0	65.000000
1	57.689414
2	59.517061
3	57.689414
4	59.517061
5	79.679951
6	79.679951
7	79.679951

Source: [AnalyticsVidhya](https://www.analyticsvidhya.com/blog/2021/06/target-encoding/)



# Hashing

- ❖ Applies hash function to reduce dimensionality
- ❖ Useful when the number of categories is extremely large

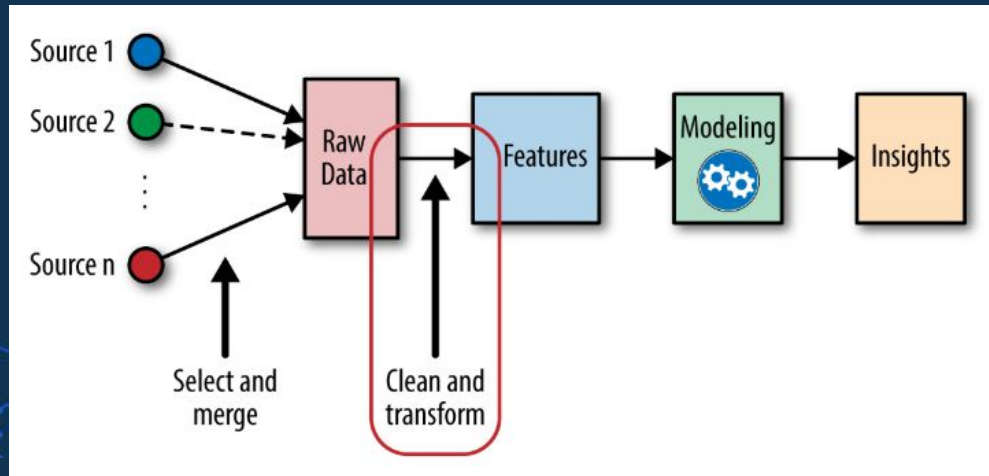
	Month
0	January
1	April
2	March
3	April
4	February
5	June
6	July
7	June
8	September

	col_0	col_1	col_2	col_3	col_4	col_5
0	0	0	0	0	1	0
1	0	0	0	1	0	0
2	0	0	0	0	1	0
3	0	0	0	1	0	0
4	0	0	0	1	0	0
5	0	1	0	0	0	0
6	1	0	0	0	0	0
7	0	1	0	0	0	0
8	0	0	0	0	1	0

Source: [AnalyticsVidhya](https://www.analyticsvidhya.com/)

# Feature Engineering

- ❖ Create informative features that improve model performance and interpretability



Source: [AnalyticsVidhya](https://www.analyticsvidhya.com)

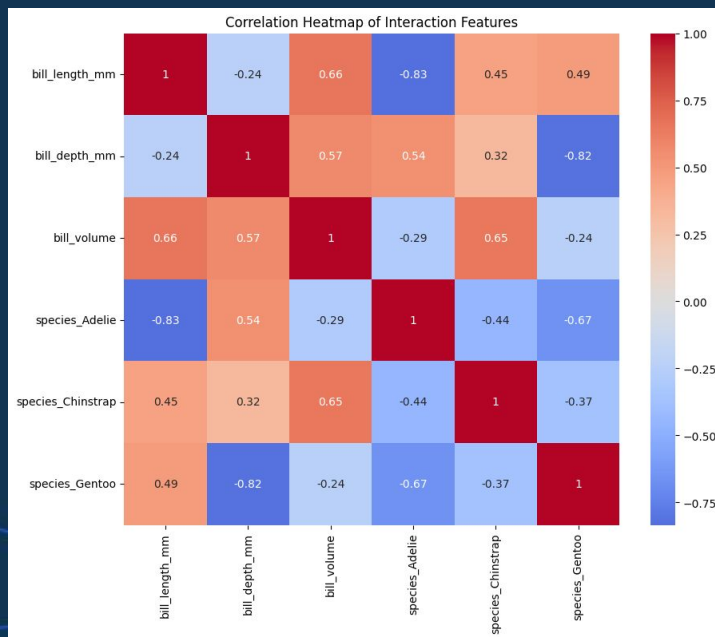
# Techniques

- ❖ **Interaction features:** Combine existing features to capture interactions
- ❖ **Polynomial features:** Generate higher-order terms to capture non-linear relationships
- ❖ **Domain-specific features:** Apply domain knowledge to create meaningful features

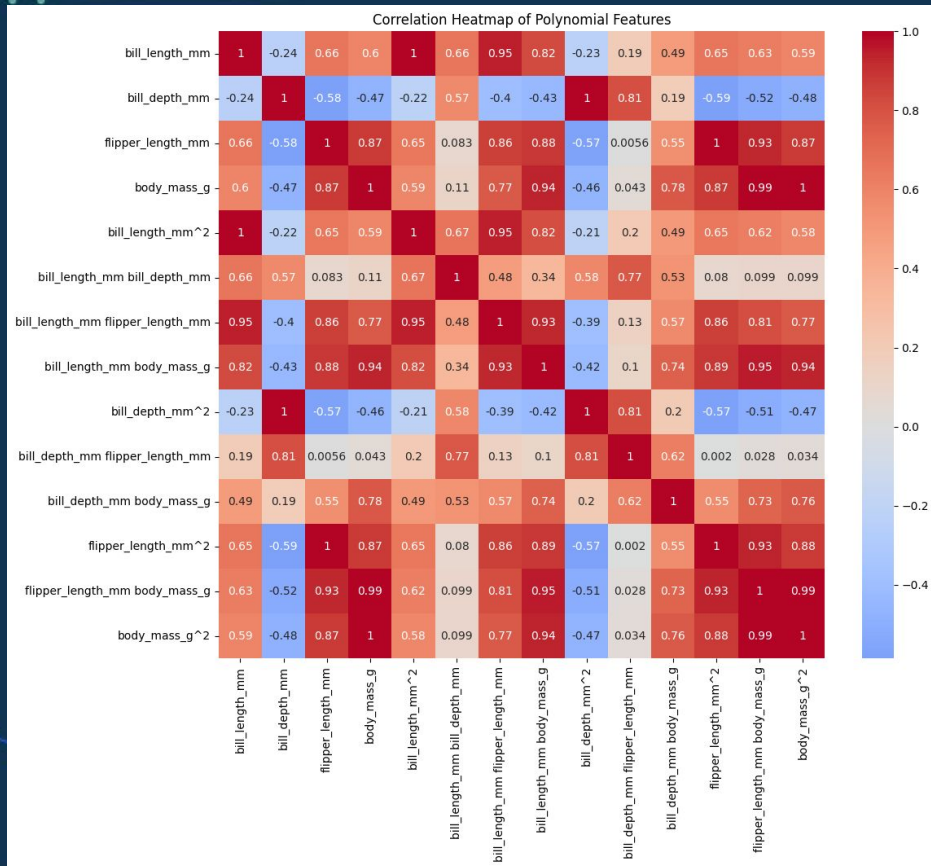
# Interaction Features

```
# Interaction features
```

```
penguins['bill_volume'] = penguins['bill_length_mm'] * penguins['bill_depth_mm']
```



# Polynomial Features



# Challenge

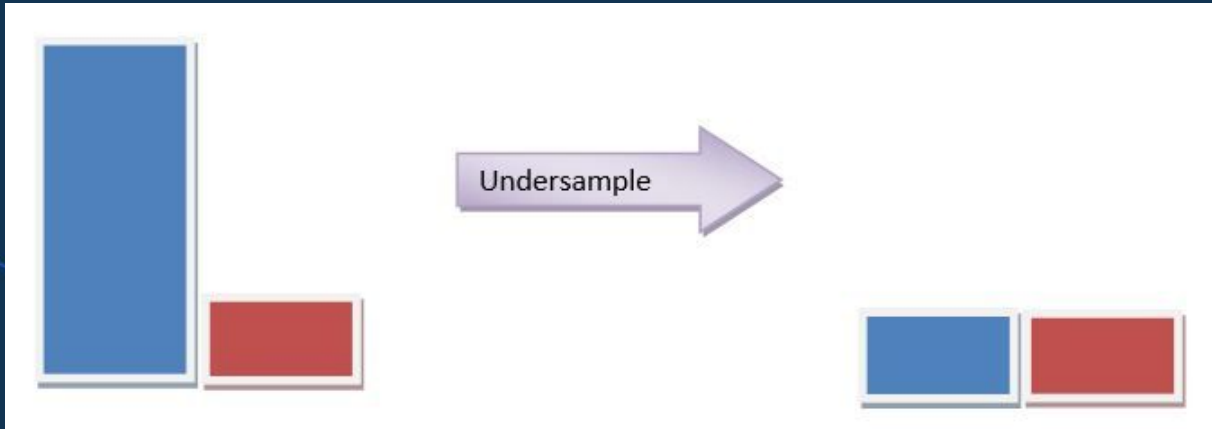
- ❖ Skewed class distribution leads to biased models and poor minority class performance



Source: [AnalyticsVidhya](https://www.analyticsvidhya.com)

# Techniques

- ❖ **Undersampling:** Reduce majority class instances
  - **Random undersampling:** Remove majority instances



Source: [AnalyticsVidhya](https://www.analyticsvidhya.com)

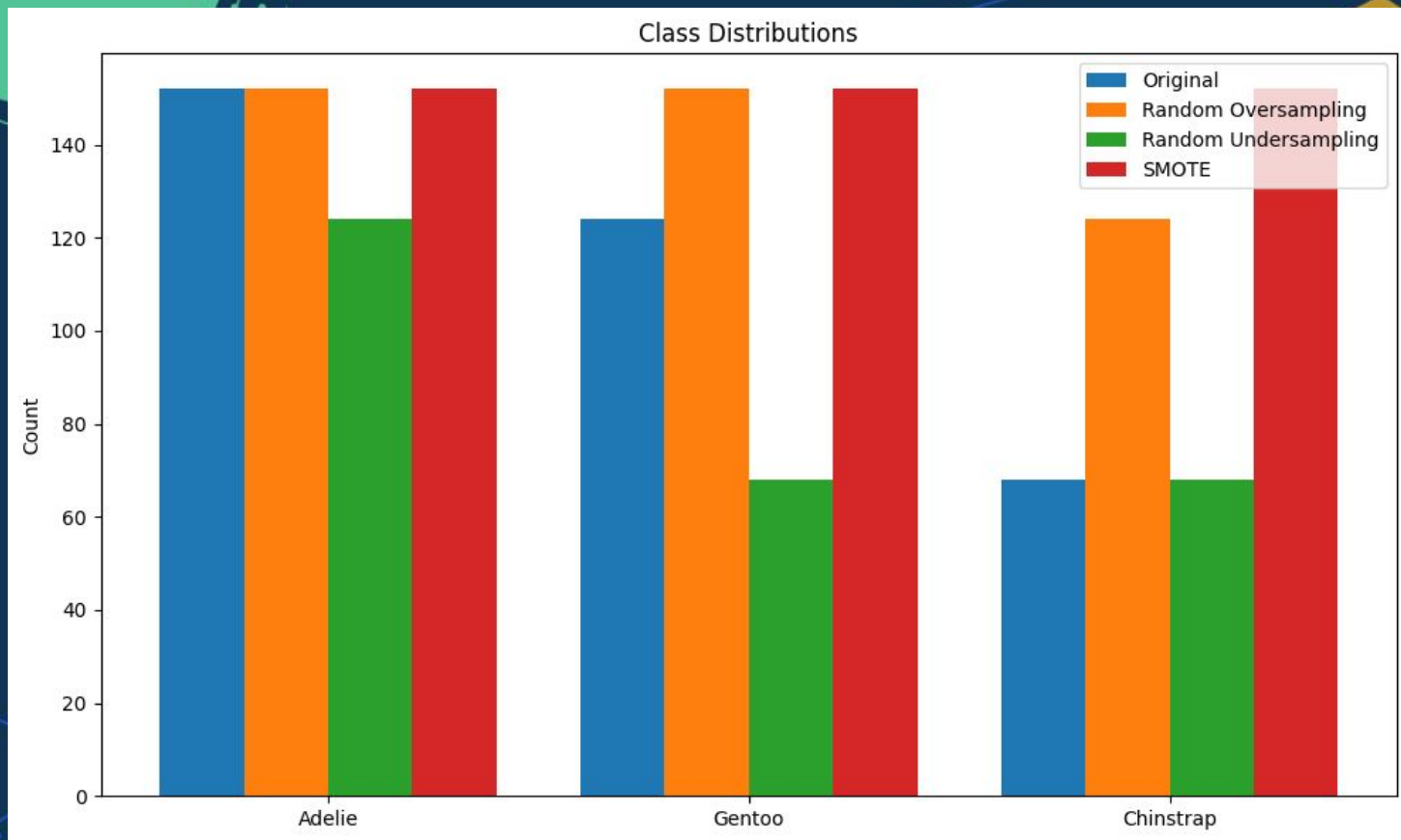
# Techniques

- ❖ **Oversampling:** Increase minority class instances
  - **Random oversampling:** Duplicate minority instances
  - **SMOTE:** Generate synthetic minority instances



Source: [AnalyticsVidhya](https://www.analyticsvidhya.com)





# Considerations

- ❖ Oversampling may lead to overfitting, especially with random oversampling
- ❖ Undersampling may discard potentially useful data



# Introduction to EDA

- ❖ **Definition:** Exploratory Data Analysis (EDA) is the process of investigating and understanding a dataset through visual and statistical techniques.
- ❖ **Importance:** EDA is a crucial first step in any data science project as it helps uncover patterns, anomalies, and relationships in the data, guiding further analysis and decision-making.






# Introduction to EDA

- ❖ **Role in the data science workflow:** EDA is performed after data collection and before model building and evaluation. It helps in understanding the data, identifying data quality issues, and selecting relevant features for modeling.



# Simple EDA Framework


## ➤ Explore Relationships:

- Analyse relationships between features and the target variable
  - Use visualisations like scatter plots, pair plots, and correlation matrices
  - Identify patterns, trends, and clusters in the data
- 



# Simple EDA Framework

## ➤ **Assess Feature Importance:**

- Determine the significance of features using statistical tests
  - Use techniques like Decision Trees or Random Forests to evaluate feature importance
  - Select relevant features based on their importance and domain knowledge
- 

# Simple EDA Framework

## ➤ Iterate and Refine:

- Iterate on the analysis based on the insights gained
- Refine the data cleaning and preprocessing steps if necessary
- Consider additional visualisations or techniques to deepen the understanding of the data



# Loading and Exploring the Dataset

- ❖ To demonstrate EDA techniques, we'll use the Iris dataset from scikit-learn.
- ❖ The Iris dataset consists of measurements of sepal length, sepal width, petal length, and petal width for three species of Iris flowers.
- ❖ We'll load the dataset using scikit-learn and create a pandas DataFrame to work with.







Source: [Wikipedia](https://en.wikipedia.org/wiki/Iris_(flower))





```
# Load the Iris dataset
```

```
iris = load_iris()
```

```
data = pd.DataFrame(data=iris.data, columns=iris.feature_names)
```

```
data['species'] = iris.target_names[iris.target]
```



# Loading and Exploring the Dataset

- ❖ After loading the dataset, we'll explore its basic properties:
  - **Shape of the dataset:** number of rows and columns
  - **Features:** the independent variables in the dataset
  - **Target variable:** the dependent variable (depends on the features) we want to predict or analyse



Dataset shape: (150, 6)

Features: Index(['sepal length (cm)', 'sepal width (cm)', 'petal length (cm)',  
'petal width (cm)', 'species'],  
dtype='object')

Target variable: Cluster

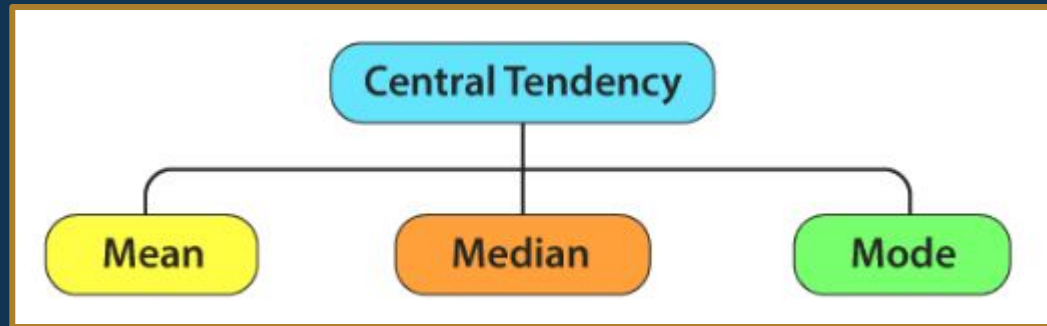
	sepal length (cm)	sepal width (cm)	petal length (cm)	petal width (cm)	\
0	5.1	3.5	1.4	0.2	
1	4.9	3.0	1.4	0.2	
2	4.7	3.2	1.3	0.2	
3	4.6	3.1	1.5	0.2	
4	5.0	3.6	1.4	0.2	

	species	Cluster
0	setosa	1
1	setosa	1
2	setosa	1
3	setosa	1
4	setosa	1

# Univariate Analysis

- ❖ Univariate analysis involves analysing each variable individually.
  - **Univariate:** involving one variate or variable quantity.
- ❖ We'll start by calculating descriptive statistics for the numeric variables using the `describe()` function.
- ❖ Descriptive statistics provide a summary of the central tendency, dispersion, and shape of the data.

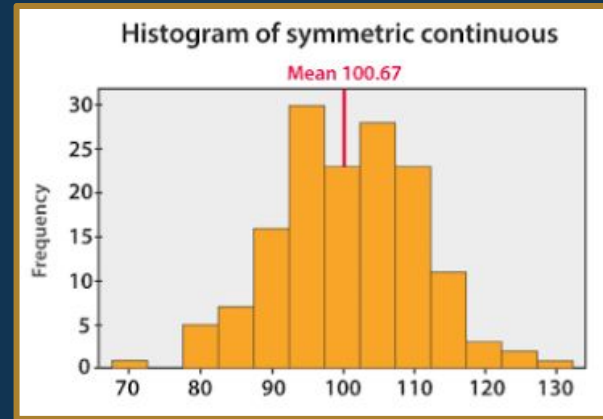
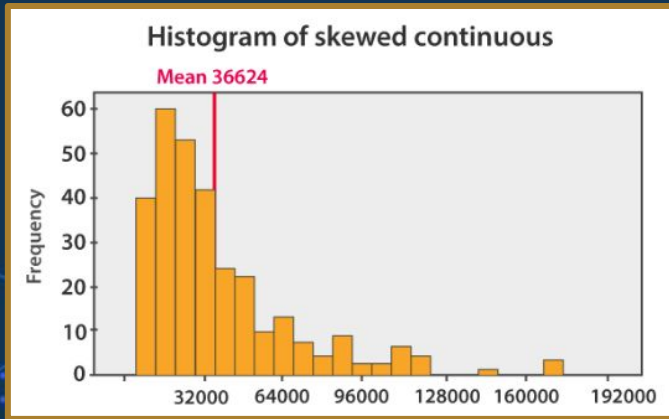
# Measures of Central Tendency





# Measures of Central Tendency

- ❖ **Mean** represents the average value of the dataset. It can be calculated as the sum of all the values in the dataset divided by the number of values.



# Measures of Central Tendency

- ❖ **Median** is the middle value of the dataset in which the dataset is arranged in the ascending order or in descending order. When the dataset contains an even number of values, then the median value of the dataset can be found by taking the mean of the middle two values.

Median odd
23
21
18
16
15
13
12
10
9
7
6
5
2

Median even
40
38
35
33
32
30
29
27
26
24
23
22
19
17

28



# Measures of Central Tendency

- ❖ **Mode** represents the frequently occurring value in the dataset. Sometimes the dataset may contain multiple modes and in some cases, it does not contain any mode at all.

Mode
5
5
5
4
4
3
2
2
1

# Univariate Analysis

```
# Univariate Analysis
```

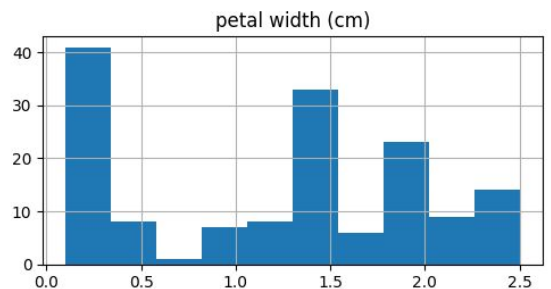
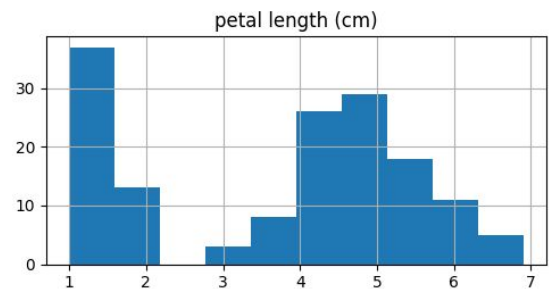
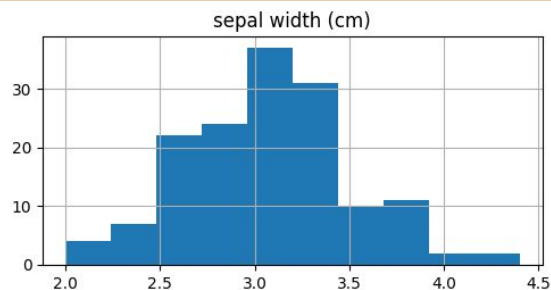
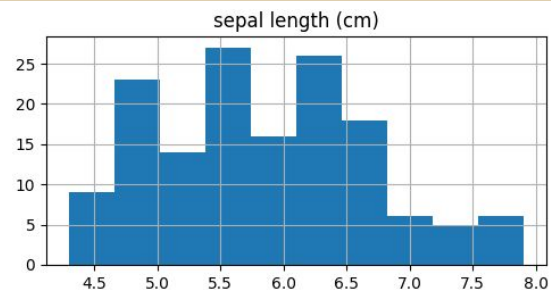
```
data.describe()
```

✓ 0.0s

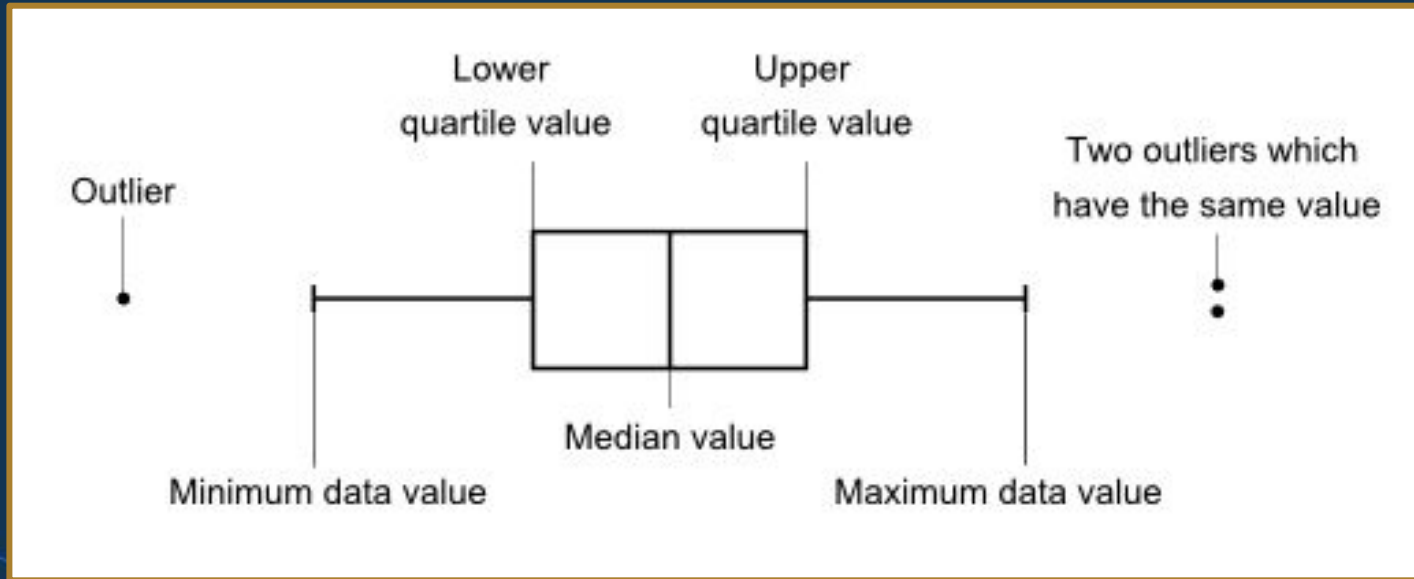
	sepal length (cm)	sepal width (cm)	petal length (cm)	petal width (cm)	Cluster
count	150.000000	150.000000	150.000000	150.000000	150.000000
mean	5.843333	3.057333	3.758000	1.199333	0.333333
std	0.828066	0.435866	1.765298	0.762238	0.472984
min	4.300000	2.000000	1.000000	0.100000	0.000000
25%	5.100000	2.800000	1.600000	0.300000	0.000000
50%	5.800000	3.000000	4.350000	1.300000	0.000000
75%	6.400000	3.300000	5.100000	1.800000	1.000000
max	7.900000	4.400000	6.900000	2.500000	1.000000

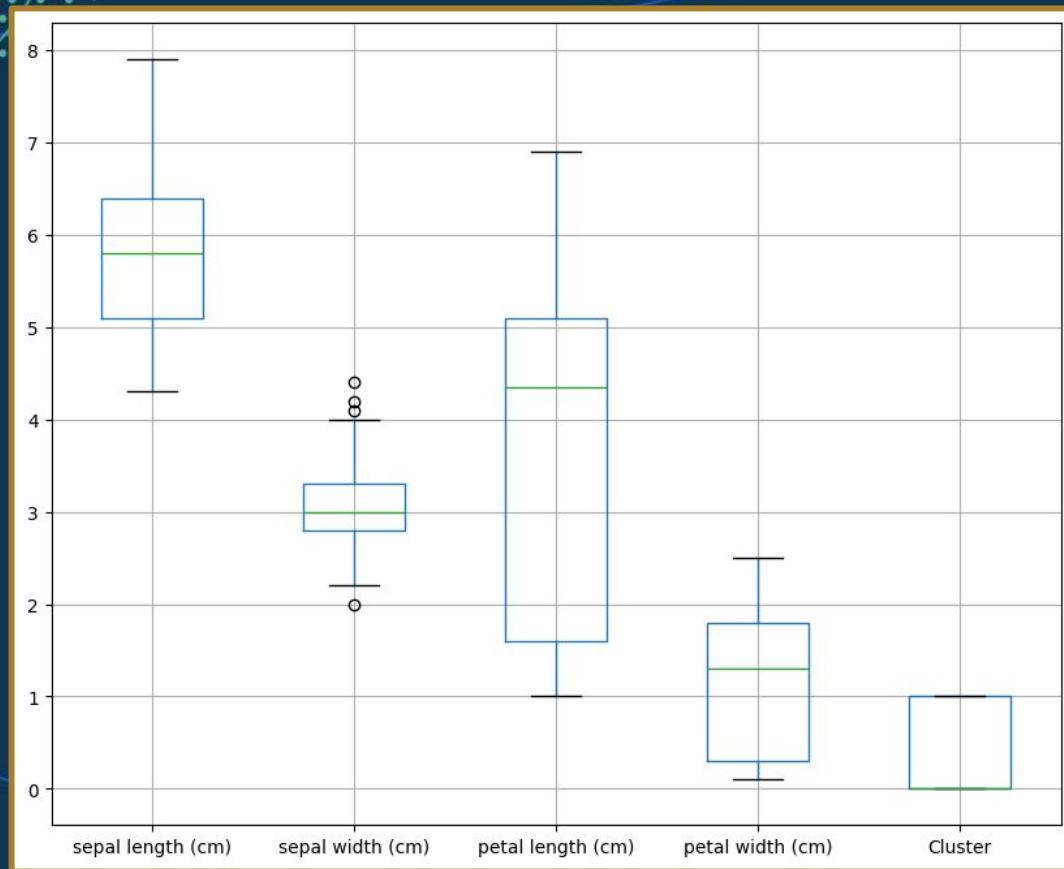
# Univariate Analysis

- ❖ Next, we'll visualise the distribution of each feature using histograms and box plots.
- ❖ **Histograms** show the frequency distribution of a variable, helping to identify the shape, central tendency, and spread of the data.
- ❖ **Box plots** provide a summary of the distribution, highlighting the median, quartiles, and outliers.



# Box Plots





# Univariate Analysis

- ❖ We'll also check for missing values in the dataset using the `isnull().sum()` function.
- ❖ Missing values can impact the analysis and need to be handled appropriately.
  - Common strategies include filling missing values with the mean, median, or mode.
  - It's usually not a good idea to just drop data, as this could skew the data and thus the results of prediction.




```
Missing values: sepal length (cm)    0
sepal width (cm)                    0
petal length (cm)                   0
petal width (cm)                    0
species                             0
Cluster                             0
dtype: int64
```





# Bivariate Analysis

- ❖ Bivariate analysis involves examining the relationship between two variables.
    - **Bivariate:** involving or depending on two variates.
  - ❖ We'll use scatter plots to visualise the relationship between features and the target variable.
  - ❖ **Scatter plots** help identify patterns, correlations, and clusters in the data.
- 



```
# Bivariate Analysis
```

```
sns.pairplot(data, hue='species')
```

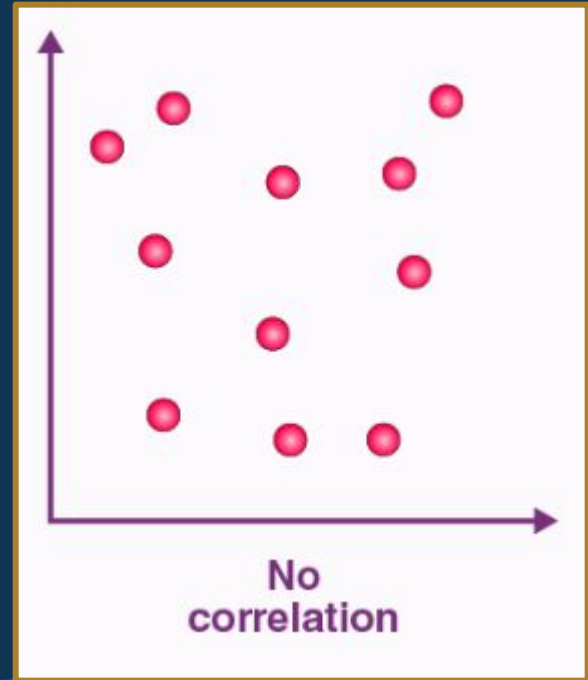
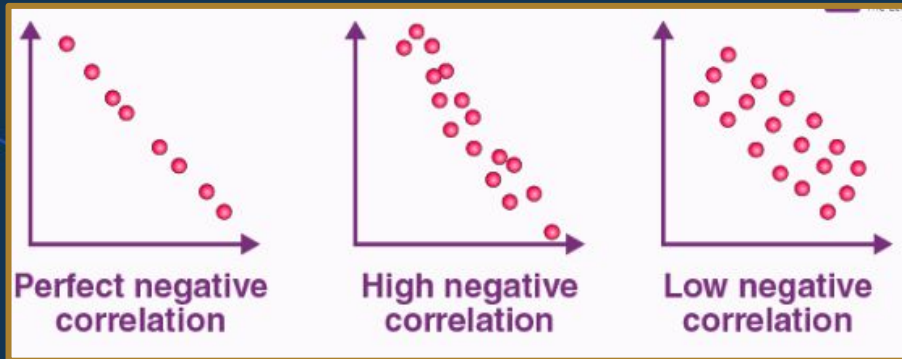
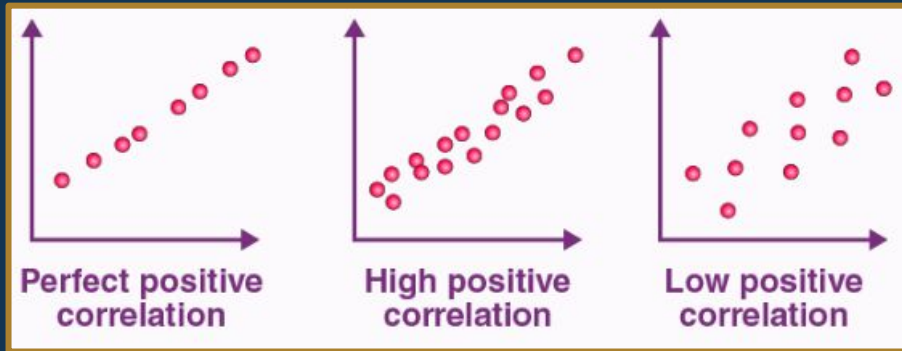
```
plt.show()
```

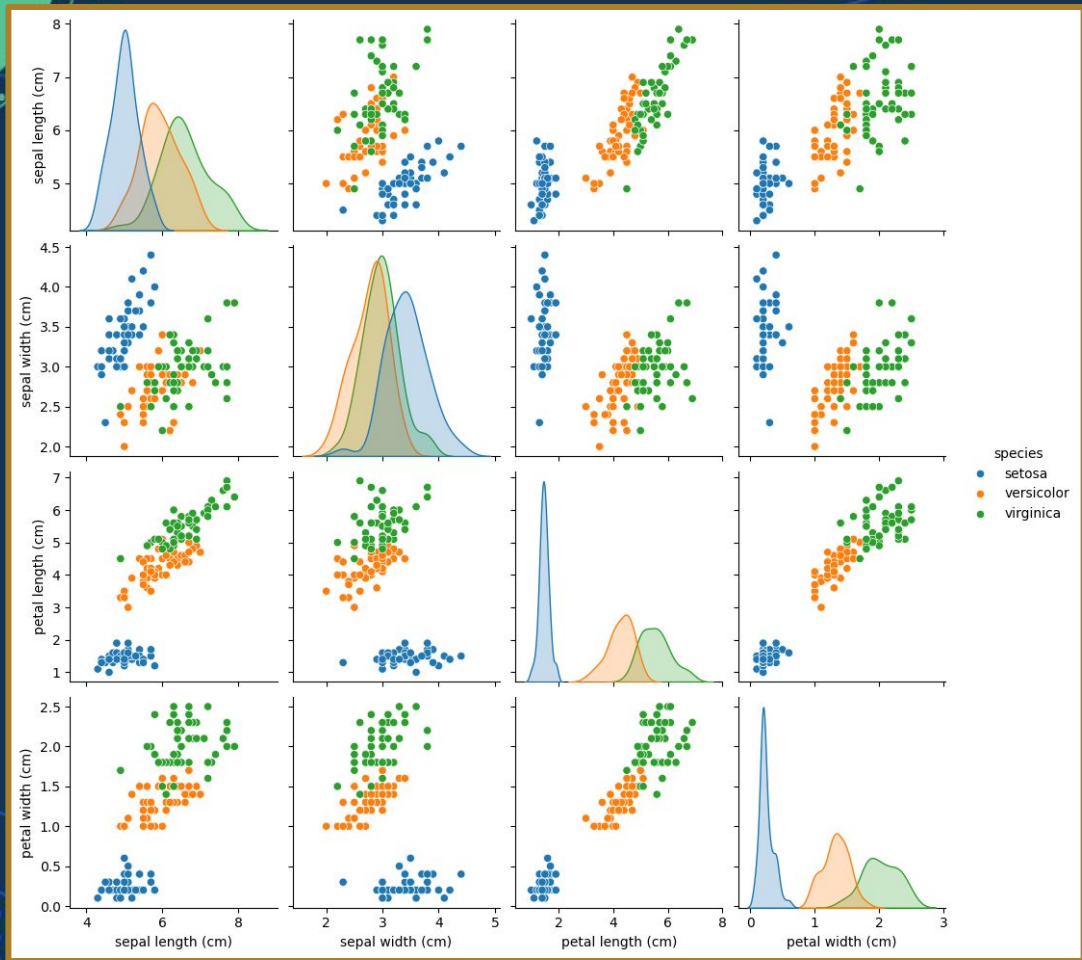
```
corr_matrix = data.iloc[:, :-1].corr()
```

```
sns.heatmap(corr_matrix, annot=True, cmap='coolwarm')
```

```
plt.show()
```

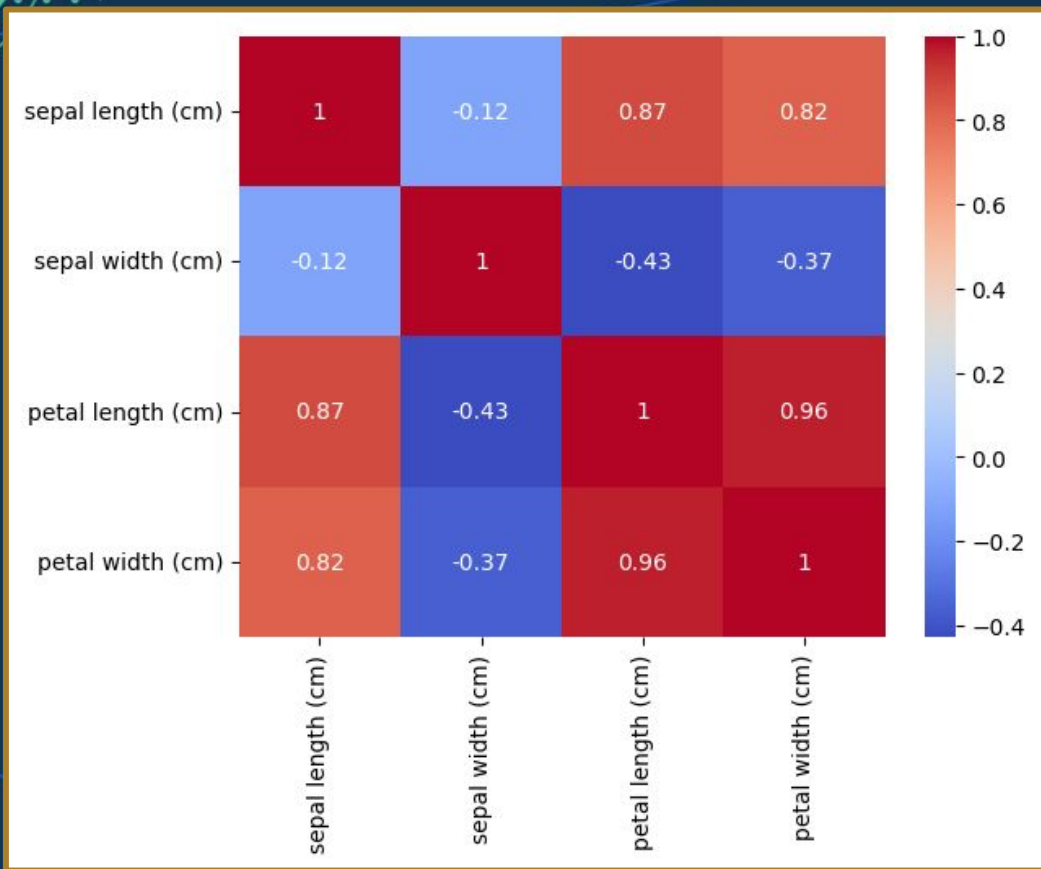
# Scatter Plot





# Bivariate Analysis

- ❖ To quantify the relationship between numeric features, we'll calculate the correlation matrix.
- ❖ The correlation matrix shows the pairwise correlation coefficients between variables.
- ❖ We'll visualise the correlation matrix using a heatmap.






# Bivariate Analysis

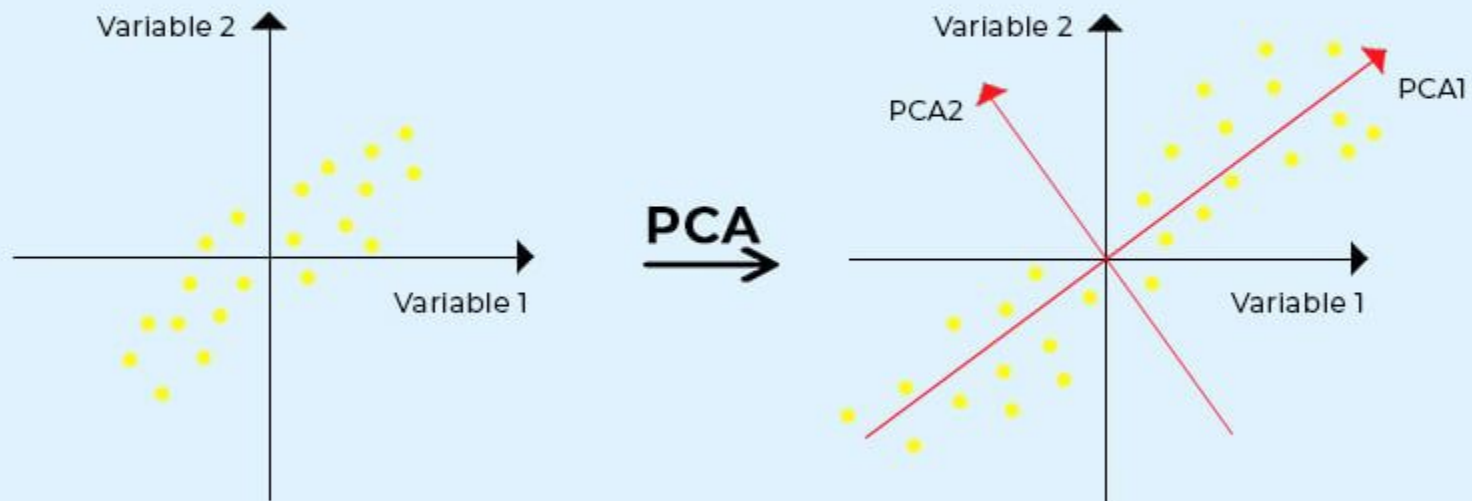
- ❖ Interpreting the correlation matrix:
  - Correlation coefficients range from -1 to 1.
  - The high positive correlations between Petal Length and Petal Width (0.96) and between Sepal Length and Petal Length (0.87) suggest that these pairs of features are strongly related and may provide similar information.
  - The low correlations between Sepal Width and the other features indicate that Sepal Width provides relatively independent information compared to the other features.



# Multivariate Analysis - PCA

- ❖ Multivariate analysis involves examining relationships among multiple variables simultaneously.
  - ❖ **Principal Component Analysis (PCA)** is a dimensionality reduction technique that transforms the original features into a new set of uncorrelated features called principal components.
  - ❖ PCA helps identify patterns and structure in high-dimensional data by finding the directions of maximum variance.
- 






Source: [AnalytixLabs](https://www.analytixlabs.com)



# PCA Steps (Python abstracts all the math)



1. Standardize the data to ensure all features have zero mean and unit variance.
  2. Compute the covariance matrix of the standardized data.
  3. Calculate the eigenvectors and eigenvalues of the covariance matrix.
  4. Sort the eigenvectors in descending order of their corresponding eigenvalues.
  5. Select the top k eigenvectors as the principal components.
  6. Transform the original data into the new feature space defined by the principal components.
- 

```
from sklearn.preprocessing import StandardScaler
from sklearn.decomposition import PCA

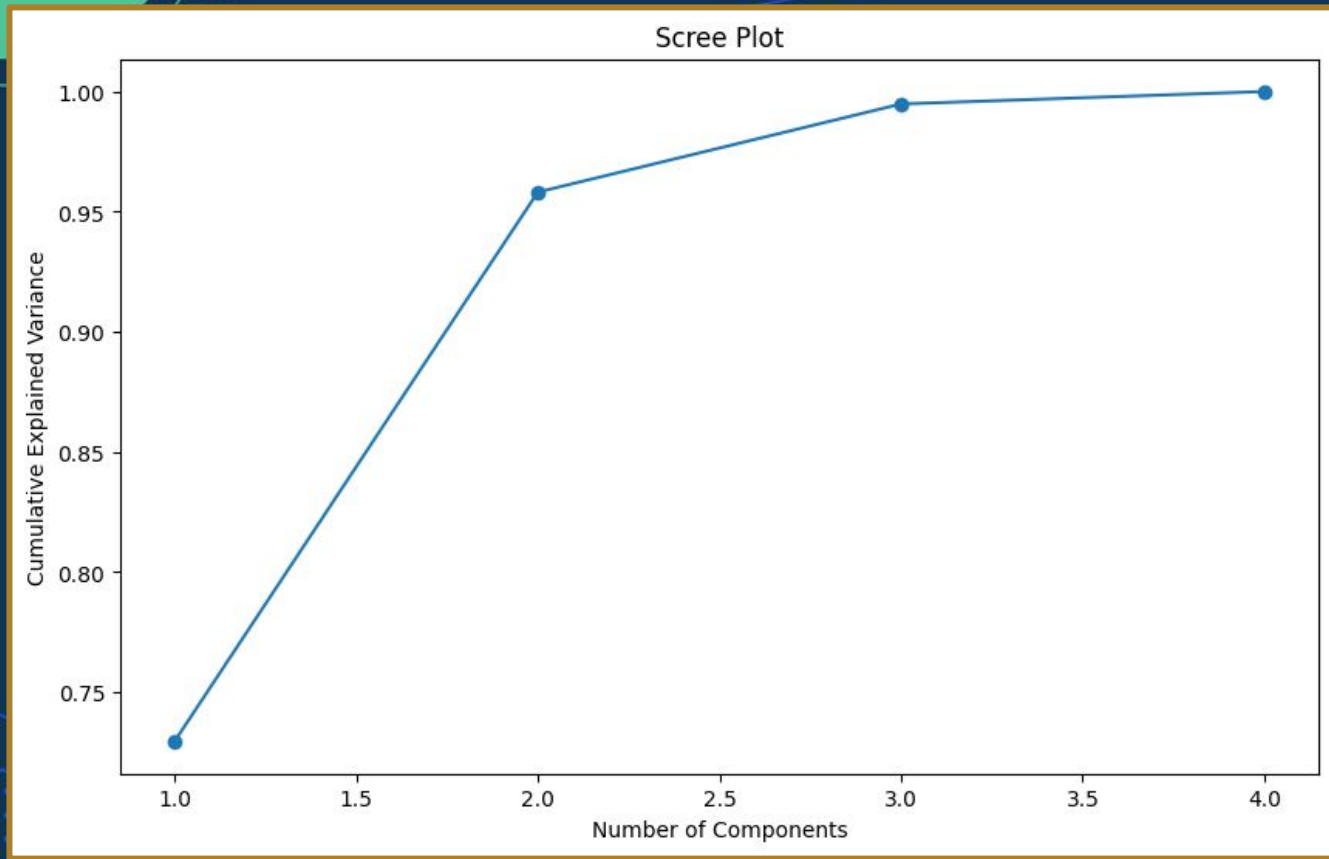
X = data.iloc[:, :-1]
y = data.iloc[:, -1]

scaler = StandardScaler()
X_scaled = scaler.fit_transform(X)

pca = PCA()
principalComponents = pca.fit_transform(X_scaled)
```

# PCA

- ❖ To determine the number of principal components to retain, we can analyse the explained variance ratio.
- ❖ The explained variance ratio represents the proportion of variance explained by each principal component.
- ❖ We can visualise the cumulative explained variance using a scree plot.



# Scree Plot

- ❖ Interpreting the scree plot:
  - Look for an elbow point where the cumulative explained variance starts to plateau.
  - Choose the number of components that capture a significant portion of the total variance (e.g., 80-90%).
  - In our case it seems to be 2 components.




# Multivariate Analysis - K-means Clustering


- ❖ K-means clustering is an unsupervised learning algorithm that partitions the data into K clusters based on similarity.
- ❖ It aims to minimize the within-cluster sum of squares (WCSS) or the Euclidean distance between data points and their cluster centroids.
- ❖ **More about this in later lectures.**








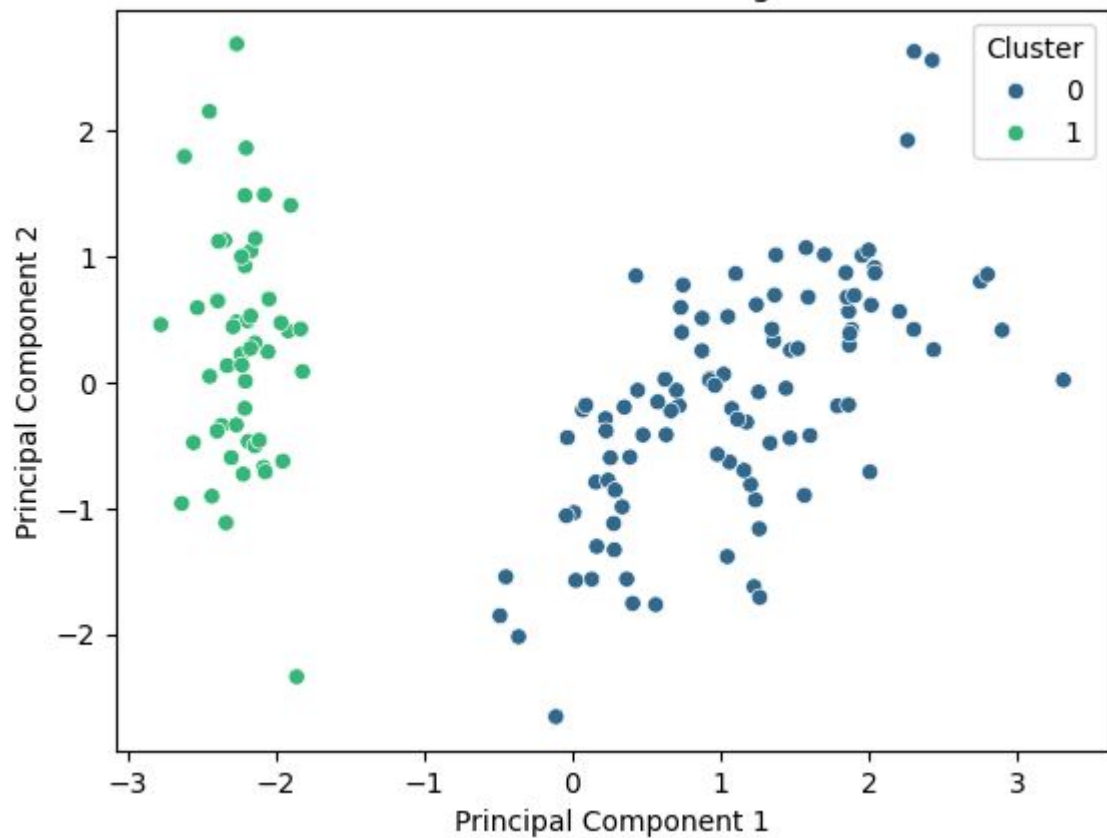
# K-means Clustering Steps (Python abstracts the math)



1. Choose the number of clusters  $K$ .
  2. Initialize  $K$  cluster centroids randomly.
  3. Assign each data point to the nearest centroid based on Euclidean distance.
  4. Update the cluster centroids by taking the mean of the data points assigned to each cluster.
  5. Repeat steps 3 and 4 until convergence or a maximum number of iterations is reached.
- 



## K-means Clustering



# Multivariate Analysis - K-means Clustering

- ❖ Interpreting the clustering results:
  - Observe the separation and compactness of the clusters.
  - Analyse the characteristics of data points within each cluster.
  - Consider the domain knowledge and interpret the meaning of the clusters.
    - In our case it seems like 2 plant species are very alike and one is distinctly different (setosa).



# Feature Importance

- ❖ Feature importance refers to the relative contribution of each feature in predicting the target variable.
- ❖ The prediction of a model is only as good as the features used to make the prediction, thus we want the most important predictors.
- ❖ We'll assess feature importance using statistical tests and machine learning techniques.

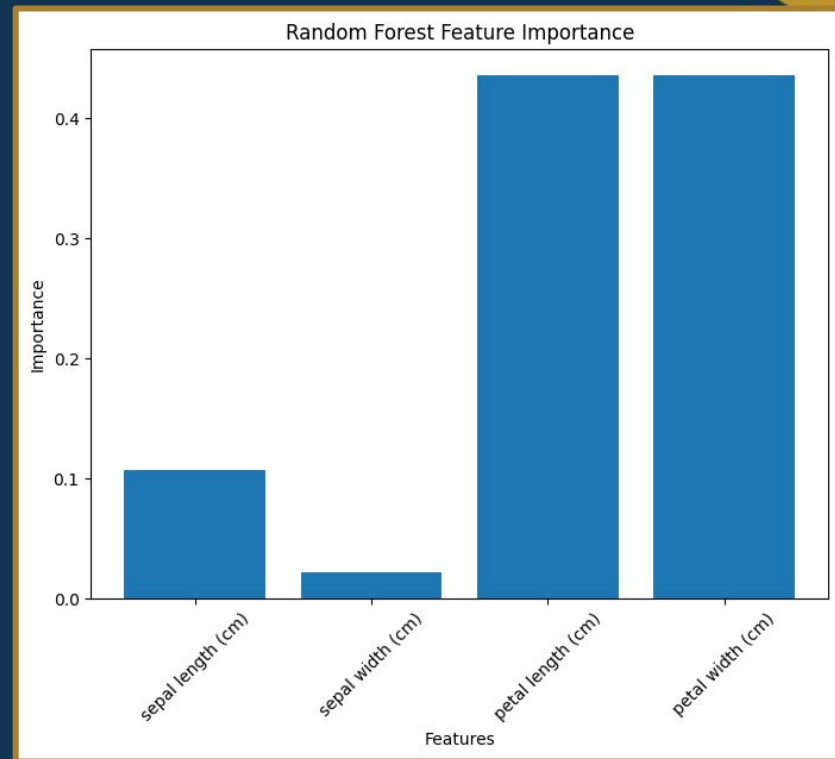
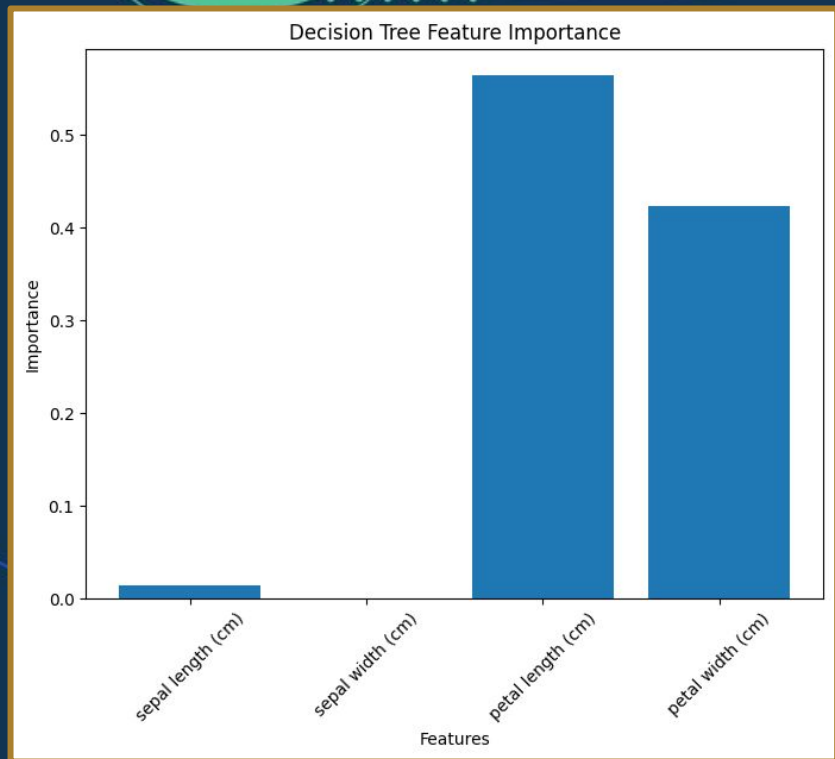




# Decision Trees and Random Forests

- ❖ Decision Trees and Random Forests are machine learning algorithms that can provide feature importance scores.
- ❖ The importance score represents the decrease in impurity or increase in information gain achieved by splitting on a particular feature.
- ❖ **More on these in dedicated lectures later on.**







# In Pandas, which function can be used to identify duplicate records in a dataset?

- A. `find_duplicates()`
- B. `duplicated()`
- C. `is_duplicate()`
- D. `has_duplicates()`



# Which of the following is a technique for standardizing inconsistent text case?

- A. `astype()`
- B. `to_datetime()`
- C. `str.upper()` or `str.lower()`
- D. `strip()`

## Summary

---

### ★ **Data Cleaning:**

Converting text to lowercase and removing whitespace for consistency.  
Standardizing categorical values to ensure uniform data representation.  
Handling missing values using appropriate imputation methods.  
Identifying and removing duplicate records to maintain data integrity.

### ★ **Data Preprocessing:**

Understanding the difference between standardization and normalization:  
One-hot encoding for categorical variables without inherent order.  
Label encoding for ordinal categorical variables.

### ★ **Exploratory Data Analysis (EDA):**

Computing summary statistics using `.describe()`.  
Visualizing data distributions using histograms and boxplots to detect outliers.  
Analyzing correlations with heatmaps to identify relationships.



# CoGrammar

## Q & A SECTION

**Please use this time to ask  
any questions relating to the  
topic, should you have any.**

# Thank you for attending



**CoGrammar**



Department  
for Education