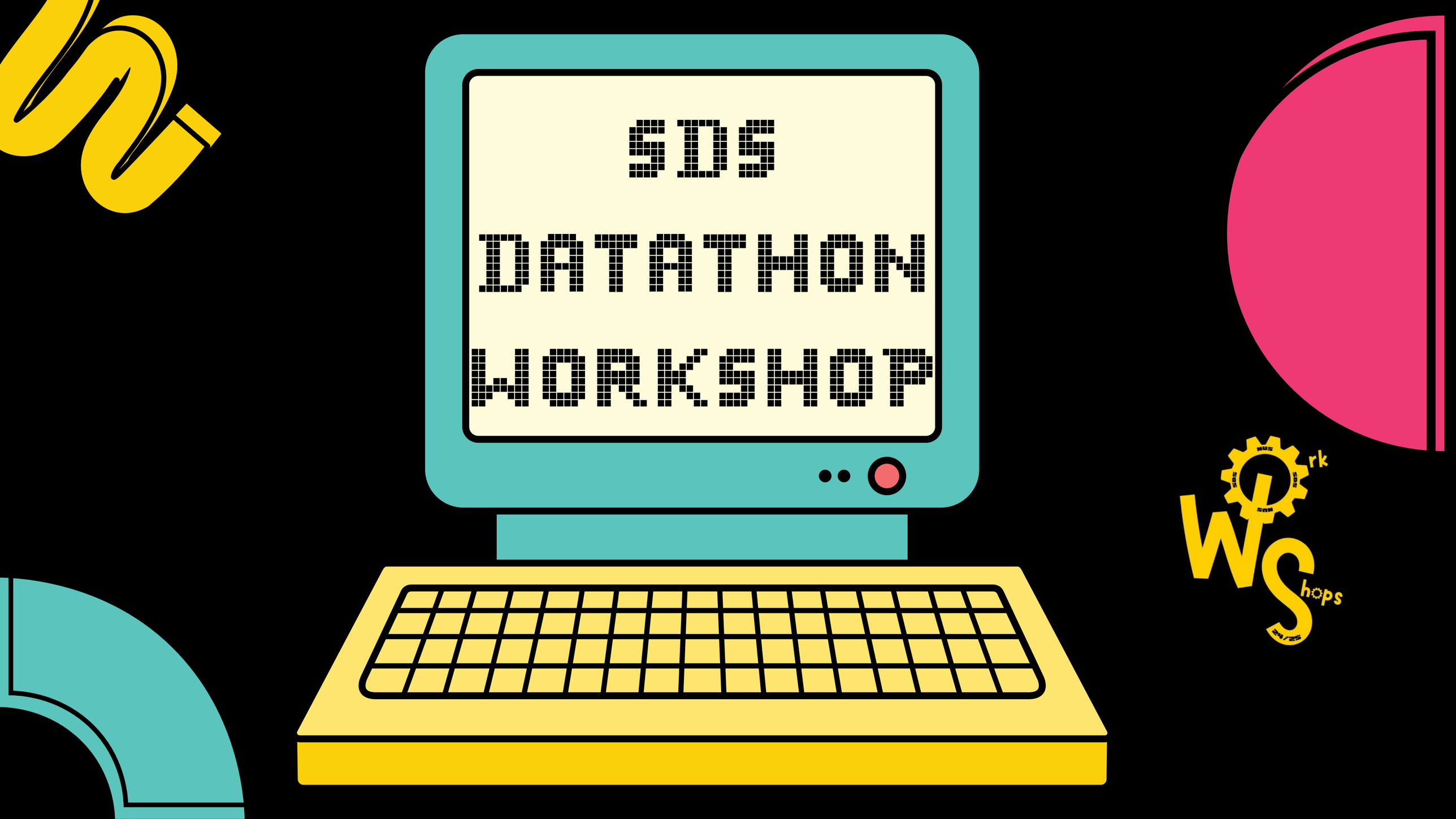

SDS
DATATHON
WORKSHOP
NUS
SDS
WS
hops
24/25

INTRODUCTION

Akshata

Clarence

DISCLAIMER

As this is only a 2 hour workshop, we will be unable to cover everything required from start to finish (its literally all of SDS' first semester workshops put together → so join our workshops next time).

However we have tried our best to touch upon all the necessary techniques and information for you to further learn by yourselves even post this workshop.

Content from previous workshops:

INTRO TO DATATHON WORKFLOW

Rarely comes with clean, ready-to-model data

Standard Workflow

load → Read data, inspect shape, columns, basic statistics

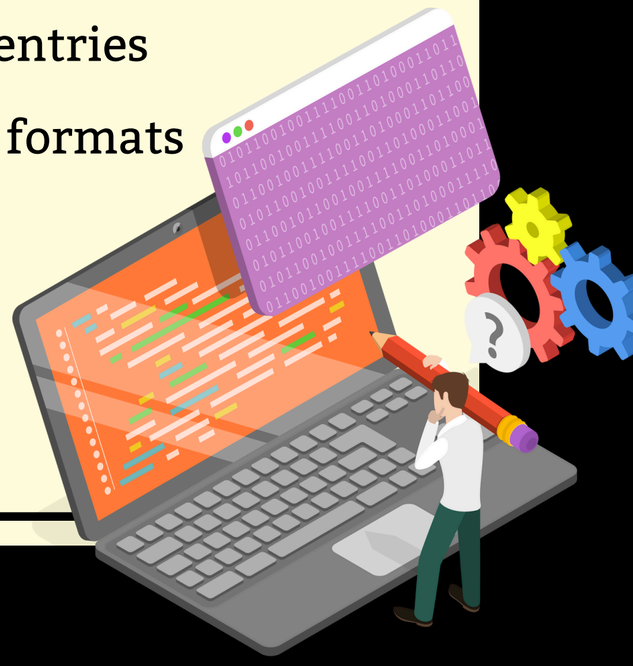
validate → Check data types, missing values, duplicates, invalid entries

clean → Fix inconsistencies, handle missingness, standardize formats

transform → Feature engineering, aggregations, encodings

analyze → EDA, pattern discovery, segmentation

model → Apply ML or NLP methods where appropriate



INTRO TO DATATHON WORKFLOW

- Datathons are non-linear and decision driven
- You will constantly trade off between:
  - speed vs correctness
  - coverage vs depth
  - simplicity vs sophistication
- Expect to loop back multiple times
  - Inspect → Question → Decide → Implement → Validate → Repeat

INTRO TO DATATHON WORKFLOW

- Mandatory Questions to Answer
  - What is one row supposed to represent?
  - Which columns are identifiers vs features?
  - Which columns cannot be trusted yet?
  - What information is missing and why?

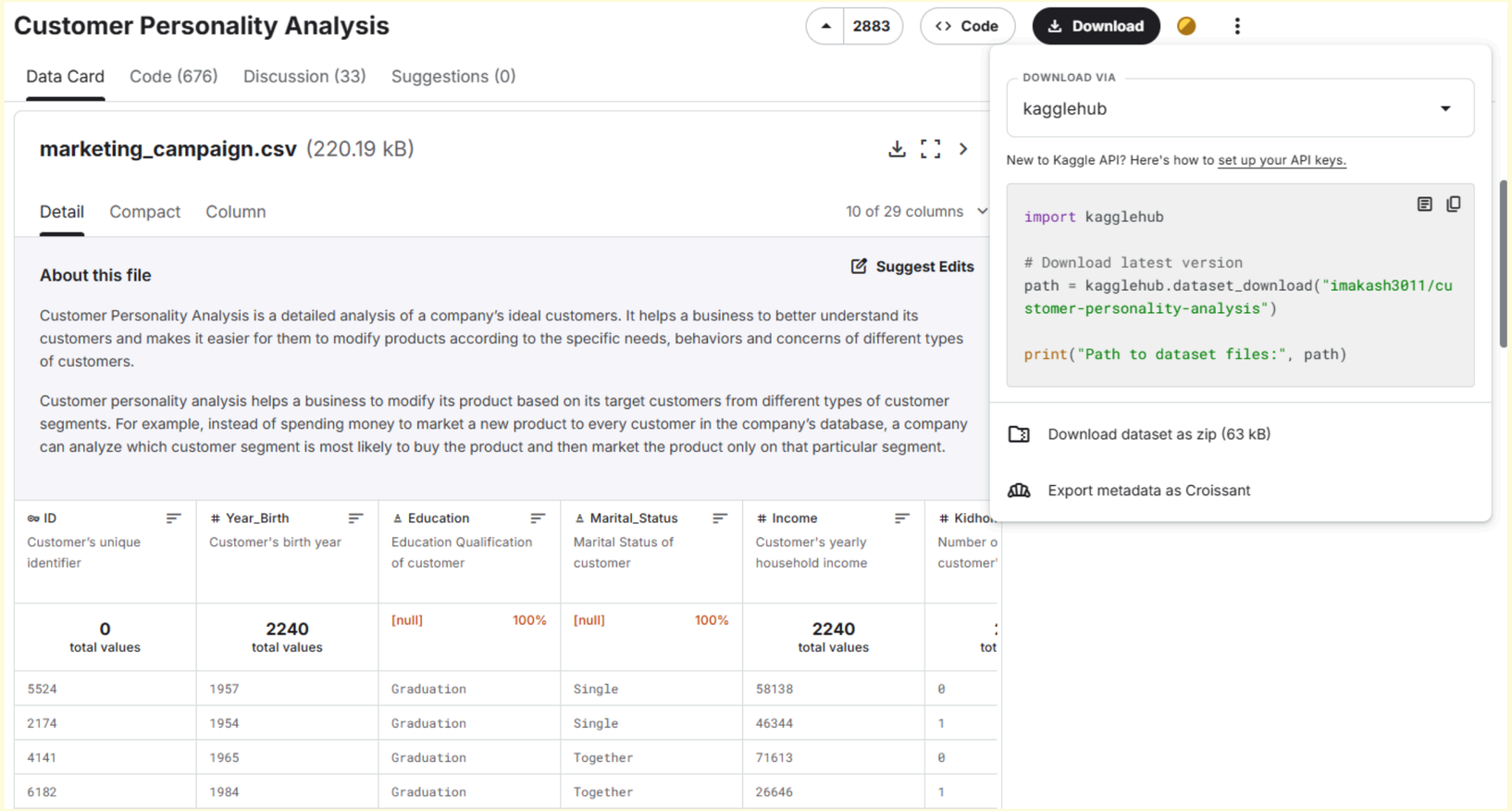
INTRO TO DATATHON WORKFLOW

Importance of fast sanity checks

- Before deep analysis:
  - Does the data make sense?
  - Are values in reasonable ranges?
  - Are there obvious data quality issues?
- Fast sanity checks prevent:
  - Debugging errors later
  - Drawing conclusions from broken data
  - Wasting time modeling noise

OVERVIEW OF THE DATA

source: https://www.kaggle.com/datasets/imakash3011/customer-personality-analysis/data



OVERVIEW OF THE DATA

1. importing and first quick glance at the data we have
    a. **df.shape** gives you the dimensions of the data
    b. **df.head()** prints the first few rows of the data

```
# LOAD DATA
df = pd.read_csv(f"{path}/marketing_campaign.csv", sep = '\t')
print("Shape:", df.shape)
df.head()
```

2. Inspecting the data orientation
    a. **df.info()** gives basic information about the data (col names, null value type etc.)
    b. **df.describe()** listing out names of columns that are numeric and categorical to see if they're "stored" correctly

```
# DATA ORIENTATION
df.info()
numeric_cols = df.select_dtypes(include=np.number).columns.tolist()
cat_cols = df.select_dtypes(include=["object"]).columns.tolist()
print("Numeric columns:", numeric_cols)
print("Categorical/Text columns:", cat_cols)
```

EXPLORATORY DATA ANALYSIS

Always answers 3 questions:

1. Can I trust this data?
2. What decisions will break my model?
3. What structure exists before I start modeling?

Univariate

Does each column make sense on its own?

Bivariate

Which variables move together?

Multivariate

Are there hidden segments or structures?

EXPLORATORY DATA ANALYSIS

Univariate

Does each column make sense on its own?

To derive the data, define and summarise it and analyse the patterns it presents. It tells you about data quality, scale, and potential transformations (e.g., log-transform skewed spend)

Numeric variables: Income, Age, Recency, Total Spend

- Methods: Histograms, .describe(), checking min/max, mean/std, skewness.
- Purpose: Identify distribution shape, central tendency, spread, outliers, or potential missingness.

Categorical variables: Education, Marital_Status

- Methods: Bar plots, value counts.
- Purpose: See dominant categories, check for imbalances, spot potential grouping issues.

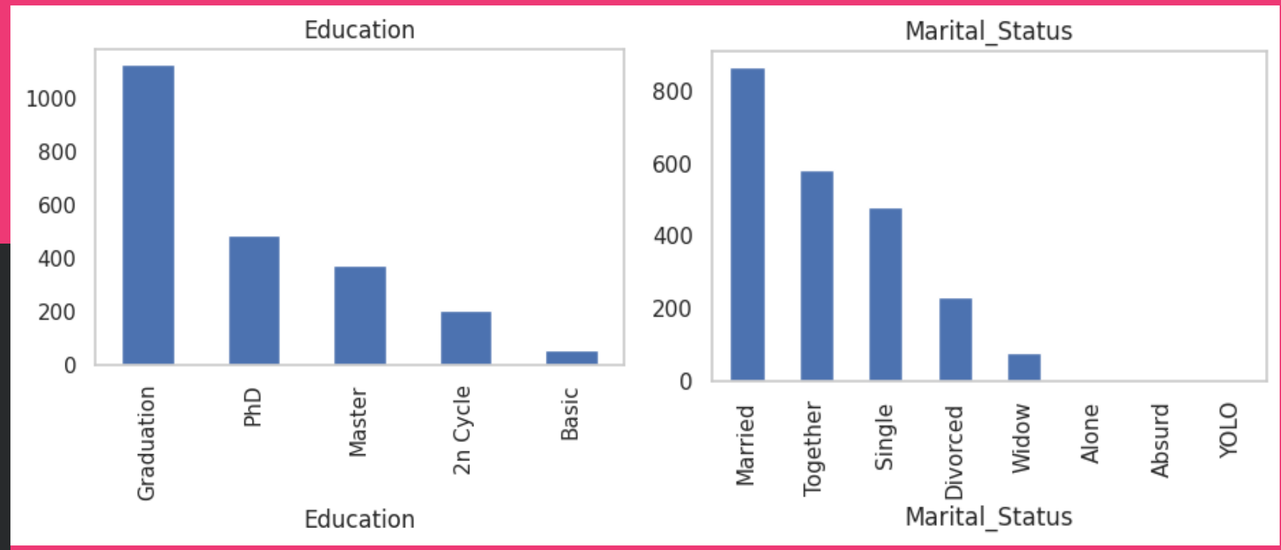
EXPLORATORY DATA ANALYSIS

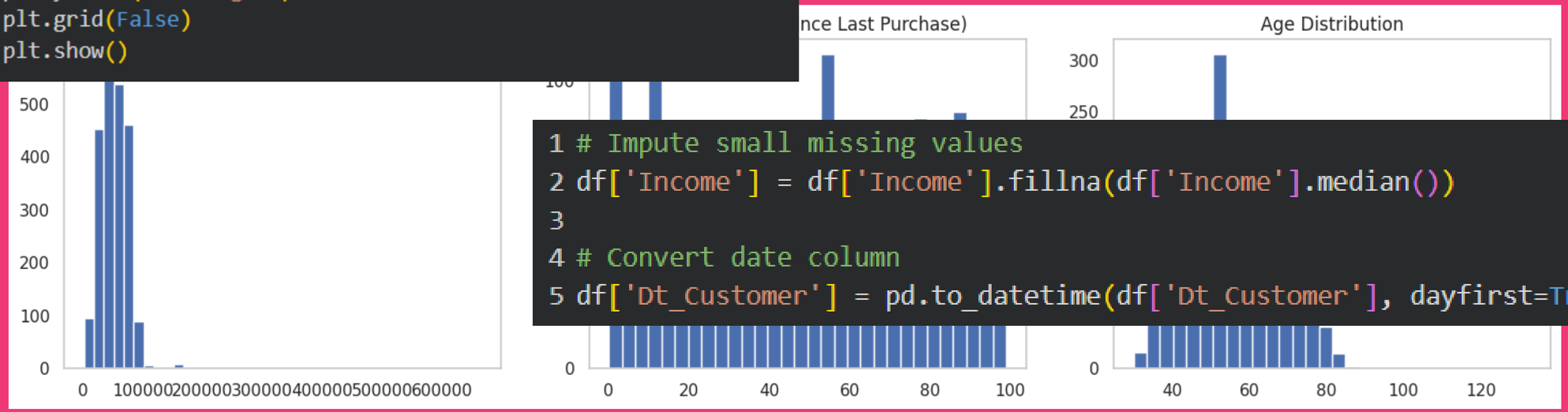
Univariate

Does each column make sense on its own?

	count	mean	std	min	25%	50%	75%	max
ID	2240.0	5592.159821	3246.662198	0.0	2828.25	5458.5	8427.75	11191.0
Year_Birth	2240.0	1968.805804	11.984069	1893.0	1959.00	1970.0	1977.00	1996.0
Income	2240.0	52237.975446	25037.955891	1730.0	35538.75	51381.5	68289.75	666666.0
Kidhom								
Teenhor								
Recenc								
MntWin								
MntFrui								

```python
# MISSING DATA ANALYSIS
missing_pct = df.isnull().mean().sort_values(ascending=False)
missing_pct[missing_pct > 0]

plt.figure(figsize=(10,4))
missing_pct[missing_pct > 0].plot(kind="bar")
plt.title("Percentage of Missing Values by Column")
plt.ylabel("Missing %")
plt.grid(False)
plt.show()
```






```python
# Impute small missing values
df['Income'] = df['Income'].fillna(df['Income'].median())

# Convert date column
df['Dt_Customer'] = pd.to_datetime(df['Dt_Customer'], dayfirst=True)
```

EXPLORATORY DATA ANALYSIS

Bivariate

Which variables move together?

1. Numeric vs. Numeric → scatter plots, correlation
    a. Example: Income vs raw_total_spend, Recency vs raw_total_spend.
    b. Purpose: Identify trends, correlations, outliers, potential feature interactions.

2. Numeric vs. Categorical → boxplots, group means
    a. Example: Average spend per Education level.
    b. Purpose: See differences across categories, detect segmentation signals.

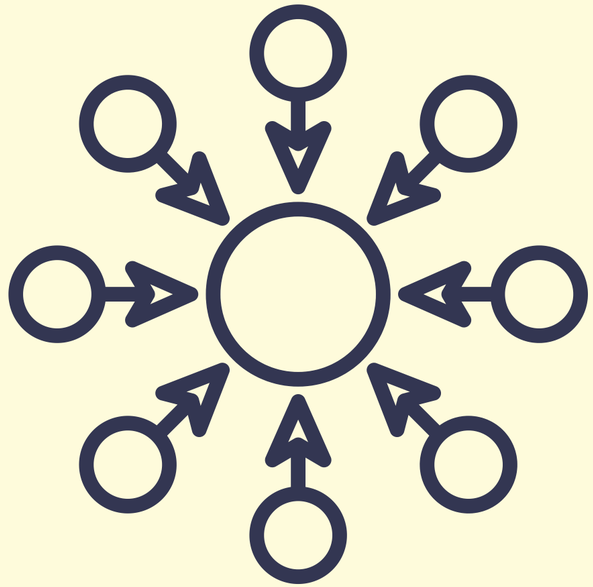
3. Categorical vs. Categorical → counts, contingency tables, heatmpas
    a. Example: Marital_Status vs Education.
    b. Purpose: Detect dependency patterns, potential multi-collinearity.

EXPLORATORY DATA ANALYSIS

Multivariate

Are there hidden segments or structures?

1. interactions between many variables
2. early clustering intuition

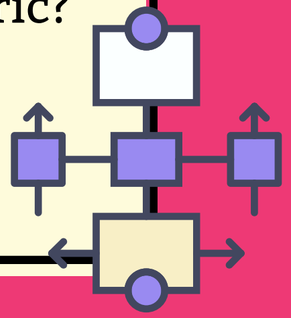


- Focus: patterns across multiple variables
- Methods: Correlation matrices, PCA, UMAP, clustering.
- Purpose:
  - Understand interactions among many variables.
  - Identify latent structure (e.g., clusters of high spenders).
- Example:
  - corr_features = ['Income', 'Age', 'Recency', 'raw_total_spend']
    sns.heatmap(df[corr_features].corr(), annot=True)
  - PCA reduces dimensionality while retaining variance.
  - UMAP captures non-linear structures for visualization.
  - Hierarchical and KMeans clustering identify natural customer segments.

EXPLORATORY DATA ANALYSIS

Schema and Data Type Checks

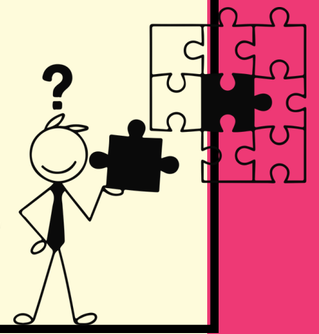
- Are numeric fields actually numeric?
- Are categorical values consistent?
- Are IDs mistakenly treated as features?



Missingness patterns

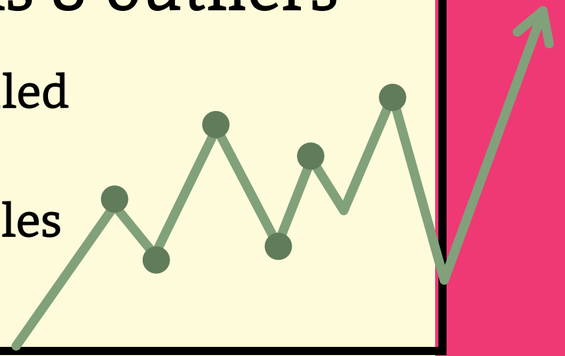
Look beyond simple counts:

- Missing by column
- Missing by row
- Missing conditional on another variable



Distribution checks & outliers

- Skewed or heavy-tailed distributions
- Zero-inflated variables
- Extreme values



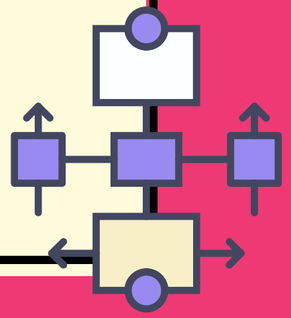
Inconsistencies and data leakage

- Duplicate entities with conflicting values
- Features that encode future information
- Aggregates computed using information not available at decision time
-

EXPLORATORY DATA ANALYSIS

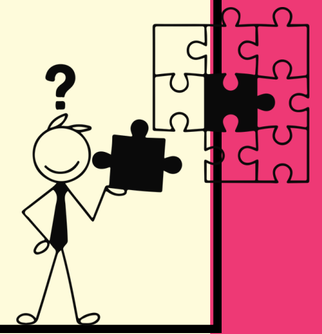
Schema and Data Type Checks

- df.info()
- df.dtypes



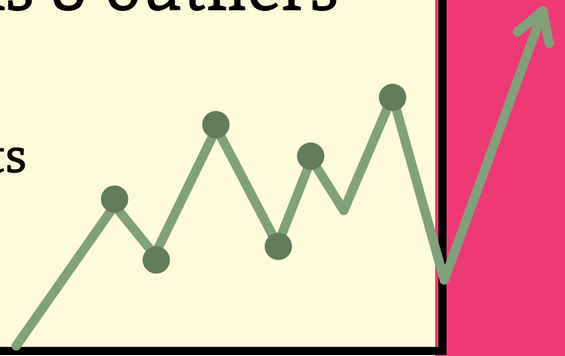
Missingness patterns

- df.isnull().mean()



Distribution checks & outliers

- .hist(), describe()
- scatterplots, boxplots



Inconsistencies and data leakage

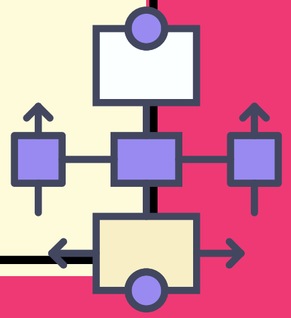
- negative spends, impossible ages
- avoiding using features that include future information or target information in predictors

EXPLORATORY DATA ANALYSIS

Schema and Data Type Checks

Common issues:

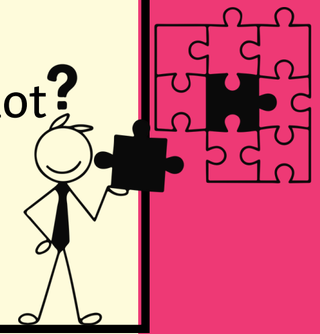
- Numbers stored as strings
- Mixed units or formats
- One-hot–like columns embedded in raw data



Missingness patterns

Why this matters:

- Missing data is often systematic, not random
- Missingness itself can carry information



Distribution checks & outliers

Questions to ask:

- Should this be log-scaled?
- Should values be capped or binned?
- Are outliers errors or valid cases?

Inconsistencies and data leakage

- Rule of thumb: If a feature feels “too good to be true”, investigate it.

No single “correct” choice.
Context matters for each case.

DATA PREPROCESSING

Action	**What is it?**	**When to Use**
Drop	Remove columns/rows	IDs, constants When data is unusable or misleading
Impute	Fill missing values	Low % missing When missingness is limited or structured
Encode	Convert categories to numerical values	ML models that need numeric values
Transform	Change the scale	Skewed distributions When you need scaling, normalization, binning etc.

No single “correct” choice.
Context matters for each case.

DATA PREPROCESSING

Action	**Where**
Drop	Drop unnecessary identifiers (ID, Year_Birth) for clustering.
Impute	df['Income'] = df['Income'].fillna(df['Income'].median())
Encode	For distance-based clustering, categorical features require appropriate encoding (one-hot, ordinal)
Transform	StandardScaler for models sensitive to scale (LR, KMeans) Convert Dt_Customer to datetime, then engineer tenure

FEATURE ENGINEERING

Models do not understand meaning — they operate on numerical representations.

Features are not neutral but rather:

- they encode beliefs about what matters
- they determine what patterns models can find

You engineer features because raw data tells you what happened, while engineered features tell the model why it happened, and the why is what drives prediction, segmentation, and insight.

FEATURE ENGINEERING

Models do not understand meaning — they operate on numerical representations.

Numerical Features

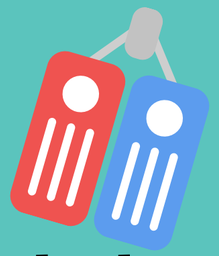
Common Techniques:

1. Scaling - standardization vs normalization
2. Ratios instead of raw values
3. Aggregations - counts, averages, rates

Examples: total_spend, web_purchases_ratio, deal_intensity

Why:

Aggregate or normalize numeric signals to capture meaningful behavior rather than raw counts



FEATURE ENGINEERING

Models do not understand meaning — they operate on numerical representations.

Categorical Features

Encoding choices depend on:

1. Cardinality (how many unique values in a column)
2. Model type

Common Approaches:

1. One-hot encoding (low-cardinality)
2. Frequency encoding
3. Grouping rare categories

Watch for:

1. Dimensionality explosion (too many columns used which slow training and overfits)
2. Leakage from target-based encodings

Example: Household size = 1 + kidhome + teenhome

FEATURE ENGINEERING

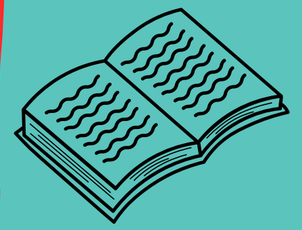
Models do not understand meaning — they operate on numerical representations.

Text Features

Raw text must be converted into structured form:

1. Bag-of-words/Count vectorization: counts how many times each word appears
2. TF-IDF (Term Frequency-Inverse Document Frequency): weighs rare words higher, common words lower
3. Embeddings: dense numeric vectors representing semantic meaning (via HuggingFace or WOrd2Vec etc.)
4. Derived features - length, keyword counts

Text is powerful but high dimensional, which means compression (dimensionality reduction/feature selection) is essential



FEATURE ENGINEERING

```
# Share of spend per category
for col in spend_cols:
    df[f'{col}_share'] = df[col] / (df['total_spend'] + 1) # +1 added to denominator to avoid division by zero
```

One of the most important features engineered

- It was misleading as raw spend: A customer spending $1000 on wine sounds premium until they spent $10000 total, making wine minor
- What share features do: they convert the spend into preference proportions, showing a customer's tastes or spending patterns rather than amount spent
  - Customer A: $500 wine / $600 total → they are wine-centric
  - Customer B: $500 wine / $5000 total → the wine isn't the main goal, they are diversified
- Note: Clustering and PCA (coming up) LOVE this as it creates a clean separation between customer types and makes the data lass dominated by income variance

FEATURE ENGINEERING

Why was it needed?

Raw columns are giving ABSOLUTE COUNTS like:

- “How much did they spend on X?”
- “How many purchases via channel Y?”
- “How many kids live at home?”

Models struggle with these because:

1. Scale dominates signal
2. High-income / high-activity customers always look “important” even if their behavior is normal for them.
3. Behavior ≠ volume
4. Two customers can spend the same total amount but behave very differently.
5. Comparability problem
6. A customer with 2 web purchases out of 2 total purchases ≠ a customer with 2 web purchases out of 20.

FEATURE ENGINEERING

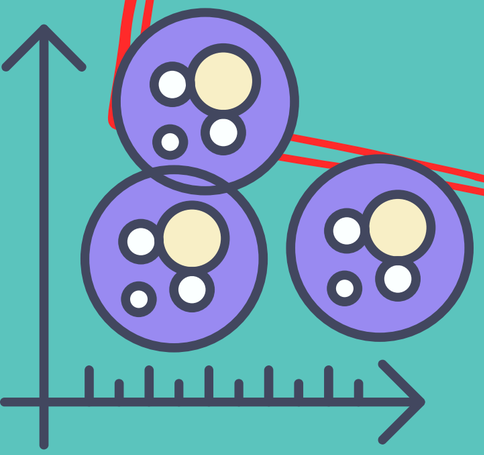
Feature engineering is often more impactful than model choice.

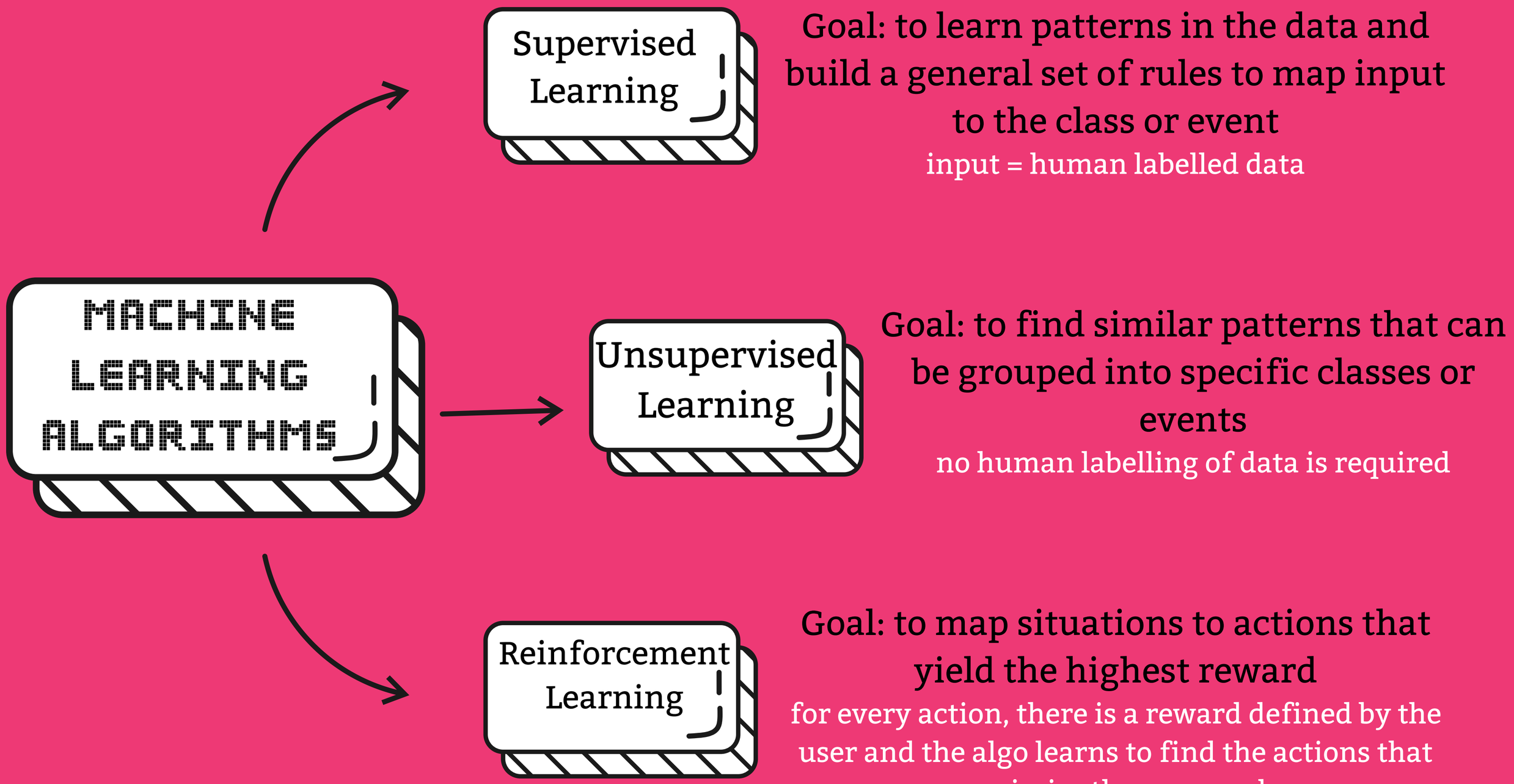
Good features:

- Simplify patterns
- Reduce noise
- Improve interpretability

How feature choices affect downstream models

- Distance-based methods are sensitive to scale
- Clustering is highly sensitive to feature selection
- Feature choices shape what patterns models can discover




MACHINE LEARNING ALGORITHMS
Supervised Learning
Goal: to learn patterns in the data and build a general set of rules to map input to the class or event
input = human labelled data
Unsupervised Learning
Goal: to find similar patterns that can be grouped into specific classes or events
no human labelling of data is required
Reinforcement Learning
Goal: to map situations to actions that yield the highest reward
for every action, there is a reward defined by the user and the algo learns to find the actions that maximise those rewards

SUPERVISED LEARNING

1. Training the model on the given data to “learn”
2. Testing or validation to improve the accuracy of the model
3. Classification/Prediction applied to new unseen data

Regression

- to predict a new y-value given a x-value
- to find the “best-fit line” for data and use its equation for predictions
- eg. prediction of housing prices

Classification

- to find the decision boundary between two classes of data
- eg. classifying email as spam/not spam

(NOT TO BE CONFUSED WITH SELF-SUPERVISED LEARNING – VERY VERY DIFFERENT

SUPERVISED LEARNING

TARGET: to predict customer spending based on history of actual spending

Linear Regression

Foundational Regression Model

Pros: Highly interpretable coefficients, fast to train and low computational cost

Why: Provides directional insights (+ve vs -ve influence), models continuous outcomes, good baseline model

$$R^2 = 1 - \frac{\sum_i (y_i - \hat{y}_i)^2}{\sum_i (y_i - \bar{y})^2}$$

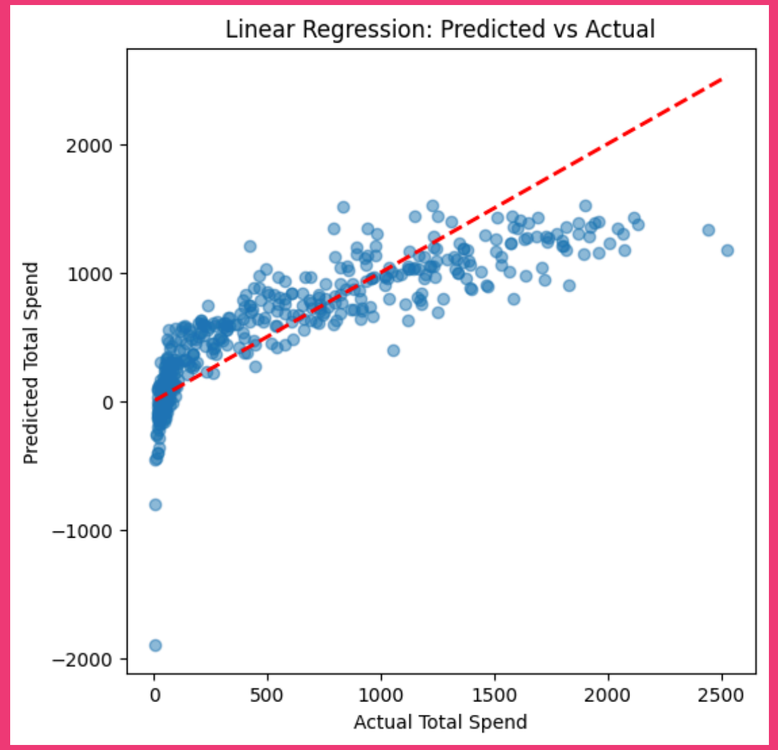
how well is the variance in the target explained

best value 1

```
Linear Regression R2 Score: 0.7286003258410843
Linear Regression RMSE: 309.9864509358606
Linear Regression MAE: 229.019518149503 3
```

root mean squared error = avg magnitude of error

mean absolute error = absolute difference blw predicted & actual values



SUPERVISED LEARNING

TARGET: predict AcceptedCmp1 - whether or not a campaign is responded to

Logistic Regression

Classification Model sensitive to feature scaling (essentially linear regression passed through a sigmoid function)

Pros: Interpretable, fast, baseline for classification

Why: Works well on scaled, engineered numeric features

Logistic Regression Accuracy: 0.9397321428571429

	precision	recall	f1-score	support
0	0.95	0.99	0.97	425
1	0.00	0.00	0.00	23
accuracy			0.94	448
macro avg	0.47	0.50	0.48	448
weighted avg	0.90	0.94	0.92	448

precision: TP/TP+FP

recall: TP/TP+FN

f1-score: 2 · precision·recall / precision+recall

actual occurances

accuracy: TP+TN / TP+TN+FP+FN

avg of metrics

↳ macro avg

weighted avg: avg of metric weighted by no. of samples per class

SUPERVISED LEARNING

TARGET: predict AcceptedCmp1 - whether or not a campaign is responded to

Random Forest
Classification Model less sensitive to scaling
Pros: Handles non-linearities, interactions, less preprocessing
Why: Can capture complex patterns missed by logistic regression

Random Forest Accuracy: 0.953125

	precision	recall	f1-score	support
0	0.96	0.99	0.98	425
1	0.62	0.22	0.32	23
accuracy			0.95	448
macro avg	0.79	0.61	0.65	448
weighted avg	0.94	0.95	0.94	448

SUPERVISED LEARNING

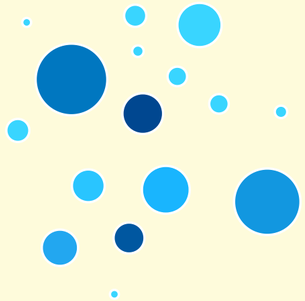
Alternatively, you can experiment with:

1. Gradient Boosting (XGBoost, LightGBM) which is stronger for tabular data
2. SVM if feature space is complex and moderately sized
3. Neural networks (not used here since its overkill for tabular marketing datasets)

```
# Optional -  Uncomment to try

'''
from xgboost import XGBClassifier

xgb = XGBClassifier(n_estimators=100, learning_rate=0.1, eval_metric='logloss', random_state=1000)
xgb.fit(X_train_scaled, y_train)

y_pred_xgb = xgb.predict(X_test_scaled)

print("XGBoost Accuracy:", accuracy_score(y_test, y_pred_xgb))
print(classification_report(y_test, y_pred_xgb))
'''
```

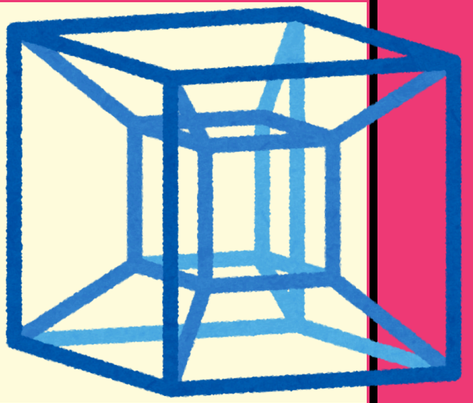
```
# Optional - Uncomment to try

'''
from sklearn.svm import SVC

svm_model = SVC(kernel='rbf', probability=True)
svm_model.fit(X_train_scaled, y_train)

y_pred_svm = svm_model.predict(X_test_scaled)

print("SVM Accuracy:", accuracy_score(y_test, y_pred_svm))
print(classification_report(y_test, y_pred_svm))
'''
```

UNSUPERVISED LEARNING

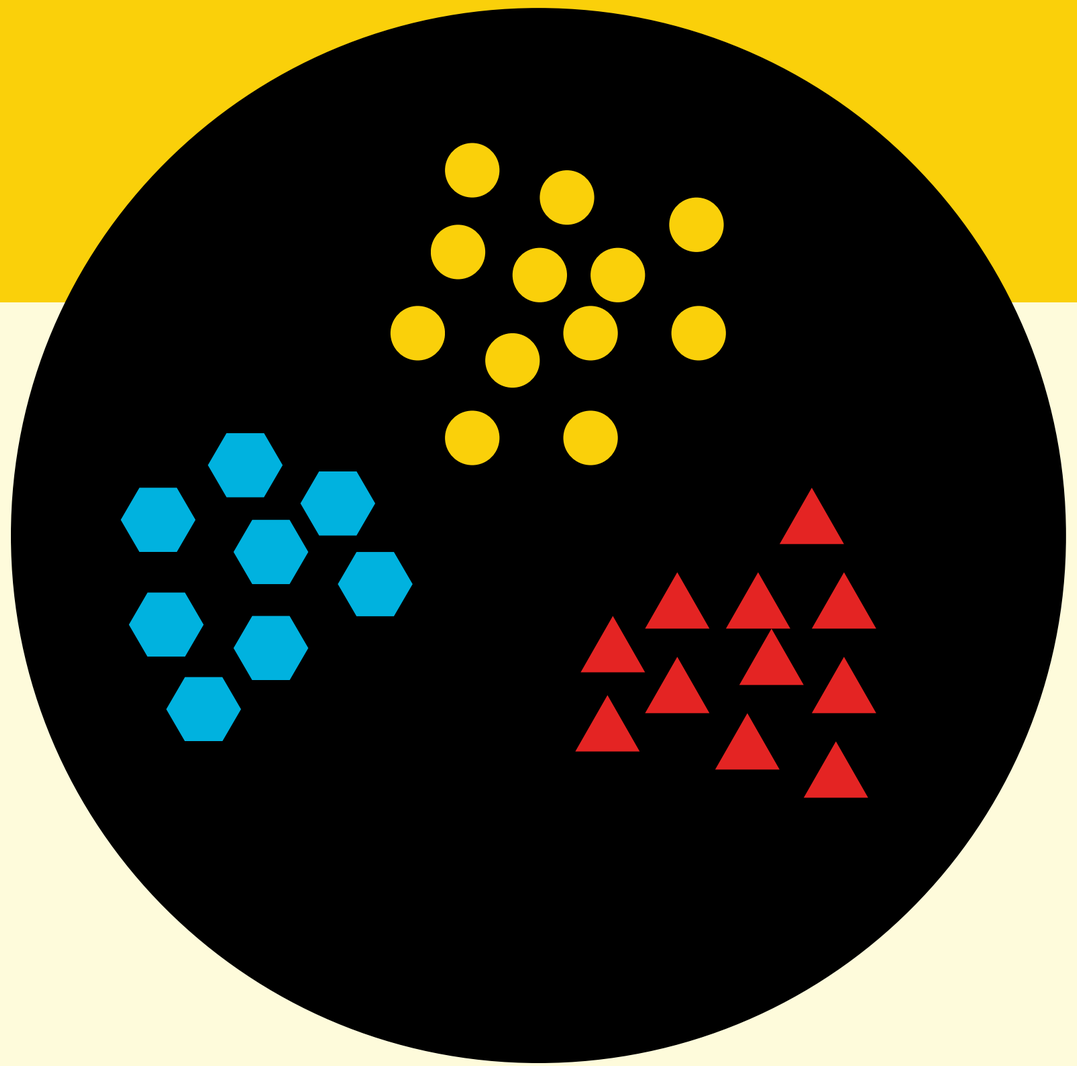


Clustering

Dimensionality Reduction



CLUSTERING



Clustering is the unsupervised learning technique of grouping similar data points together into clusters based on their characteristics, without requiring labelled data

<u>Aim</u>

Uncover natural grouping or structure in unlabelled data

CLUSTERING TECHNIQUES

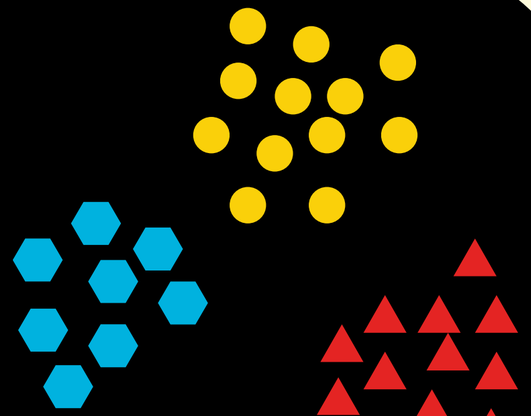
1

K-means Clustering

2

Hierarchical Clustering

KMEANS



K-Means in a Nutshell

- A popular clustering algorithm where K refers to the desired number of clusters
- Groups points based on distance to cluster centres

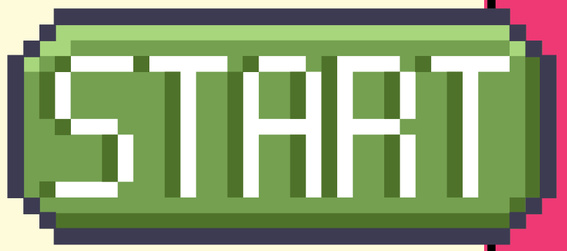
4 steps to K-Means

- Initialisation
- Assignment
- Update
- Repeat

4 STEPS TO KMEANS

Initialisation

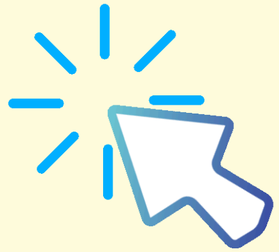
Start by assigning k points as the cluster centroid



Assignment

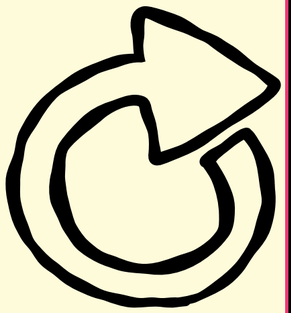
Compute the Euclidean distance of each point to each cluster centroid

Choose nearest cluster



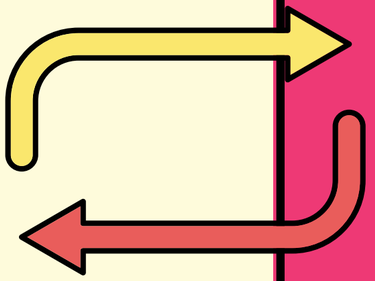
Update

Revise the cluster centroid as the mean of the data points categorised inside the cluster



Repeat

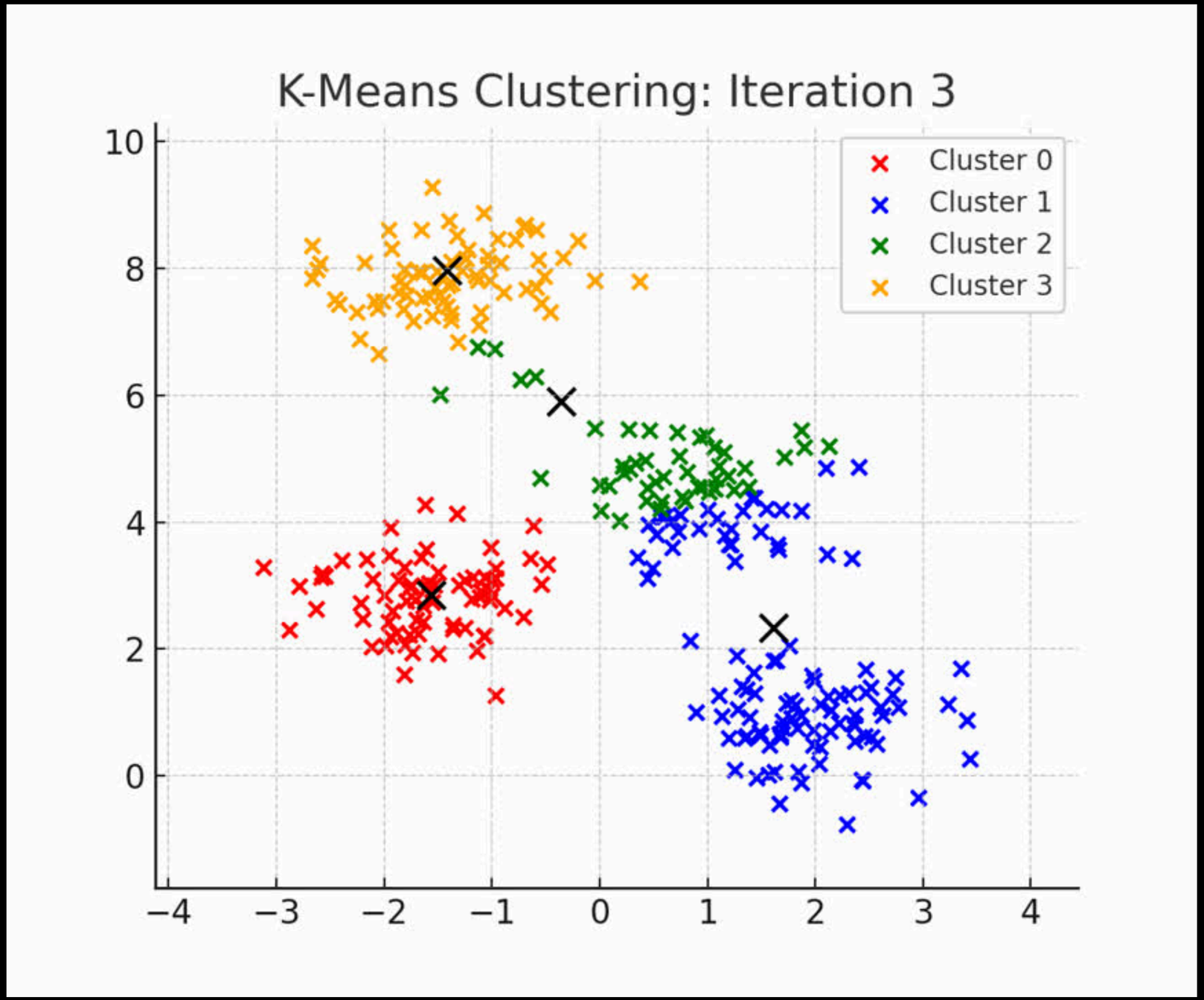
Repeat the Assignment and Updating steps until there are no change to cluster membership



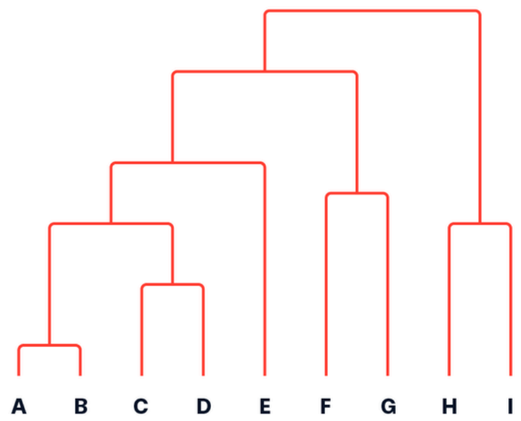
K-MEANS

Worked Example

Data	1	2	3	4	5	6	7	8	9
Centroid 1	4.13	1.25	3.84	4.69	2.69	4.21	7.98	2.36	8.93
Centroid 2	5.36	9.65	2.36	14.85	12.3	7.24	3.36	8.51	4.21
Centroid 3	4.21	4.23	8.98	1.23	1.14	7.86	5.63	12.85	4.36
Assignment	**1**	**1**	**2**	**3**	**3**	**1**	**2**	**1**	**2**


K-Means Clustering: Iteration 3
Cluster 0
Cluster 1
Cluster 2
Cluster 3
10
8
6
4
2
0
−4
−3
−2
−1
0
1
2
3
4

HIERARCHICAL



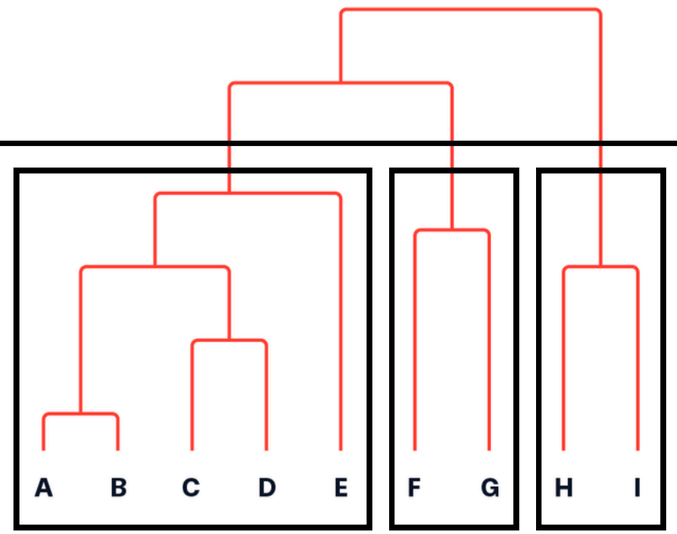
Hierarchical Clustering in a Nutshell

- A clustering method to group similar data based on their distance or other similarity metrics
- Builds a tree-like structure known as a dendrogram
- Number of clusters is determined visually by the analyst

Agglomerative clustering

- Each data point starts off as an individual cluster
- Using similarity metrics, the most similar clusters are grouped
- Repeat until there is only 1 cluster

INTERPRETING DENDROGRAMS



- Clusters are determined by drawing a line through the dendrogram
- The number of subtrees is the number of clusters

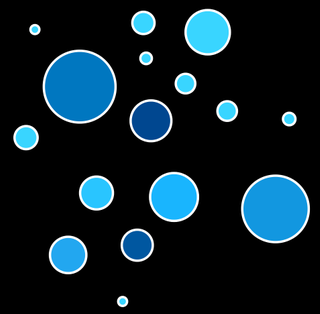
Cluster 1	A,B,C,D,E
Cluster 2	F,G
Cluster 3	H,I

CLUSTERING AT WORK

How does clustering benefit your workflow

Why is this effictive?

Grouping similar observations together transforms complex datasets into coherent segments that are easier for humans and LLMs to reason about. Therefore, they often offer more actionable insights.



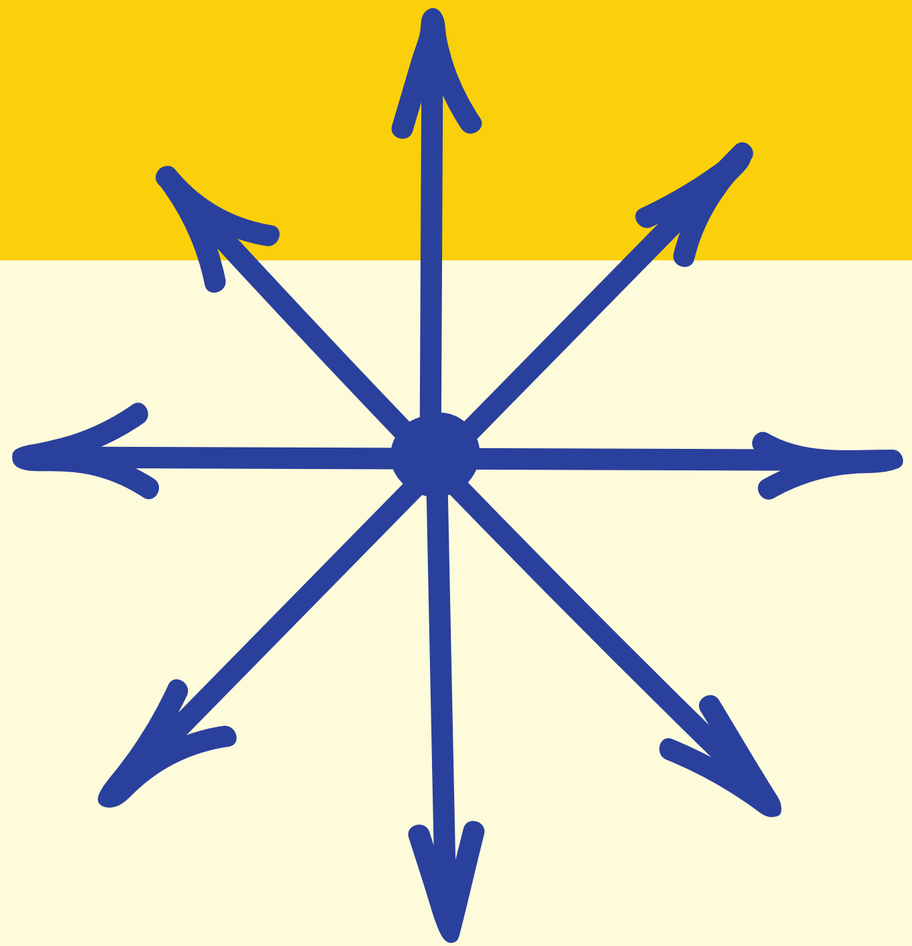
When to use

- Large datasets with no explicit target variable
- Goal is to discover patterns or isolate groups
- Individual-level analysis is too noisy and complex
- Preparing downstream tasks like profiling

Benefits to Workflow

- Segregate large datasets into smaller categories that produces more actionable insights
- Unexpected patterns require deeper analysis
- These patterns often offer more meaningful insights

DIMENSIONALITY REDUCTION

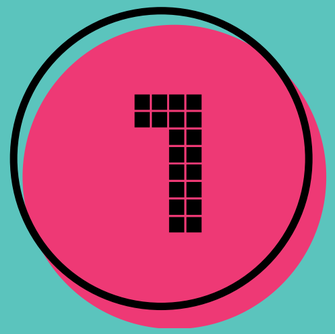


Dimensionality Reduction focusses on reducing the number of features in a data set by combining features, removing features or projecting data into a lower dimension space

<u>Aim</u>

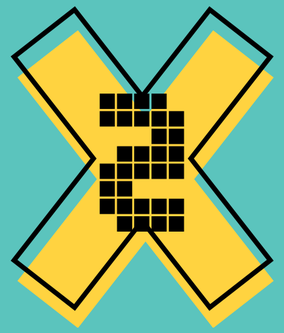
Represent the dataset with less dimensions without losing too much information

DIMENSIONALITY REDUCTION



PCA

Linear Data



UMAP

Non-linear Data

PCA

Problem

- Datasets with more dimensions (columns) require more computation, longer run time thus operationally expensive

- **Concentration of Distance Phenomenon**: At high dimensions, distances between points tend to become similar, making Euclidean distance less discriminative

- Problem of Multicollinearity: 2 or more variables are highly correlated with each other
  - Difficult to determine variable responsible for change
  - Focusing on variables with strong relationships to the outcome helps reduce redundancy and improve interpretability

PCA

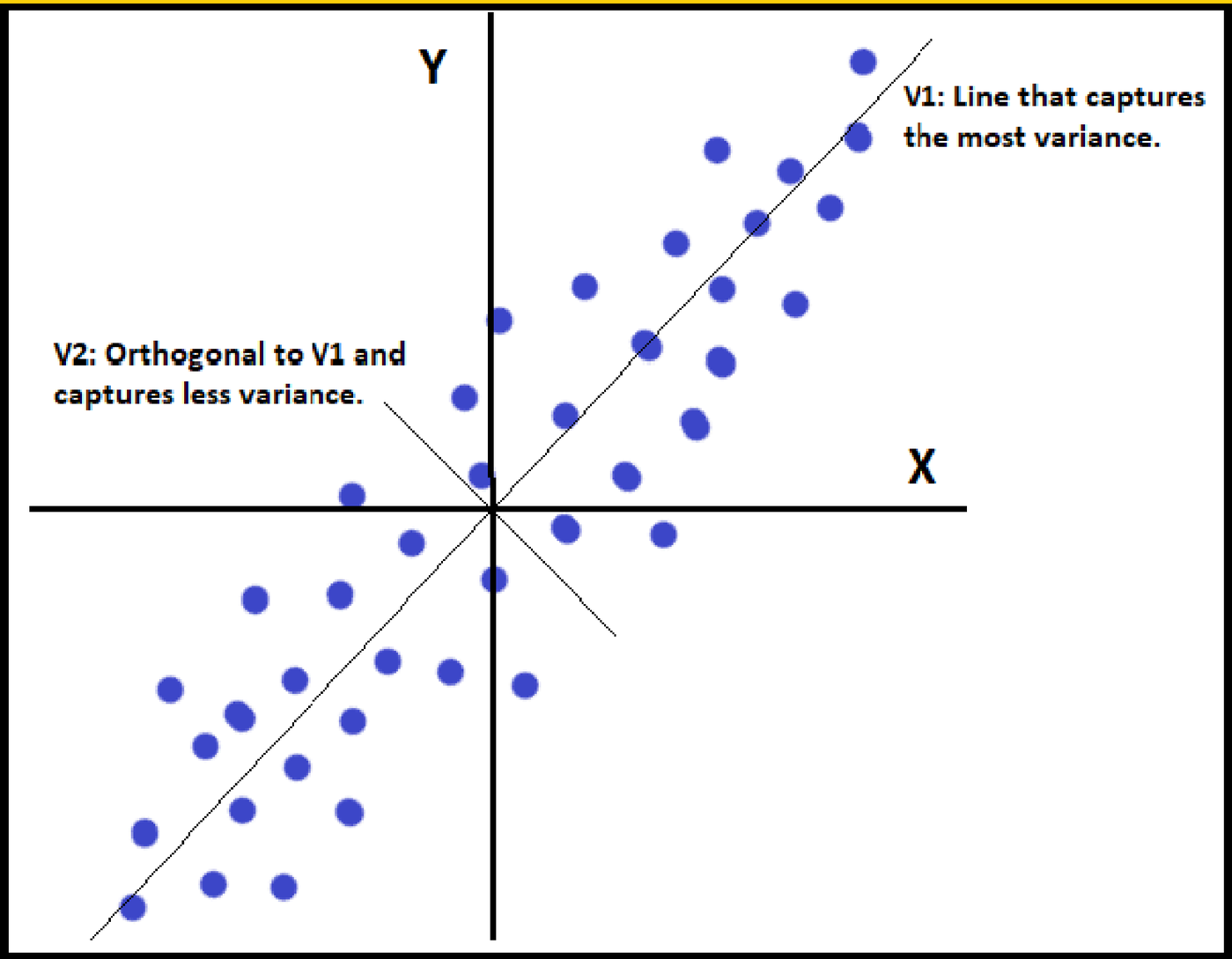
How PCA solves the problem

- Create new uncorrelated variables corresponding to the directions along which the data vary the most
- These new variables are linear combinations of original features, known as principal components
- Principal components are ordered according to the amount of variation in the data it explains

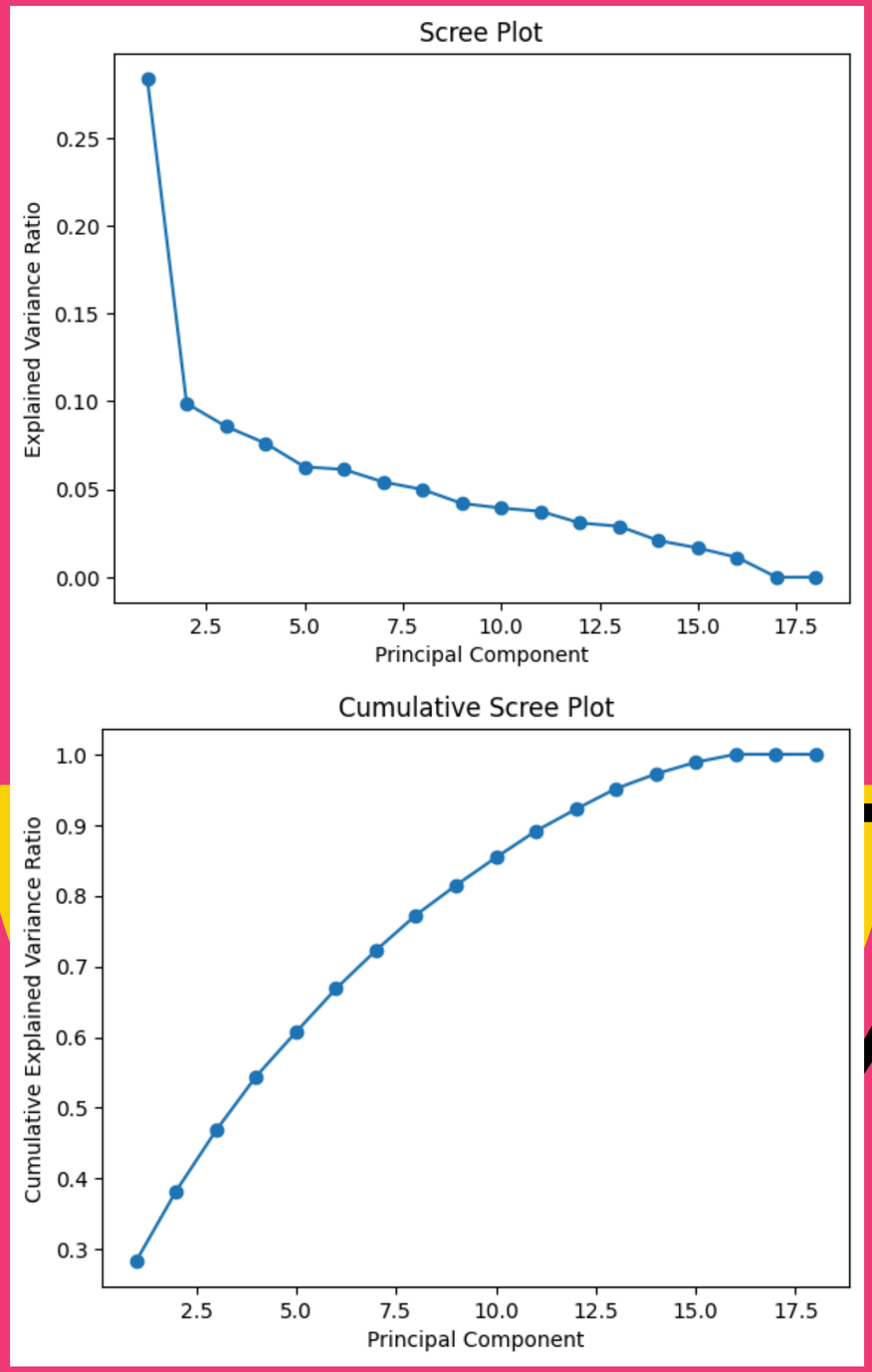
More on PCA

- https://www.geeksforgeeks.org/data-analysis/principal-component-analysis-pca/

VISUALISING PCA



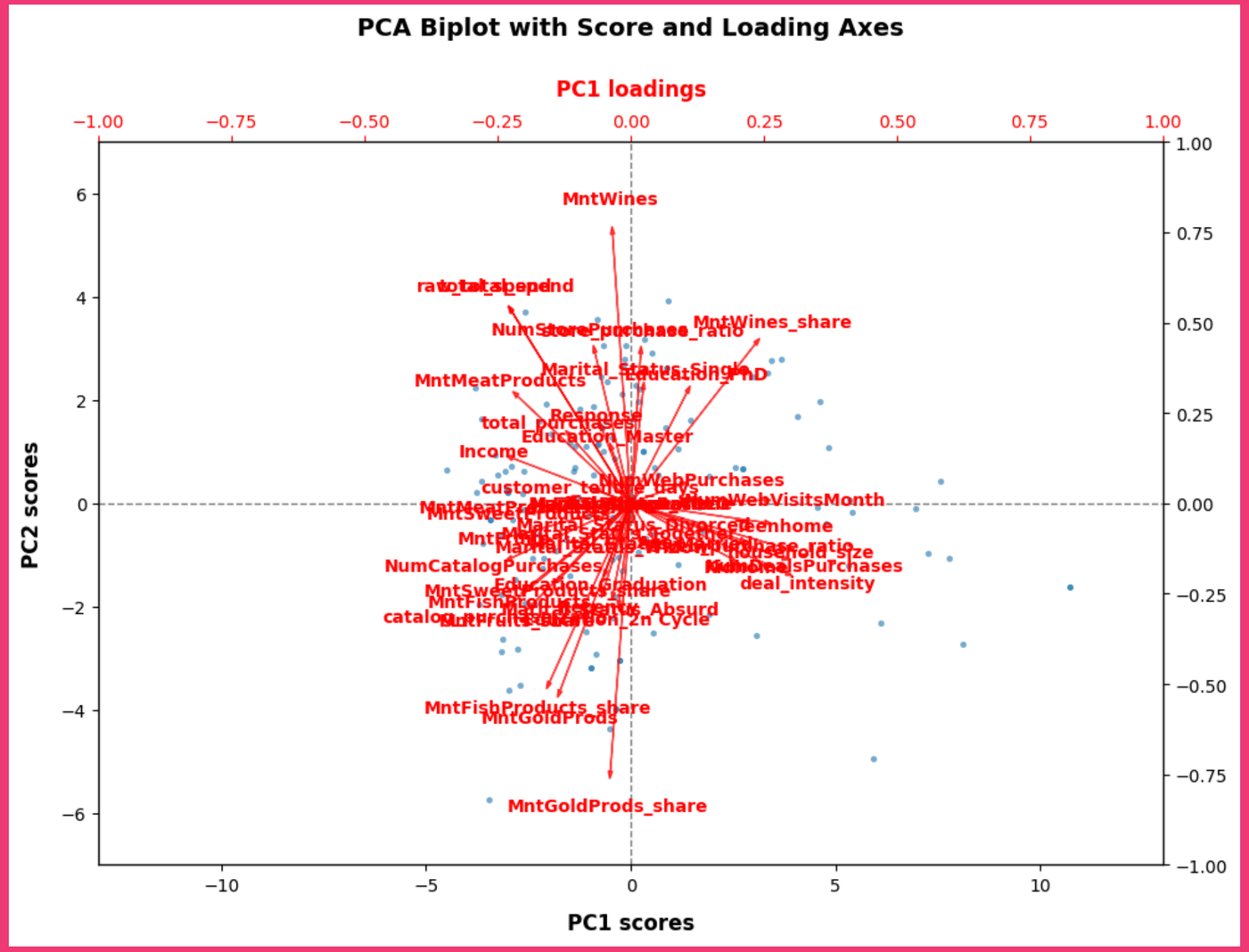
- The original data exists in two dimensions [X, Y]
- Dimensionality reduction transforms the data into fewer dimensions while preserving the most important information
- PCA finds a new axis that captures as much variation in the data as possible
- This new axis (V1), called PC1, represents a weighted combination of X and Y
- Instead of describing the data using X and Y separately, PCA lets us describe it using one new variable that summarizes both



SCREE PLOT

- A scree plot shows how much variance each PC explains
- PCs are ordered from most variance explained to least
- The total variance explained across all components equals 100%
- A cumulative plot helps verify how much total variance is captured as components are added
- The elbow point indicates where adding more components yields diminishing returns

BIPLOT



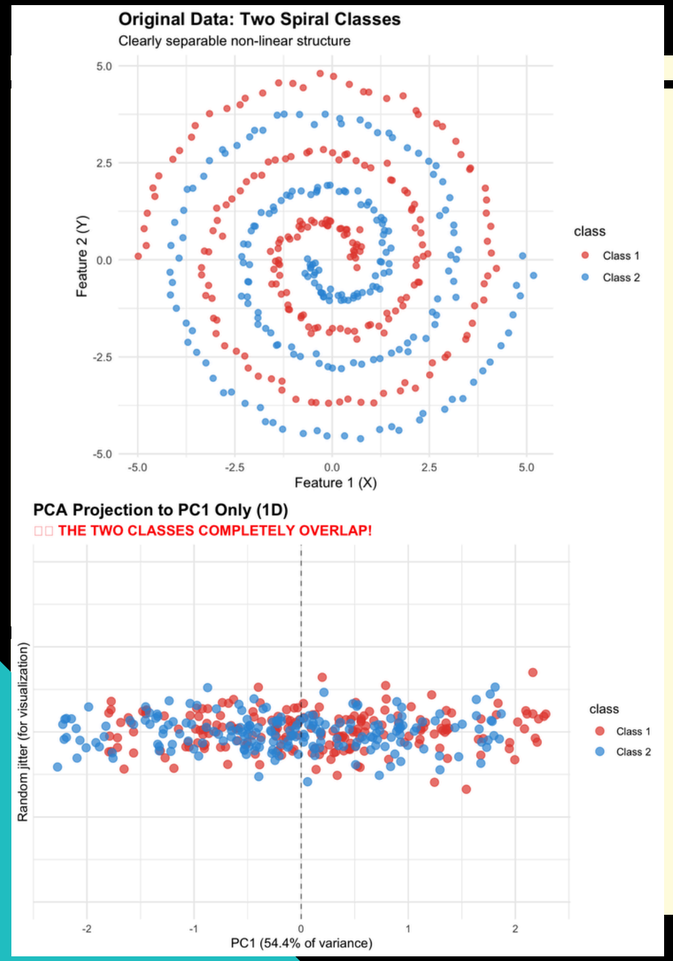
What is a biplot

- Plots data points along the first two principal components (PC1 and PC2)
- Feature loadings show how original variables contribute to each component

Interpreting Biplots

1. *Data Points (Observations)*
   - Visible groupings may indicate natural clusters in the data
2. *Feature Vectors (Loadings)*
   - Length of the arrow indicates the strength of that feature's contribution
   - Direction shows how the feature relates to the principal components
3. *Feature Relationships (Heuristic)*
   - Angle between feature vectors indicates correlation:
   - 0° → Strong positive correlation
   - 90° → Little or no correlation
   - 180° → Strong negative correlation

LIMITATIONS OF PCA



- PCA struggles with non-linear data
- Non-linear data refer to data points that do not lie on a flat hyperplane
- Solutions:
  - Kernel PCA
  - t-SNE or UMAP

UMAP

Uniform Manifold Approximation and Projection

What UMAP Does

- UMAP is a non-linear dimensionality reduction technique primarily used for visualization and clustering
- It assumes high-dimensional data lies on a lower-dimensional structure (manifold)
- The goal is to preserve local relationships between data points

How UMAP Works (An intuition)

- Identifies nearest neighbors for each data point in high-dimensional space
- Learns how strongly points are connected based on local similarity
- Projects the data into 2D or 3D while keeping similar points close together
- Adjusts the layout so the low-dimensional structure resembles the original data

POINTS TO NOTE

On using UMAP

- UMAP is typically used as a visualisation technique
- Often used in conjunction with K-means clustering
- Not intuitive to use UMAP results as a basis for predictions unlike PCA

More on UMAP

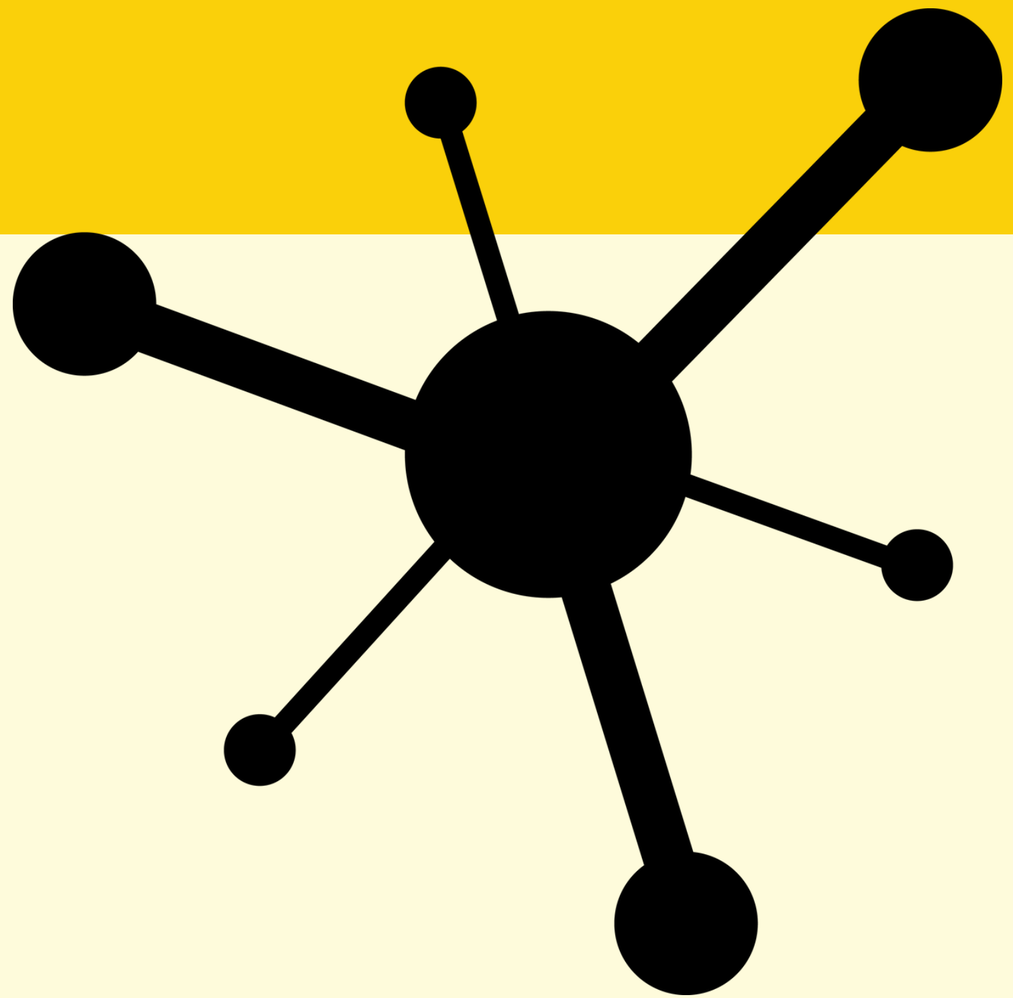
- https://umap-learn.readthedocs.io/en/latest/
- https://www.youtube.com/watch?v=eN0wFzBA4Sc

UNSUPERVISED WALKTHROUGH

Transition to our Google Colab For:

- Coded Examples of kmeans clustering
- Coded Examples of hierarchical clustering
- Coded Examples of PCA
- Coded Examples of UMAP

SIMILARITY METRICS



Computers need a rule to measure how close one data point is to another. The more features that two data entries have in common, the more similar they will be.

<u>Aim</u>

Discuss some useful similarity metrics that might improve unsupervised learning

METRICS

1

Euclidean Distance

2

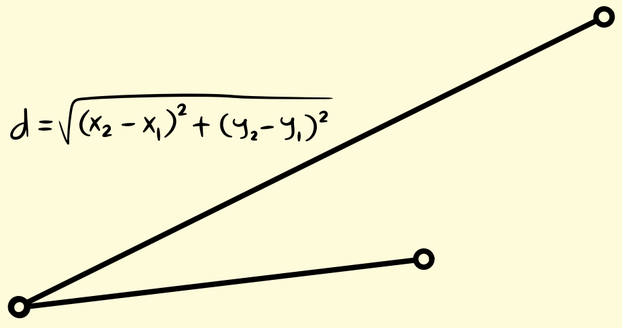
Cosine Similarity

EUCLIDEAN DISTANCE

Distance formulas

$$D = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2 + \dots}$$

$$= \left\| \begin{pmatrix} x_2 \\ y_2 \end{pmatrix} - \begin{pmatrix} x_1 \\ y_1 \end{pmatrix} \right\|$$



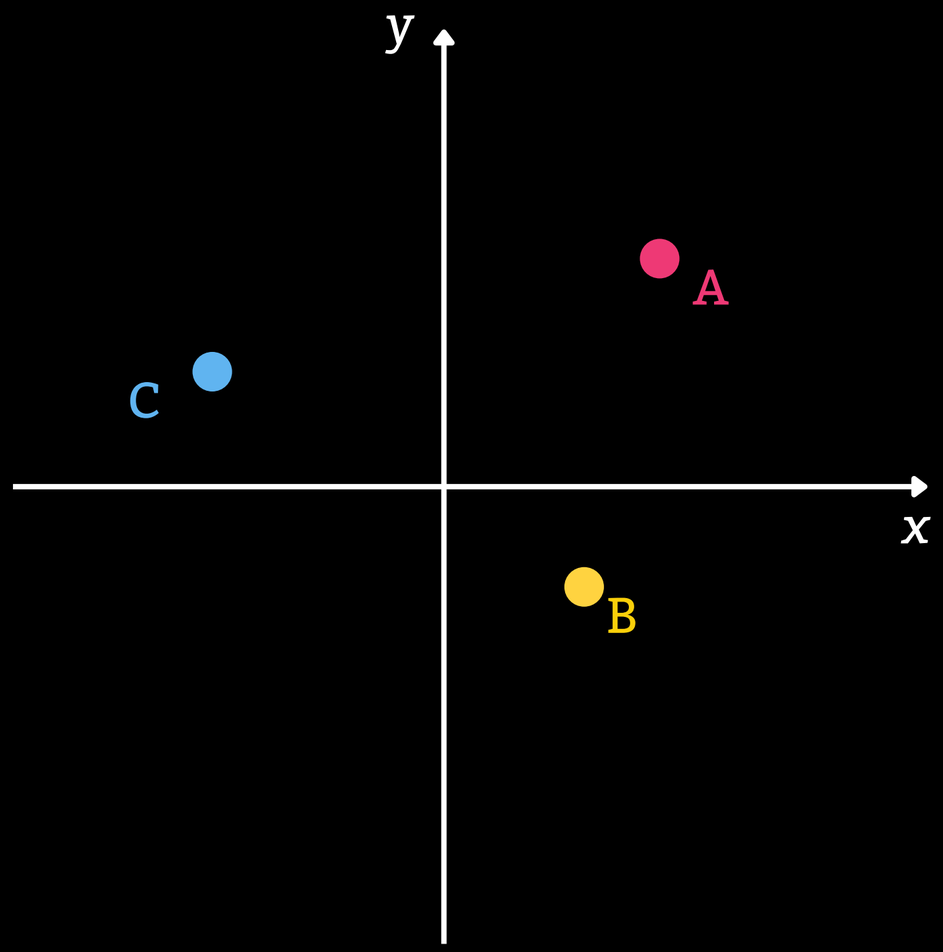
Calculating Euclidean Distance

- Straight-line distance between two points in the Euclidean space
- The length of the line segment that connects the two points
- Equivalent to the L2-norm of the vector that connects two points

Similarity Metric

- The closer the points are in the euclidean space, the more similar they are
- Euclidean distance treats all dimensions equally (assumes features are on comparable scales) – feature scaling is impt!

WORKED EXAMPLE



Point	x	y
A	3	3
B	2	-1
C	-3	2

$$D = \sqrt{(3-2)^2 + (3-(-1))^2}$$

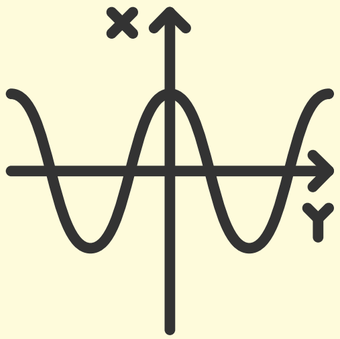
$$= 4.12\,(3sf)$$

The Euclidean distance between A and B is 4.12 units

COSINE SIMILARITY

Cosine formulas

$$\cos(\theta) = \frac{A^T B}{\|A\|\|B\|}$$



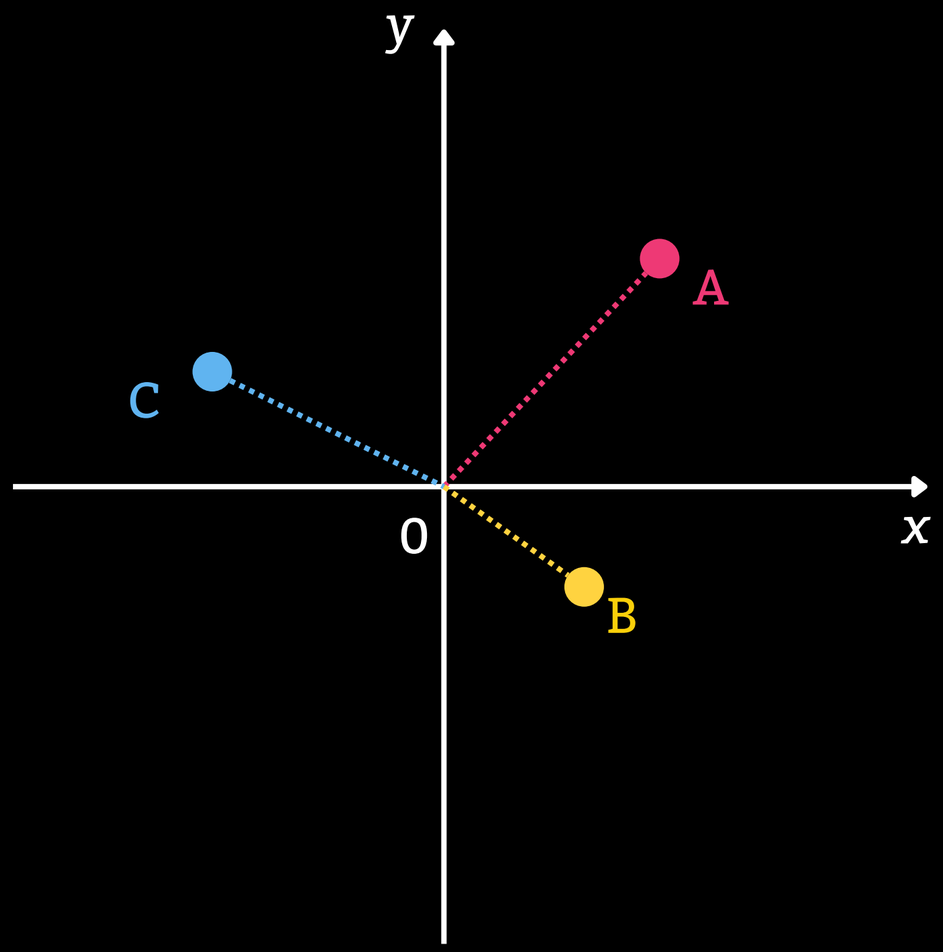
How Cosine Similarity Works

- Cosine similarity measures how similar two vectors are based on their direction, not their magnitude
- Widely used in text analysis, document comparison, search, and recommendation systems
- Values range from –1 to 1

Interpreting Cosine Similarity

- 1 → Vectors point in the same direction
  Highly similar content: “cat” and “kitten”
- 0 → Vectors are orthogonal
  No detectable similarity signal: “walnuts” and “Labubu”
- –1 → Vectors point in opposite directions
  Rare. usually indicates dissimilarity, not true semantic opposites

WORKED EXAMPLE



Point	x	y
A	3	3
B	2	-1
C	-3	2

$$\cos \angle A0B = \frac{\begin{pmatrix}3\\3\end{pmatrix}^T \begin{pmatrix}2\\-1\end{pmatrix}}{\sqrt{3^2+3^2} \times \sqrt{2^2+(-1)^2}}$$

$$= 0.316$$

TEXT ANALYSIS

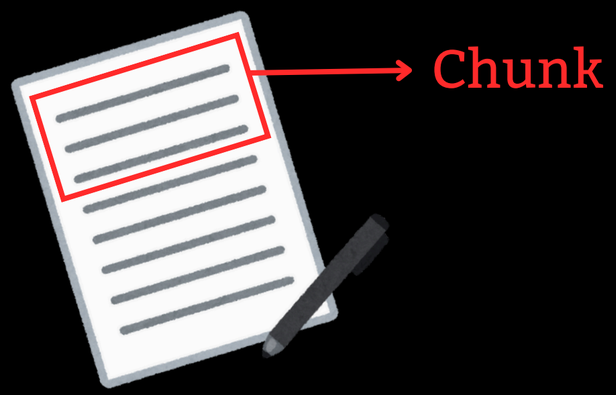
Cosine Similarity in Text Analysis

1. Represent Text as Vectors
    a. Word embeddings (Word2Vec, GloVe)
    b. Sentence/ document embeddings (from transformer models)
2. Cosine similarity is used to measure the angle between text vectors
    a. Each sentence is now represented by a vector
    b. Vectors can be compared to each other
    c. Texts with similar content will have high cosine similarity scores

RAG

Retrieval Augmented Generation

Why is this effictive?
RAG retrieves the most relevant information from a large dataset, then feed it into an LLM to generate answers. This ensures responses are contextually accurate, grounded in real data, and reduces hallucinations



A little bit about Retrieval Augmented Generation (RAG)

- Retrieves relevant data from documents or information in a knowledge base
- Augments LLM generation by feeding retreived information into prompts for context-aware responses
- Reduce hallucinations by limiting the reliance on information from model's training

How RAG works

- Documents are split into chunks and vectorised
- A user submits a question, which is also vectorised
- The question vector is compared to the vectorised chunks using cosine similarity
- The most similar chunks are inserted into a prompt to generate context-aware responses

LARGE LANGUAGE MODELS

MODEL	COMPANY
GPT	OpenAI
Claude	Antropic
Deepseek	DeepSeek
Gemini	Google DeepMind
LLaMA	Meta AI
Nova	Amazon
ERNIE	Baidu
Qwen	Alibaba
Hunyuan	Tencent
Mistral AI	Mistral AI

AN INTRODUCTION

A category of Deep Learning Models known as Transformers that is capable of understanding and generating natural language (IBM)

TRANSFORMERS

Neural Networks

A machine learning model that stacks “neurons” in layers

Learns weights from data to map inputs to outputs (IBM)

Other Neural Networks:
RNN, CNN, LSTM, GAN

Attention

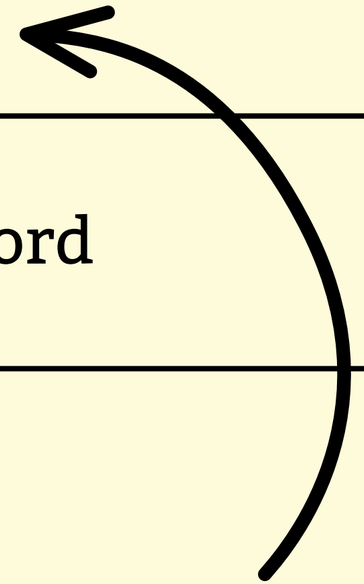
The attention mechanism directs deep learning models to prioritise the most relevant parts of the input data

Visualisation Tool:
https://poloclub.github.io/transformer-explainer/

HOW LLMS WORK

Predicting the next word in a sequence

The chicken crossed the road because		
it	**29.43%**	Next Word
he	21.25	
the	17.80%	**it**
of	16.53%	
she	15.00%	

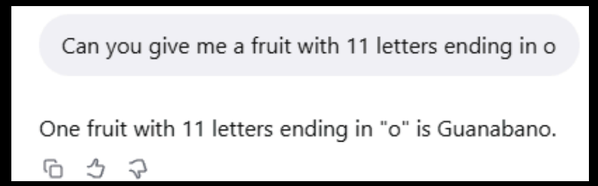


At their core, LLMs predict the next token in a sequence, but modern LLMs use attention mechanisms to consider the entire context

PROMPT ENGINEERING

Why Prompt Engineering

1. Structure a question to illicit the proper response from the LLM



2. Avoid Hallucinations
    a. Hallucinations are false results from LLMs
    b. A result of model's probabilistic generation process and training on diverse, sometimes contradictory Internet data
3. Provide Clear, Specific and Complete instructions to LLMs
    a. Clear - Tasks easily understood and retained by Attention
    b. Specific - Scope of the task is defined
    c. Complete - Encompassing all tasks you desire to be fulfilled

PROMPT STRUCTURE

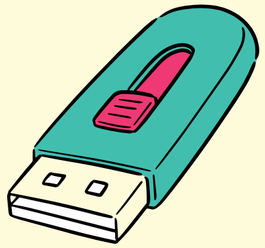
Role Assignment

Placing a persona upon the LLM

Task Definition

Define clearly what you want the LLM to achieve

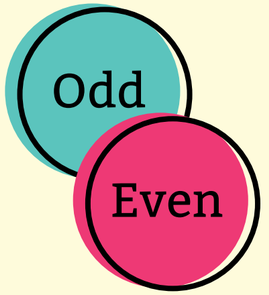
Context Provision



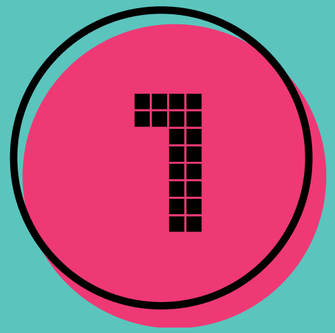
Provide the information that you want the LLM to act on

Response Formatting

Defining the structure of the LLM’s response

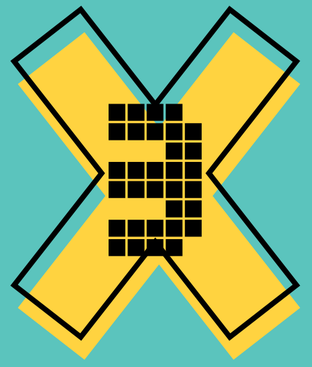


PROMPTING TECHNIQUES

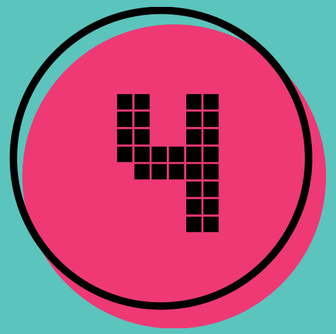


Content Grounding

Few-Shot Prompting



Chain-of-Thought



Directional Stimulus Prompting

CONTENT GROUNDING

Providing Specific Information for the LLM to work on

Why is this effictive?

The LLM will not use the generic information from the internet that is part of its training. Instead, it will only use the content that you provide it. This forms the basis of RAG pipelines.

When to use

- The content is not publicly available on the internet
- You want the model to treat the context as the single source of truth
- Reduce hallucinations and model making assumptions

How it works

- Content that you want the LLM to work on (ie. business data) has to be inserted into your prompt
- This can be done dynamically through Python's formatting

EXAMPLE

Ungrounded Prompt:

You are a security analyst

What are the key security requirements of our company's password policy?

Role Assignment

Context Provision

Task Definition

Grounded Prompt:

You are a security analyst

Use only the information provided in the triple inverted commas to answer the question:

'''Company Password Policy v3.2:

- Passwords must be at least 14 characters long

- Must include uppercase, lowercase, numbers, and symbols '''

Question: What are the key security requirements of our company's password policy?

FEW-SHOT PROMPTING

Providing Examples For the LLM to Model After

Zero-Shot Vs One-Shot Vs Few-Shot

The number of examples used in your prompt

0 = No Examples
One = One Example Used
Few = Two or More

Why Bother?

- Examples help to demonstrate the task to the model
- Model is able to understand the context and expected output better
- Increased likelihood of obtaining a response that matches your desired result

The number of examples matter

- Increasing the number of examples “few-shot” generally results in a better result
- Tasks that are nuanced, requires specific formatting or has varied inputs

EXAMPLE

You are a customer support analyst
Classify the sentiment of each message as Positive, Neutral, or Negative

Examples of messages and expected sentiment:
Input: “The app is easy to use and works perfectly.”
Output: Positive
Input: “The app works, but the interface could be better.”
Output: Neutral
Input: “The app crashes every time I open it.”
Output: Negative

Question: “The app drains my battery in 10 minutes”

Role Assignment

Task Definition

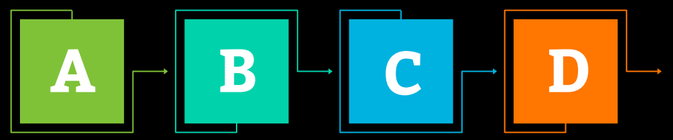
Context Provision:
Few Shot Examples

CHAIN-OF-THOUGHT

Step-by-step reasoning process

Why is this effictive?

It forces the LLM to produce a response that has a clear, effective, and logical reasoning structure.



When to use

- Excellent technique for complex tasks like multi-step reasoning

How it works

- It facilitates problem-solving by guiding the model though a step-by-step reasoning process
- Forces the LLM to break down the problem into a few smaller components
- This boosts the model’s ability to accurately solve multi-step problems

EXAMPLE

You are an economics professor
Follow the steps provided exactly as outlined below to solve the problem

Problem:
A perfectly competitive firm has the following cost functions in the short run:

- Total Cost (TC) = $50 + 10q + q^2$

Find the short run supply curve

Steps to follow:

1. Identify the fixed cost and variable cost from the total cost function
2. Derive the marginal cost (MC) function
3. Derive the average variable cost (AVC) function
4. State the condition under which a firm supplies output in the short run

Role Assignment

Task Definition

Context Provision:
Chain-of-Thought
Reasoning

DIRECTIONAL STIMULUS PROMPTING

Providing hints and keywords to generate desired outputs

Why is this effictive?

It constrains the model's reasoning and content selection to align closely with the user's intended outcome.

When to use

- Tasks requires a specific set of responses
- Context-sensitive

How it works

- Define explicit criteria that govern how the model must respond
- Forces the model to include specific concepts, facts, or reasoning patterns
- Produce outputs that closely adhere to the user-directed content

EXAMPLE

You are a business analyst.
Summarise the company information provided below.

Your summary must explicitly include and address the following keywords:

- Revenue growth
- Operating costs
- Market expansion
- Key risks

Company Data:
XYZ Corp reported a 12% increase in annual revenue driven by strong product sales in Asia. Operating costs rose due to higher logistics and labor expenses.
The company entered two new regional markets but faces increased competition and regulatory uncertainty.

Role Assignment

Task Definition

Context Provision:
Directional Stimulus Prompting

SPECIFYING FORMAT

Defining how LLMs write its responses

You are a business analyst.
Analyze the company information provided below and present your response exactly in the following format.

'''Summary:
- <one sentence overview>
Financials:
- Revenue: <value or description>
- Costs: <value or description>
Risks:
- <list up to 2 risks>'''
Instructions:

- Use the headings exactly as shown
- Do not add or remove sections
- Do not include extra commentary outside the format

Importance

- Getting consistent output for every request
- Ensures that all required segments are incuded in the response
- Ease of refinement

Enhancements

- Use one-shot examples to demonstrate the expected output format
- Incorporate structured step-by-step reasoning to improve clarity and consistency

CODED EXAMPLES

Google colab

Transition to our Google Colab For:

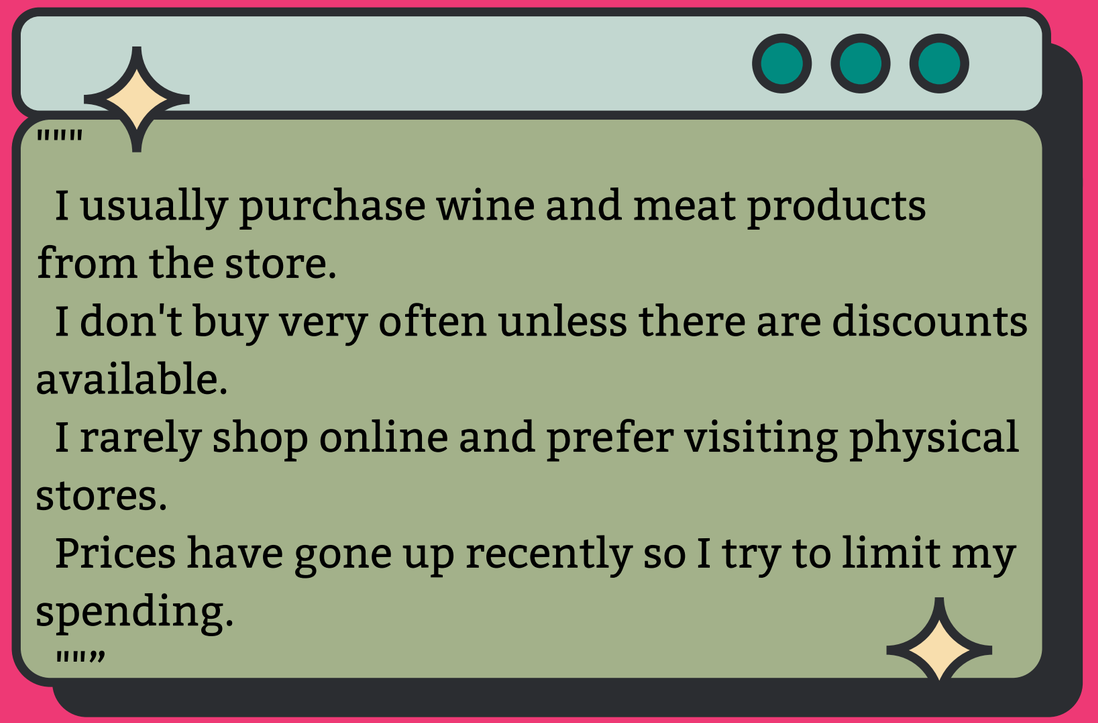
1. Coded Examples of Context Grounding
2. Coded Examples of Few Shot Prompting
3. Coded Examples of Chain-of-Thought Prompting
4. Coded Examples of Directional Stimulus Prompting

TEXT ANALYSIS

What and Why?

Many datasets include open-ended survey questions, company descriptions and/or feedback text, in which case text analysis helps you:

- Summarise large text fields
- Categorise responses
- Extract structured insights



TEXT SUMMARIZATION

USE CASES

1. Condense long text fields
2. Compare groups or segments
3. Extract high-level themes

BEST PRACTICES

1. Chunk long documents
2. Summarise per unit before aggregating (per group not row)
3. Keep prompts constrained

SUMMARY: The customer primarily buys wine and meat products from the store but purchases infrequently due to the lack of discounts. They generally prefer shopping at physical stores instead of online and have been reducing their spending due to recent price increases.
Summary Confidence: 0.9

TEXT CLASSIFICATION

TASKS INCLUDE

1. Topic assignments
2. Sentiment or stance detection
3. Category labelling

KEY CONSIDERATIONS

1. Define labels clearly
2. Enforce structured outputs
3. Validate results with samples

CLASSIFICATION: { "spending_level": "Medium", "price_sensitivity": "High", "preferred_channel": "Store" }
Classification Confidence: 0.85

INFORMATION EXTRACTION

GOAL

To convert unstructured text into structured data
Eg. key phrases, entities, attributes

TECHNIQUES

1. Explicit schemas
2. JSON-formatted outputs
3. One task per prompt

EXTRACTION: { "mentioned_products": ["wine", "meat"], "mentions_discounts": null, "mentions_price_concern": true, "purchase_channel": "physical stores" }
Explanation: The mentioned products are explicitly stated as "wine and meat products", and the customer states that they prefer shopping in physical stores. However, the text does not explicitly mention discounts or prices being mentioned other than the comment about "unless there are discounts available". So the mentions for discounts and purchase price concern are marked as null until more context is provided to confirm either true or false.
Extraction Confidence: 0.83

TEXT ANALYSIS

Confidence Scoring

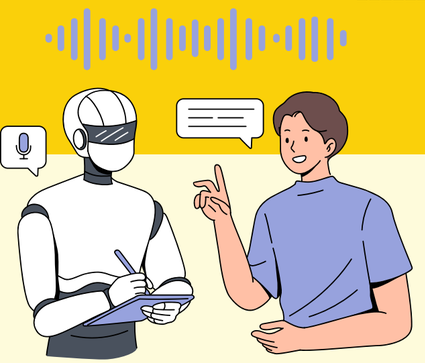
LLMs do not output probabilities, so confidence is approximated using evidence counting.

Rule-based confidence logic:

- count how many explicit signals appear in the text
- more signals → higher confidence
- ambiguity → lower confidence

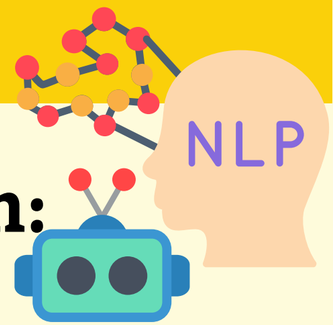
This way the output generated is transparent, explainable and defendable.

LLMS VS CLASSICAL NLP



Use LLMs when:

- Language is nuanced
- Labels are ambiguous
- Interpretability is important



Use Classical NLP when:

- Labels are simple
- Dataset is large
- Cost and speed matter

LLM and NLP are often complementary with hybrid approaches often being used, its not just either/or

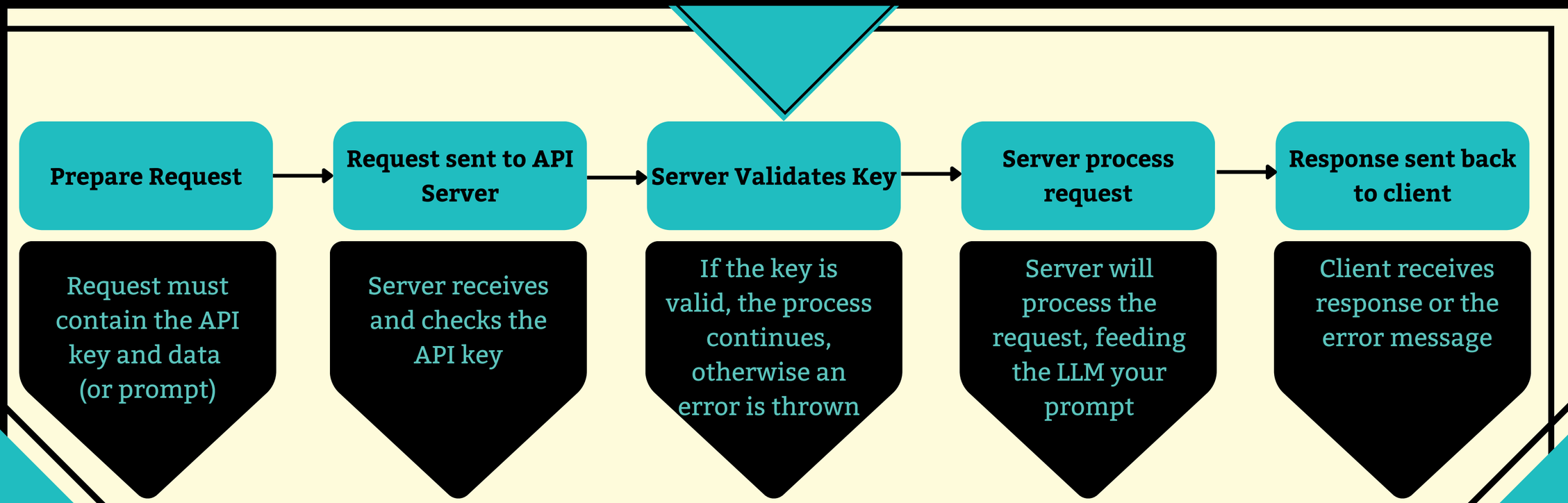
LLMs are augmentation tools not replacements for analysis.

Application Programming Interface

API CALLS

- Allows programmers to integrate data, services, and capabilities from other applications
- API grants us access to a transformer-based language model provided by a developer platform (e.g., OpenAI)
- Think: Functional Abstraction
- We send a structured prompt as input and receive the model's generated response as output
- API keys act as unique credentials used to authenticate API requests

API KEYS AT WORK



SECURITY. SECURITY. SECURITY.

API keys identify you as a user

- ❌ Never expose API keys in client-side or public code
- ❌ Never share your API key with anyone outside your team
- ❌ Never ignore monitoring and rotation

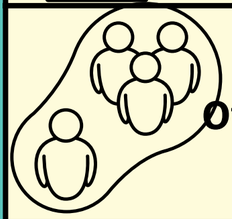
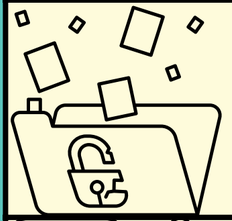
Most LLM developers charge the use of their API
If your API key lands in the hands of unauthorised users, you will end up paying for their usage

API WALKTHROUGH

Using HuggingFace as an example:

- Create our API key
- Deploy our API key
- Retrieve a response with our API key

PITFALLS AND CAUTIONS

PROBLEM	WHAT HAPPENS?	MITIGATION
 **Hallucinations**	• LLMs generate plausible but incorrect outputs • More common when data is sparse or ambiguous	• Constrain prompts • Allow “unknown” outputs • Ground responses with data excerpts
 **Overgeneralisation**	• Small samples lead to sweeping conclusions • Clusters treated as universal truths	• Check segment sizes • Compare across groups • Acknowledge uncertainty
 **Data Leakage**	• Features that indirectly encode outcomes • Using information unavailable at inference time	Check: “Would i realistically know this value when making a decision?”
 **Treating LLM Output as Ground Truth**	• Explanations ≠ evidence • Generated insights must be validated against data	Rule: every claim should be traced back to data

SLIDES WILL ALSO BE ON

Github

https://github.com/NUS-SDS-Workshops/AY-25-26-Public

All SDS workshops code and slides will be on there. Do consider ⭐ starring ⭐ to stay updated!

OUR NEXT WORKSHOP

Collaboration with DSEC Society: Time Series Workshop

27th January 2026 (Week 3)
Tuesday
7pm-9pm

MORE WORKSHOP COMING THIS SEM

Week 3: Collab with DSEC Society - Time Series Workshop
Week 4: Cloud Workshop
Week 5: Collab with Math Society - Matlab Workshop
Week 5: Collab with Product Club - Product Analytics Workshop
Week 9: Collab with AI Society - RL/NLP Workshop
Week 10: MEGA WORKSHOP (will reveal soon)

So follow our telegram @nussds or instagram @nus.sds to stay updated!

FEEDBACK FORM

Please help fill in so we can improve future workshops for you!

THANK YOU AND ALL THE BEST!