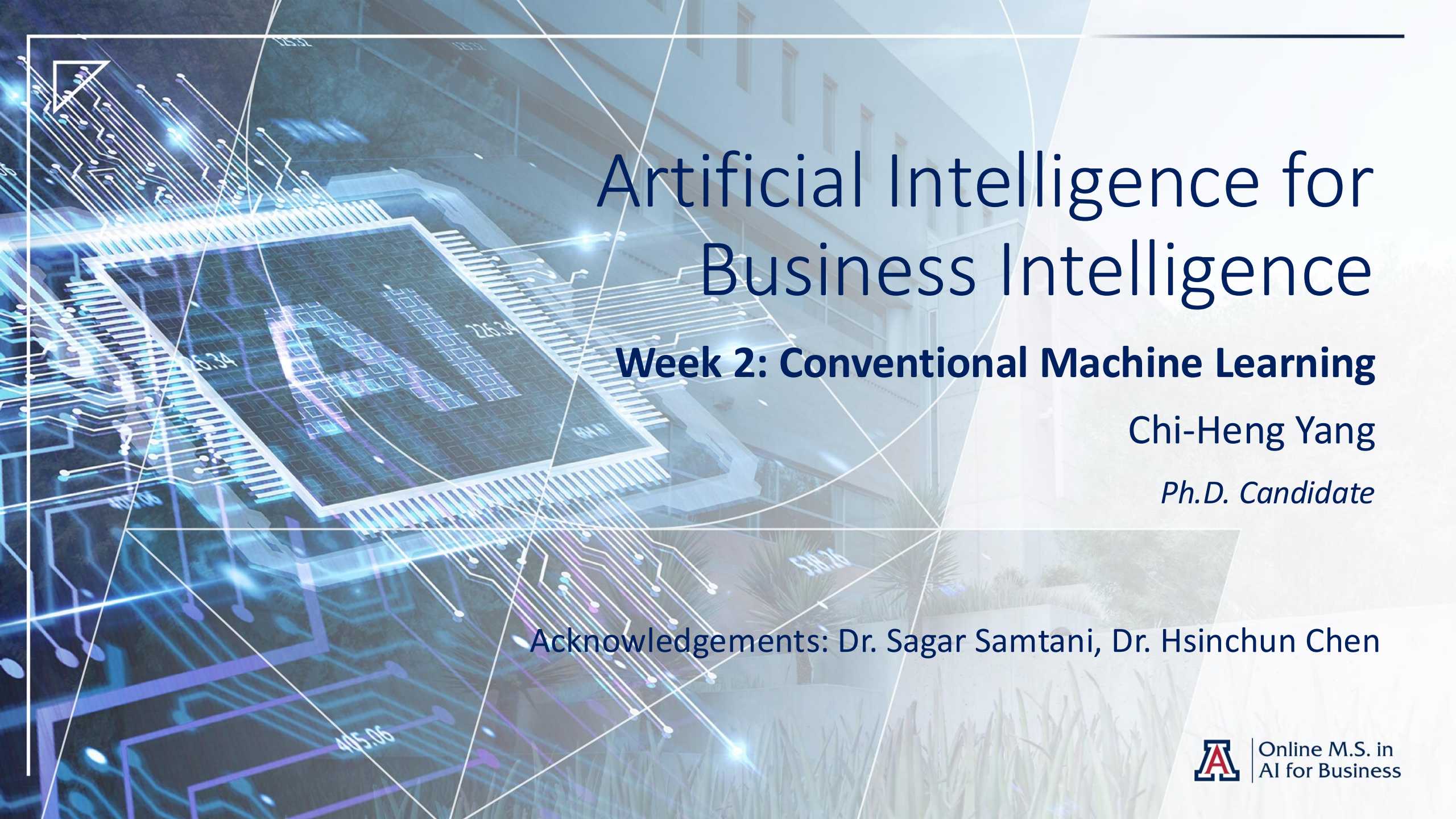


Artificial Intelligence for Business Intelligence

Week 2: Conventional Machine Learning

Chi-Heng Yang

Ph.D. Candidate

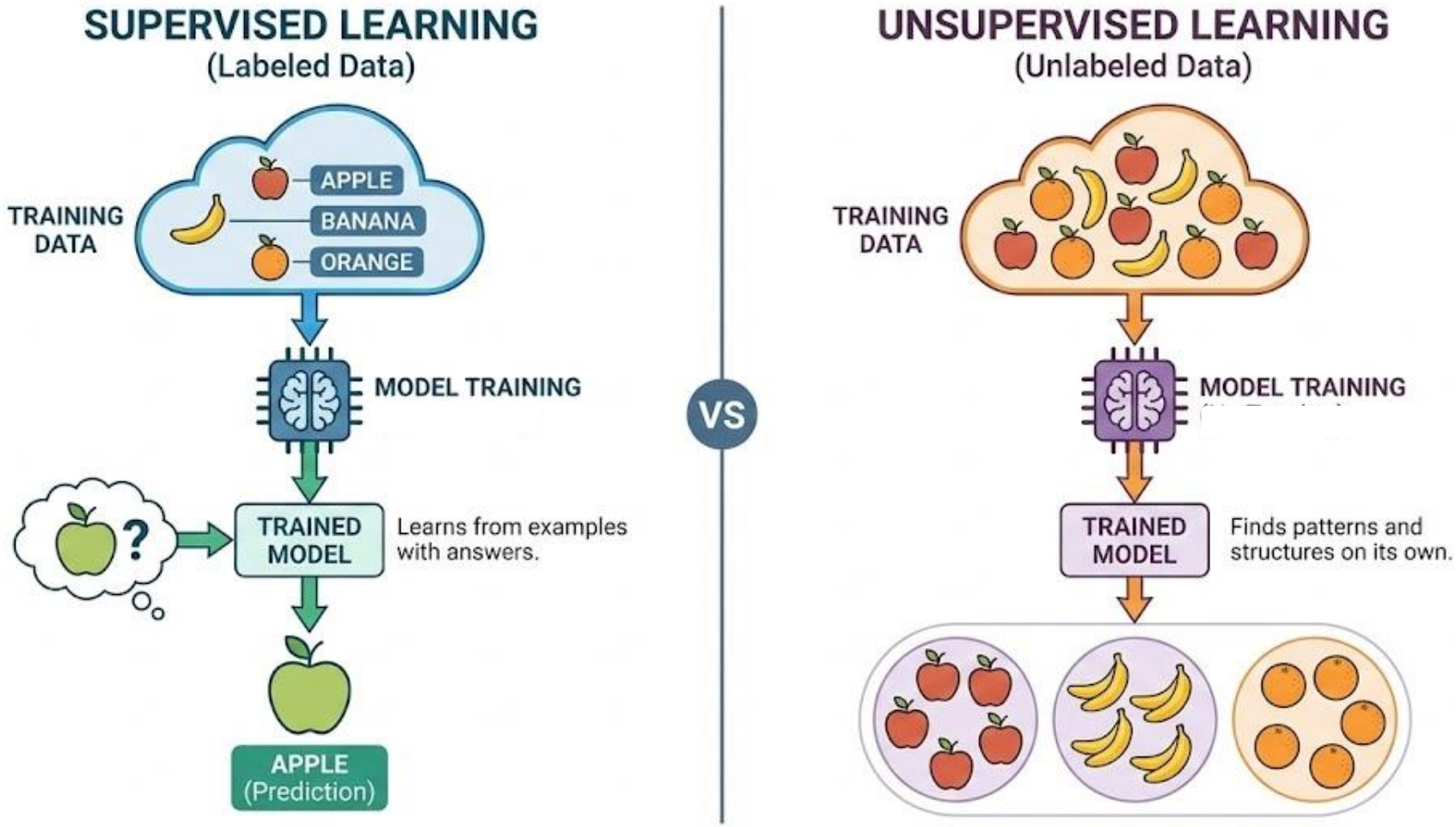
Acknowledgements: Dr. Sagar Samtani, Dr. Hsinchun Chen

Online M.S. in AI for Business

Today’s Learning Objectives

- Understand the core differences between supervised and unsupervised learning.
- Explain how **Decision Trees** and **Support Vector Machines (SVM)** perform classification.
- Compare **K-means** and **DBSCAN** for identifying customer segments or irregular data patterns.
- Understand **Topic Modeling** to discover hidden topics and themes in document collections.

Machine Learning



Classification – Problem and Data

- You are a financial institution that evaluates thousands of loan applications daily
  - Determine whether a **new loan applicant** should be approved based on their financial profile.

- We have historical **loan application data** containing applicant features such as income, credit history, employment status, loan amount, dependents, and marital status.

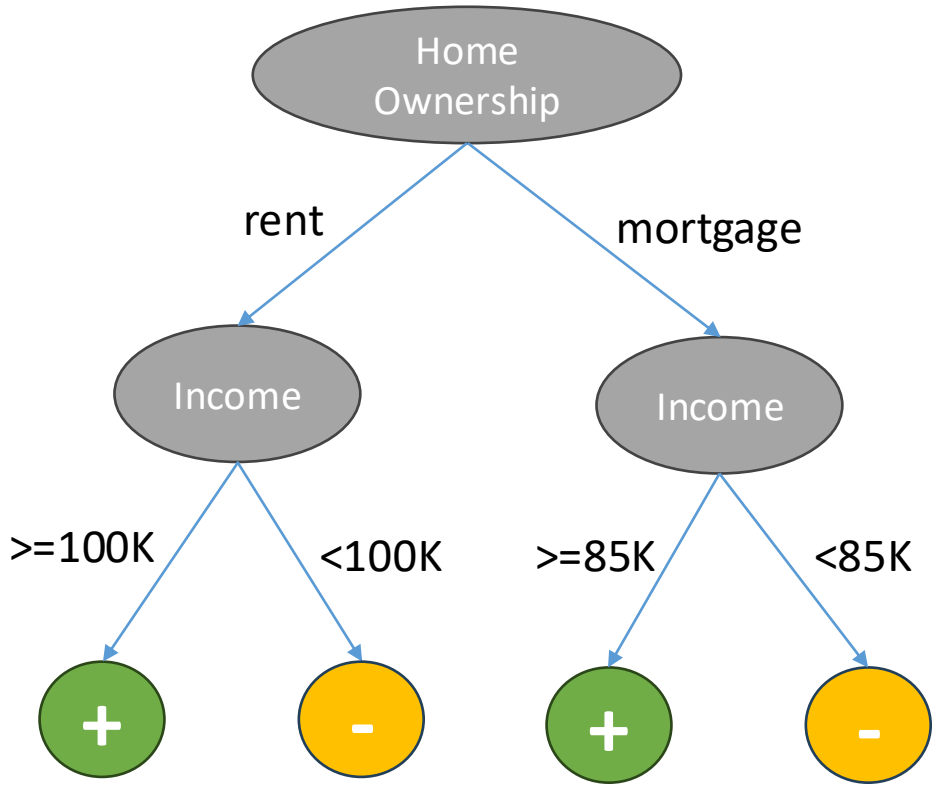
- Not all applicants pose equal risk
  - How should we jointly consider those factors to determine whether to approve their applications?

Classification – Analytic Description

- We have a bunch of features for each data point.
  - Income, loan amount, and employment status for each applicant.
- We have the result (class) based on those features for each data point
  - Those with high income, low loan amount usually get approved
- We want to find such criteria and apply them to future applicants.
  - For example, if their income × 5 >= loan amount, we should approve.
- How do we find such criteria based on historical approval data?

Classification – Decision Tree

- A **decision tree** is a classification procedure that:
  - Breaks down a decision problem into **a sequence of tests**
  - Each internal node = test on an attribute
  - Each branch = outcome of the test
  - Each leaf = classification



- How do we generate such tree?

Classification – Decision Tree

- ID3 (Iterative Dichotomiser 3) is an algorithm invented by Ross Quinlan used to generate a decision tree from a dataset.

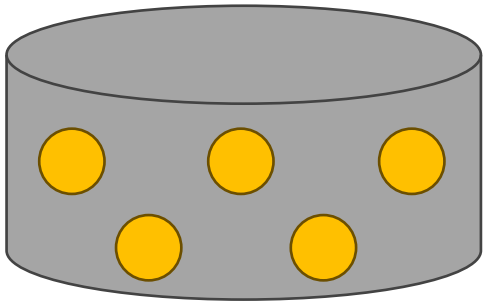
- Given a dataset with multiple instances with:
  - A **target class** (e.g., approved, not approved)
  - A set of **features** (income, education level)

- At each step, it selects the attribute (make a decision node) to divide the dataset based on **entropy**.

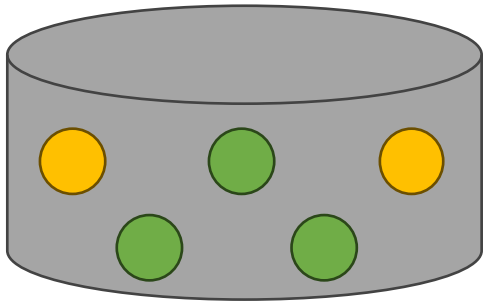
Classification – Decision Tree

- **Entropy** is a measure of the amount of uncertainty in the dataset.

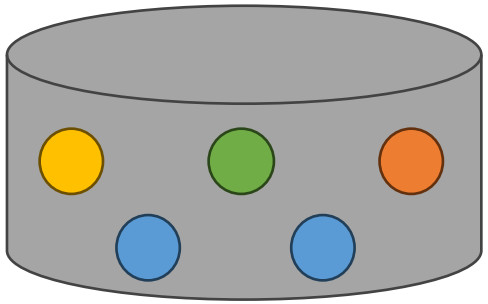
- Imagine that you take out a ball from each jar, how uncertain is your outcome?



100% Yellow



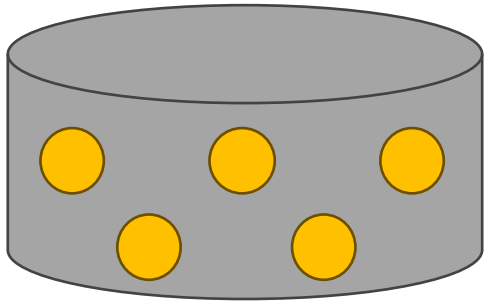
60% Green
40% Yellow



40% Blue
20% Yellow
20% Green
20% Orange

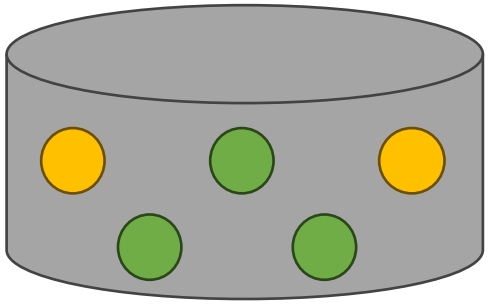
Classification – Decision Tree

- Entropy $H = -\sum_{i=1}^{n} p_i \log_2(p_i)$



$P_{yellow} = 1$

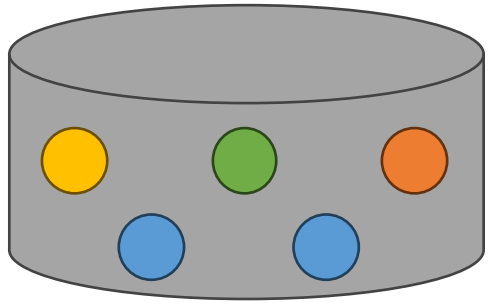
$H = -\left(1.0 \cdot \log_2(1.0)\right) = 0$



$P_{green} = 0.6$

$P_{yellow} = 0.4$

$H = -\left(0.6 \log_2(0.6) + 0.4 \log_2(0.4)\right) = 0.971$



$P_{blue} = 0.4$

$P_{yellow} = 0.2$

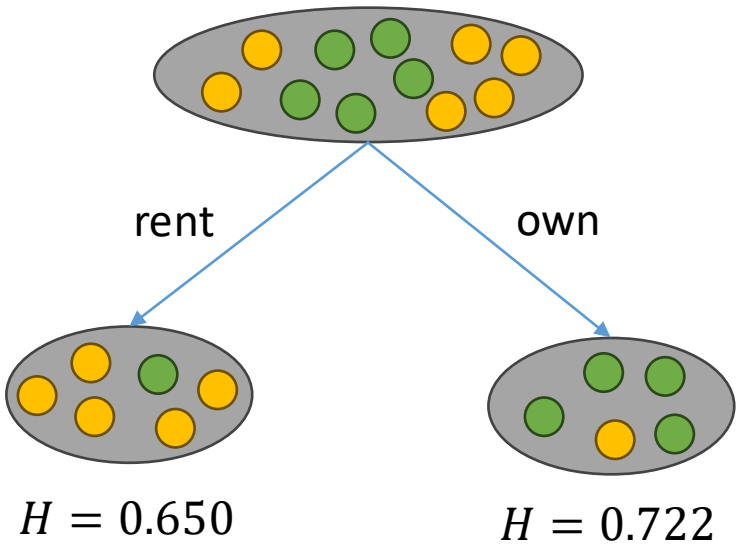
$P_{green} = 0.2$

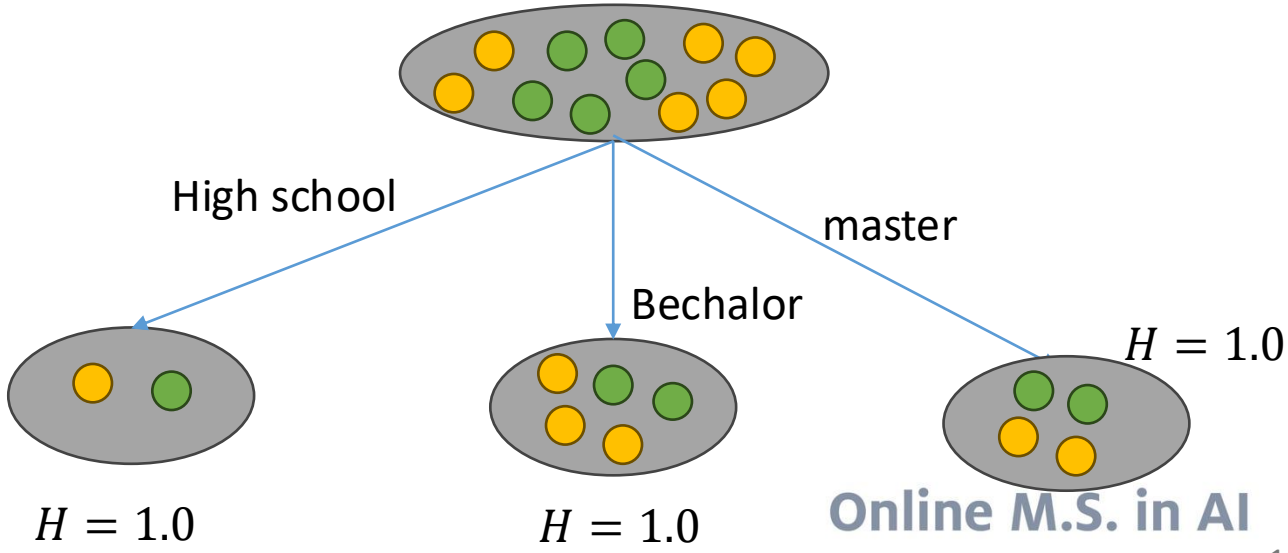
$P_{orange} = 0.2$

$$H = -\left(0.4 \log_2(0.4) + 0.2 \log_2(0.2) + 0.2 \log_2(0.2) + 0.2 \log_2(0.2)\right) = 1.922$$

Classification – Decision Tree

- When creating branches, we calculate the entropy before dividing.
  - For each attribute, compute the sum of entropy after dividing.
- We choose the criteria that lead to the largest uncertainty reduction (largest decrease in entropy).





Classification – Decision Tree

- In summary, ID3 works as follow:

- At each node
  - For each attribute, calculate entropy reduction
  - Choose the attribute that reduces entropy the most
  - Create branches and repeat the same process for each new node

- We stop when:
  - There are no more attributes; or
  - There is no more entropy (we can perfectly predict the class)

Classification – Decision Tree

- Other algorithms with a different splitting method:
  - **C4.5**: Handles continuous attributes and missing values
  - **CART**: Uses *Gini index* (classification) or *MSE* (regression)
  - **CHAID**: Uses *chi-square* significance testing to choose splits

- Decision trees are useful because they:
  - Are **interpretable** (rule-like decisions)
  - Map naturally to human reasoning (if–then rules)
  - Are computationally efficient
  - Can represent complex classification boundaries by successive splits

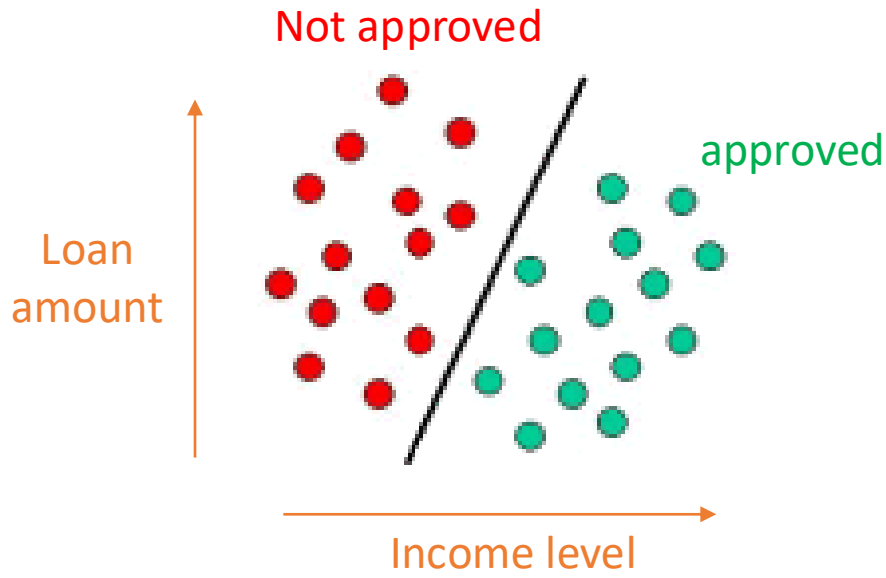
Classification – Decision Tree

- They are especially suitable for classification tasks that require:
  - **Interpretability**: we can see the exact rules that lead to the decision.
  - **Categorical attribute**: they handle categorical features naturally without encoding.

- However, decision trees have some limitations:
  - **Highly prone to overfitting:** A single tree can memorize the training data, creating overly complex decision boundaries that don't generalize well.
  - **Greedy splits:** Trees make the best local split at each step, which doesn't guarantee the best overall model.

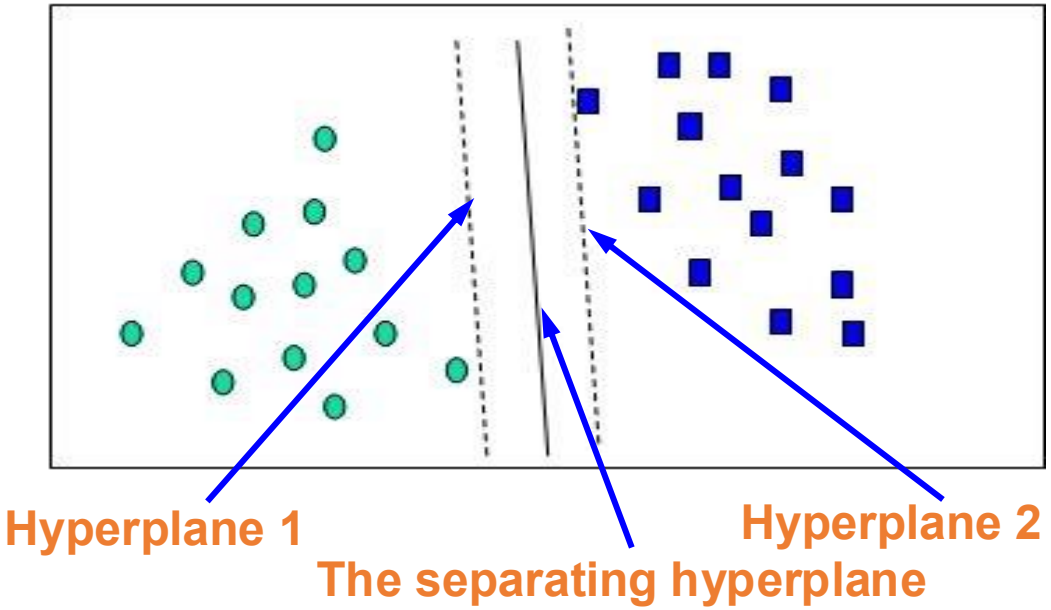
Classification – SVM

- Support Vector Machines (SVMs) are a set of machine learning approaches used for classification and regression.
  - Developed by Vladimir Vapnik and his co-workers at AT&T Bell Labs in the mid 90's.
- SVM is based on the concept of decision planes that define decision boundaries.
- A decision plane is one that separates a set of objects having different class memberships.



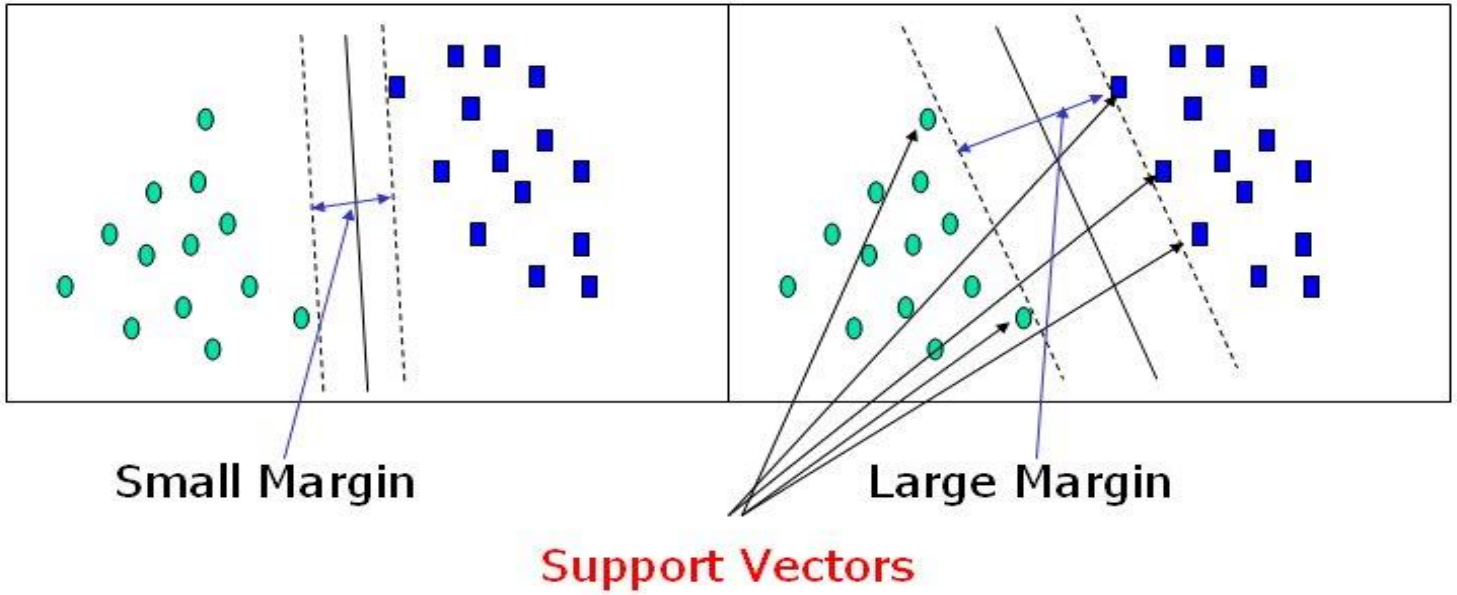
Classification – SVM

- SVM aims to find the **best decision plane** between classes.
- SVM views the input data as two sets of vectors in an *n*-dimensional space.
- It constructs a **separating hyperplane** in that space, one which maximizes the *margin* between the two data sets.
- To calculate the margin, two parallel hyperplanes are constructed, one on each side of the separating hyperplane.



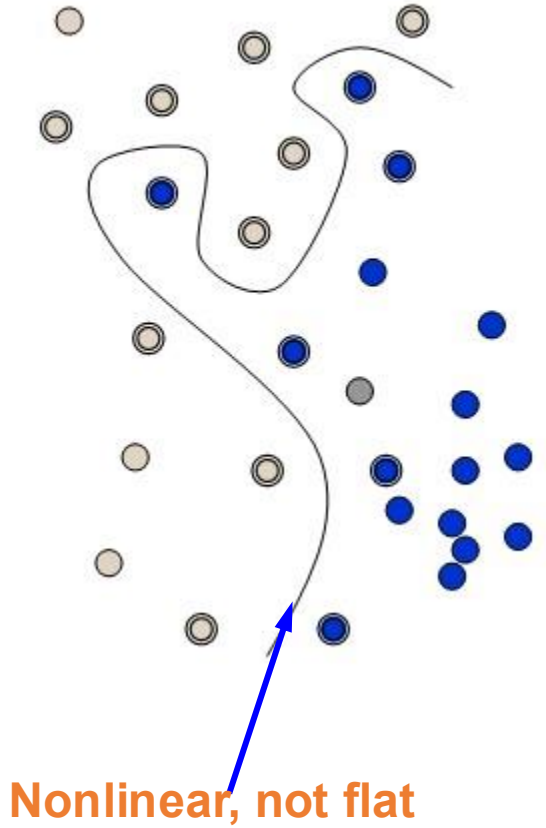
Classification – SVM

- A good separation is achieved by the hyperplane that has the largest distance to the neighboring data points of both classes.
- The vectors (points) that constrain the width of the margin are the *support vectors*.
- An SVM analysis finds the line (or, in general, hyperplane) that is oriented so that the margin between the support vectors is maximized.



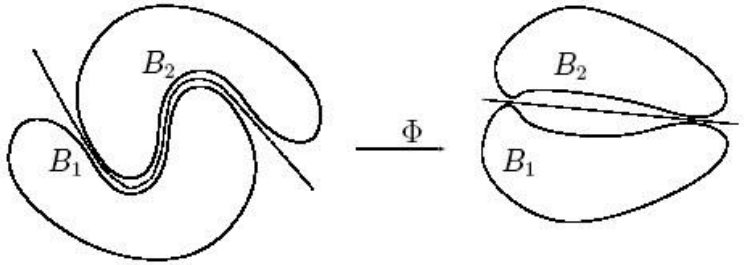
Classification – SVM

- The simplest way to divide two groups is with a straight line, flat plane, or an N-dimensional hyperplane.
- But what if the points are separated by a nonlinear region?
  - Straight Line or flat plane dose not fit anymore.
- Rather than fitting nonlinear curves to the data, SVM handles this by using a *kernel function* to map the data into a different space where a hyperplane can be used to do the separation.



Classification – SVM

- Kernel function Φ: map data into a different space to enable linear separation.



- Kernel function allows SVM models to perform separations even with very complex boundaries.
  - Linear kernel (equivalent to no mapping)
  - Polynomial kernel
  - Radial basis kernel
  - Convolution kernel
  - Your own kernel function

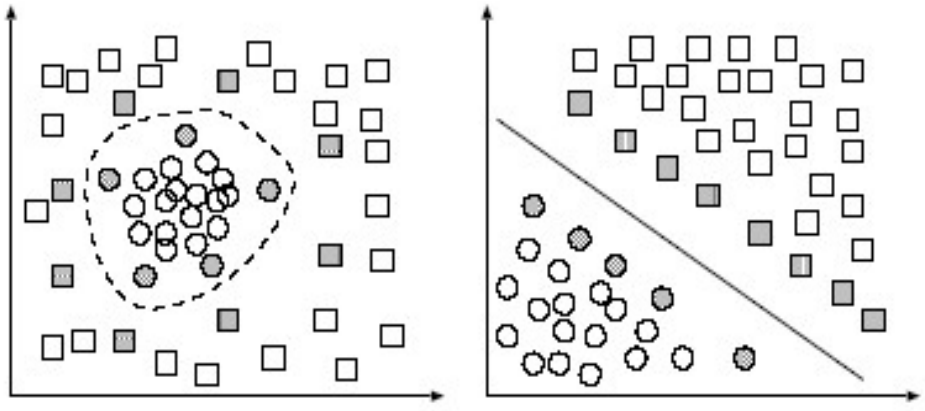
Classification – SVM

- Example: the **Radial Basis Function (RBF)** kernel

- Using the RBF kernel, the data is implicitly mapped into a higher-dimensional space.
  - In this new space, the classes **become linearly separable**.

- A simple **linear hyperplane** can now separate the transformed data.

Radial basis function: exp(-gamma*|u-v|^2)



(a) Radial Basis Function (b) RBF mapping

Separable classification with Radial Basis kernel functions in different space. Left: original space. Right: feature space.

Classification – SVM

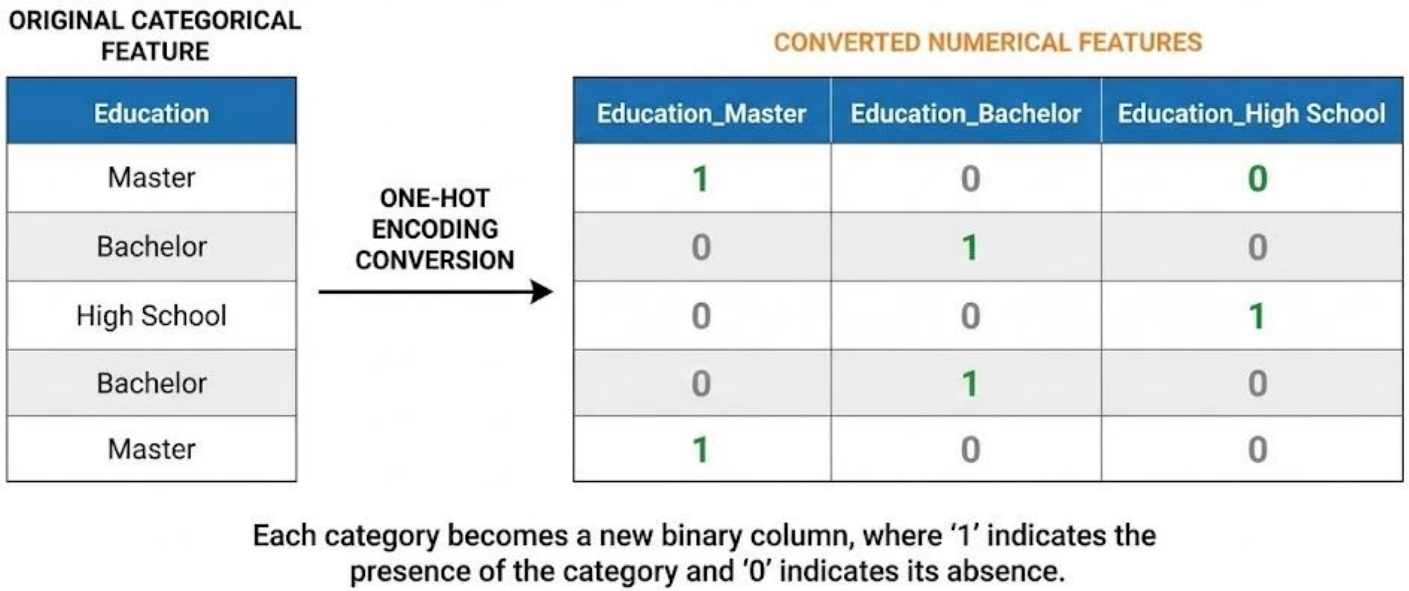
- SVM is suitable for classification tasks that:
  - Work primarily with **numerical features**
  - Benefit from a **large-margin decision boundary**

- What if there are categorical features?
  - Education (Master, Bachelor, High School)
  - Home ownership (rent, mortgage, …)

- SVMs do **not** handle raw categorical data directly.

Classification – SVM

- Convert categorical variables into numerical features using **one-hot encoding techniques.**



Classification – Python Example

- **Python example**: loan application classification using Decision Tree and SVM
  - Refer to supplemental materials (*classification.ipynb, loan_data.csv*)
- **Data**: past loan application records with customer information and decisions
- **Key Packages**: sklearn.tree, sklearn.svm

LOAN_ID	APPLICANT_INCOME	LOAN_AMOUNT	CREDIT_HISTORY	LOAN_STATUS
LP001002	5849	NaN	1	Y
LP001003	4583	128	1	N
LP001005	3000	66	1	Y
LP001006	2583	120	1	Y
LP001008	6000	141	1	Y

Clustering – Motivation

- A retailer has the data of their own customers:
  - Age
  - Gender
  - Annual income
  - Spending score

- It currently treats all customers uniformly with the same marketing, promotions, and engagement strategies.

Clustering – Motivation

- However, different groups respond differently to promotions, pricing, and product offerings.

- The retailer wants to identify **distinct customer segments.**
  - They can tailor offers, allocate budget wisely, retain high-value customers, and re-engage under-performers.

Clustering

- Clustering algorithm groups data points into clusters (groups) such that data points within the same cluster are similar to each other.
- There are no predefined labels/classes. The cluster pattern is identified solely based on the attributes of data points.
- For example, we might find several clusters of customers:
  - **Premium Loyalists**: High income, high spending; ideal for VIP perks and exclusive offers.
  - **Price-Sensitive Professionals**: High income, low spending; need engagement and targeted incentives.
  - **Impulsive Shoppers**: Young, lower income, high spending; respond to trends and promotions.

Clustering – K-means

- K-means algorithm is a simple iterative method to partition a given dataset into a user-specified number of clusters k.
  - Each instance is represented as a vector (e.g., a vector consists of age, annual income, and spending score).

- **Initialization**: we randomly select k data points as the centroid for each cluster.

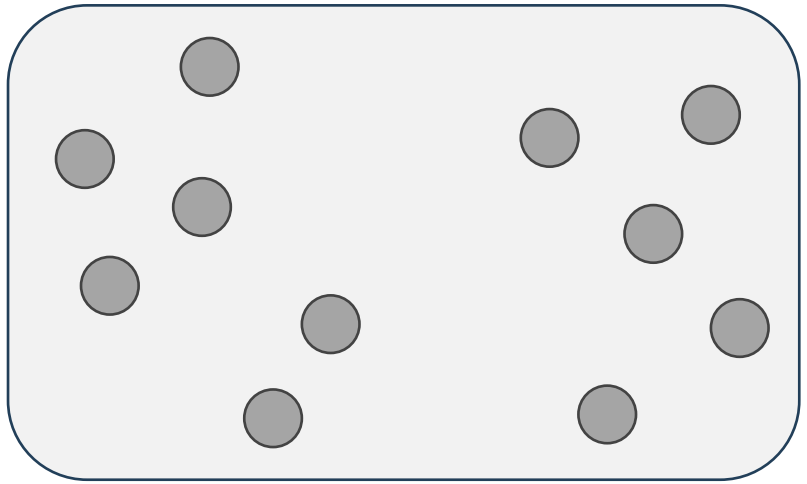
- **Iterate**: K-means algorithm iterates between two steps:
  1. **Data assignment**: each data point is assigned to its closest centroid.
  2. **Relocation of means**: each cluster representative is relocated to the center (mean) of all data points assigned to it.

- **Converge**: Stop when the data assignment (step 1) no longer changes.

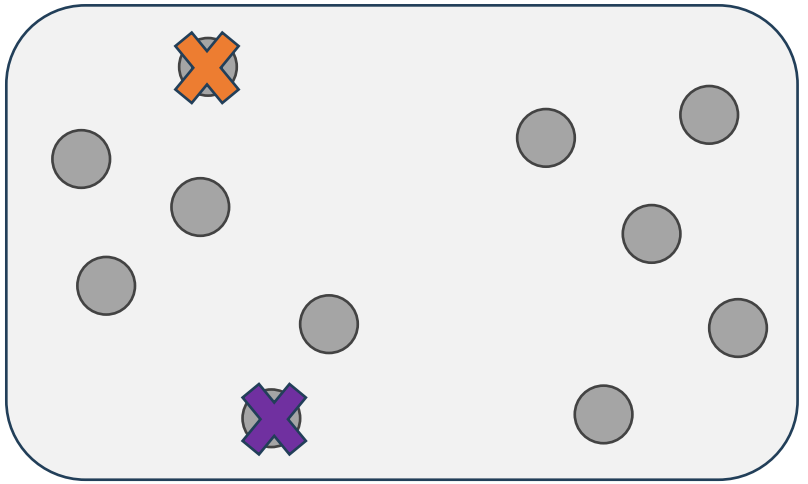
Clustering – K-means

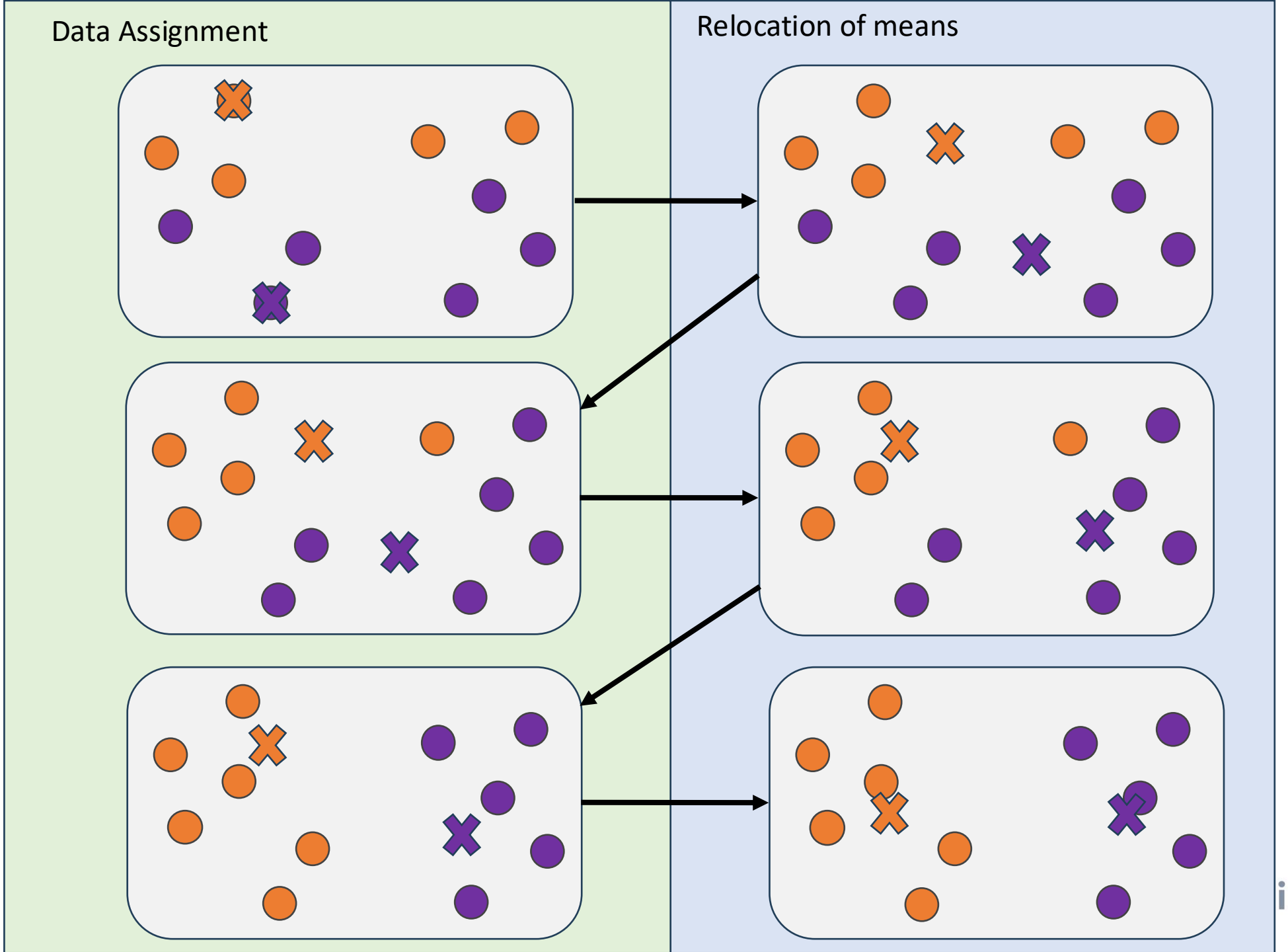
- Consider this example with 11 entities.

Dataset

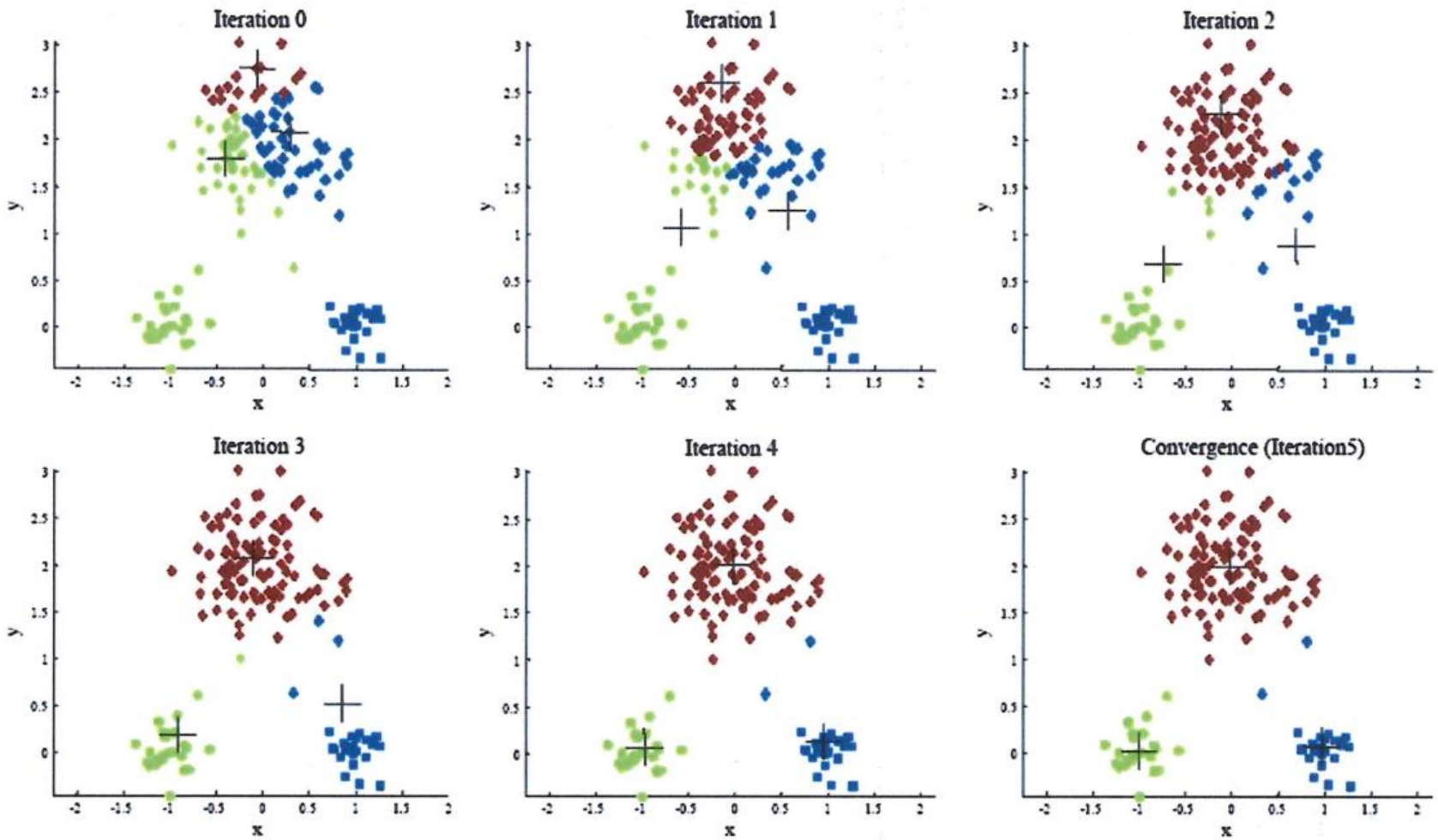


Initialization



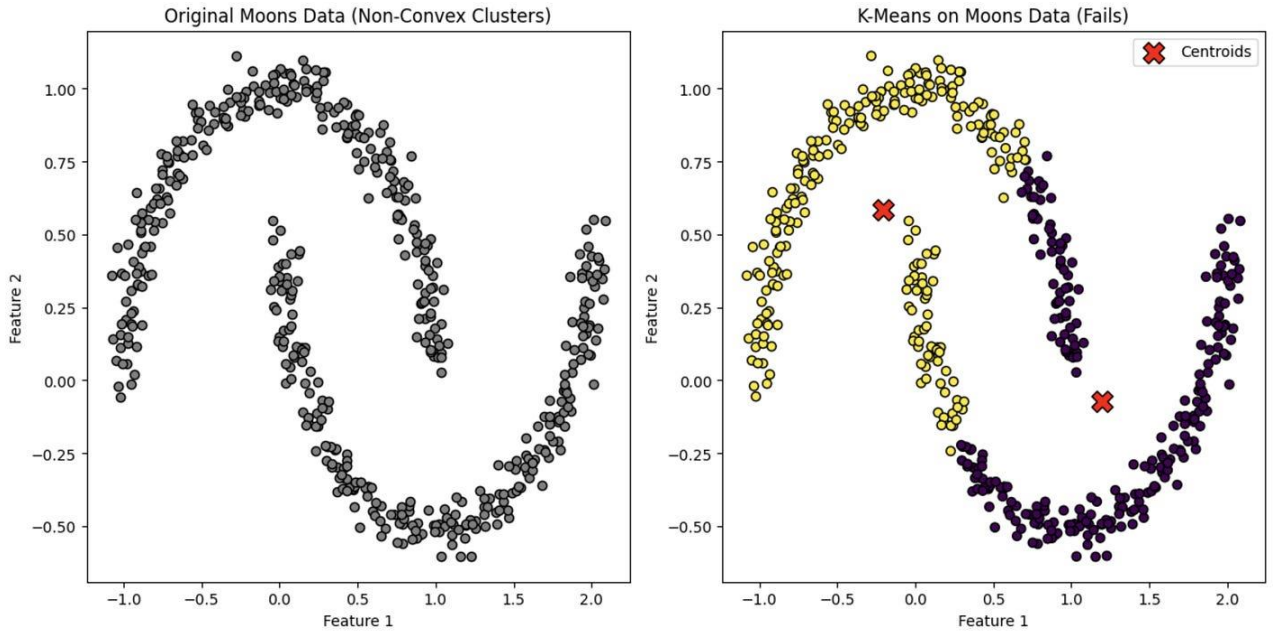

Data Assignment
Relocation of means

Clustering – K-means



Clustering – K-means

- K-means has some limitations:
  - The number of cluster k has to be decided in advance
  - Clustering result depending on the initial partition
  - No guarantee on global optimum
  - It doesn't work well when clusters are not compact



Clustering - DBScan

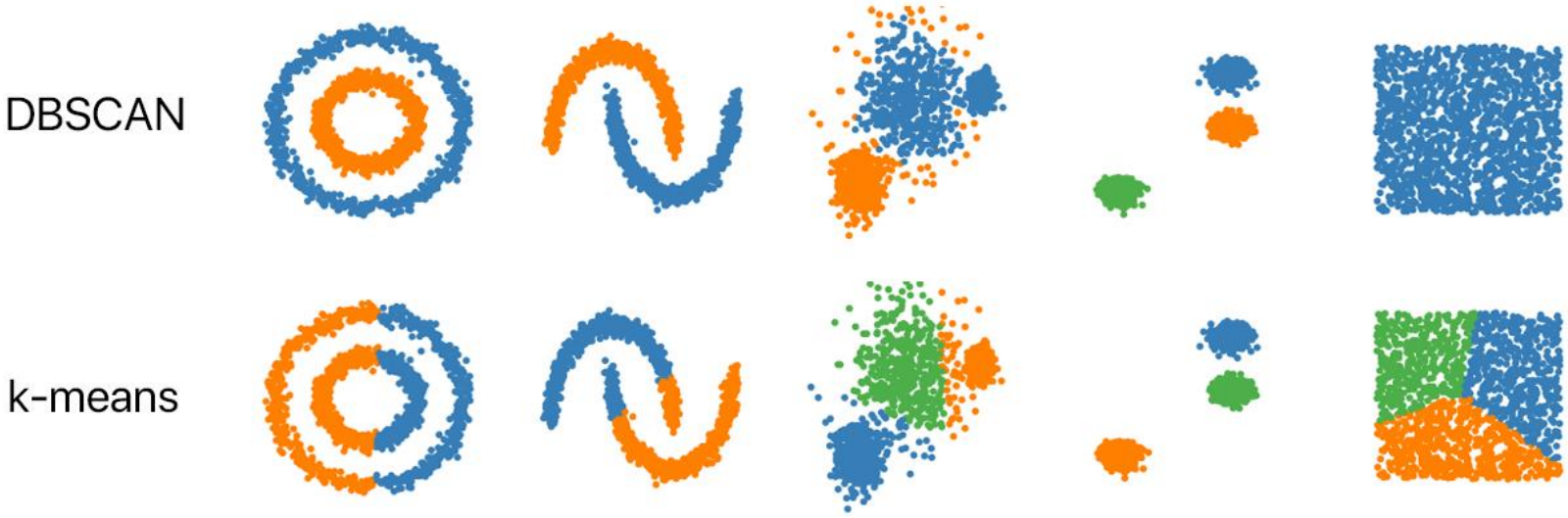
- Real-world data often contains:
  - Clusters of irregular shapes
  - Clusters with different densities

- DBSCAN was created to solve these issues by:
  - Detecting dense regions of points automatically
  - Requiring no predefined number of clusters

- This makes it ideal for messy, real customer or product data where clusters are not clean or well separated.

Clustering - DBSCAN

- DBSCAN algorithm is a density-based method to discover clusters of arbitrary shape and identify noise in a dataset
- Core idea:
  - Clusters are formed where data points are **densely packed**
  - Points that lie in sparse regions are marked as **noise/outliers**.



Clustering - DBSCAN

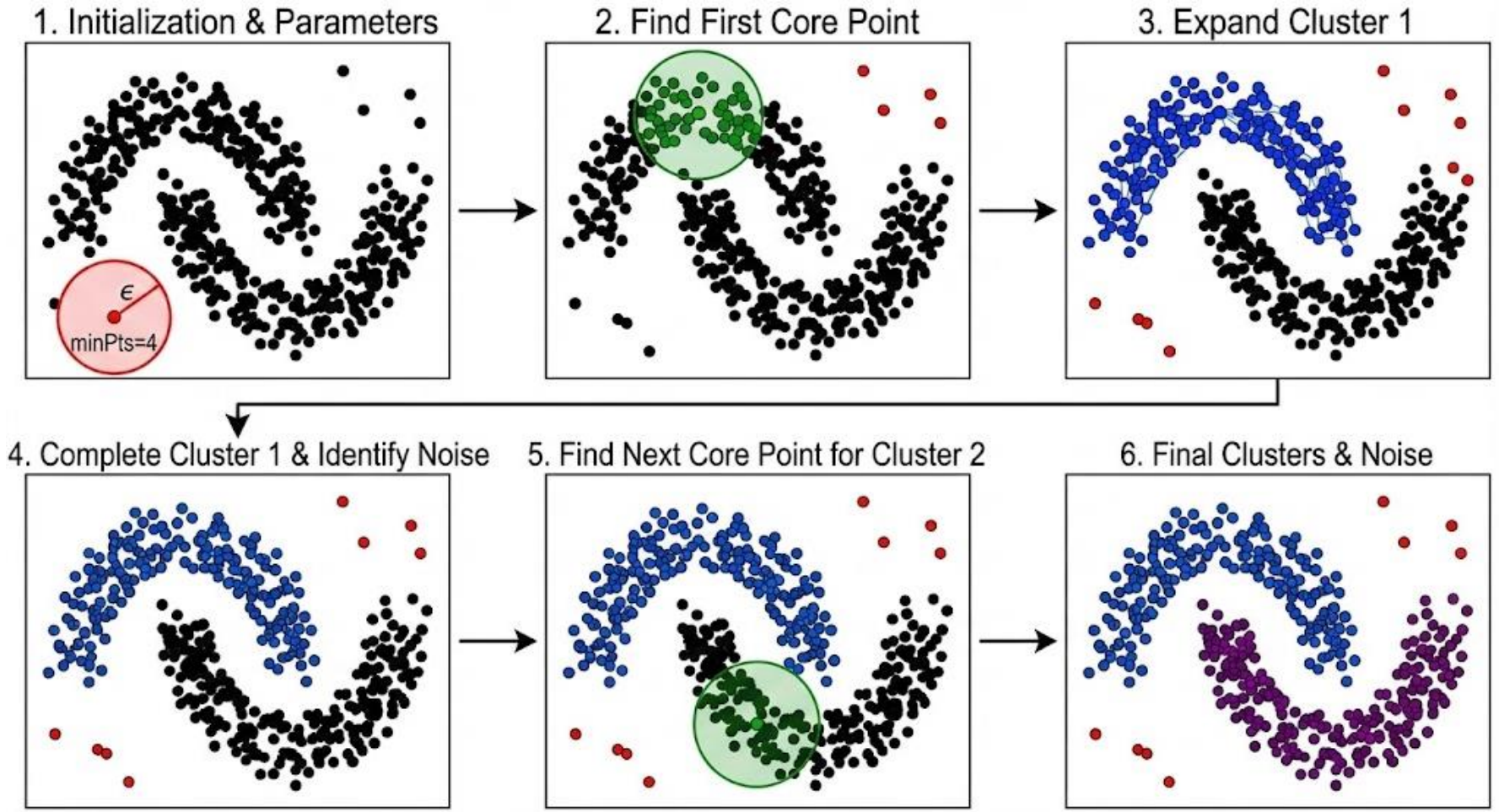
- Parameter initialization
  - ϵ: the radius defining a neighborhood around a point
  - **minPts**: minimum number of points required to form a dense region

- Iterate through two steps:
  - **Neighborhood check**: For each point, checks how many points fall within distance ϵ
    - If ≥ minPts → the point is a core point
    - If < minPts but near a core point → border point
    - Otherwise → noise

  - **Cluster expansion**: When a core point is found, create a new cluster and **expands** it by recursively adding all points that are density-reachable (directly or through chains of core points).

- Stop until all points are visited

Clustering – DBSCAN



Clustering – DBSCAN

- **Advantages** of DBSCAN:
  - The number of clusters does not need to be decided in advance.
  - Can identify clusters of arbitrary and irregular shapes.
  - Robust to outliers as it explicitly labels noise.

- But there are still some **limitations**:
  - It doesn’t work well when clusters have varying densities.
  - Sensitive to the choice of input parameters.
  - Decreased effectiveness in high-dimensional spaces.

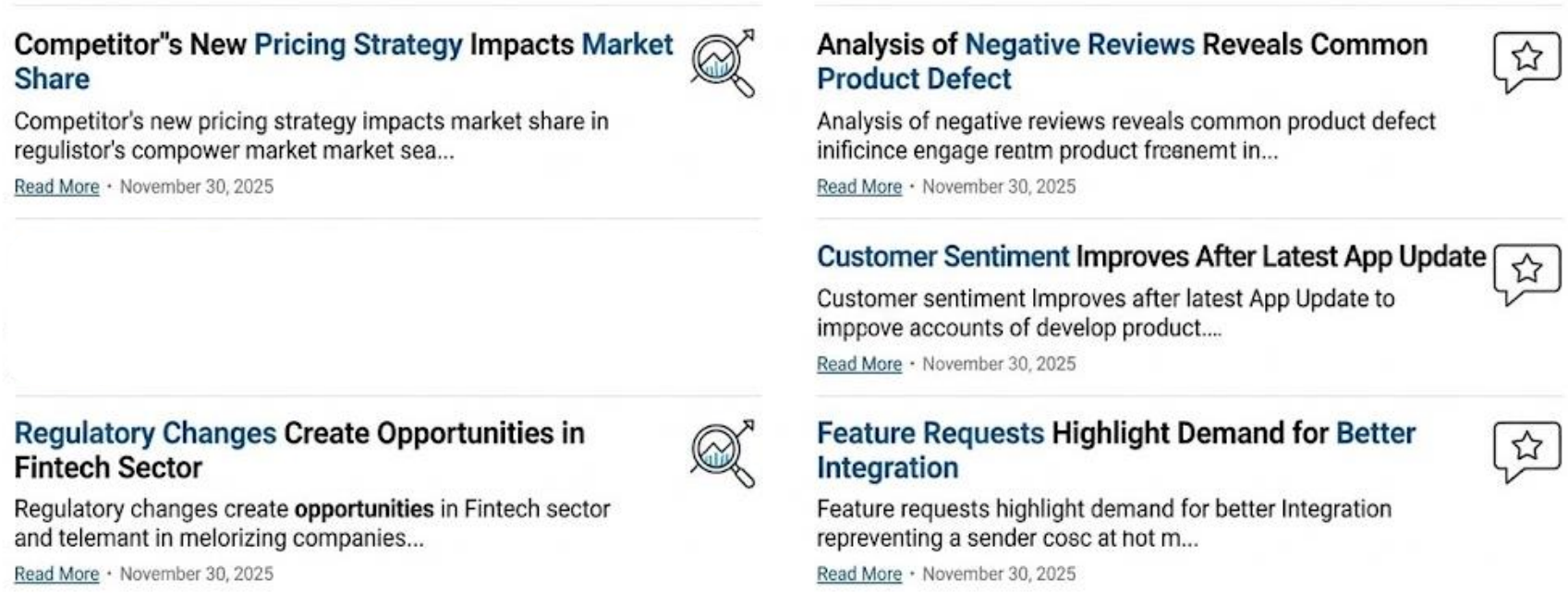
Clustering – Python Example

- **Python example**: customer segmentation
  - Refer to supplemental materials (*clustering.ipynb, Mall_Customers.xls*)
- **Data**: mall customer information (e.g., age, income, spending)
- **Key Packages**: sklearn.cluster

Topic Modeling

- So far, we have been dealing with numerical or categorical data.
  - But some data is in textual format (e.g., news articles).

- Suppose you have some news articles, how do you know what they are discussing?

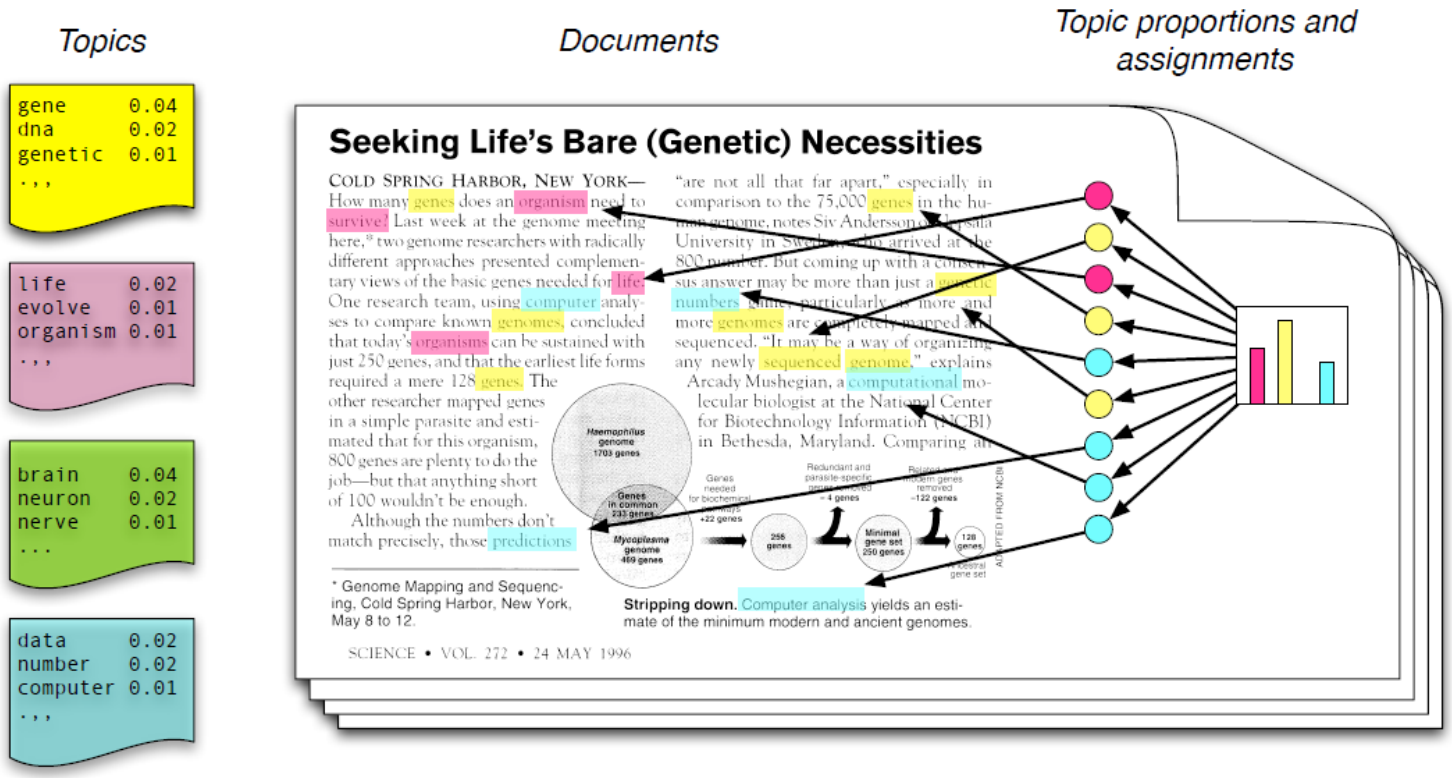


Topic Modeling

- Topic models are a suite of algorithms for discovering the main themes from an unstructured collection of documents.
- Topic models can be applied to massive collections of documents to automatically organize, understand, search, and summarize large electronic archives.
- Latent Dirichlet Allocation (LDA), a technique based on Bayesian Modeling, is the most commonly used nowadays.

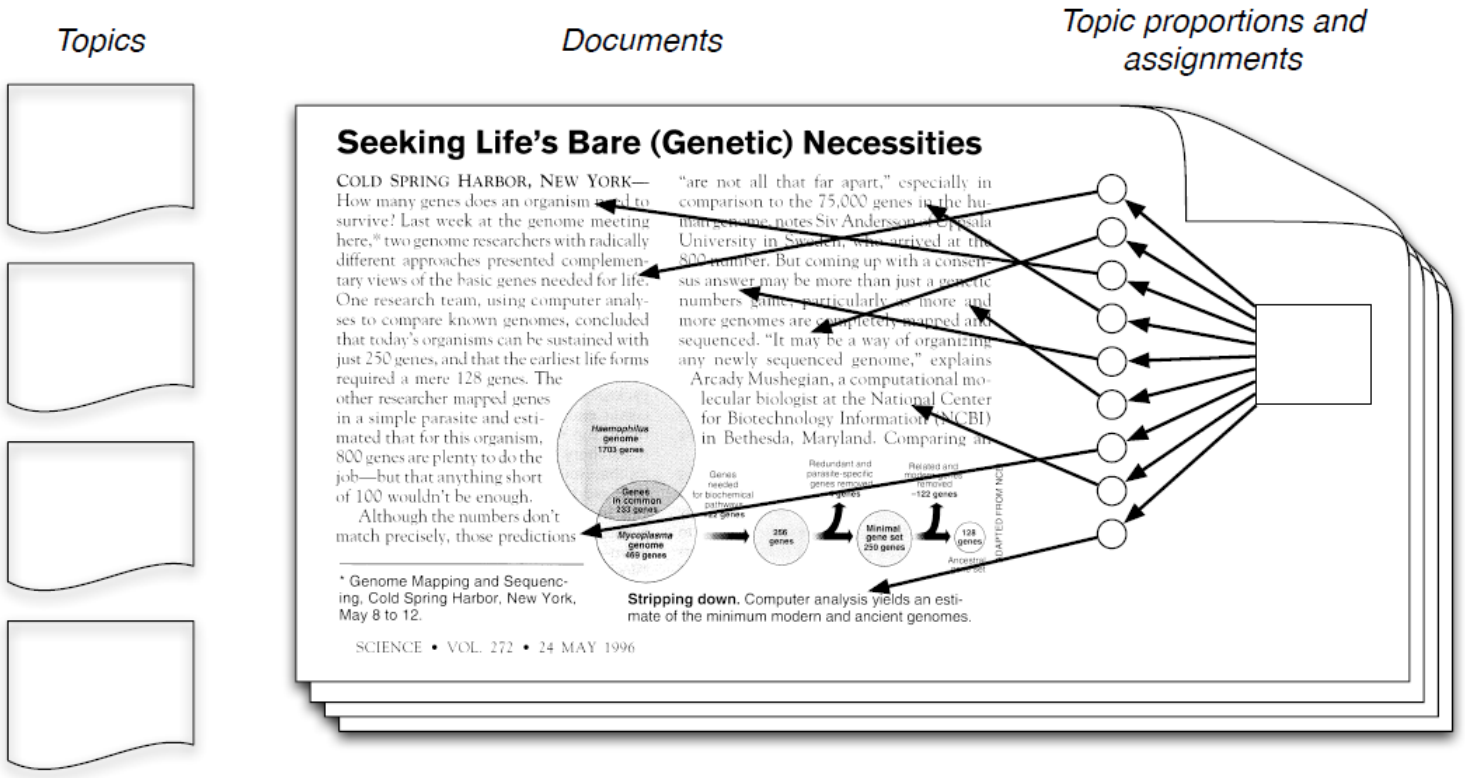
Topic Modeling – LDA

- Each **topic** is a distribution of words.
- Each **document** is a mixture of corpus-wide topics.
- Each **word** is drawn from one of those topics.



Topic Modeling – LDA

- But in reality, we only observe documents. The other structures are hidden variables. Our goal is to **infer** the hidden variables.

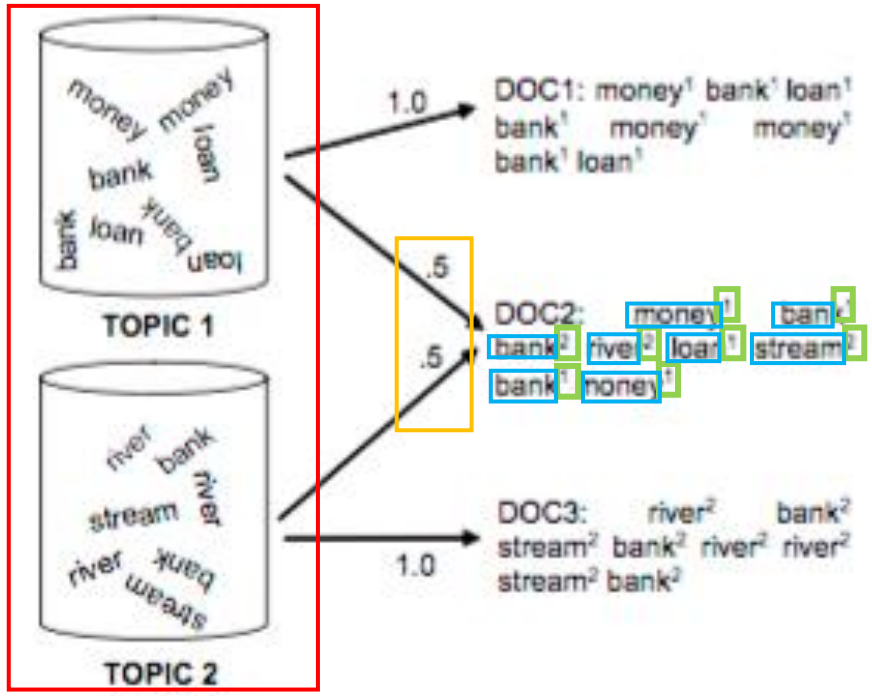


Topic Modeling – LDA

- Assumptions on all variables:
  - *Word*: the basic unit of discrete data
  - *Document*: a collection of words (*exchangeability* assumption)
  - *Corpus*: a collection of documents
  - *Topic* (hidden): a distribution over words & the number of topics K is known.
- Assumptions on how texts are generated:

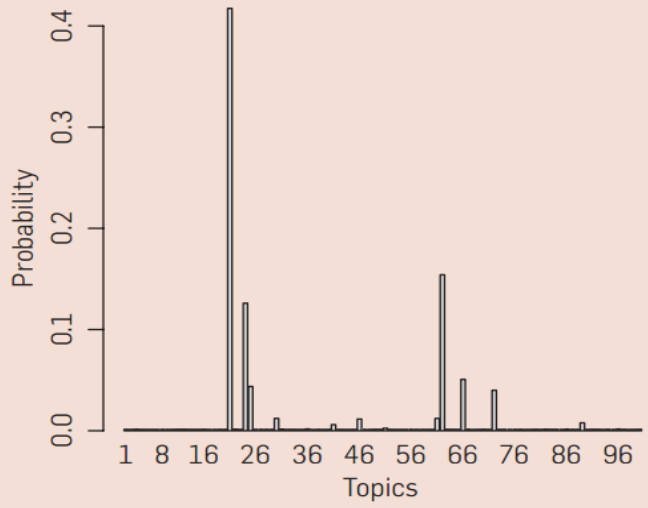
For each topic k,
draw a multinomial over words $\beta_k \sim Dir(\eta)$
For each document d,

- Draw a document topic proportion $\boldsymbol{\theta}_d \sim Dir(\alpha)$
- For each word $w_{d,n}$:
  - Draw a topic $z_{d,n} \sim Multi(\boldsymbol{\theta}_d)$
  - Draw a word $w_{d,n} \sim Multi(\beta_{z_{d,n}})$



Topic Modeling – LDA

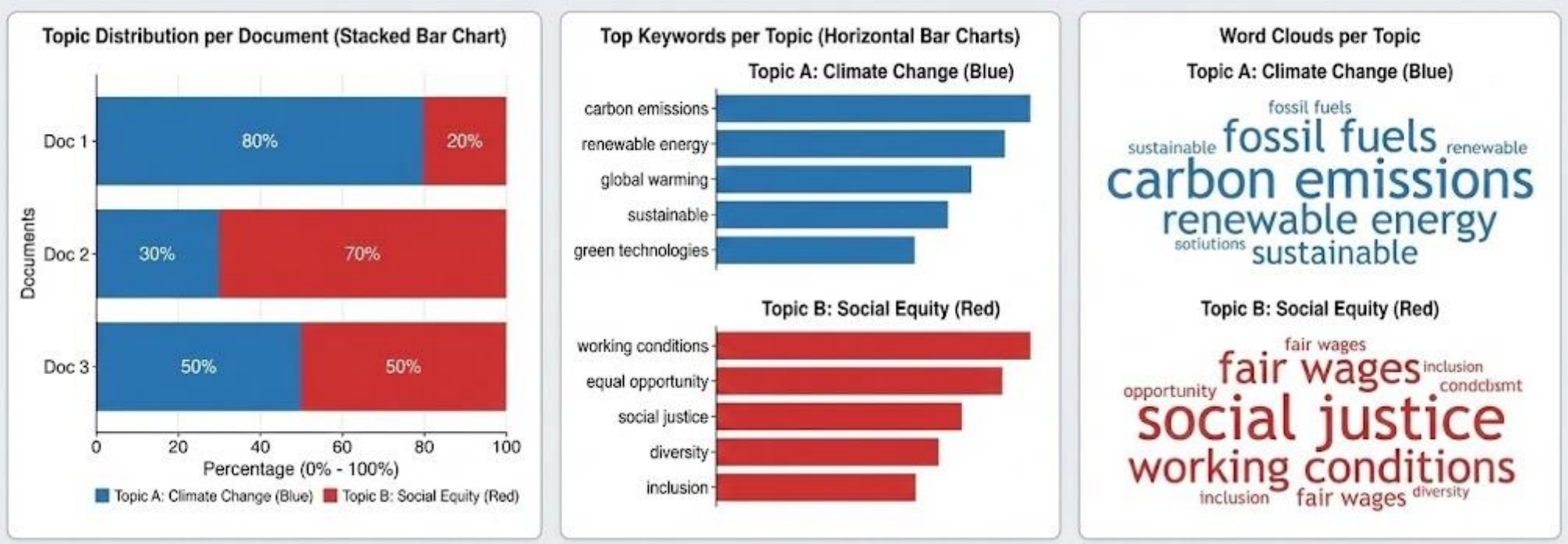
The resulting output from an LDA model would be sets of topics containing keywords which would then be manually labeled. On the left are the inferred topic proportions for the example article from the pervious figure.



"Genetics"	"Evolution"	"Disease"	"Computers"
human	evolution	disease	computer
genome	evolutionary	host	models
dna	species	bacteria	information
genetic	organisms	diseases	data
genes	life	resistance	computers
sequence	origin	bacterial	system
gene	biology	new	network
molecular	groups	strains	systems
sequencing	phylogenetic	control	model
map	living	infectious	parallel
information	diversity	malaria	methods
genetics	group	parasite	networks
mapping	new	parasites	software
project	two	united	new
sequences	common	tuberculosis	simulations

Topic Modeling – Visualization

- Topic modeling can be visualized through
  - **Topic Distribution per Document** to show how dominant each topic is in each document
  - **Topic–Word Bar Charts** to show word importance in each topic
  - **Word Clouds per Topic** to get a brief view of the topic.



Review Questions

- In the ID3 algorithm, how is entropy used to determine which attribute should be the next decision node?
- What is a "separating hyperplane," and why does SVM aim to maximize the margin between support vectors?
- Which clustering algorithm is better suited for irregular, "non-compact" shapes, and why?
- In topic modeling, what is the relationship between a "topic" and a "distribution of words"?

Next Week

Problem Identification

Stakeholder

Scenario

Current Approach

Limitation

Data Collection and Extraction

Data Collection

Data Extraction

AI-enabled Analytics

Machine Learning

Deep Learning

Advanced Techniques

(**LLM**, GNN, TSFM, RL)

Visualization

System and User Interface

Course of Action

Online M.S. in AI