



DATA SCIENCE IN PYTHON

Unsupervised Learning

★★★★★ *With Expert Data Science Instructor Alice Zhao*



*Copyright Maven Analytics, LLC

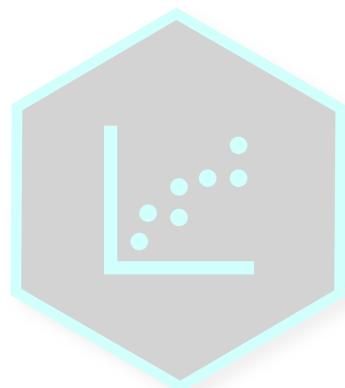
ABOUT THIS SERIES

This is **Part 4** of a **5-Part series** designed to take you through several applications of data science using Python, including **data prep & EDA, regression, classification, unsupervised learning & NLP**



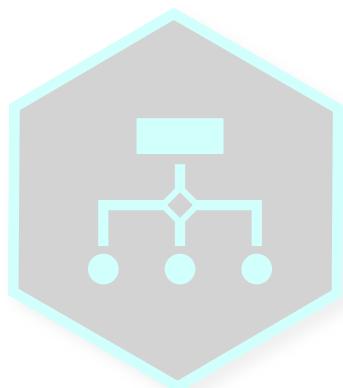
PART 1

Data Prep & EDA



PART 2

Regression



PART 3

Classification



PART 4

Unsupervised
Learning



PART 5

Natural Language
Processing

COURSE STRUCTURE



This is a **project-based** course for students looking for a practical, hands-on approach to learning data science and applying unsupervised learning models with Python

Additional resources include:

-  **Downloadable PDF** to serve as a helpful reference when you're offline or on the go
-  **Quizzes & Assignments** to test and reinforce key concepts, with step-by-step solutions
-  **Interactive demos** to keep you engaged and apply your skills throughout the course

COURSE OUTLINE

1

Intro to Data Science

Introduce the fields of data science and machine learning, review essential skills, and introduce each phase of the data science workflow

2

Unsupervised Learning 101

Review the basics of unsupervised learning, including key concepts, types of techniques and applications, and its place in the data science workflow

3

Pre-Modeling Data Prep

Recap the data prep & EDA steps required to apply unsupervised learning models, including restructuring data, engineering features, and more

4

Clustering

Apply several clustering techniques in Python and learn to interpret the results using metrics, visualizations, and domain expertise

5

PROJECT: Clustering

Apply three clustering techniques on a data set, interpret the outputs, and compare the results to make a final recommendation

6

Anomaly Detection

Understand where anomaly detection fits in the data science workflow, and apply several anomaly detection techniques in Python

COURSE OUTLINE

7

Dimensionality Reduction

Understand where dimensionality reduction fits in the data science workflow, apply several techniques in Python, and interpret the results

8

Recommenders

Recognize the variety of approaches for creating recommenders and the required data structures, then apply multiple techniques in Python

9

PROJECT: Recommenders

Use matrix factorization to create a recommender, interpret the results, and make recommendations

10

Unsupervised Learning Review

Review and compare unsupervised learning algorithms, including their strengths and weaknesses, as well as common applications and use cases

11

FINAL PROJECT

Apply multiple unsupervised learning techniques including clustering and dimensionality reduction on a single data set to make recommendations

INTRODUCING THE COURSE PROJECT



THE **SITUATION**

You've just been hired as an Associate Data Scientist for the **HR Analytics** team at a medium-sized software company that's trying to increase employee retention



THE **ASSIGNMENT**

You have access to the company's employee database, including demographic info, performance history, tenure at the company, attrition, and more

Your task is to use **unsupervised learning techniques** to define employee segments and make recommendations to increase retention within each one



THE **OBJECTIVES**

1. **Prepare** the data for unsupervised modeling
2. **Segment** the employees using clustering
3. **Visualize** the clusters using dimensionality reduction
4. **Explore** the employees within each cluster
5. **Recommend** next steps to increase retention



SETTING EXPECTATIONS



This course covers **clustering & dimensionality reduction** techniques

- We will review both the theory and application of unsupervised learning models, including K-Means Clustering, Hierarchical Clustering, DBSCAN, Principal Component Analysis (PCA), and t-SNE



We'll also dive into popular applications: **anomaly detection & recommenders**

- Anomaly detection and recommender techniques include statistical and supervised learning approaches as well, but we will be focusing on popular unsupervised learning approaches in this course



We'll use **Jupyter Notebook** as our primary coding environment

- Jupyter Notebook is free to use, and the industry standard for conducting data analysis with Python



You do **NOT** need to be a Python expert to take this course

- It is strongly recommended that you complete the first course in this series, Data Prep & EDA, but we will teach the relevant math and Python code for applying unsupervised learning models in this course

INSTALLATION & SETUP

INSTALLATION & SETUP



In this section we'll install Anaconda and introduce **Jupyter Notebook**, a user-friendly coding environment where we'll be coding in Python

TOPICS WE'LL COVER:

[Installing Anaconda](#)

[Launching Jupyter](#)

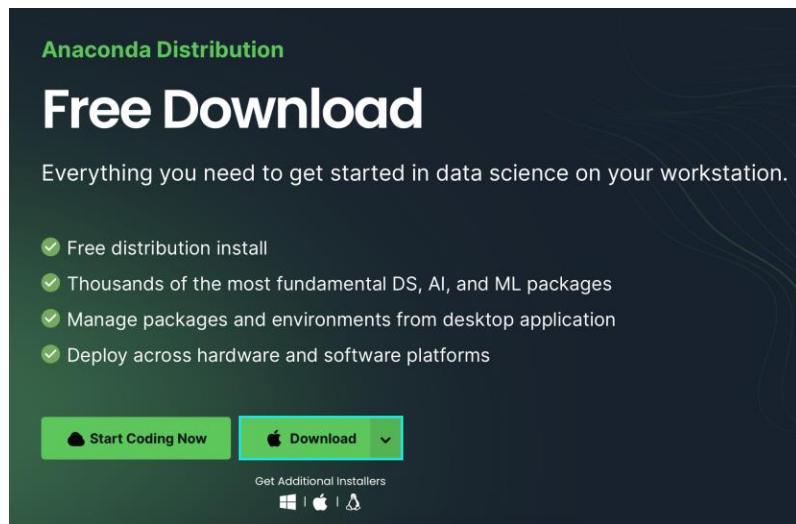
GOALS FOR THIS SECTION:

- Install Anaconda and launch Jupyter Notebook
- Get comfortable with the Jupyter Notebook environment and interface

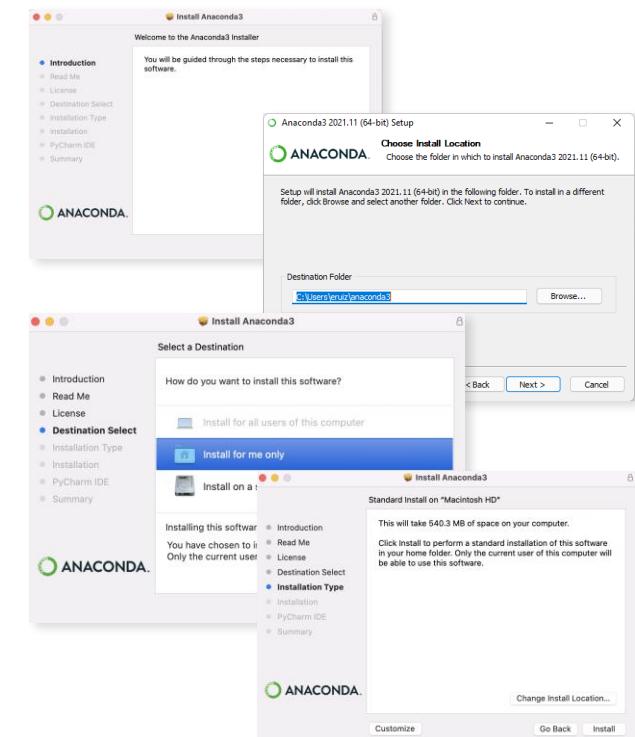


INSTALL ANACONDA (MAC)

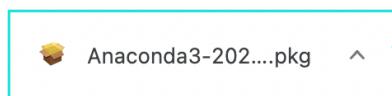
1) Go to anaconda.com/products/distribution and click



3) Follow the **installation steps**
(default settings are OK)



2) Launch the downloaded Anaconda **pkg** file





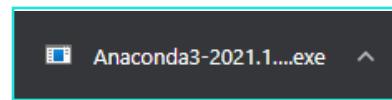
INSTALL ANACONDA (PC)

1) Go to anaconda.com/products/distribution and click

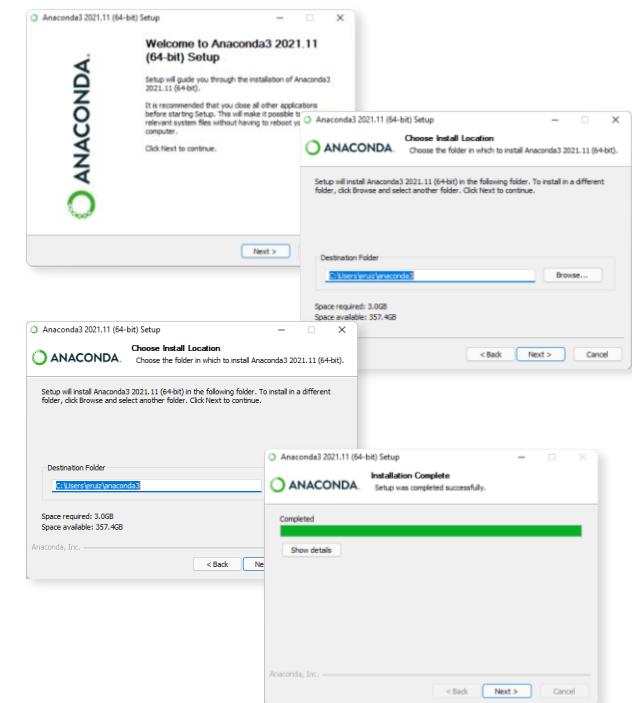


The screenshot shows the Anaconda Distribution website. On the left, there's a sidebar with 'Installing Anaconda' and 'Launching Jupyter'. The main content area has a dark background with green text. It says 'Anaconda Distribution' at the top, followed by 'Free Download' in large white letters. Below that, it says 'Everything you need to get started in data science on your workstation.' A bulleted list follows: '✓ Free distribution install', '✓ Thousands of the most fundamental DS, AI, and ML packages', '✓ Manage packages and environments from desktop application', and '✓ Deploy across hardware and software platforms'. At the bottom, there are two green buttons: 'Start Coding Now' and 'Download'. Below the buttons, it says 'Get Additional Installers' and shows icons for Windows, Mac, and Linux.

2) Launch the downloaded Anaconda **exe** file



3) Follow the **installation steps**
(default settings are OK)





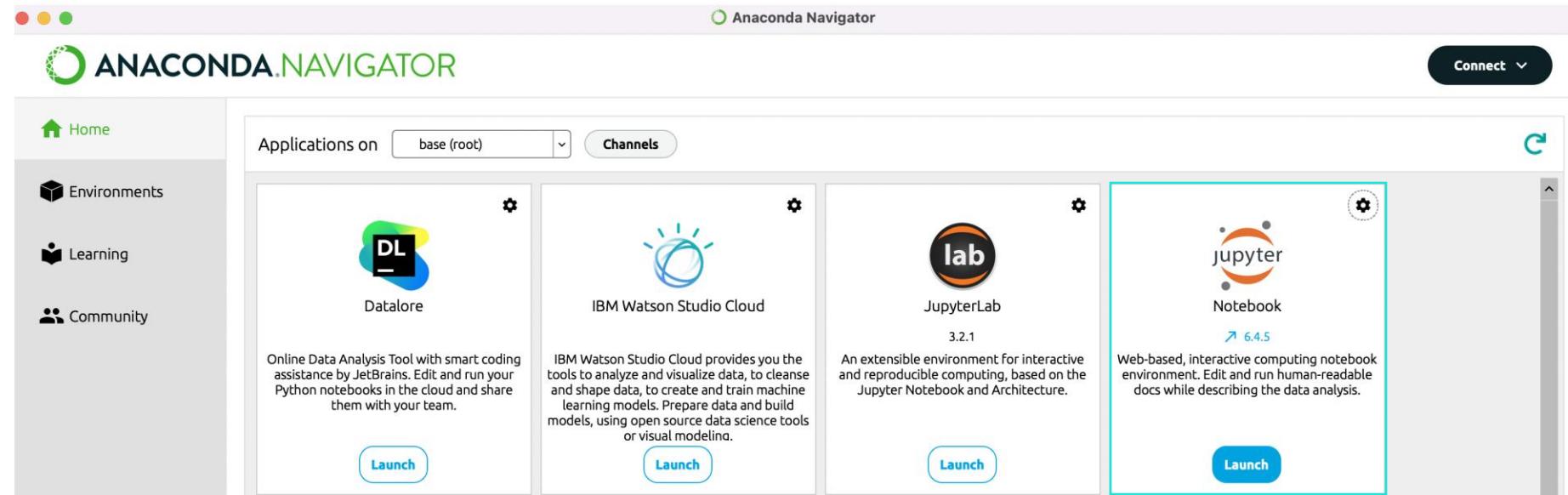
LAUNCHING JUPYTER

Installing
Anaconda

Launching
Jupyter

1) Launch **Anaconda Navigator**

2) Find **Jupyter Notebook** and click **Launch**



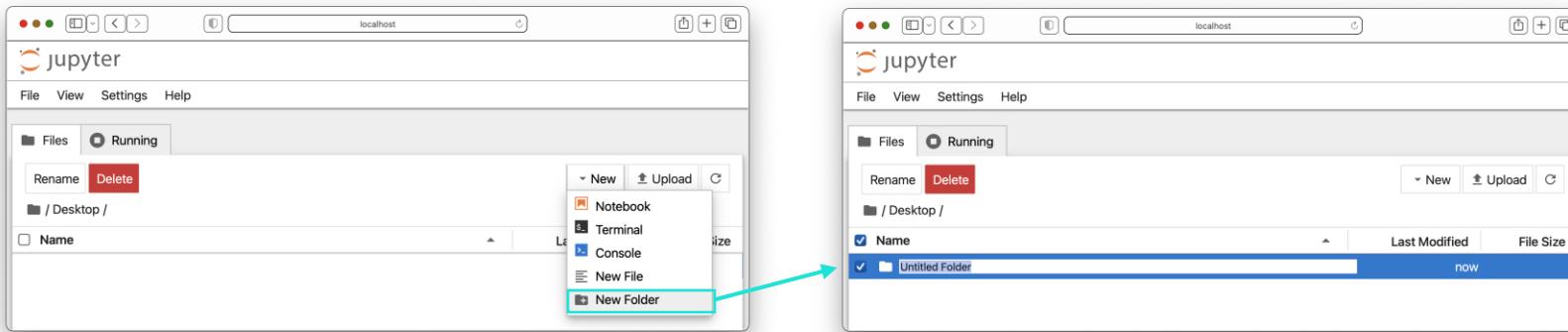


YOUR FIRST JUPYTER NOTEBOOK

Installing
Anaconda

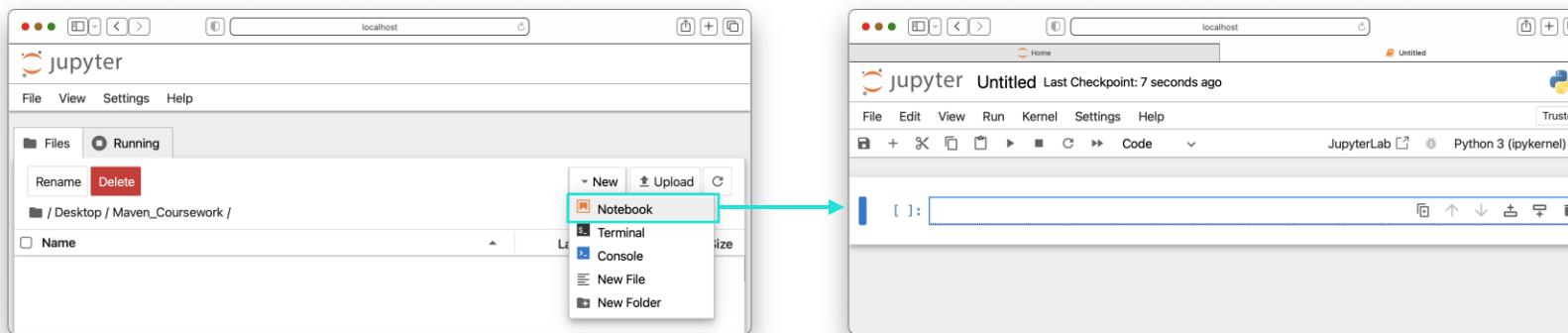
Launching
Jupyter

- Once inside the Jupyter interface, **create a folder** to store your notebooks for the course



NOTE: You can rename your folder by clicking "Rename" in the top left corner

- Open your new coursework folder and **launch your first Jupyter notebook!**



NOTE: You can rename your notebook by clicking on the title at the top of the screen



THE NOTEBOOK SERVER

Installing
Anaconda

Launching
Jupyter

NOTE: When you launch a Jupyter notebook, a terminal window may pop up as well; this is called a **notebook server**, and it powers the notebook interface

```
Last login: Tue Jan 25 14:04:12 on ttys002
(base) chrissb@Chriss-MBP ~ % jupyter notebook
[I 2022-01-26 08:45:53.886 LabApp] JupyterLab extension loaded from /Users/chrissb/opt/anaconda3/lib/python3.9/site-packages/jupyterlab
[I 2022-01-26 08:45:53.886 LabApp] JupyterLab application directory is /Users/chrissb/opt/anaconda3/share/jupyter/lab
[I 08:45:53.890 NotebookApp] Serving notebooks from local directory: /Users/chrissb
[I 08:45:53.890 NotebookApp] Jupyter Notebook 6.4.5 is running at:
[I 08:45:53.890 NotebookApp] http://localhost:8888/?token=3159cf032d9e6841d04910e257db2b24b6df6dfc878d6d5f
[I 08:45:53.890 NotebookApp] or http://127.0.0.1:8888/?token=3159cf032d9e6841d04910e257db2b24b6df6dfc878d6d5f
[I 08:45:53.890 NotebookApp] Use Control-C to stop this server and shut down all kernels (twice to skip confirmation).
[C 08:45:53.893 NotebookApp]

To access the notebook, open this file in a browser:
file:///Users/chrissb/Library/Jupyter/runtime/nbserver-27175-open.html
Or copy and paste one of these URLs:
http://localhost:8888/?token=3159cf032d9e6841d04910e257db2b24b6df6dfc878d6d5f
or http://127.0.0.1:8888/?token=3159cf032d9e6841d04910e257db2b24b6df6dfc878d6d5f
[W 08:46:05.829 NotebookApp] Notebook Documents/Maven_Coursework/Python_Intro.ipynb
```

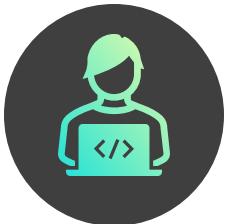


If you close the server window,
your notebooks will not run!

Depending on your OS, and method of launching Jupyter, one may not open – as long as you can run your notebooks, don't worry!

INTRO TO DATA SCIENCE

INTRO TO DATA SCIENCE



In this section we'll **introduce the field of data science**, discuss how it compares to other data fields, and walk through each phase of the data science workflow

TOPICS WE'LL COVER:

[What is Data Science?](#)

[Essential Skills](#)

[Machine Learning](#)

[Data Science Workflow](#)

GOALS FOR THIS SECTION:

- Compare data science and machine learning with other common data analytics fields
- Introduce supervised and unsupervised learning, and examples of each technique
- Review the machine learning landscape and commonly used algorithms
- Discuss essential skills, and review each phase of the data science workflow



WHAT IS DATA SCIENCE?

What is Data Science?

Essential Skills

Machine Learning

Data Science Workflow

Data science is about using *data* to make smart decisions



Wait, isn't that **business intelligence** ?

Yes! The differences lie in the **types of problems** you solve, and **tools and techniques** you use to solve them:

What happened?

- Descriptive Analytics
- Data Analysis
- Business Intelligence

What's going to happen?

- Predictive Analytics
- Data Mining
- **Data Science**



DATA SCIENCE SKILL SET

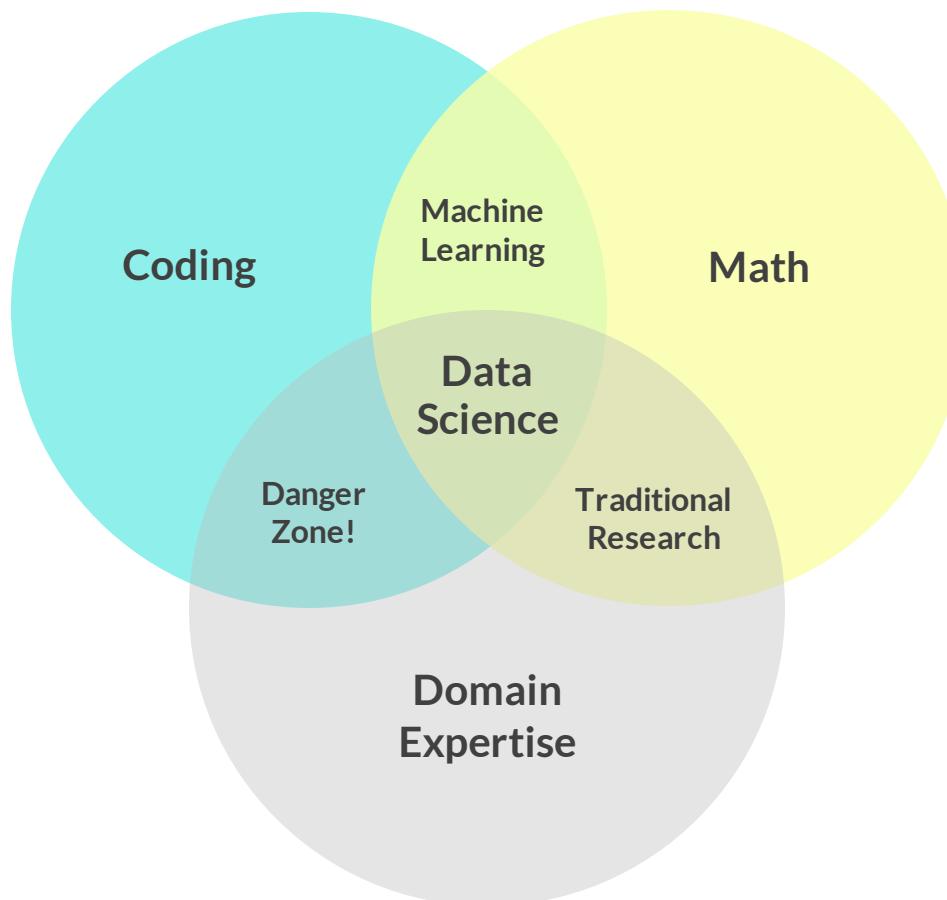
Data science requires a blend of **coding**, **math**, and **domain expertise**

What is Data Science?

Essential Skills

Machine Learning

Data Science Workflow



The key is in applying these along with soft skills like:

- Communication
- Problem solving
- Curiosity & creativity
- Grit
- Googling prowess



Data scientists & analysts approach problem solving in similar ways, but data scientists will often work with larger, more complex data sets and utilize advanced algorithms



WHAT IS MACHINE LEARNING?

What is Data Science?

Essential Skills

Machine Learning

Data Science Workflow

Data scientists use **machine learning** algorithms to enable computers to learn and make decisions from data

Machine learning algorithms fall into two broad categories:

Supervised Learning

Using historical data to predict the future



What will house prices look like for the next 12 months?



How can I flag suspicious emails as spam?

Unsupervised Learning

Finding patterns and relationships in data



How can I segment my customers?



Which TV shows should I recommend to each user?



COMMON ALGORITHMS

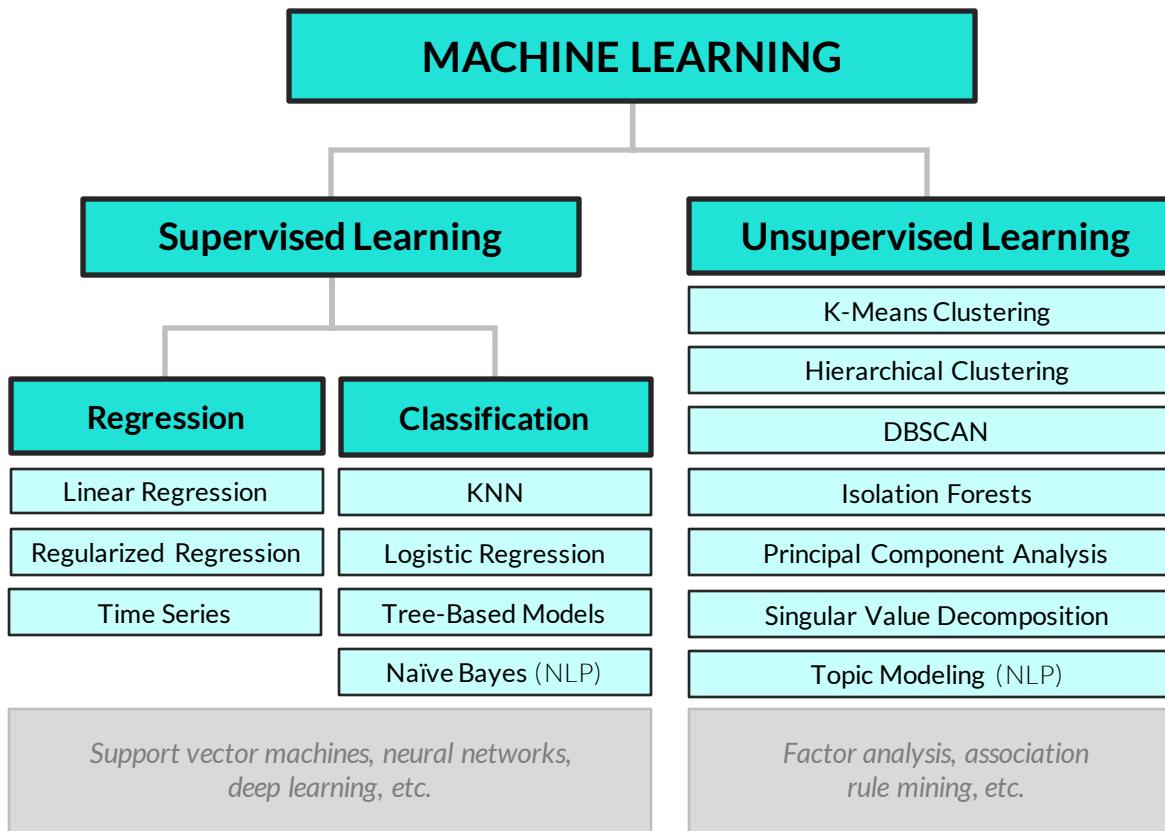
What is Data Science?

Essential Skills

Machine Learning

Data Science Workflow

These are some of the most **common machine learning algorithms** that data scientists use in practice



Another category of machine learning algorithms is called **reinforcement learning**, which is commonly used in robotics and gaming

Fields like **deep learning** and **natural language processing** utilize both supervised and unsupervised learning techniques



DATA SCIENCE WORKFLOW

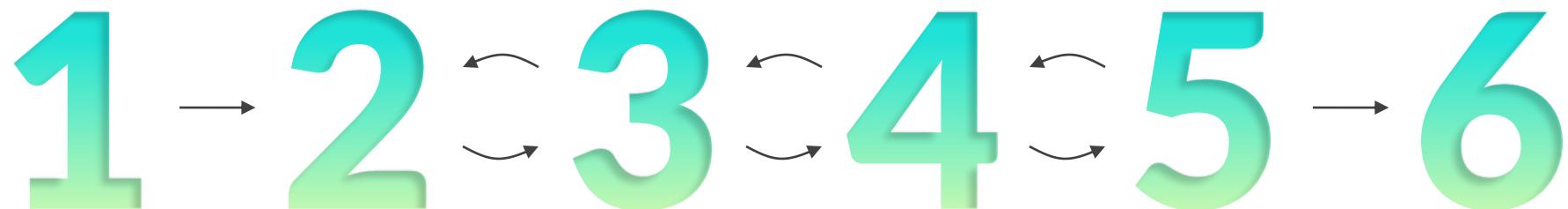
What is Data Science?

Essential Skills

Machine Learning

Data Science Workflow

The **data science workflow** consists of scoping the project, gathering, cleaning and exploring the data, applying models, and sharing insights with end users



This is not a linear process! You'll likely go back to further gather, clean and explore your data



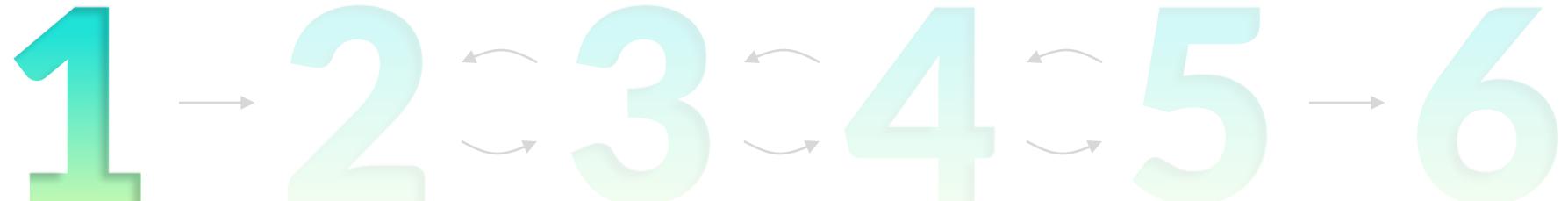
STEP 1: SCOPING A PROJECT

What is Data Science?

Essential Skills

Machine Learning

Data Science Workflow



Projects don't start with *data*, they start with a **clearly defined scope**:

- Who are your end users or stakeholders?
- What business problems are you trying to help them solve?
- Is this a supervised or unsupervised learning problem? (*do you even need data science?*)
- What data do you need for your analysis?



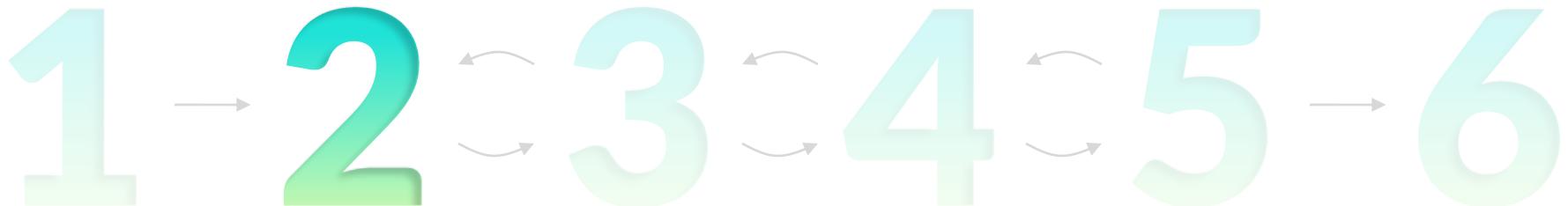
STEP 2: GATHERING DATA

What is Data Science?

Essential Skills

Machine Learning

Data Science Workflow



A project is only as strong as the underlying data, so **gathering the right data** is essential to set a proper foundation for your analysis

Data can come from a variety of sources, including:

- Files (flat files, spreadsheets, etc.)
- Databases
- Websites
- APIs



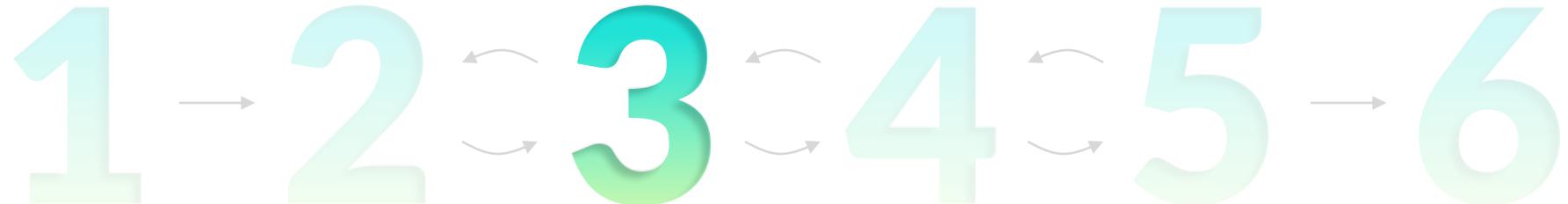
STEP 3: CLEANING DATA

What is Data Science?

Essential Skills

Machine Learning

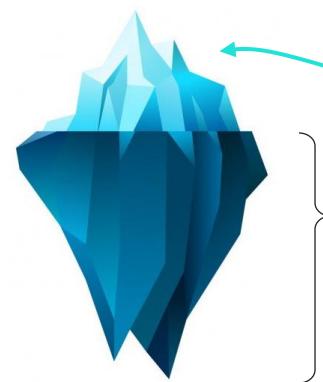
Data Science Workflow



A popular saying within data science is “garbage in, garbage out”, which means that **cleaning data** properly is key to producing accurate and reliable results

Data cleaning tasks may include:

- Resolving formatting issues
- Correcting data types
- Imputing missing data
- Restructuring the data



Building models
The flashy part of data science

Cleaning data
Less fun, but very important
(Data scientists estimate that around 50-80% of their time is spent here!)



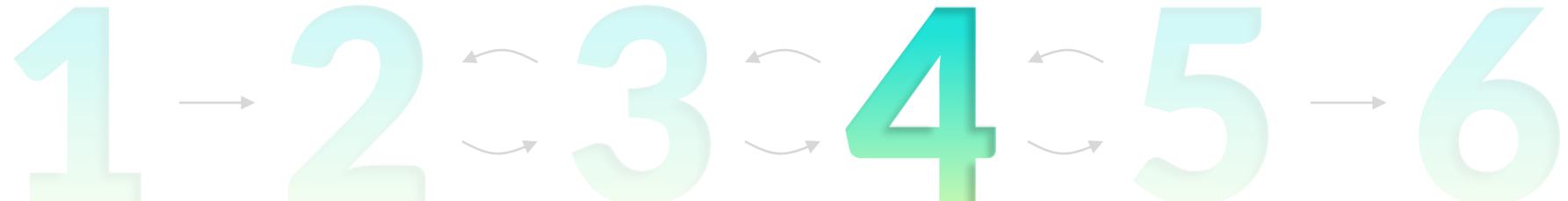
STEP 4: EXPLORING DATA

What is Data Science?

Essential Skills

Machine Learning

Data Science Workflow



Scoping a Project

Gathering Data

Cleaning Data

Exploring Data

Modeling Data

Sharing Insights

Exploratory data analysis (EDA) is all about exploring and understanding the data you're working with before applying models or algorithms

EDA tasks may include:

- Slicing & dicing the data
- Profiling the data
- Visualizing the data



A good number of the **final insights** that you share will come from the exploratory analysis phase!



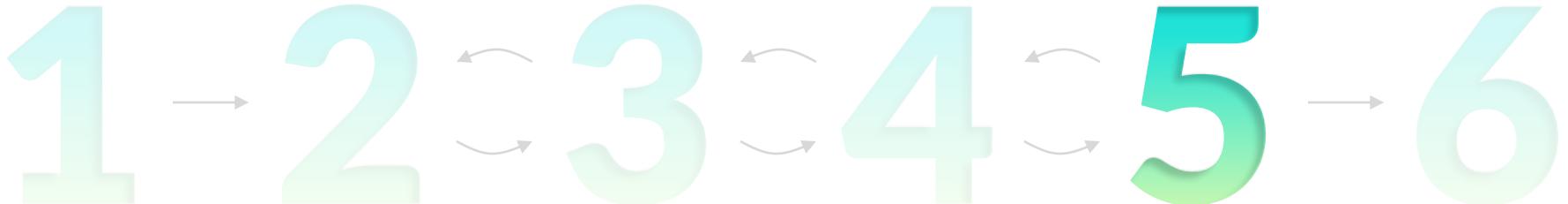
STEP 5: MODELING DATA

What is Data Science?

Essential Skills

Machine Learning

Data Science Workflow



Scoping a Project

Gathering Data

Cleaning Data

Exploring Data

Modeling Data

Sharing Insights

Modeling data involves structuring and preparing data for specific modeling techniques, and applying those models to make predictions or discover patterns

Modeling tasks include:

- Feature selection & engineering
- Fitting models
- Interpreting results



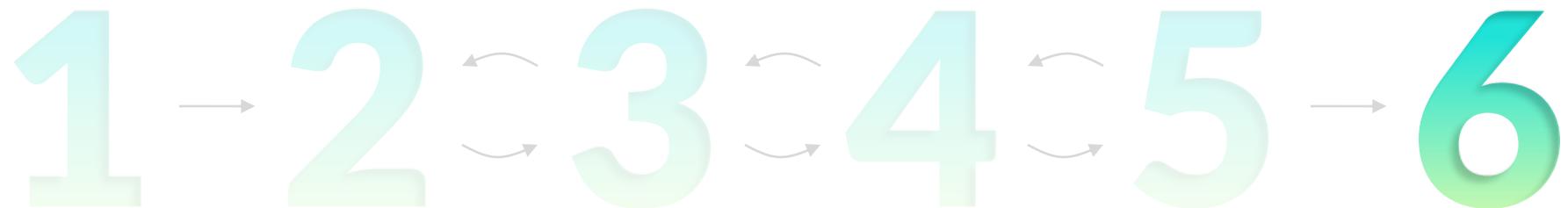
With fancy new algorithms introduced every year, you may feel the need to learn and apply the latest and greatest techniques

In practice, **simple is best**; businesses & leadership teams appreciate solutions that are easy to understand, interpret and implement



STEP 6: SHARING INSIGHTS

- What is Data Science?
- Essential Skills
- Machine Learning
- Data Science Workflow



Scoping a Project Gathering Data Cleaning Data Exploring Data Modeling Data Sharing Insights

The final step of the workflow involves summarizing your key findings and **sharing insights** with end users or stakeholders:

- Reiterate the problem
- Summarize the results of your analysis
- Share recommendations and next steps
- Focus on potential impact, not technical details



Even with all the technical work that's been done, it's important to remember that the focus here is on **non-technical solutions**

NOTE: Another way to share results is to deploy your model, or put it into production



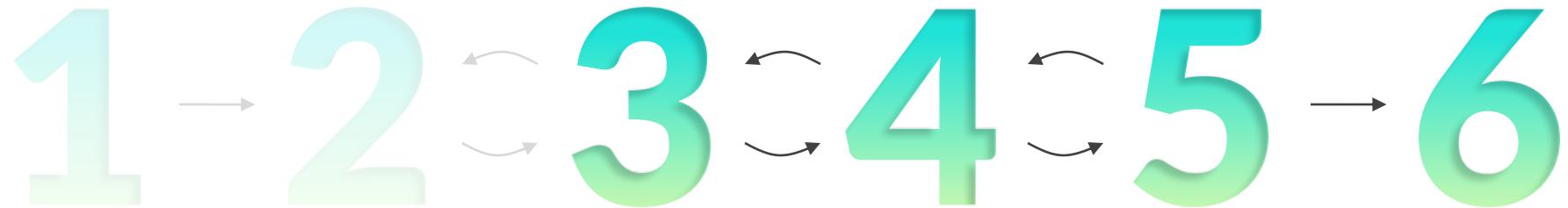
UNSUPERVISED LEARNING

What is Data Science?

Essential Skills

Machine Learning

Data Science Workflow



Scoping a Project

Gathering Data

Cleaning Data

Exploring Data

Modeling Data

Sharing Insights

DATA PREP & EDA

UNSUPERVISED LEARNING

Unsupervised learning is used to discover patterns & relationships within data

These techniques can be utilized during the **cleaning, exploratory or modeling** phase of the data science workflow, and all findings can be shared as insights

KEY TAKEAWAYS



Data science is about using data to make smart decisions

- *Supervised learning techniques use historical data to predict the future, and unsupervised learning techniques use algorithms to find patterns and relationships*



Data scientists have both **coding** and **math** skills along with **domain expertise**

- *In addition to technical expertise, soft skills like communication, problem-solving, curiosity, creativity, grit, and Googling prowess round out a data scientist's skillset*



The **data science workflow** starts with defining a clear scope

- *Once the project scope is defined, you can move on to gathering and cleaning data, performing exploratory data analysis, preparing data for modeling, applying algorithms, and sharing insights with end users*



Unsupervised learning techniques are used for finding patterns in data

- *Data scientists are often tasked with finding patterns and relationships in data, which can happen during the cleaning, exploratory or modeling phase of the data science workflow*

UNSUPERVISED LEARNING 101

UNSUPERVISED LEARNING 101



In this section we'll cover the basics of **unsupervised learning**, including key concepts, techniques & applications, and where it can be used within the data science workflow

TOPICS WE'LL COVER:

Unsupervised Learning

Techniques & Applications

Data Science Workflow

GOALS FOR THIS SECTION:

- Introduce the basics of unsupervised learning
- Review key terminology and concepts
- Understand the different techniques and applications of unsupervised learning
- Revisit the data science workflow and identify where unsupervised learning fits within it



UNSUPERVISED LEARNING 101

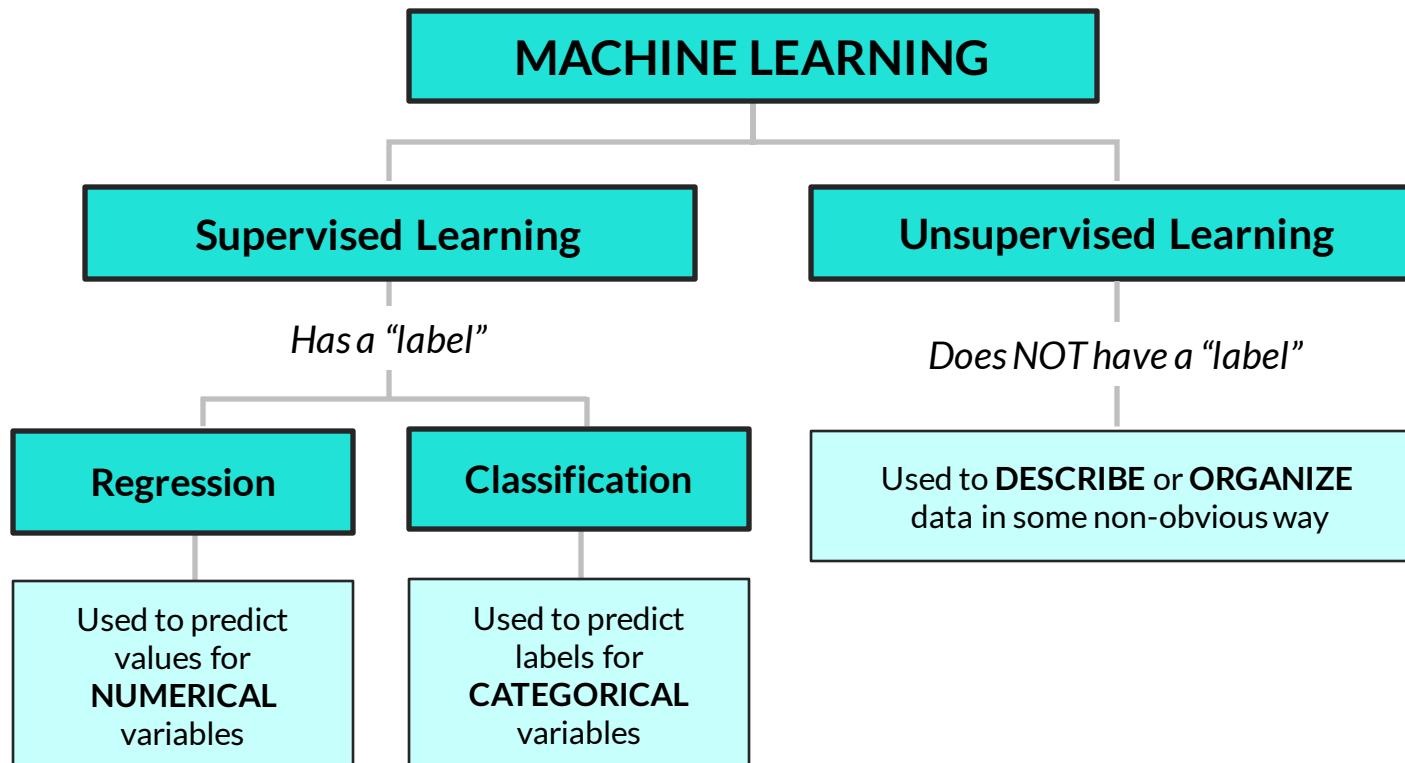
Unsupervised Learning

Techniques & Applications

Data Science Workflow

Unsupervised learning is about **finding insights & patterns** hidden in the data

- Unlike regression or classification, we don't care about splitting our data into train / test sets and making predictions, we just care about *understanding* the relationships in our data





UNSUPERVISED LEARNING 101

Unsupervised Learning

Techniques & Applications

Data Science Workflow

EXAMPLE

Clustering customers based on listening behavior

Each row
represents a
customer

These are **features** (what goes into the model)

Customer	Music Streaming Hours	Podcast Listening Hours
Aria	46	9
Chord	38	10
Harmony	44	17
Melody	19	50
Reed	7	44
Viola	16	52
Rock	5	19
Piper	10	11
Allegra	17	9

← Note that there is **NO target**



How can we segment
these customers?



UNSUPERVISED LEARNING 101

EXAMPLE

Clustering customers based on listening behavior

Unsupervised Learning

Techniques & Applications

Data Science Workflow

Each row represents a customer →

These are **features** (what goes into the model)

Customer	Music Streaming Hours	Podcast Listening Hours
Aria	46	9
Chord	38	10
Harmony	44	17
Melody	19	50
Reed	7	44
Viola	16	52
Rock	5	19
Piper	10	11
Allegra	17	9

← Note that there is **NO target**

Cluster 1
Music lovers

Cluster 2
Podcast enthusiasts

Cluster 3
Casual listeners



UNSUPERVISED LEARNING 101

EXAMPLE

Clustering customers based on listening behavior

Unsupervised Learning

Techniques & Applications

Data Science Workflow

These are
features



We can clearly segment our customers into **three clusters**:

- Music lovers
- Podcast enthusiasts
- Casual listeners



UNSUPERVISED LEARNING TECHNIQUES

Unsupervised Learning

Techniques & Applications

Data Science Workflow

There are two popular categories of unsupervised learning **techniques**:



Clustering

Identifying groups (or *clusters*) of data points that are similar to one another but distinct from other groups

Common techniques:

- K-Means Clustering
- Hierarchical Clustering
- DBSCAN (Density-Based Clustering)

Applications:

- Clustering / Segmentation
- Anomaly Detection
- Recommenders



Dimensionality Reduction

Reducing the number of columns (or *dimensions*) in a data set while losing as little information as possible

Common techniques:

- PCA (Principal Component Analysis)
- t-SNE (t-Stochastic Neighbor Embedding)
- SVD (Singular Value Decomposition)

Applications:

- Feature Extraction
- Data Visualization
- Recommenders



UNSUPERVISED LEARNING APPLICATIONS

Unsupervised Learning

Techniques & Applications

Data Science Workflow

These are two common **applications** of unsupervised learning techniques:



Anomaly Detection

Identifying rare points in a data set that deviate significantly from the rest

Unsupervised learning techniques:

- Clustering Techniques
- Isolation Forests

Other techniques:

- Statistical Analysis
- Time Series Analysis



Recommenders

Suggesting items to users based on their preferences or behaviors

Unsupervised learning techniques:

- Clustering Techniques
- Dimensionality Reduction Techniques

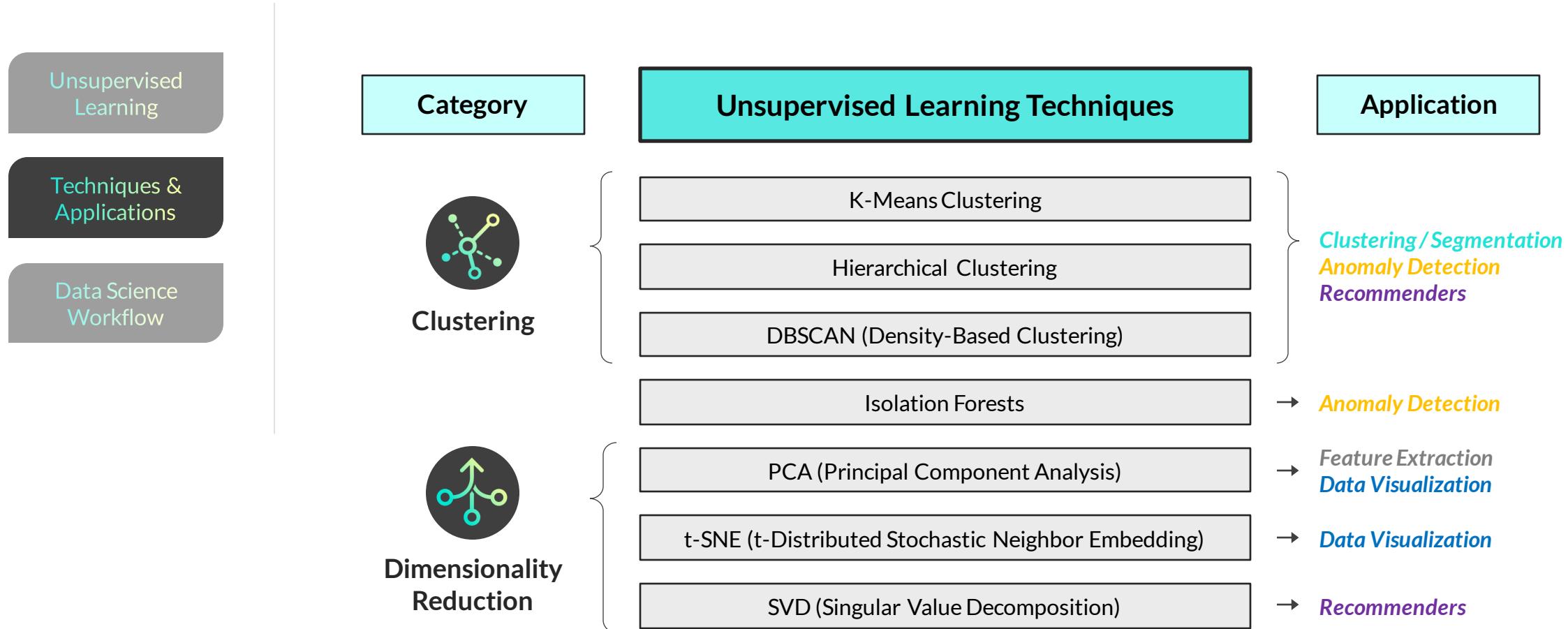
Other techniques:

- Distance Measures
- Supervised Learning



STRUCTURE OF THIS COURSE

We'll cover unsupervised learning **techniques & applications** in this order:



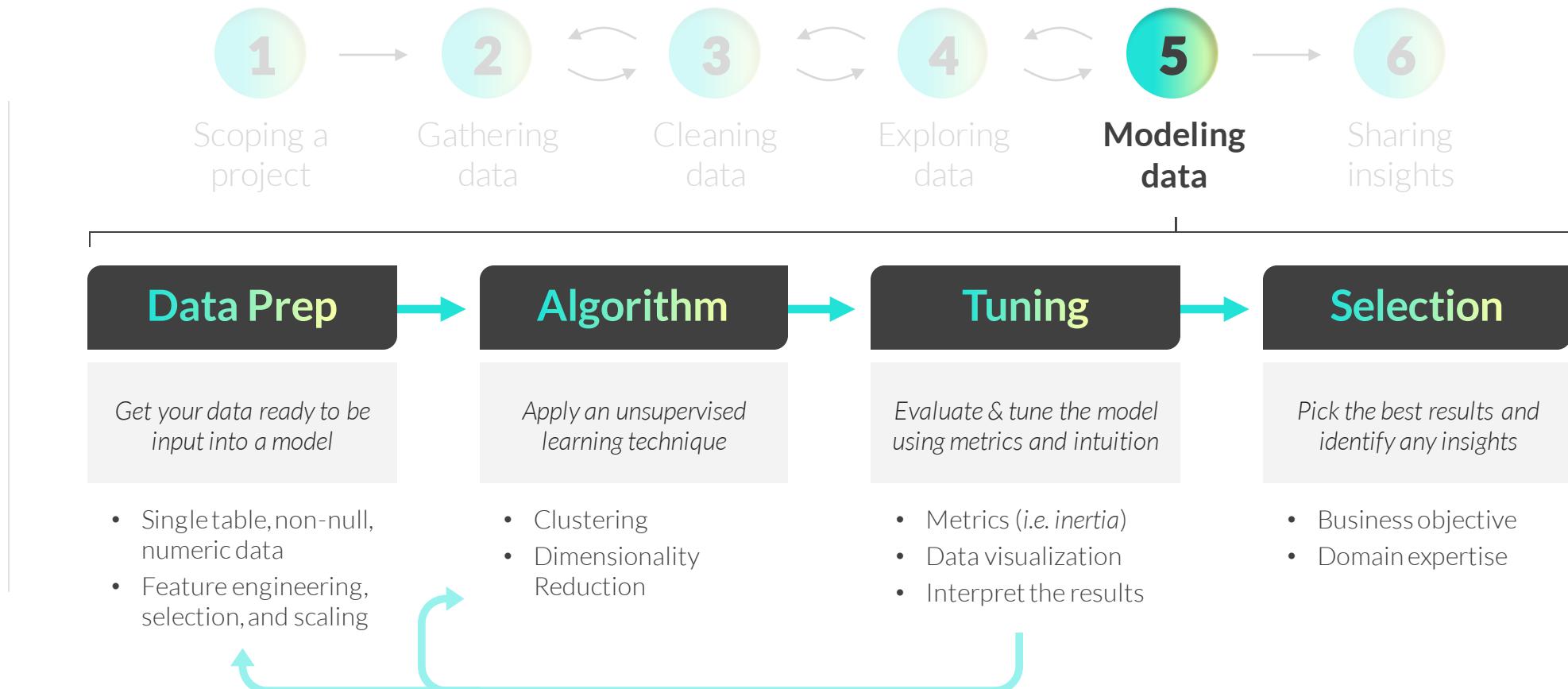


UNSUPERVISED LEARNING WORKFLOW

Unsupervised Learning

Techniques & Applications

Data Science Workflow



The main differences compared to supervised learning are the **lack of data splitting** (having training and test sets) and the focus on **evaluation based on intuition** (versus mainly metrics)



UNSUPERVISED LEARNING WORKFLOW

Unsupervised Learning

Techniques & Applications

Data Science Workflow



Anomaly Detection

*Can be used as a machine learning alternative to **Outlier Detection***

- Outlier detection is typically done using statistics or plots
- By using anomaly detection, you can catch unusual data points and patterns before applying other ML algorithms

PCA / t-SNE

*Can be used as a machine learning alternative to **Data Visualization***

- Data is typically visualized using two dimensions (x and y-axes)
- By using PCA or t-SNE, high-dimensional data can also be visualized in two dimensions

PCA

*Can be used as an ML alternative to **Feature Selection / Engineering***

- Feature selection or engineering is typically done manually by removing features or applying transformations to create them
- By using PCA, multiple fields can be mathematically reduced to fewer fields, which is called feature extraction

KEY TAKEAWAYS



Unsupervised learning is used to **find patterns & relationships** in data

- *There are no predictions or labels with unsupervised learning – we are just trying to better understand the data's non-obvious structure, organization, and relationships between data points*



Unsupervised learning has a **different mindset** than supervised learning

- *Unlike supervised learning, unsupervised learning does not require splitting the data into a training and test set, and the evaluation is based on a heavy-dose of domain expertise in addition to metrics*



There are **multiple applications** for unsupervised learning **techniques**

- *While the two main categories of unsupervised learning techniques fall under clustering and dimensionality reduction, these techniques can be applied to segmentation, anomaly detection, recommenders, and more*



The techniques can be used at **multiple steps** of the data science workflow

- *In addition to using unsupervised learning techniques during the modeling step of the data science workflow, select techniques can also be used during the data cleaning, exploration, and feature engineering phases*

PRE-MODELING DATA PREP

PRE-MODELING DATA PREP



In this section we'll review the **data prep** steps required before applying unsupervised learning algorithms, including making sure the rows and columns are set up properly

TOPICS WE'LL COVER:

Data Prep Steps

Row Granularity

Column Preparation

Feature Engineering

Feature Selection

Feature Scaling

GOALS FOR THIS SECTION:

- Learn Python techniques to adjust row granularity and make sure all values are non-null and numeric
- Apply feature engineering techniques for unsupervised learning models
- Understand why identifier columns should be excluded from modeling, but used as reference
- Identify situations where feature scaling is needed



PRE-MODELING DATA PREP

Data Prep Steps

Row Granularity

Column Preparation

Feature Engineering

Feature Selection

Feature Scaling

These are common **data prep** steps required to transform your source data into a format that can be directly input into an unsupervised learning model:

- 1 Setting the correct **row granularity**
- 2 Ensuring each column is **non-null** and **numeric**
- 3 **Engineering features** for modeling
- 4 **Selecting features** and excluding **identifier columns**
- 5 **Scaling features** for distance-based unsupervised learning algorithms



SETTING THE CORRECT ROW GRANULARITY

Data Prep Steps

Row Granularity

Column Preparation

Feature Engineering

Feature Selection

Feature Scaling

To **set the correct row granularity**, think about the question you're trying to answer and determine what one row (observation) of your table will look like

GOAL

Cluster **customers** based on listening behavior



Customer	Genre	# Songs
Aria	Pop	50
Aria	Indie	48
Aria	Rock	1
Chord	Pop	15
Chord	Indie	36
Harmony	Pop	10
Harmony	Indie	5
Harmony	Rock	3
Melody	Rock	2
Reed	Rock	5



Customer	# Pop Songs	# Indie Songs	# Rock Songs
Aria	50	48	1
Chord	15	36	0
Harmony	10	5	3
Melody	0	0	2
Reed	0	0	5

→ This table has the correct data, but it needs to be reshaped so that each row represents **one customer** (that's what we're clustering!)



RESHAPING DATA

Data Prep Steps

Row Granularity

Column Preparation

Feature Engineering

Feature Selection

Feature Scaling

Common ways to **reshape data** to the correct row granularity are:

- Using **.groupby()** to combine all records with the same ID column into a single row

df

	Customer	Genre	# Songs
0	Aria	Pop	50
1	Aria	Indie	48
2	Aria	Rock	1
3	Chord	Pop	15
4	Chord	Indie	36
5	Harmony	Pop	10
6	Harmony	Indie	5
7	Harmony	Rock	3
8	Melody	Rock	2
9	Reed	Rock	5

```
# use group by to sum up the songs for each customer  
(df.groupby('Customer')['# Songs']  
    .sum()  
    .reset_index())
```

Customer # Songs

0	Aria	99
1	Chord	51
2	Harmony	18
3	Melody	2
4	Reed	5

Now each row represents a customer!



RESHAPING DATA

Data Prep Steps

Row Granularity

Column Preparation

Feature Engineering

Feature Selection

Feature Scaling

Common ways to **reshape data** to the correct row granularity are:

- Using **.groupby()** to combine all records with the same ID column into a single row
- Using **.pivot()** to transform data from a “long” format to a “wide” format

df

	Customer	Genre	# Songs
0	Aria	Pop	50
1	Aria	Indie	48
2	Aria	Rock	1
3	Chord	Pop	15
4	Chord	Indie	36
5	Harmony	Pop	10
6	Harmony	Indie	5
7	Harmony	Rock	3
8	Melody	Rock	2
9	Reed	Rock	5

```
# use pivot to turn the genres into columns  
(df.pivot(index='Customer',  
         columns='Genre',  
         values='# Songs')  
 .fillna(0)  
 .reset_index())
```

	Genre	Customer	Indie	Pop	Rock
0	Aria	48.0	50.0	1.0	
1	Chord	36.0	15.0	0.0	
2	Harmony	5.0	10.0	3.0	
3	Melody	0.0	0.0	2.0	
4	Reed	0.0	0.0	5.0	

Now each row represents a customer, and we still have the genre information!



The opposite of **.pivot()** is **.melt()**, which is often used to transform data from a “wide” format to a “long” format

ASSIGNMENT: SET THE CORRECT ROW GRANULARITY

 **NEW MESSAGE**
March 4, 2024

From: Cindy Cinema (Lead Data Scientist)
Subject: Please format data for analysis

Hi,

I hear you're the new associate data scientist on the team – welcome!

We're currently working on a project to segment students based on their entertainment preferences.

Could you format the data in this spreadsheet so that it's at the correct row granularity for student-level analysis?

Thanks!
Cindy

 entertainment.xlsx

Reply **Forward**

Key Objectives

1. Read the Excel file into a Pandas DataFrame
2. Check the number of rows and columns
3. Determine the row granularity needed
4. Apply the correct DataFrame transformation
5. Save the transformation as a new DataFrame
6. Check the number of rows and columns

Hint: The new DataFrame should have 150 rows



PREPARING COLUMNS FOR MODELING

Data Prep Steps

Row Granularity

Column Preparation

Feature Engineering

Feature Selection

Feature Scaling

1

All values should be **non-null**

- Identify missing (or null) values using `df.info()` or `df.isna()`
- Resolve them by either removing them or imputing the values

2

All values should be **numeric**

- Convert fields from text data types to numeric data types
- Turn fields into numeric fields using conditional logic with `np.where()`
- Turn categorical fields into numeric fields using dummy variables



PRO TIP: There are some algorithms that can handle null and non-numeric values, including tree-based models and some classification models, but it's still best practice to prepare the data this way



IDENTIFYING MISSING DATA

Data Prep Steps

Row Granularity

Column Preparation

Feature Engineering

Feature Selection

Feature Scaling

```
# null values in each column  
customers.info()
```

```
<class 'pandas.core.frame.DataFrame'>  
RangeIndex: 8 entries, 0 to 7  
Data columns (total 7 columns):  
 #   Column      Non-Null Count  Dtype     
---  --          --          --  
 0   Name        8 non-null     object    
 1   Age         6 non-null    float64  
 2   Followers   5 non-null    float64  
 3   Income      8 non-null    object    
 4   Sign Up Date 8 non-null   object    
 5   Discount    8 non-null    object    
 6   Education Level 8 non-null object    
dtypes: float64(2), object(5)  
memory usage: 580.0+ bytes
```

Compare the total entries with the non-null count for each column

You can use `.any(axis=1)` to return the rows with null values

```
# cells with null values  
customers.isna()
```

	Name	Age	Followers	Income	Sign Up Date	Discount	Education Level
0	False	False		False	False	False	False
1	False	False		False	False	False	False
2	False	False	True	False	False	False	False
3	False	False	True	False	False	False	False
4	False	False		False	False	False	False
5	False	True		False	False	False	False
6	False	True	True	False	False	False	False
7	False	False		False	False	False	False

This is **True** for any null values

```
# rows with null values  
customers[customers.isna().any(axis=1)]
```

	Name	Age	Followers	Income	Sign Up Date	Discount	Education Level
2	Harmony	26.0	NaN	\$120,000	4/25/23	No	Graduate School
3	Melody	47.0	NaN	\$450,000	5/5/23	No	College
5	Selena	NaN	1.0	\$62,000	8/26/23	No	College
6	Stefani	NaN	NaN	\$81,000	9/24/23	No	College



HANDLING MISSING DATA

Data Prep Steps

Row Granularity

Column Preparation

Feature Engineering

Feature Selection

Feature Scaling

There are multiple ways to **handle missing data** in a DataFrame:

- The `.dropna()` method **removes** rows or columns with missing data

customers

	Name	Age	Followers	Income	Sign Up Date	Discount	Education Level
0	Aria	25.0	0.0	\$45,000	5/18/23	Yes	College
1	Chord	19.0	12.0	\$28,000	8/23/23	Yes	High School
2	Harmony	26.0	NaN	\$120,000	4/25/23	No	Graduate School
3	Melody	47.0	NaN	\$450,000	5/5/23	No	College
4	Reed	52.0	0.0	\$75,000	6/14/23	Yes	High School
5	Selena	NaN	1.0	\$62,000	8/26/23	No	College
6	Stefani	NaN	NaN	\$81,000	9/24/23	No	College
7	Taylor	33.0	52.0	\$60,000	9/8/23	No	High School

```
# drop rows with null values  
customers.dropna()
```

	Name	Age	Followers	Income	Sign Up Date	Discount	Education Level
0	Aria	25.0	0.0	\$45,000	5/18/23	Yes	College
1	Chord	19.0	12.0	\$28,000	8/23/23	Yes	High School
4	Reed	52.0	0.0	\$75,000	6/14/23	Yes	High School
7	Taylor	33.0	52.0	\$60,000	9/8/23	No	High School

Note the index values are skipping the dropped rows,
but you can fix this by chaining on a `.reset_index()`



PRO TIP: The `.dropna()` method defaults to dropping rows, but you can drop columns instead by specifying `.dropna(axis=1)`



HANDLING MISSING DATA

Data Prep Steps

Row Granularity

Column Preparation

Feature Engineering

Feature Selection

Feature Scaling

There are multiple ways to **handle missing data** in a DataFrame:

- The `.dropna()` method **removes** rows or columns with missing data
- The `.fillna()` method **imputes** missing data with an appropriate value

customers

	Name	Age	Followers	Income	Sign Up Date	Discount	Education Level
0	Aria	25.0	0.0	\$45,000	5/18/23	Yes	College
1	Chord	19.0	12.0	\$28,000	8/23/23	Yes	High School
2	Harmony	26.0	NaN	\$120,000	4/25/23	No	Graduate School
3	Melody	47.0	NaN	\$450,000	5/5/23	No	College
4	Reed	52.0	0.0	\$75,000	6/14/23	Yes	High School
5	Selena	NaN	1.0	\$62,000	8/26/23	No	College
6	Stefani	NaN	NaN	\$81,000	9/24/23	No	College
7	Taylor	33.0	52.0	\$60,000	9/8/23	No	High School



What values can we fill in here?

- The **median age** would remove the impact of outliers
- Experience tells us most customers have **0 followers**

#impute age with the median

```
customers['Age'] = customers.Age.fillna(customers.Age.median())
```

#impute followers with 0

```
customers['Followers'] = customers.Followers.fillna(0)
```

customers

	Name	Age	Followers	Income	Sign Up Date	Discount	Education Level
0	Aria	25.0	0.0	\$45,000	5/18/23	Yes	College
1	Chord	19.0	12.0	\$28,000	8/23/23	Yes	High School
2	Harmony	26.0	0.0	\$120,000	4/25/23	No	Graduate School
3	Melody	47.0	0.0	\$450,000	5/5/23	No	College
4	Reed	52.0	0.0	\$75,000	6/14/23	Yes	High School
5	Selena	29.5	1.0	\$62,000	8/26/23	No	College
6	Stefani	29.5	0.0	\$81,000	9/24/23	No	College
7	Taylor	33.0	52.0	\$60,000	9/8/23	No	High School



CONVERTING TO NUMERIC

Data Prep Steps

Row Granularity

Column Preparation

Feature Engineering

Feature Selection

Feature Scaling

Pandas will often read in numeric fields as text (object) data types

You can use **pd.to_numeric()** to convert them to numeric fields

customers

	Name	Age	Followers	Income	Sign Up Date	Discount	Education Level
0	Aria	25.0	0.0	\$45,000	5/18/23	Yes	College
1	Chord	19.0	12.0	\$28,000	8/23/23	Yes	High School
2	Harmony	26.0	0.0	\$120,000	4/25/23	No	Graduate School
3	Melody	47.0	0.0	\$450,000	5/5/23	No	College
4	Reed	52.0	0.0	\$75,000	6/14/23	Yes	High School
5	Selena	29.5	1.0	\$62,000	8/26/23	No	College
6	Stefani	29.5	0.0	\$81,000	9/24/23	No	College
7	Taylor	33.0	52.0	\$60,000	9/8/23	No	High School

The **dollar sign** and **comma** cause this to be read in as text

customers.info()

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 8 entries, 0 to 7
Data columns (total 7 columns):
 #   Column           Non-Null Count  Dtype  
--- 
 0   Name             8 non-null      object 
 1   Age              8 non-null      float64
 2   Followers        8 non-null      float64
 3   Income           8 non-null      object 
 4   Sign Up Date    8 non-null      object 
 5   Discount         8 non-null      object 
 6   Education Level 8 non-null      object 
dtypes: float64(2), object(5)
memory usage: 576.0+ bytes
```



CONVERTING TO NUMERIC

Data Prep Steps

Row Granularity

Column Preparation

Feature Engineering

Feature Selection

Feature Scaling

Pandas will often read in numeric fields as text (object) data types

You can use **pd.to_numeric()** to convert them to numeric fields

```
customers['Income'] = customers.Income.str.replace('$', '').str.replace(',', '')
```

```
customers
```

	Name	Age	Followers	Income	Sign Up Date	Discount	Education Level
0	Aria	25.0	0.0	45000	5/18/23	Yes	College
1	Chord	19.0	12.0	28000	8/23/23	Yes	High School
2	Harmony	26.0	0.0	120000	4/25/23	No	Graduate School
3	Melody	47.0	0.0	450000	5/5/23	No	College
4	Reed	52.0	0.0	75000	6/14/23	Yes	High School
5	Selena	29.5	1.0	62000	8/26/23	No	College
6	Stefani	29.5	0.0	81000	9/24/23	No	College
7	Taylor	33.0	52.0	60000	9/8/23	No	High School

```
customers.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 8 entries, 0 to 7
Data columns (total 7 columns):
 #   Column           Non-Null Count  Dtype  
--- 
 0   Name            8 non-null      object 
 1   Age             8 non-null      float64 
 2   Followers       8 non-null      float64 
 3   Income          8 non-null      object 
 4   Sign Up Date    8 non-null      object 
 5   Discount        8 non-null      object 
 6   Education Level 8 non-null      object 
dtypes: float64(2), object(5)
memory usage: 576.0+ bytes
```

Even though we removed the punctuation,
we still need to convert to numeric



CONVERTING TO NUMERIC

Data Prep Steps

Row Granularity

Column Preparation

Feature Engineering

Feature Selection

Feature Scaling

Pandas will often read in numeric fields as text (object) data types

You can use **pd.to_numeric()** to convert them to numeric fields

```
customers['Income'] = pd.to_numeric(customers.Income)
```

```
customers
```

	Name	Age	Followers	Income	Sign Up Date	Discount	Education Level
0	Aria	25.0	0.0	45000	5/18/23	Yes	College
1	Chord	19.0	12.0	28000	8/23/23	Yes	High School
2	Harmony	26.0	0.0	120000	4/25/23	No	Graduate School
3	Melody	47.0	0.0	450000	5/5/23	No	College
4	Reed	52.0	0.0	75000	6/14/23	Yes	High School
5	Selena	29.5	1.0	62000	8/26/23	No	College
6	Stefani	29.5	0.0	81000	9/24/23	No	College
7	Taylor	33.0	52.0	60000	9/8/23	No	High School

```
customers.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 8 entries, 0 to 7
Data columns (total 7 columns):
 #   Column           Non-Null Count  Dtype  
--- 
 0   Name            8 non-null      object 
 1   Age             8 non-null      float64
 2   Followers       8 non-null      float64
 3   Income          8 non-null      int64  
 4   Sign Up Date    8 non-null      object 
 5   Discount        8 non-null      object 
 6   Education Level 8 non-null      object 
dtypes: float64(2), int64(1), object(4)
memory usage: 576.0+ bytes
```





CONVERTING TO DATETIME

Data Prep Steps

Row Granularity

Column Preparation

Feature Engineering

Feature Selection

Feature Scaling

Pandas will often read in date fields as text (object) data types as well

You can use **pd.to_datetime()** to convert them to datetime fields

customers

	Name	Age	Followers	Income	Sign Up Date	Discount	Education Level
0	Aria	25.0	0.0	45000	5/18/23	Yes	College
1	Chord	19.0	12.0	28000	8/23/23	Yes	High School
2	Harmony	26.0	0.0	120000	4/25/23	No	Graduate School
3	Melody	47.0	0.0	450000	5/5/23	No	College
4	Reed	52.0	0.0	75000	6/14/23	Yes	High School
5	Selena	29.5	1.0	62000	8/26/23	No	College
6	Stefani	29.5	0.0	81000	9/24/23	No	College
7	Taylor	33.0	52.0	60000	9/8/23	No	High School

These look like dates, but they're actually being read in as text

customers.info()

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 8 entries, 0 to 7
Data columns (total 7 columns):
 #   Column           Non-Null Count  Dtype  
--- 
 0   Name            8 non-null      object 
 1   Age             8 non-null      float64 
 2   Followers       8 non-null      float64 
 3   Income          8 non-null      int64  
 4   Sign Up Date    8 non-null      object 
 5   Discount        8 non-null      object 
 6   Education Level 8 non-null      object 
dtypes: float64(2), int64(1), object(4)
memory usage: 576.0+ bytes
```



CONVERTING TO DATETIME

Data Prep Steps

Row Granularity

Column Preparation

Feature Engineering

Feature Selection

Feature Scaling

Pandas will often read in date fields as text (object) data types as well

You can use **pd.to_datetime()** to convert them to datetime fields

```
customers['Sign Up Date'] = pd.to_datetime(customers['Sign Up Date'], format='%m/%d/%y')
```

customers

	Name	Age	Followers	Income	Sign Up Date	Discount	Education Level
0	Aria	25.0	0.0	45000	2023-05-18	Yes	College
1	Chord	19.0	12.0	28000	2023-08-23	Yes	High School
2	Harmony	26.0	0.0	120000	2023-04-25	No	Graduate School
3	Melody	47.0	0.0	450000	2023-05-05	No	College
4	Reed	52.0	0.0	75000	2023-06-14	Yes	High School
5	Selena	29.5	1.0	62000	2023-08-26	No	College
6	Stefani	29.5	0.0	81000	2023-09-24	No	College
7	Taylor	33.0	52.0	60000	2023-09-08	No	High School

customers.info()

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 8 entries, 0 to 7
Data columns (total 7 columns):
 #   Column           Non-Null Count  Dtype  
--- 
 0   Name            8 non-null      object 
 1   Age             8 non-null      float64
 2   Followers       8 non-null      float64
 3   Income          8 non-null      int64  
 4   Sign Up Date    8 non-null      datetime64[ns]
 5   Discount        8 non-null      object 
 6   Education Level 8 non-null      object 
dtypes: datetime64[ns](1), float64(2), int64(1)
memory usage: 576.0+ bytes
```



While a datetime field can't be input into a model, its components can be extracted as numeric values and saved as new fields for modeling (more on this next)



EXTRACTING DATETIME COMPONENTS

Data Prep Steps

Row Granularity

Column Preparation

Feature Engineering

Feature Selection

Feature Scaling

```
#extract the month  
customers['Sign Up Month'] = customers['Sign Up Date'].dt.month  
  
#extract the day of the week (0 = monday)  
customers['Sign Up Day'] = customers['Sign Up Date'].dt.dayofweek  
  
customers
```

	Name	Age	Followers	Income	Sign Up Date	Discount	Education Level	Sign Up Month	Sign Up Day
0	Aria	25.0	0.0	45000	2023-05-18	Yes	College	5	3
1	Chord	19.0	12.0	28000	2023-08-23	Yes	High School	8	2
2	Harmony	26.0	0.0	120000	2023-04-25	No	Graduate School	4	1
3	Melody	47.0	0.0	450000	2023-05-05	No	College	5	4
4	Reed	52.0	0.0	75000	2023-06-14	Yes	High School	6	2
5	Selena	29.5	1.0	62000	2023-08-26	No	College	8	5
6	Stefani	29.5	0.0	81000	2023-09-24	No	College	9	6
7	Taylor	33.0	52.0	60000	2023-09-08	No	High School	9	4



Even though month and day of week look like numeric values, it's inaccurate to say that 8>5 (August is better than May). These values often go through an additional process called **binning**, which we'll cover in the feature engineering section.



CALCULATING BASED ON A CONDITION

Data Prep Steps

Row Granularity

Column Preparation

Feature Engineering

Feature Selection

Feature Scaling

```
customers[['Name', 'Discount']]
```

	Name	Discount
0	Aria	Yes
1	Chord	Yes
2	Harmony	No
3	Melody	No
4	Reed	Yes
5	Selena	No
6	Stefani	No
7	Taylor	No

```
import numpy as np
```

```
#turn the field into 1 (Yes) and 0 (No) values
customers['Discount'] = np.where(customers['Discount'] == 'Yes', 1, 0)

customers
```

	Name	Age	Followers	Income	Discount	Education Level	Sign Up Month	Sign Up Day
0	Aria	25.0	0.0	45000	1	College	5	3
1	Chord	19.0	12.0	28000	1	High School	8	2
2	Harmony	26.0	0.0	120000	0	Graduate School	4	1
3	Melody	47.0	0.0	450000	0	College	5	4
4	Reed	52.0	0.0	75000	1	High School	6	2
5	Selena	29.5	1.0	62000	0	College	8	5
6	Stefani	29.5	0.0	81000	0	College	9	6
7	Taylor	33.0	52.0	60000	0	High School	9	4



PRO TIP: np.where() is quite versatile and can also be used to set values other than 1/0 (other numeric values, text values, etc.), set more than two values, set values for another column, and more



DUMMY VARIABLES

Data Prep Steps

Row Granularity

Column Preparation

Feature Engineering

Feature Selection

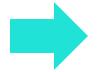
Feature Scaling

A **dummy variable** is a field that only contains ones and zeros to represent the presence (1) or absence (0) of a value, also known as one-hot encoding

- They are used to transform categorical fields into multiple numeric fields

These dummy variables are **numeric representations** of the "Education Level" field

Name	Age	Education Level
Aria	25	College
Chord	19	High School
Harmony	26	Graduate School
Melody	47	College
Reed	52	High School



Name	Age	Education Level	College	Graduate School	High School
Aria	25	College	1	0	0
Chord	19	High School	0	0	1
Harmony	26	Graduate School	0	1	0
Melody	47	College	1	0	0
Rock	52	High School	0	0	1



DUMMY VARIABLES

Data Prep Steps

Row Granularity

Column Preparation

Feature Engineering

Feature Selection

Feature Scaling

Use `pd.get_dummies()` to create dummy variables in Python

```
customers[['Education Level']]
```

Education Level	
0	College
1	High School
2	Graduate School
3	College
4	High School
5	College
6	College
7	High School

```
# create dummy variables  
pd.get_dummies(customers['Education Level'])
```

	College	Graduate School	High School
0	True	False	False
1	False	False	True
2	False	True	False
3	True	False	False
4	False	False	True
5	True	False	False
6	True	False	False
7	False	False	True





DUMMY VARIABLES

Data Prep Steps

Row Granularity

Column Preparation

Feature Engineering

Feature Selection

Feature Scaling

Use `pd.get_dummies()` to create dummy variables in Python

```
customers[['Education Level']]
```

Education Level	
0	College
1	High School
2	Graduate School
3	College
4	High School
5	College
6	College
7	High School

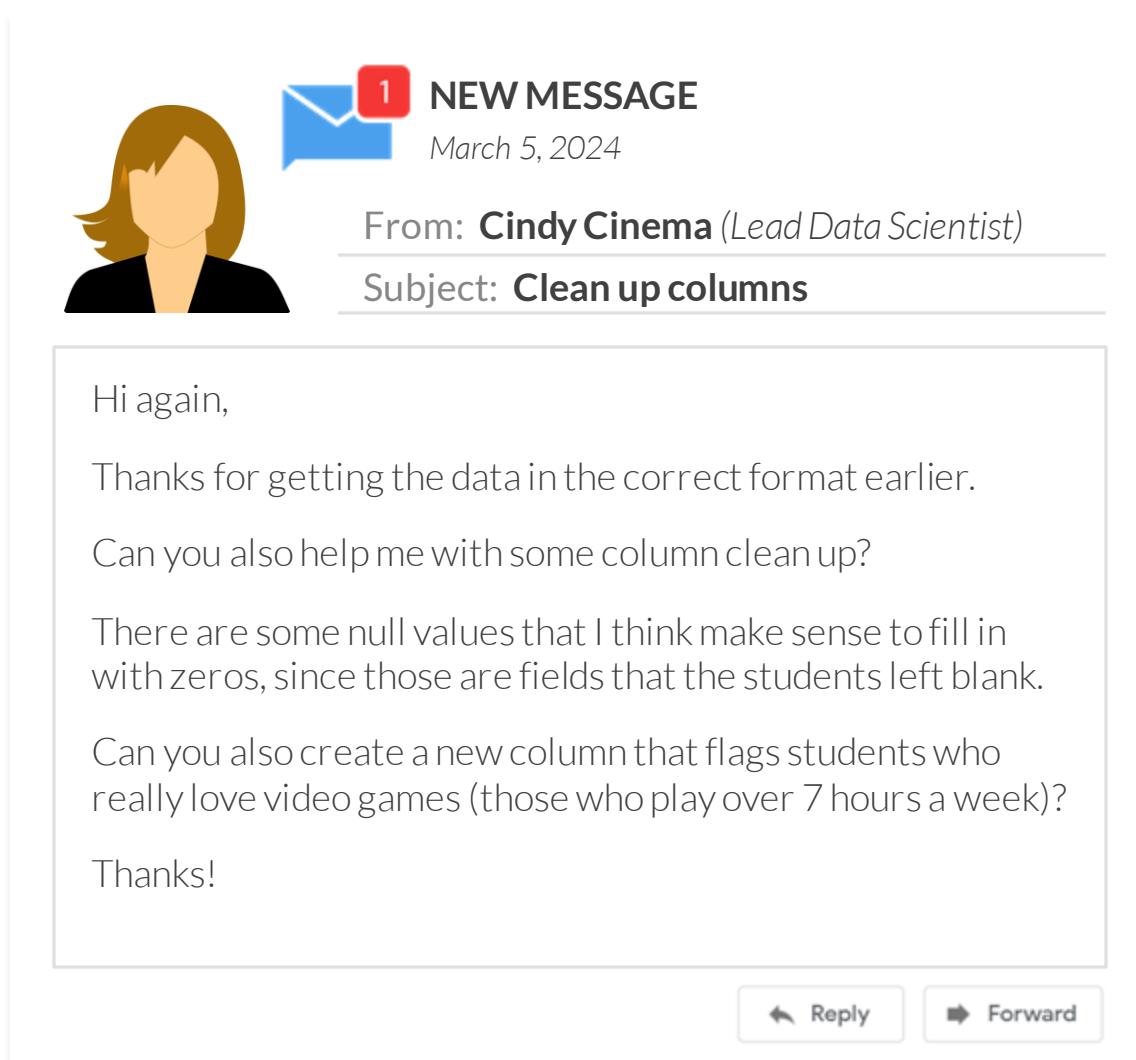
```
# make them 1s and 0s  
pd.get_dummies(customers['Education Level']).astype(int)
```

	College	Graduate School	High School
0	1	0	0
1	0	0	1
2	0	1	0
3	1	0	0
4	0	0	1
5	1	0	0
6	1	0	0
7	0	0	1



Use `.astype(int)` to convert TRUE/FALSE values to 1/0

ASSIGNMENT: PREPARE COLUMNS FOR MODELING



The image shows a simulated email inbox interface. On the left, there is a profile picture of a woman with short brown hair. Next to it is a blue envelope icon with a red notification bubble containing the number '1'. To the right of the icon, the text 'NEW MESSAGE' is displayed in bold capital letters. Below that, the date 'March 5, 2024' is shown. The email details are listed below: 'From: Cindy Cinema (Lead Data Scientist)' and 'Subject: Clean up columns'. The main body of the email contains the following text:

Hi again,
Thanks for getting the data in the correct format earlier.
Can you also help me with some column clean up?

There are some null values that I think make sense to fill in with zeros, since those are fields that the students left blank.

Can you also create a new column that flags students who really love video games (those who play over 7 hours a week)?

Thanks!

At the bottom of the email window, there are two buttons: 'Reply' with a left arrow icon and 'Forward' with a right arrow icon.

Key Objectives

1. Find the missing values
2. Fill in the missing values with zeros
3. Create a new column called **video_game_lover**
 - Set the value to 1 if a student played more than 7 hours of video games each week
 - Set the value to 0 otherwise



FEATURE ENGINEERING

Data Prep Steps

Row Granularity

Column Preparation

Feature Engineering

Feature Selection

Feature Scaling

Feature engineering is the process of creating columns that you think will be helpful inputs for improving a model (*help segment, recommend, etc.*)

`df.head()`

	Customer	Genre	# Songs
0	Aria	Pop	50
1	Aria	Indie	48
2	Aria	Rock	1
3	Chord	Pop	15
4	Chord	Indie	36



`model_df.head()`

	Name	Age	# Songs	Pct_Pop
0	Aria	25.0	99	0.505051
1	Chord	19.0	51	0.294118
2	Harmony	26.0	18	0.555556
3	Melody	47.0	2	0.000000
4	Reed	52.0	5	0.000000

This DataFrame:

- Is at the correct row granularity
- Contains non-null and numeric values
- Includes newly engineered features
- Is ready for modeling

Common feature engineering techniques:

- All the data prep steps so far
- Applying calculations
- Binning values
- Identifying proxy variables



Once you have prepared the rows & columns, the data is technically ready for modeling

However, being deliberate about **engineering new features** can be the difference between a good model and a great one



FEATURE ENGINEERING DURING DATA PREP

Data Prep Steps

Row Granularity

Column Preparation

Feature Engineering

Feature Selection

Feature Scaling

Everything we've done so far is technically a **feature engineering** technique, since we've created new columns, or features, along the way

1

Setting the correct row granularity

- **Feature aggregation:** aggregating multiple rows into a single row

2

Ensuring each column is non-null and numeric

- **Handling missing data:** imputing missing values
- **Categorical encoding:** turning categorical values into numeric values



APPLYING CALCULATIONS

Data Prep Steps

Row Granularity

Column Preparation

Feature Engineering

Feature Selection

Feature Scaling

You can combine columns using **calculations** to create new features

songs_genres

	Customer	# Songs	Indie	Pop	Rock
0	Aria	99	48	50	1
1	Chord	51	36	15	0
2	Harmony	18	5	10	3
3	Melody	2	0	0	2
4	Reed	5	0	0	5
5	Selena	60	20	20	20
6	Stefani	15	2	5	8
7	Taylor	121	19	89	13

create a new column for % pop

```
songs_genres['Pct_Pop'] = songs_genres['Pop'] / songs_genres['# Songs']  
songs_genres
```



	Customer	# Songs	Indie	Pop	Rock	Pct_Pop
0	Aria	99	48	50	1	0.505051
1	Chord	51	36	15	0	0.294118
2	Harmony	18	5	10	3	0.555556
3	Melody	2	0	0	2	0.000000
4	Reed	5	0	0	5	0.000000
5	Selena	60	20	20	20	0.333333
6	Stefani	15	2	5	8	0.333333
7	Taylor	121	19	89	13	0.735537



BINNING VALUES

Data Prep Steps

Row Granularity

Column Preparation

Feature Engineering

Feature Selection

Feature Scaling

You can group numerical features into **bins** or discrete categories if it makes more sense for your analysis

```
model_df[['Name', 'Sign Up DOW']]
```

	Name	Sign Up DOW
0	Aria	3
1	Chord	2
2	Harmony	1
3	Melody	4
4	Reed	2
5	Selena	5
6	Stefani	6
7	Taylor	4

```
# create an indicator for weekend or not
```

```
model_df['Weekend'] = np.where(customers['Sign Up DOW'].isin([5, 6]), 1, 0)
```

```
model_df[['Name', 'Sign Up DOW', 'Weekend']]
```

	Name	Sign Up DOW	Weekend
0	Aria	3	0
1	Chord	2	0
2	Harmony	1	0
3	Melody	4	0
4	Reed	2	0
5	Selena	5	1
6	Stefani	6	1
7	Taylor	4	0





PROXY VARIABLES

Data Prep Steps

Row Granularity

Column Preparation

Feature Engineering

Feature Selection

Feature Scaling

A **proxy variable** is a feature meant to approximately represent another variable

- They are used when a feature is either difficult to gather or engineer into a new feature

"Sign Up Month" is numeric, but
may not be good for modeling
(August is not better than May)

	Name	Sign Up Month	Avg_Temp
0	Aria	5	66
1	Melody	5	66
2	Chord	8	81
3	Selena	8	81
4	Harmony	4	56
5	Reed	6	77
6	Stefani	9	74
7	Taylor	9	74

Instead of turning it into dummy variables or binning the months, you can use a **proxy variable** like the average temperature each month



You may not be able to engineer proxy variables from existing data, but they can be gathered from **externalsources**



FEATURE ENGINEERING TIPS

Data Prep Steps

Row Granularity

Column Preparation

Feature Engineering

Feature Selection

Feature Scaling

1

Anyone can apply an algorithm, but only someone with **domain expertise** can engineer relevant features, which is what makes a great model

2

You want your data to be long, not wide (**many rows, few columns**), so remember to:

- Try to collect as many observations (rows) of data as you can
- Only select the most meaningful features for modeling

3

Once you start modeling, you're bound to find things you missed during data prep and will **continue to engineer features**

ASSIGNMENT: FEATURE ENGINEERING

 **NEW MESSAGE**
March 6, 2024

From: Cindy Cinema (Lead Data Scientist)
Subject: Engineer new features

Hi!

We'd like to add a few features before we begin modeling.

Can you create the following:

- A column that sums up the total hours of entertainment each student consumes weekly
- A column that calculates the percent of entertainment consumed that's on screens

Thanks again,
Cindy

[Reply](#) [Forward](#)

Key Objectives

1. Create a column called **total_entertainment** that sums up all the types of entertainment for each student
2. Create a column called **pct_screen** that calculates the percent of entertainment that's on screens (everything except for books) for each student



EXCLUDING IDENTIFIERS

Data Prep Steps

Row Granularity

Column Preparation

Feature Engineering

Feature Selection

Feature Scaling

Columns containing names or IDs should be **excluded from modeling**, but remembered for interpretation down the line

features / inputs
(what goes into a model)

Customer	Age	Discount	Pct_Pop
Aria	25	1	51
Chord	19	1	29
Harmony	26	0	56
Melody	47	0	0
Reed	52	1	0

features / inputs
(what goes into a model)

House ID	Price	Bedrooms
1	350,000	2
2	500,000	3
3	180,000	0
4	270,000	2
5	245,000	1

While it's important to remember these fields, they should not be included when the data is input into a model



FEATURE SELECTION

Data Prep Steps

Row Granularity

Column Preparation

Feature Engineering

Feature Selection

Feature Scaling

More features does not always mean a better model, so it's important to select a **subset of the features** for modeling



How do we select the “right” number of features?

- **Use your intuition** and think about the goal for your analysis – which features would do the best job predicting / segmenting / etc.?
- **Start simple** with perhaps two or three features for your model, then assess the results, and continue to make the model more complex and assess
- **Don't worry** about getting this right the first time – modeling is a lot of trial and error!



There are also feature selection and extraction techniques that use machine learning
– we will be covering some of them in the **dimensionality reduction** section



FEATURE SELECTION

Data Prep Steps

Row Granularity

Column Preparation

Feature Engineering

Feature Selection

Feature Scaling

More features does not always mean a better model, so it's important to select a **subset of the features** for modeling

	Age	Followers	Income	Discount	College	Graduate School	High School	# Songs	Indie	Pop	Rock	Pct_Pop	Weekend	Avg_Temp
0	25.000000	0.0	45000	1	1	0	0	99	48	50	1	0.505051	0	66
1	19.000000	12.0	28000	1	0	0	1	51	36	15	0	0.294118	0	66
2	26.000000	0.0	120000	0	0	1	0	18	5	10	3	0.555556	0	81
3	47.000000	0.0	450000	0	1	0	0	2	0	0	2	0.000000	0	81
4	52.000000	0.0	75000	1	0	0	1	5	0	0	5	0.000000	0	56
5	33.666667	1.0	62000	0	1	0	0	60	20	20	20	0.333333	1	77
6	33.666667	0.0	81000	0	1	0	0	15	2	5	8	0.333333	1	74
7	33.000000	52.0	60000	0	0	0	1	121	19	89	13	0.735537	0	74



My domain expertise & gut feel is telling me that the best way to differentiate customers is by their age, the total number of songs they listen to and the types of songs they listen to. These three features are what I'll include in my first round of modeling

ASSIGNMENT: FEATURE SELECTION

 **NEW MESSAGE**
March 7, 2024

From: Cindy Cinema (Lead Data Scientist)
Subject: Narrow down features

Hi again,
We're almost ready for modeling! Just a few more steps.
Can you save the student name column as its own Series that we can use to reference later on?
Can you create a final modeling DataFrame that includes only the three new features that we engineered – video_game_lover, total_entertainment and pct_screen?
Thanks!
Cindy

Reply **Forward**

Key Objectives

1. Save the **name** column of the DataFrame as its own Series for reference
2. Save the three new columns you engineered as its own DataFrame for modeling – **video_game_lover**, **total_entertainment** and **pct_screen**



SCALING FEATURES

Data Prep Steps

Row Granularity

Column Preparation

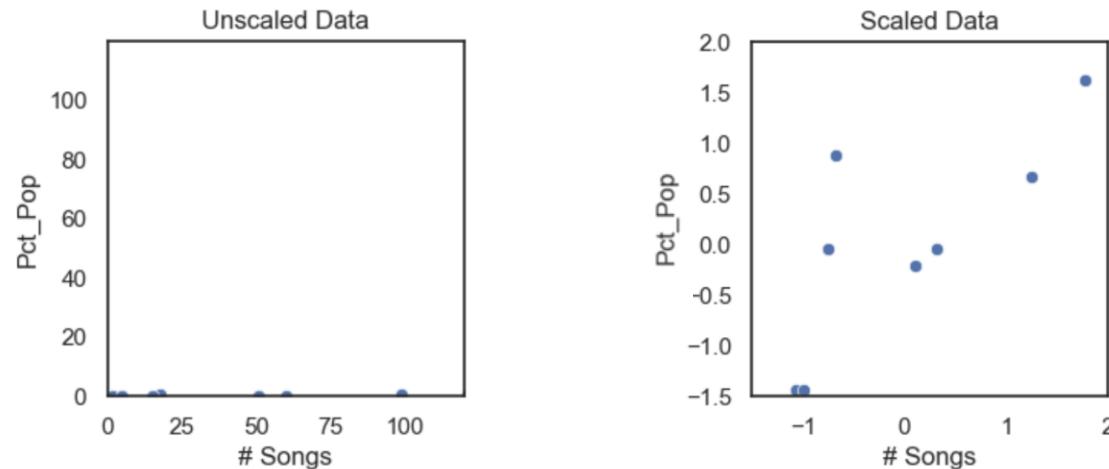
Feature Engineering

Feature Selection

Feature Scaling

Scaling, as the name implies, requires setting all input features on a similar scale

- Scaling is an optional feature engineering step which depends on the ML algorithm
- Common techniques for scaling include **normalization** and **standardization**



PRO TIP: Many unsupervised learning techniques use distance-based calculations, which makes scaling a required data prep step before modeling



NORMALIZATION

Data Prep Steps

Row Granularity

Column Preparation

Feature Engineering

Feature Selection

Feature Scaling

Normalization transforms all columns to be between 0 and 1 (or between -1 and 1)

- This is typically used when the column distributions are unknown or mixed

model_df_subset

	Age	# Songs	Pct_Pop
0	25.000000	99	0.505051
1	19.000000	51	0.294118
2	26.000000	18	0.555556
3	47.000000	2	0.000000
4	52.000000	5	0.000000
5	33.666667	60	0.333333
6	33.666667	15	0.333333
7	33.000000	121	0.735537

```
# normalization with sklearn
from sklearn.preprocessing import MinMaxScaler

mm_scaler = MinMaxScaler()
normalized = mm_scaler.fit_transform(model_df_subset)
pd.DataFrame(normalized, columns=model_df_subset.columns)
```

	Age	# Songs	Pct_Pop
0	0.181818	0.815126	0.686642
1	0.000000	0.411765	0.399868
2	0.212121	0.134454	0.755306
3	0.848485	0.000000	0.000000
4	1.000000	0.025210	0.000000
5	0.444444	0.487395	0.453184
6	0.444444	0.109244	0.453184
7	0.424242	1.000000	1.000000

Use sklearn's **MinMaxScaler** function and the **.fit_transform** method to normalize DataFrame columns

Normalization equation

$$\frac{x - x_{min}}{x_{max} - x_{min}}$$



STANDARDIZATION

Data Prep Steps

Row Granularity

Column Preparation

Feature Engineering

Feature Selection

Feature Scaling

Standardization transforms all columns to have a mean of 0 and standard deviation of 1

- This is typically used when the column distributions are *normal* (bell curve)

model_df_subset

	Age	# Songs	Pct_Pop
0	25.000000	99	0.505051
1	19.000000	51	0.294118
2	26.000000	18	0.555556
3	47.000000	2	0.000000
4	52.000000	5	0.000000
5	33.666667	60	0.333333
6	33.666667	15	0.333333
7	33.000000	121	0.735537

```
# standardization with sklearn
from sklearn.preprocessing import StandardScaler

std_scaler = StandardScaler()
standardized = std_scaler.fit_transform(model_df_subset)
pd.DataFrame(standardized, columns=model_df_subset.columns)
```

	Age	# Songs	Pct_Pop
0	-0.834272	1.257265	0.6666614
1	-1.411845	0.110496	-0.209823
2	-0.738010	-0.677908	0.876466
3	1.283496	-1.060164	-1.431898
4	1.764807	-0.988491	-1.431898
5	0.000000	0.325515	-0.046880
6	0.000000	-0.749581	-0.046880
7	-0.064175	1.782868	1.624299

Use sklearn's **StandardScaler** function and the **.fit_transform** method to standardize DataFrame columns

Standardization equation

$$\frac{x - \bar{x}_{mean}}{x_{std}}$$

ASSIGNMENT: FEATURE SCALING

 **NEW MESSAGE**
March 8, 2024

From: Cindy Cinema (Lead Data Scientist)
Subject: Feature scaling request

Hi, I have one final request for you.

We plan on clustering the data using a distance-based algorithm, so the data needs to be scaled.

Using the DataFrame with the three features for modeling, can you scale the columns so that they all have a mean of 0 and a standard deviation of 1?

Once you do that, we should be all set with our data prep steps and ready for modeling.

Thanks for your help this week!

Reply **Forward**

Key Objectives

1. Scale the features in the modeling DataFrame so they all have a mean of 0 and a standard deviation of 1
2. Save the output as a final DataFrame that's ready for modeling

KEY TAKEAWAYS



Data prep is required before applying unsupervised learning techniques

- *This includes setting the correct row granularity, making sure all columns are non-null and numeric, engineering useful features, selecting the best ones, and scaling them*



Both the **rows & columns** of a DataFrame must be prepared for modeling

- *There are several techniques that are commonly used to prepare rows and columns for modeling, including using .groupby(), .pivot(), .fillna(), np.where() and pd.get_dummies()*



Feature engineering can be the difference between a good and a great model

- *There are many techniques for feature engineering, including aggregating values, handling missing data, categorical encoding, applying calculations, binning data, using proxy variables – and then selecting features*



Feature scaling is required for models that use distance-based calculations

- *Many unsupervised learning techniques use distance-based calculations, and it's a good idea to try multiple scaling techniques, including normalization and standardization, to see which one performs best*

CLUSTERING

CLUSTERING



In this section we'll introduce the fundamentals of **clustering** and compare three popular clustering techniques: K-Means Clustering, Hierarchical Clustering, and DBSCAN

TOPICS WE'LL COVER:

Clustering Basics

K-Means Clustering

Hierarchical Clustering

DBSCAN

Comparing Models

GOALS FOR THIS SECTION:

- Learn how clustering models fundamentally work
- Use Python to apply different clustering models and interpret their results
- Compare & contrast popular clustering techniques



CLUSTERING BASICS

Clustering Basics

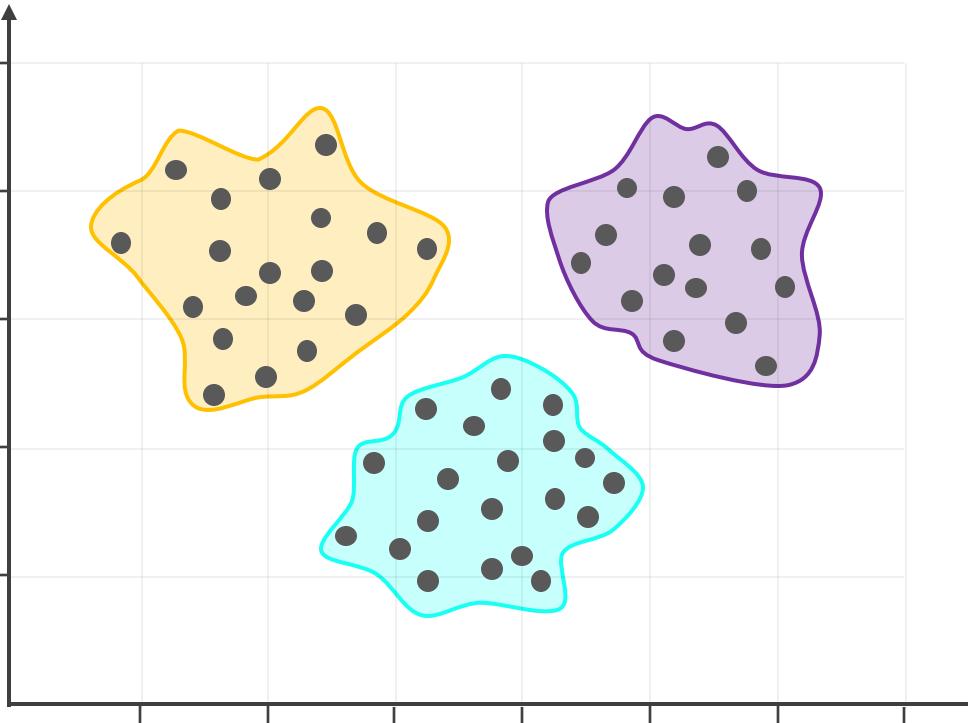
K-Means
Clustering

Hierarchical
Clustering

DBSCAN

Comparing
Models

Clustering allows you to find concentrations or groups of observations which are similar to one another but distinct from other groups





CLUSTERING WORKFLOW

Clustering Basics

K-Means Clustering

Hierarchical Clustering

DBSCAN

Comparing Models

The general **clustering workflow** consists of the following steps:

Data Prep

Get your data ready to be input into an ML model

- Single table, non-null and numeric data
- Feature engineering, selection, and scaling

Modeling

Apply a clustering algorithm

- K-Means Clustering
- Hierarchical Clustering
- DBSCAN

Tuning

Evaluate & tune the model using metrics and intuition

- Metrics (*i.e. inertia*)
- Data visualization
- Interpret the results

Selection

Pick the best results and identify any insights

- Business objective
- Domain expertise



Remember, there's no "right" answer or single optimization metric when it comes to clustering; the best outputs are the ones which help you **answer the question at hand** and **make practical, data-driven business decisions**



K-MEANS CLUSTERING

Clustering Basics

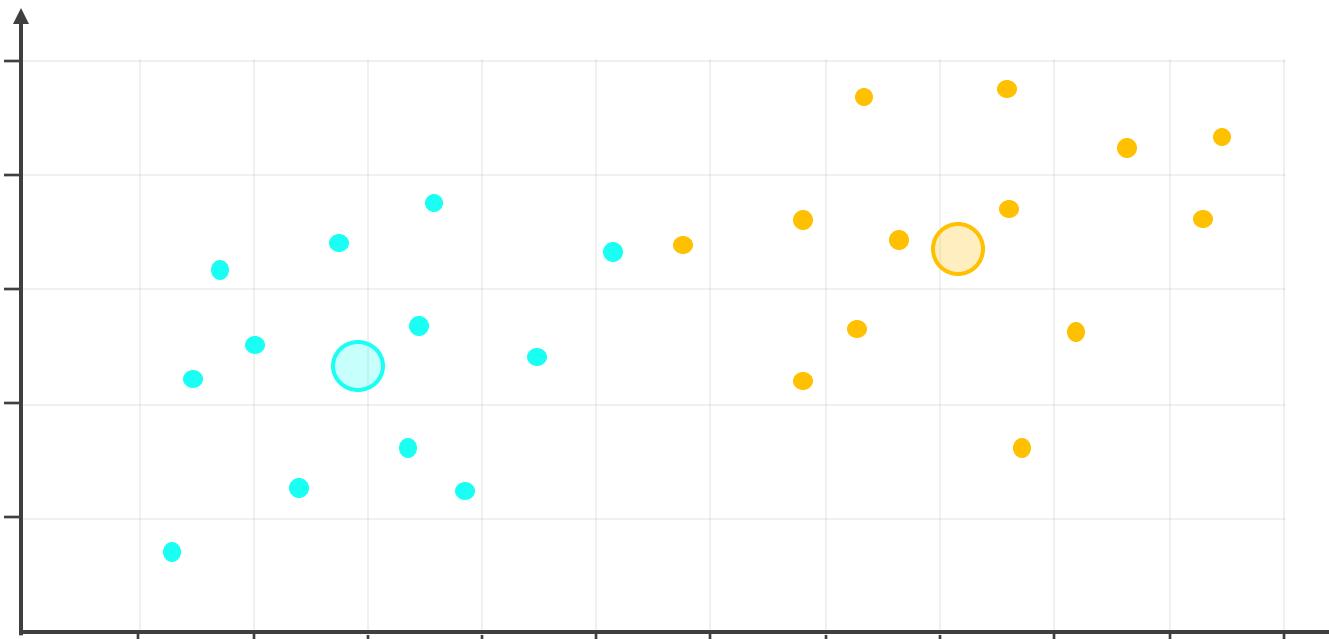
K-Means
Clustering

Hierarchical
Clustering

DBSCAN

Comparing
Models

K-Means Clustering is a popular algorithm which assigns each observation in a data set to a specific cluster, where “K” represents the number of clusters





K-MEANS CLUSTERING

Clustering Basics

K-Means
Clustering

Hierarchical
Clustering

DBSCAN

Comparing
Models

K-Means Clustering is a popular algorithm which assigns each observation in a data set to a specific cluster, where “K” represents the number of clusters

Here's how it works:

1. Select “K” arbitrary locations in a scatter plot as cluster centers (or **centroids**), and assign each observation to a cluster based on the closest centroid
2. Recalculate and relocate each centroid to the mean of the observations assigned to it, then reassign each observation to its new closest centroid
3. Repeat the process until observations no longer change clusters

Example use cases:

- Identifying customer segments for targeted marketing campaigns
- Clustering store locations based on factors like sales, ratings, size, etc.



K-MEANS CLUSTERING

Clustering Basics

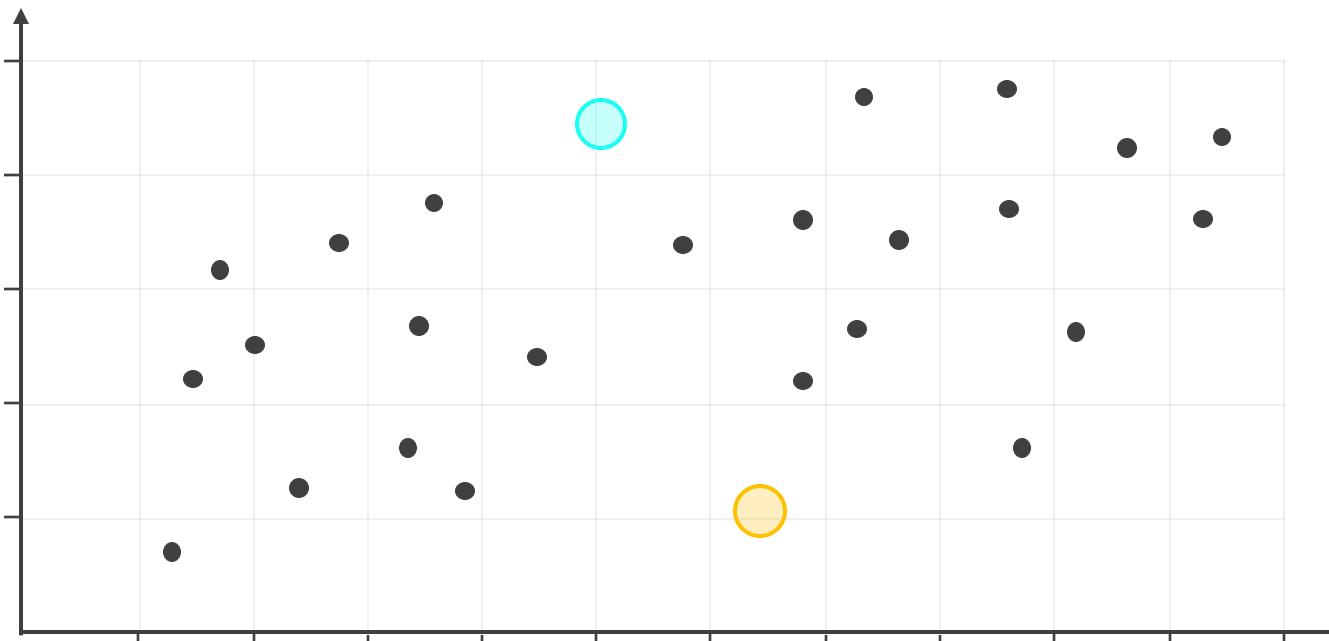
K-Means
Clustering

Hierarchical
Clustering

DBSCAN

Comparing
Models

STEP 1: Determine what you think might be an appropriate number of clusters (in this case 2), and select arbitrary locations as initial centroids





K-MEANS CLUSTERING

Clustering Basics

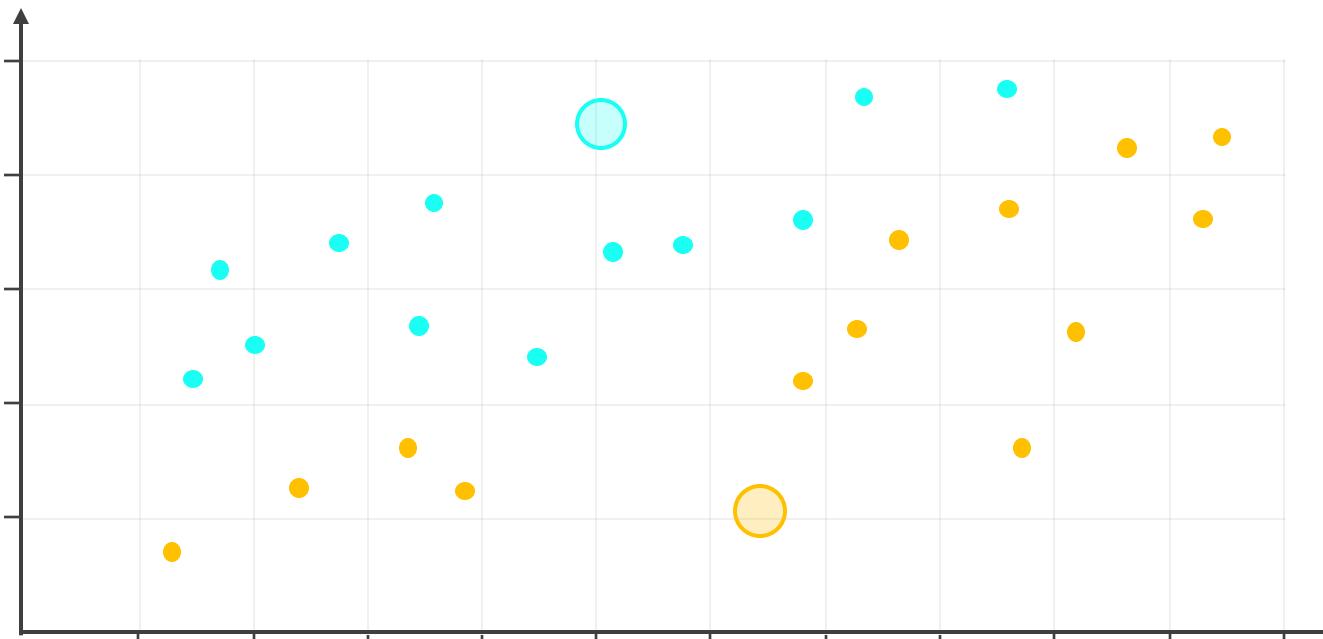
K-Means
Clustering

Hierarchical
Clustering

DBSCAN

Comparing
Models

STEP 2: Assign each observation to a cluster, based on the closest centroid





K-MEANS CLUSTERING

Clustering Basics

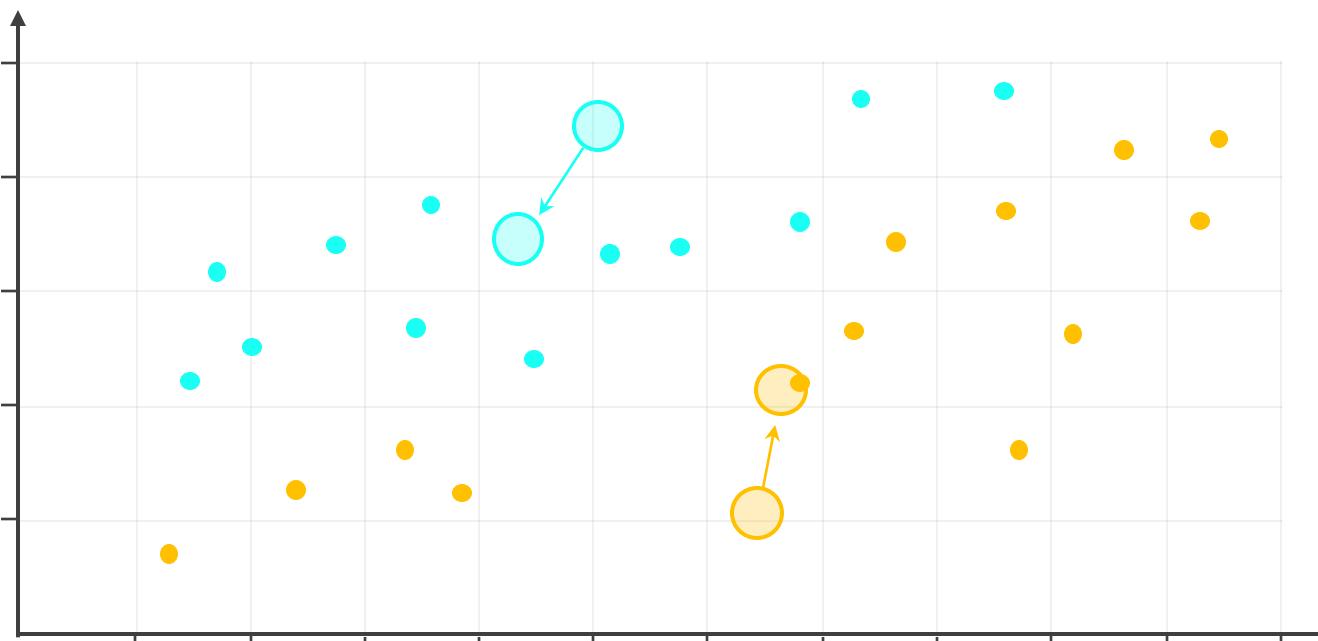
K-Means
Clustering

Hierarchical
Clustering

DBSCAN

Comparing
Models

STEP 3: Relocate each centroid to the mean of its assigned observations, and reassign each observation to the new closest centroid





K-MEANS CLUSTERING

Clustering Basics

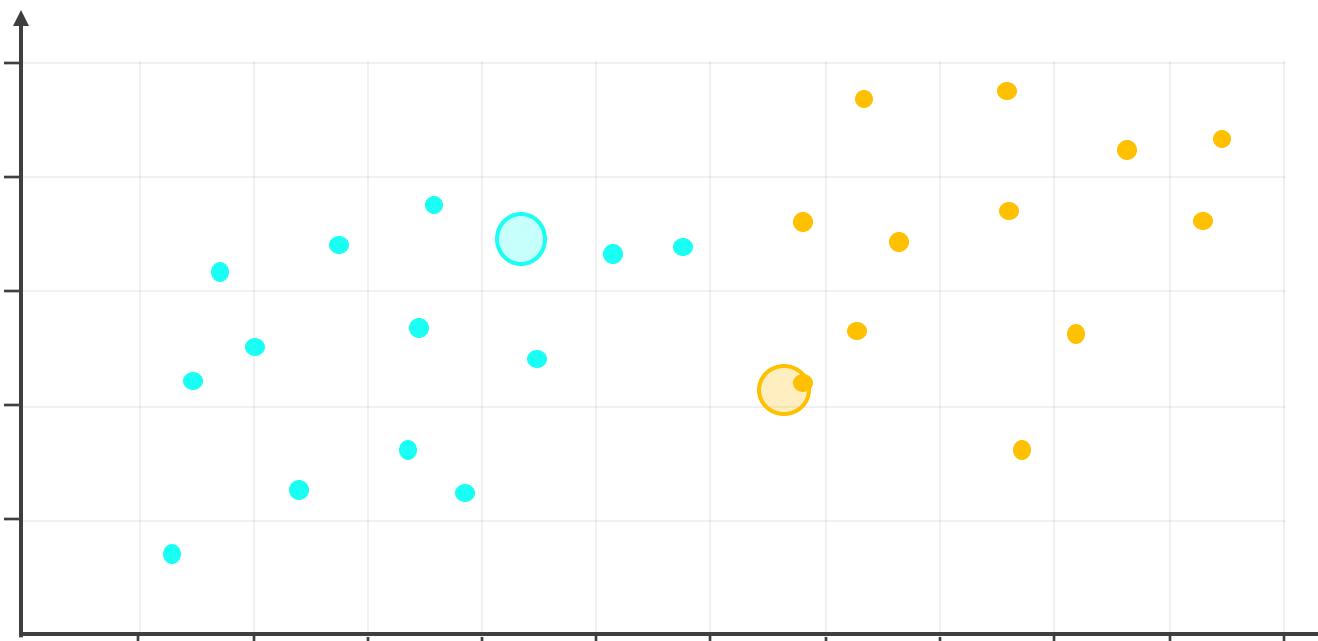
K-Means
Clustering

Hierarchical
Clustering

DBSCAN

Comparing
Models

STEP 3: Relocate each centroid to the mean of its assigned observations, and reassign each observation to the new closest centroid





K-MEANS CLUSTERING

Clustering Basics

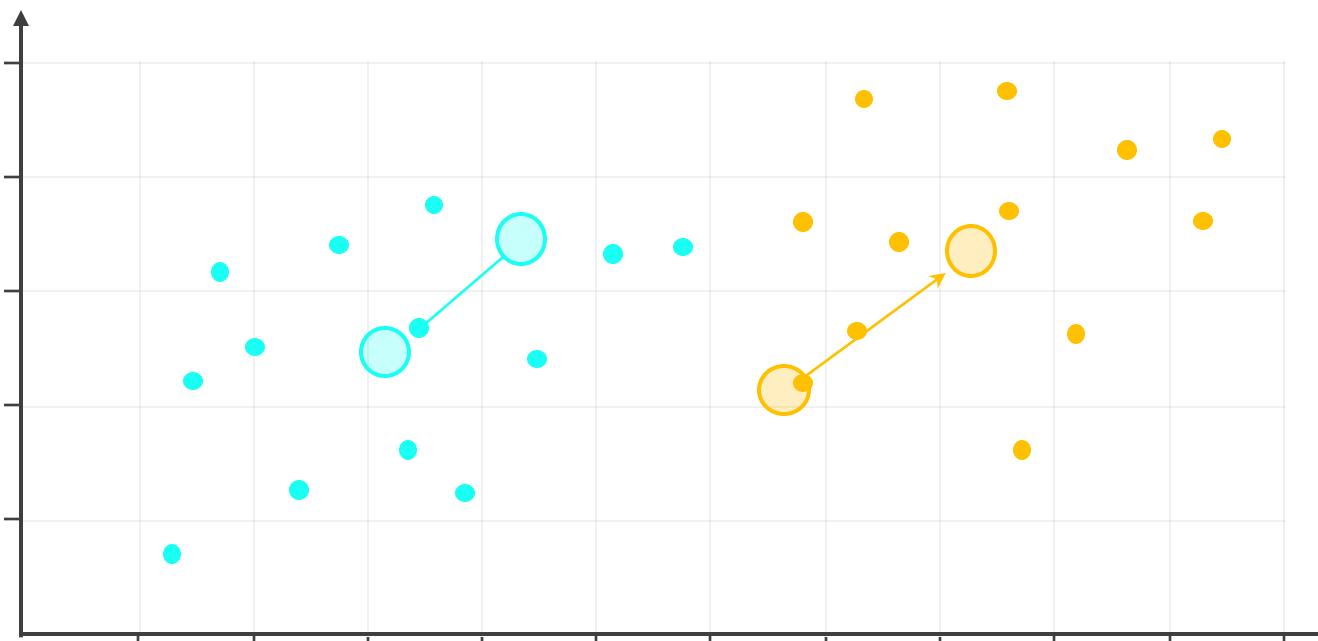
K-Means
Clustering

Hierarchical
Clustering

DBSCAN

Comparing
Models

STEP 4: Continue to relocate each centroid to the mean of its assigned observations, until the clusters no longer change





K-MEANS CLUSTERING

Clustering Basics

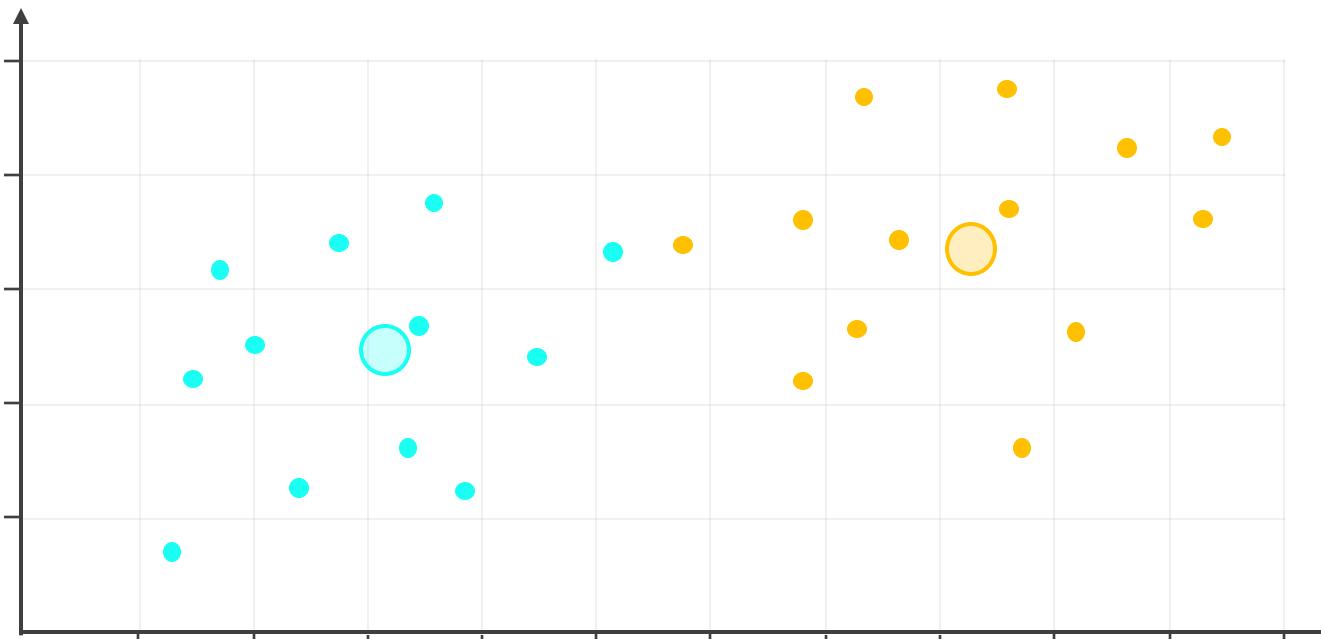
K-Means
Clustering

Hierarchical
Clustering

DBSCAN

Comparing
Models

STEP 4: Continue to relocate each centroid to the mean of its assigned observations, until the clusters no longer change





K-MEANS CLUSTERING

Clustering Basics

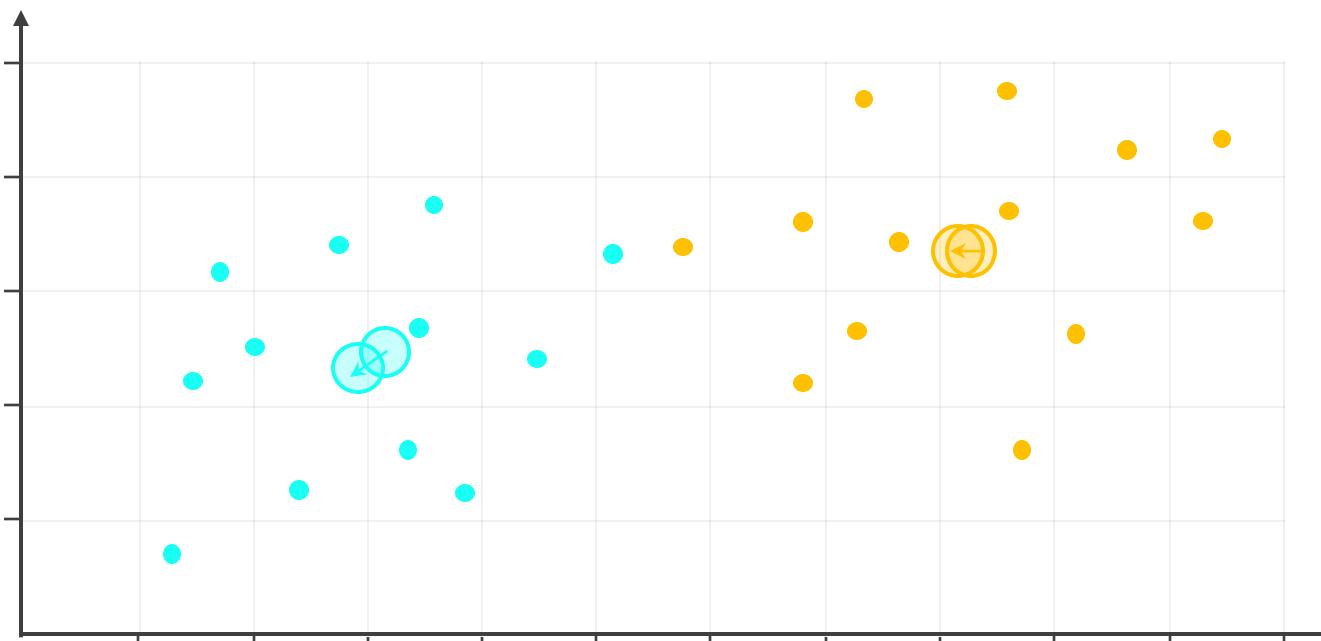
K-Means
Clustering

Hierarchical
Clustering

DBSCAN

Comparing
Models

STEP 4: Continue to relocate each centroid to the mean of its assigned observations, until the clusters no longer change





K-MEANS CLUSTERING

Clustering Basics

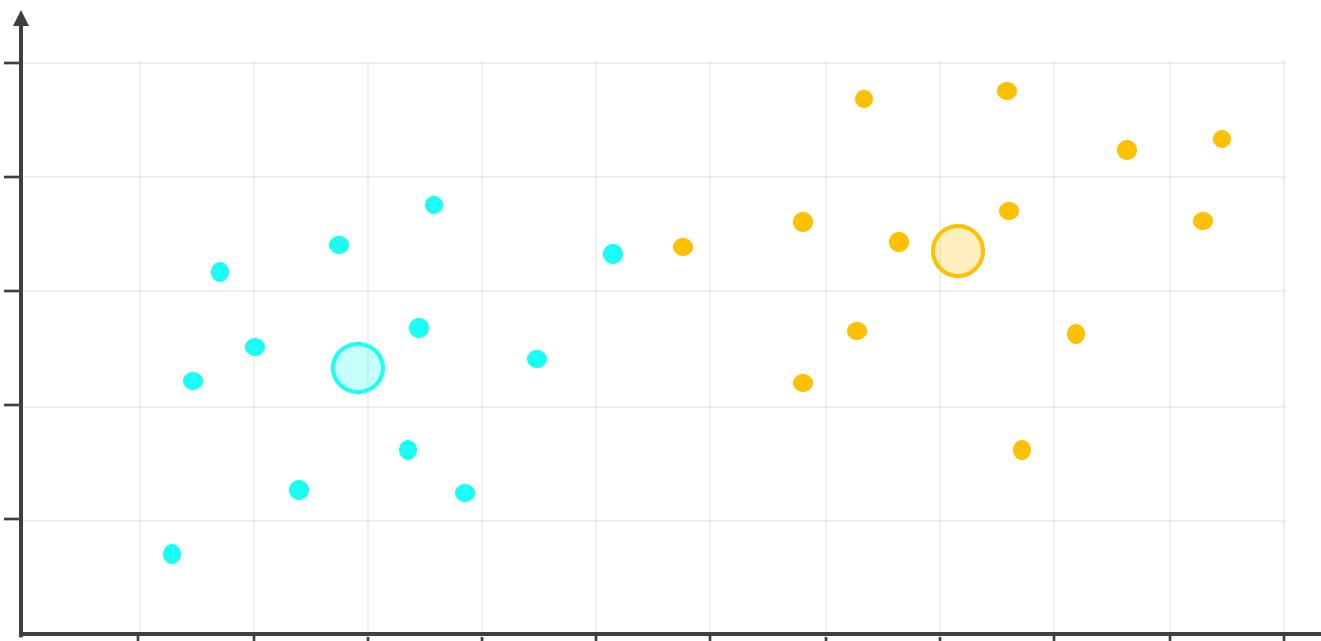
K-Means
Clustering

Hierarchical
Clustering

DBSCAN

Comparing
Models

STEP 4: Continue to relocate each centroid to the mean of its assigned observations, until the clusters no longer change





K-MEANS CLUSTERING IN PYTHON

Clustering Basics

K-Means
Clustering

Hierarchical
Clustering

DBSCAN

Comparing
Models

You can use sklearn's **KMeans()** function to perform K-Means Clustering

```
from sklearn.cluster import KMeans  
  
kmeans = KMeans(n_clusters=2, n_init='auto', random_state=42)
```

The "k" number of clusters to identify (default is 8)

The number of models to fit with different initial centroids, returning the best result ("auto" will fit one model)

Setting a random_state value guarantees the same results each time the model is fit



PRO TIP: It's typically a good idea to start with 2 clusters and build up from there, comparing the results



K-MEANS CLUSTERING IN PYTHON

Clustering Basics

K-Means
Clustering

Hierarchical
Clustering

DBSCAN

Comparing
Models

EXAMPLE

Clustering high school students based on entertainment preferences so the local library can use targeted ads to encourage teenagers to read more

`df.head(10)`

	name	books	tv_shows	video_games
0	Aaliyah	0.5	4.6	4.9
1	Abigail	0.0	4.5	4.8
2	Addison	0.5	4.5	5.0
3	Adeline	3.5	4.5	6.6
4	Alana	2.8	3.8	5.6
5	Alexander	5.8	4.6	6.9
6	Alivia	4.2	4.5	6.7
7	Amara	3.2	4.5	5.6
8	Amelia	0.0	4.6	4.9
9	Annabelle	4.0	4.1	6.0



Some students spend almost no time reading each week



Students prefer to spend most of their time playing video games



K-MEANS CLUSTERING IN PYTHON

EXAMPLE

Clustering high school students based on entertainment preferences so the local library can use targeted ads to encourage teenagers to read more

Clustering Basics

K-Means Clustering

Hierarchical Clustering

DBSCAN

Comparing Models

```
# import kmeans from sklearn
from sklearn.cluster import KMeans

# fit a kmeans model with 2 clusters
kmeans = KMeans(n_clusters=2, n_init='auto', random_state=42)
kmeans.fit(data)
```

KMeans

Once the model is fit, you can view the cluster that each row as been assigned to using the `.labels_` attribute



VISUALIZING K-MEANS CLUSTERING

Clustering Basics

K-Means Clustering

Hierarchical Clustering

DBSCAN

Comparing Models

```
# import plotting libraries
import matplotlib.pyplot as plt
import seaborn as sns
from mpl_toolkits.mplot3d import Axes3D

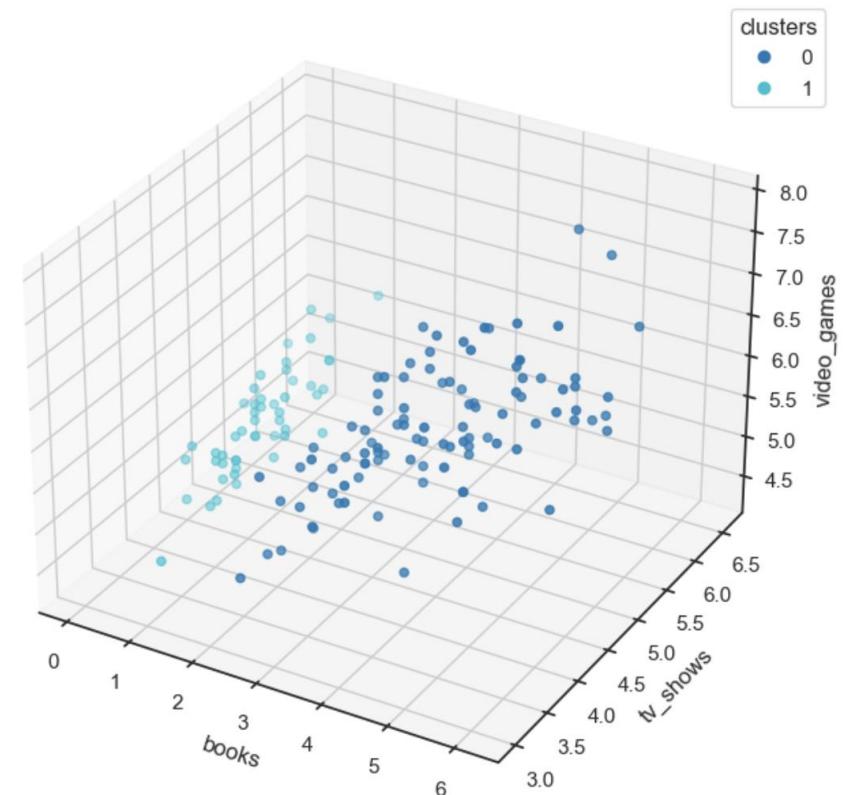
# combine the data and cluster labels
cluster_labels = pd.Series(kmeans.labels_, name='cluster')

# create a clean dataframe
df = pd.concat([data, cluster_labels], axis=1)

# create a 3d scatter plot
fig = plt.figure(figsize=(8, 6))
ax = Axes3D(fig)
fig.add_axes(ax)

# specify the data and labels
sc = ax.scatter(df['books'], df['tv_shows'], df['video_games'],
                 c=df['cluster'], cmap='tab10')
ax.set_xlabel('books')
ax.set_ylabel('tv_shows')
ax.set_zlabel('video_games')

# add a legend
plt.legend(*sc.legend_elements(), title='clusters',
           bbox_to_anchor=(1.05, 1));
```





INTERPRETING K-MEANS CLUSTERING

Clustering Basics

K-Means
Clustering

Hierarchical
Clustering

DBSCAN

Comparing
Models

You can **interpret the results** of a K-Means model using the `.cluster_centers_` attribute and your intuition

```
# view the column names  
data.columns  
  
Index(['books', 'tv_shows', 'video_games'], dtype='object')
```

```
# view the cluster centers  
kmeans.cluster_centers_
```

```
array([[4.192, 4.314, 6.262],  
       [0.596, 5.13 , 5.006]])
```

Students in the first cluster spend on average:

- 4.2 hours reading books
- 4.3 hours watching TV shows
- 6.3 hours playing video games



These students consume a good amount of each type of content, so we could name them **“consumers of all types of entertainment”**

Students in the second cluster spend on average:

- 0.6 hours reading books
- 5.1 hours watching TV shows
- 5 hours playing video games



These students don't read many books, so we could name them **“non-readers”**



VISUALIZING CLUSTER CENTERS

Visualizing the cluster centers in a heat map can help interpret them

Clustering Basics

K-Means Clustering

Hierarchical Clustering

DBSCAN

Comparing Models

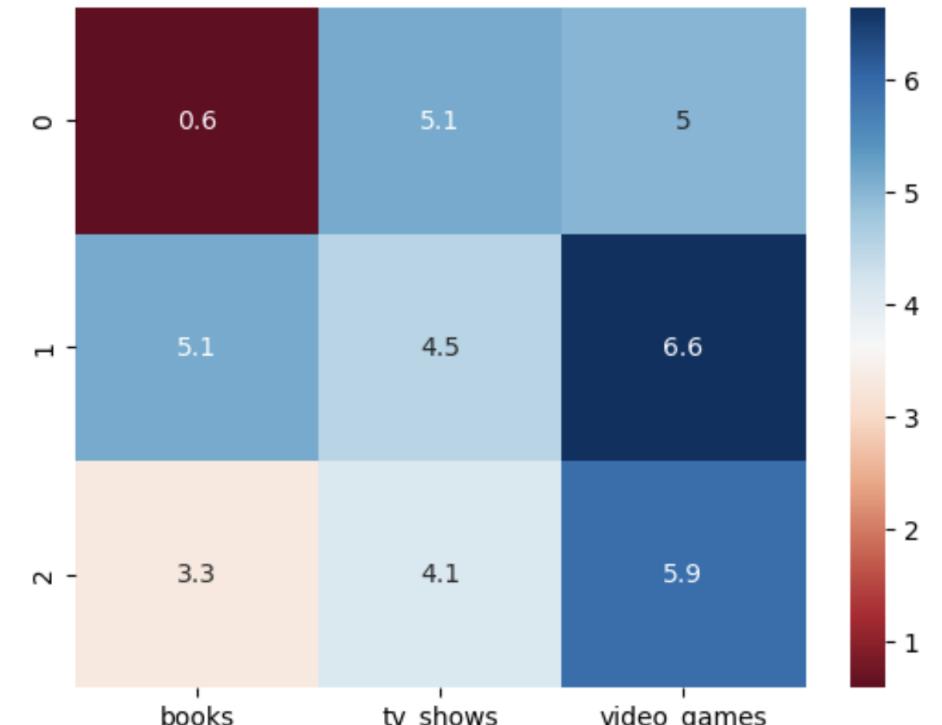
```
# view the cluster centers  
kmeans3.cluster_centers_
```

```
array([[0.596      , 5.13       , 5.006     ],  
       [5.14375    , 4.52708333, 6.63958333],  
       [3.31346154, 4.11730769, 5.91346154]])
```

```
# view the cluster centers in a dataframe  
cluster_centers3 = pd.DataFrame(kmeans3.cluster_centers_,  
                                  columns=data.columns)  
cluster_centers3
```

	books	tv_shows	video_games
0	0.596000	5.130000	5.006000
1	5.143750	4.527083	6.639583
2	3.313462	4.117308	5.913462

```
# view the cluster centers in a heatmap  
import seaborn as sns  
sns.heatmap(cluster_centers3, cmap='RdBu', annot=True);
```



Cluster 0 students don't read many books
Cluster 1 students consume a lot of entertainment
Cluster 2 students prefer video games to books

ASSIGNMENT: K-MEANS CLUSTERING

 **1 NEW MESSAGE**
March 11, 2024

From: Clyde Clusters (Sr. Data Scientist)
Subject: K-Means help

Hi there!

Our client, Maven Supermarket, would like to set up cereal displays around their store based on various niches of cereals (i.e. geared towards kids, those with dietary restrictions, etc.).

As a starting point, can you fit a K-Means Clustering model on the attached cereal data set? I recommend using two clusters. Let me know how you decide to interpret the two clusters.

Thanks!
Clyde

 cereal.csv Reply Forward

Key Objectives

1. Read in the cereal.csv file
2. Prep the data by dropping the name and manufacturer columns
3. Fit a K-Means Clustering model with 2 clusters
4. Interpret the cluster centers



INERTIA

Clustering Basics

K-Means
Clustering

Hierarchical
Clustering

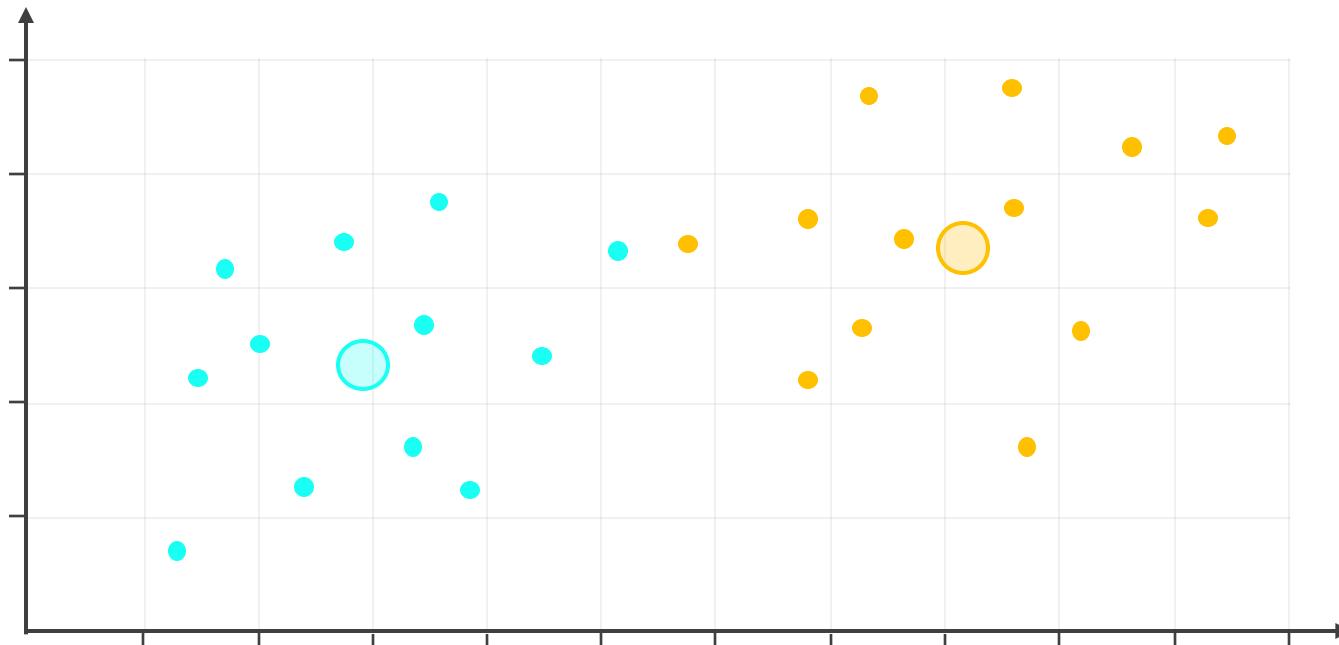
DBSCAN

Comparing
Models



How do we know what's the “right” number of clusters (K)?

- While there is no “right” or “wrong” number of clusters, you can use the **inertia** (aka within-cluster sum of squares or WCSS) to help inform your decision





INERTIA

Clustering Basics

K-Means
Clustering

Hierarchical
Clustering

DBSCAN

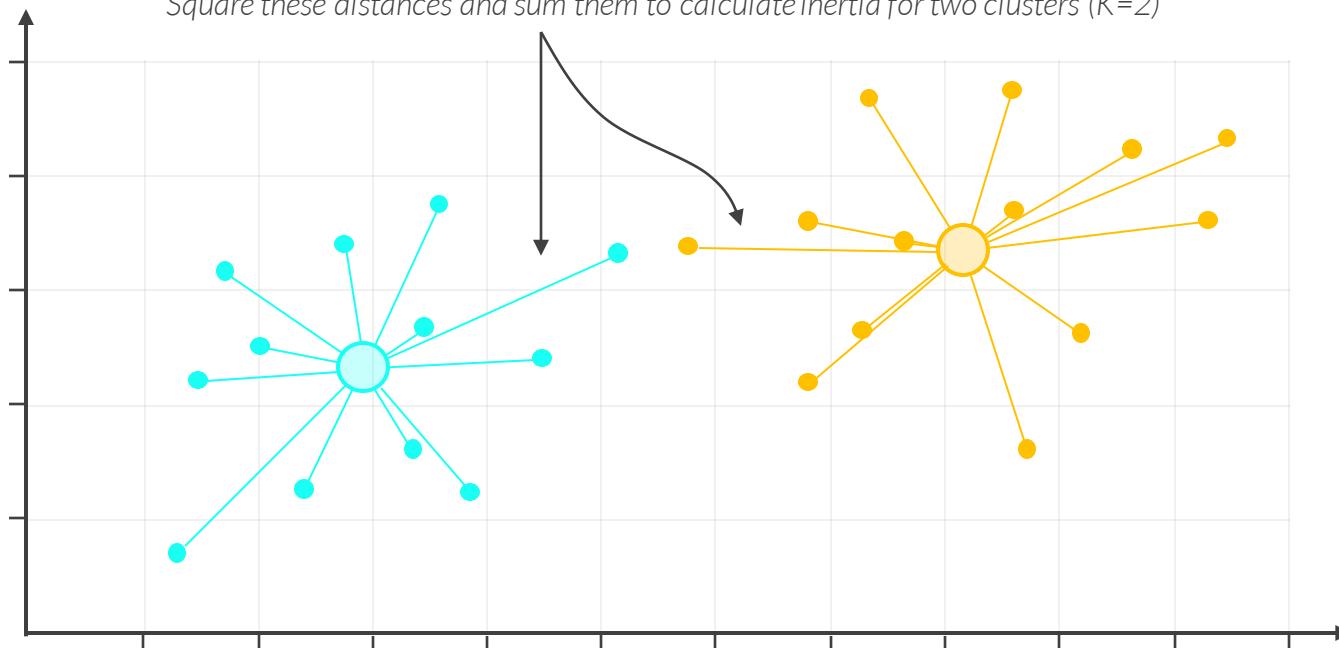
Comparing
Models



How do we know what's the “right” number of clusters (K)?

- While there is no “right” or “wrong” number of clusters, you can use the **inertia** (aka within-cluster sum of squares or WCSS) to help inform your decision

Square these distances and sum them to calculate inertia for two clusters ($K=2$)





INERTIA

Clustering Basics

K-Means
Clustering

Hierarchical
Clustering

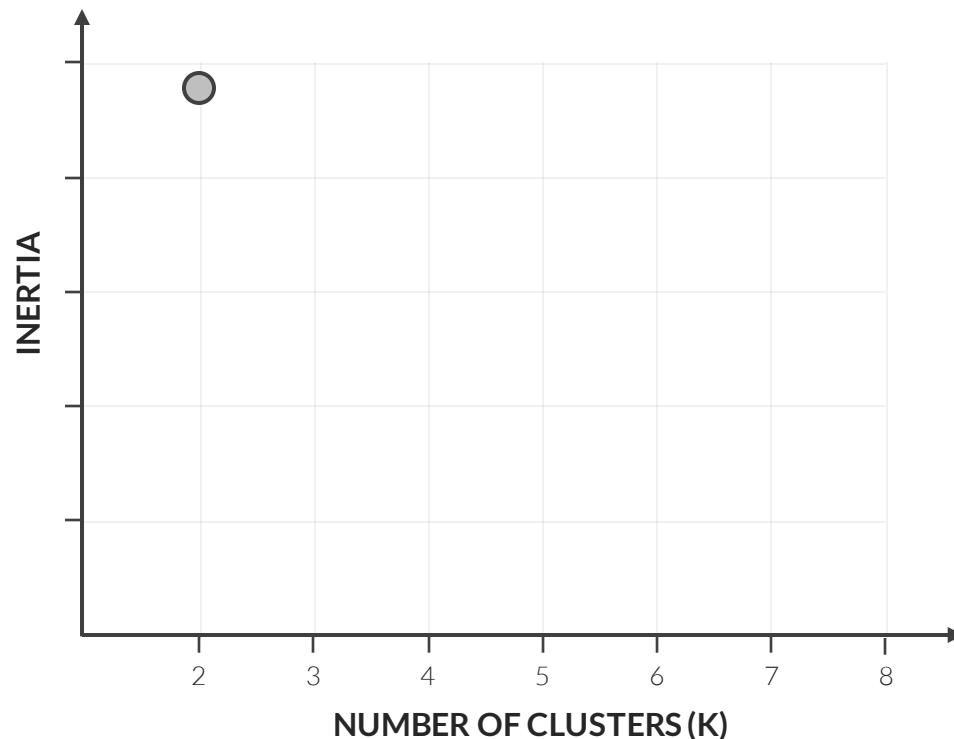
DBSCAN

Comparing
Models



How do we know what's the “right” number of clusters (K)?

- While there is no “right” or “wrong” number of clusters, you can use the **inertia** (aka within-cluster sum of squares or WCSS) to help inform your decision





INERTIA

Clustering Basics

K-Means Clustering

Hierarchical Clustering

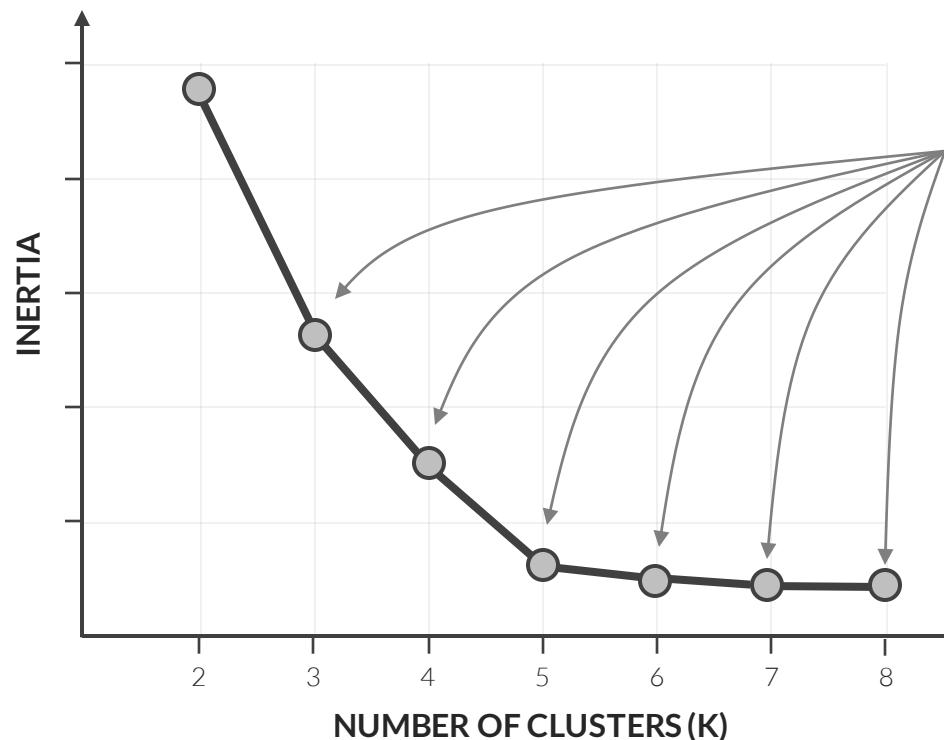
DBSCAN

Comparing Models



How do we know what's the “right” number of clusters (K)?

- While there is no “right” or “wrong” number of clusters, you can use the **inertia** (aka within-cluster sum of squares or WCSS) to help inform your decision



Rerun the model with additional clusters, and plot the inertia for each value of K



INERTIA

Clustering Basics

K-Means Clustering

Hierarchical Clustering

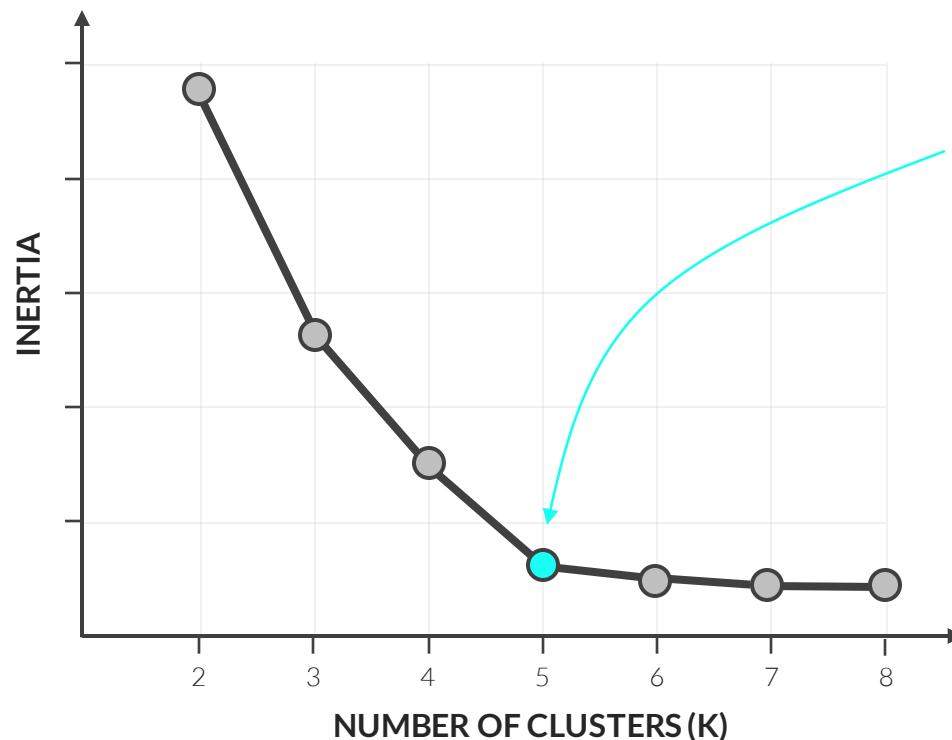
DBSCAN

Comparing Models



How do we know what's the “right” number of clusters (K)?

- While there is no “right” or “wrong” number of clusters, you can use the **inertia** (aka within-cluster sum of squares or WCSS) to help inform your decision



Look for an “elbow” or inflection point, where adding another cluster has a relatively small impact on inertia (in this case where $K=5$)



PRO TIP: Think of this as a guideline, not a strict rule



PLOTTING INERTIA IN PYTHON

Clustering Basics

K-Means Clustering

Hierarchical Clustering

DBSCAN

Comparing Models

```
# fit k-means models for 2-15 clusters, note the inertia scores
inertia_values = []

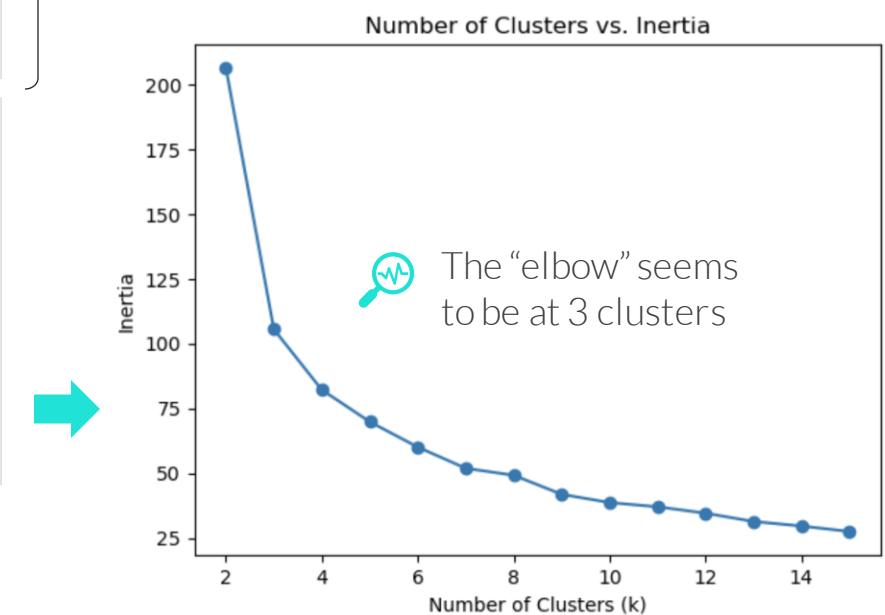
for k in range(2, 16):
    kmeans = KMeans(n_clusters=k, n_init=10, random_state=30)
    kmeans.fit(data)
    inertia_values.append(kmeans.inertia_)

# plot the inertia values
import matplotlib.pyplot as plt

# turn the list into a series for plotting
inertia_series = pd.Series(inertia_values, index=range(2, 16))

# plot the data
inertia_series.plot(marker='o')
plt.xlabel("Number of Clusters (k)")
plt.ylabel("Inertia")
plt.title("Number of Clusters vs. Inertia");
```

Fit models using 2 to 15 clusters and append their inertia values to a list



ASSIGNMENT: INERTIA PLOT

 **1 NEW MESSAGE**
March 12, 2024

From: Clyde Clusters (Sr. Data Scientist)
Subject: More K-Means help

Hi again!

Thanks for sharing the results of your 2-cluster model – they make a lot of sense, so let's continue down this path.

Can you fit 14 K-Means Clustering models, using 2-15 clusters, and plot their inertia values?

Once you find the “elbow”, let me know how many clusters we should use and what you recommend naming those clusters.

Thanks!
Clyde

Reply **Forward**

Key Objectives

1. Write a loop to fit K-Means Clustering models with 2 to 15 clusters
2. Create a plot with the number of clusters on the x-axis and the inertia on the y-axis
3. Identify the elbow of the plot
4. Fit a K-Means Clustering model on the specific number of clusters at the elbow
5. Interpret the cluster centers using a heat map



TUNING A K-MEANS MODEL

Part of the clustering workflow is to go back to various data prep and modeling steps to **tune a model** before selecting the best one

Clustering Basics

K-Means
Clustering

Hierarchical
Clustering

DBSCAN

Comparing
Models

Data Prep

- **Data cleaning:** Removing outliers, etc.
- **Feature engineering:** Creating relevant features, etc.
- **Feature selection:** More features does not mean a better model!
- **Scaling:** K-Means is a distance-based algorithm, so it's a good idea to scale the data

Modeling

- **Trying a different number of clusters:** If your clusters are very different with each run, K-Means with that specific number of clusters may not be the best fit for your data, so try using a different number of clusters
- **Trying other clustering models:** K-Means works best when the clusters are mostly circular in shape, but algorithms like Hierarchical Clustering (*up next!*) can address this

ASSIGNMENT: TUNING A K-MEANS MODEL

  **NEW MESSAGE**
March 13, 2024

From: Clyde Clusters (Sr. Data Scientist)
Subject: K-Means Fine Tuning

Hi again,

I realized that we could potentially improve our model by selecting a subset of the features and scaling them.

Can you:

- Remove the “Fat” column
- Standardize the four remaining columns
- Run the same code fitting 14 different models, plot their inertia values, and then interpret the “best” clusters

Thanks!

[Reply](#) [Forward](#)

Key Objectives

1. Remove the “Fat” column
2. Standardize the remaining columns
3. Repeat the steps from the previous inertia plot assignment
 - a) Write a loop to fit K-Means Clustering models with 2 to 15 clusters
 - b) Create a plot with the number of clusters on the x-axis and the inertia on the y-axis
 - c) Identify the elbow of the plot
 - d) Fit a K-Means Clustering model on the specific number of clusters at the elbow
 - e) Interpret the cluster centers using a heat map



SELECTING THE BEST MODEL

Clustering Basics

K-Means
Clustering

Hierarchical
Clustering

DBSCAN

Comparing
Models

There is no definitive best model when clustering, but a good model has clusters that **make sense, capture patterns, and help solve the business problem**

You can explore the clusters using data-based approaches such as:

- Comparing the cluster assignments for the data set and individual rows of data
- Comparing metrics of the various models – inertia, silhouette score (*coming soon!*), etc.
- Testing the clustering models on unseen data (*more on this at the end of this section!*)



PRO TIP: Sometimes the exact cluster assignments don't matter as much as the actionable recommendations you can make based on the clustering models



SELECTING THE BEST MODEL

Clustering Basics

K-Means
Clustering

Hierarchical
Clustering

DBSCAN

Comparing
Models

EXAMPLE

Clustering high school students based on entertainment preferences so the local library can use targeted ads to encourage teenagers to read more

```
# view the number of students in each cluster for model 1
model1_names.value_counts()
```

```
model1_clusters
Prefer Video Games to Books      52
Non-Readers                      50
Entertainment Enthusiasts        48
Name: count, dtype: int64
```

```
# view the number of students in each cluster for model 2
model2_names.value_counts()
```

```
model2_clusters
Typical Students                  52
Less Entertainment (Few Books)    50
Less Screens                       36
Entertainment Enthusiasts (Many Video Games) 12
Name: count, dtype: int64
```

```
# compare the cluster assignments and means of both models
(cluster_names.groupby(['model1_clusters', 'model2_clusters'])
[[ 'books', 'tv_shows', 'video_games']]
.mean())
```

model1_clusters		model2_clusters		
		books	tv_shows	video_games
Entertainment Enthusiasts	Entertainment Enthusiasts (Many Video Games)	5.125000	4.691667	7.475000
	Less Screens	5.150000	4.472222	6.361111
Non-Readers	Less Entertainment (Few Books)	0.596000	5.130000	5.006000
Prefer Video Games to Books	Typical Students	3.313462	4.117308	5.913462



Model 1 has 3 evenly-sized clusters, while model 2 has more specific clusters



The “less screens” and “entertainment enthusiasts” groups are quite similar and could possibly be combined into a single cluster

ASSIGNMENT: SELECTING THE BEST K-MEANS MODEL

 NEW MESSAGE
March 14, 2024

From: Clyde Clusters (Sr. Data Scientist)
Subject: Final K-Means Model

Hello, thanks for all your help with modeling so far!

As a reminder, our original goal was to help our client, Maven Supermarket, set up cereal displays around their store based on various niches of cereals.

Looking at the models that you built, can you compare them and let me know which clusters make the most sense?

Once you do, I'll pass along your recommendations to the Maven Supermarket team.

Thanks!

[Reply](#) [Forward](#)

Key Objectives

1. Compare two models:
 - a) Label each row in your original data set with a cluster name from the unstandardized data model and a cluster name from standardized data model
 - b) How many cereals fall into each cluster?
 - c) Decide on the best model for our client
2. Recommend a specific number of displays and suggest a few cereals that should be shown in each display



HIERARCHICAL CLUSTERING

Clustering Basics

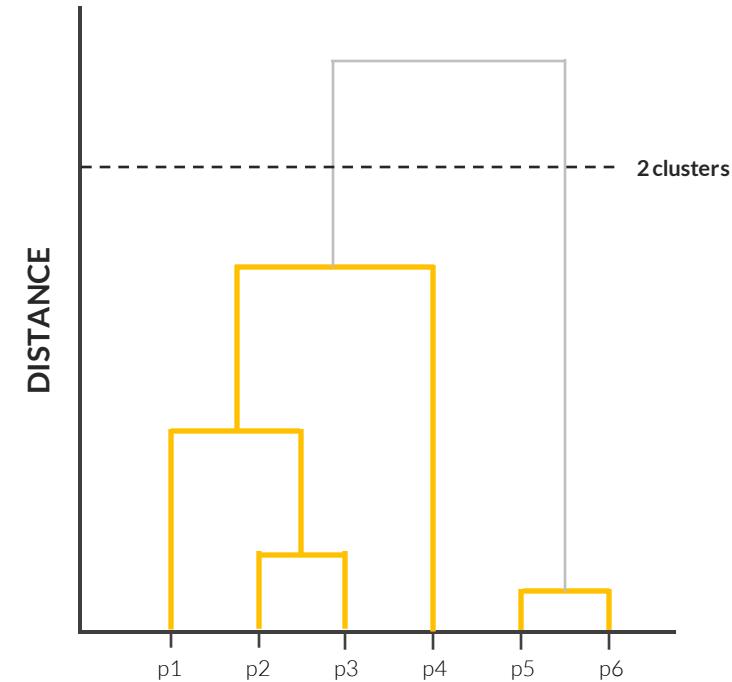
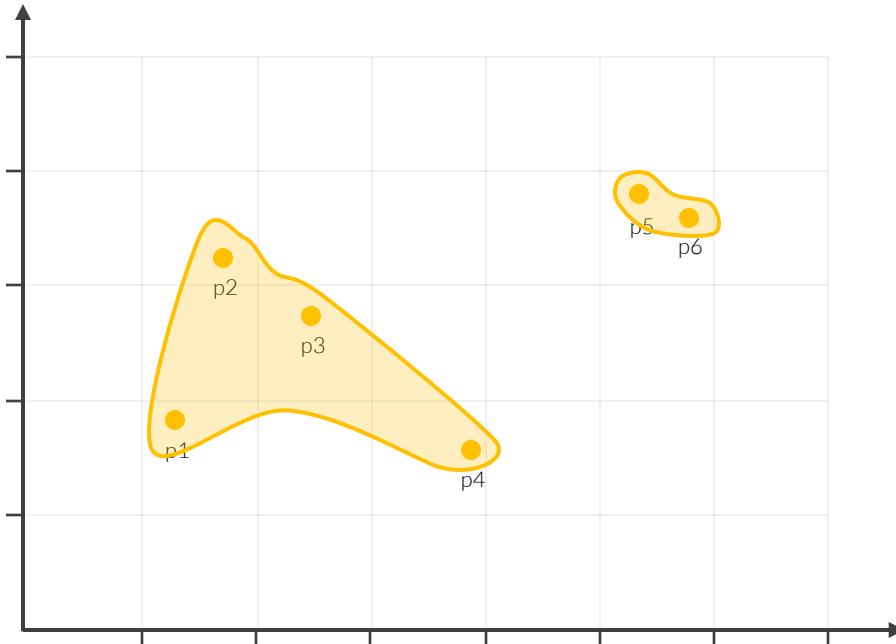
K-Means
Clustering

Hierarchical
Clustering

DBSCAN

Comparing
Models

Hierarchical Clustering is a clustering technique that creates clusters by grouping similar data points together





HIERARCHICAL CLUSTERING

Clustering Basics

K-Means
Clustering

Hierarchical
Clustering

DBSCAN

Comparing
Models

Hierarchical Clustering is a clustering technique that creates clusters by grouping similar data points together*

Here's how it works:

1. Within a scatter plot, find the 2 closest points, and group them into a cluster
2. Then find the next two closest points or clusters, and group them to a cluster
3. Repeat the process of combining the closest pairs of points or clusters until you eventually end up with one single cluster

This process is visualized using a tree diagram called a **dendrogram**, which shows the hierarchical relationship between clusters

*This is known as agglomerative or “bottom-up” clustering (vs. divisive or “top-down” clustering, which is much less common)



HIERARCHICAL CLUSTERING

Clustering Basics

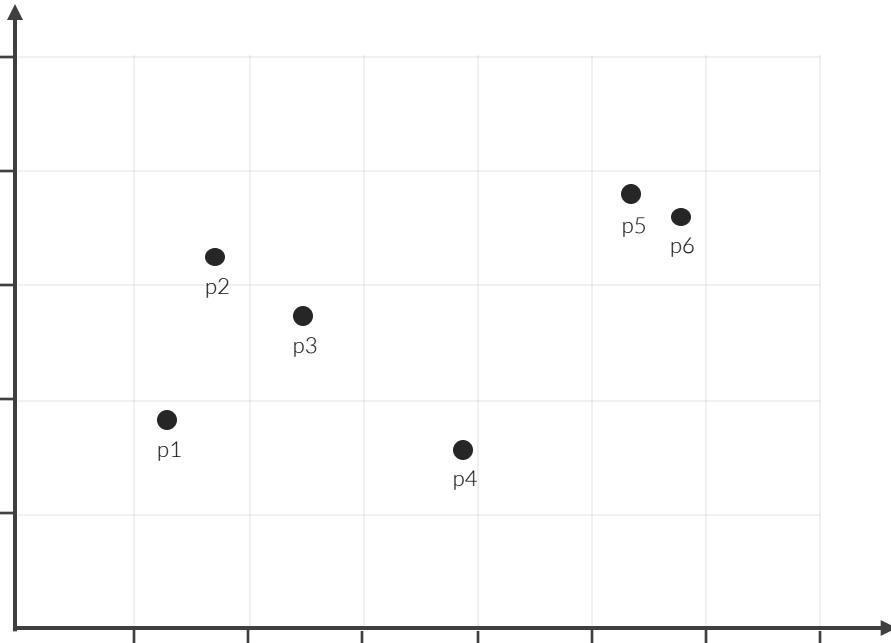
K-Means
Clustering

Hierarchical
Clustering

DBSCAN

Comparing
Models

STEP 1: Find the two closest points, and group them into a cluster



How do you define “closest”?



HIERARCHICAL CLUSTERING

Clustering Basics

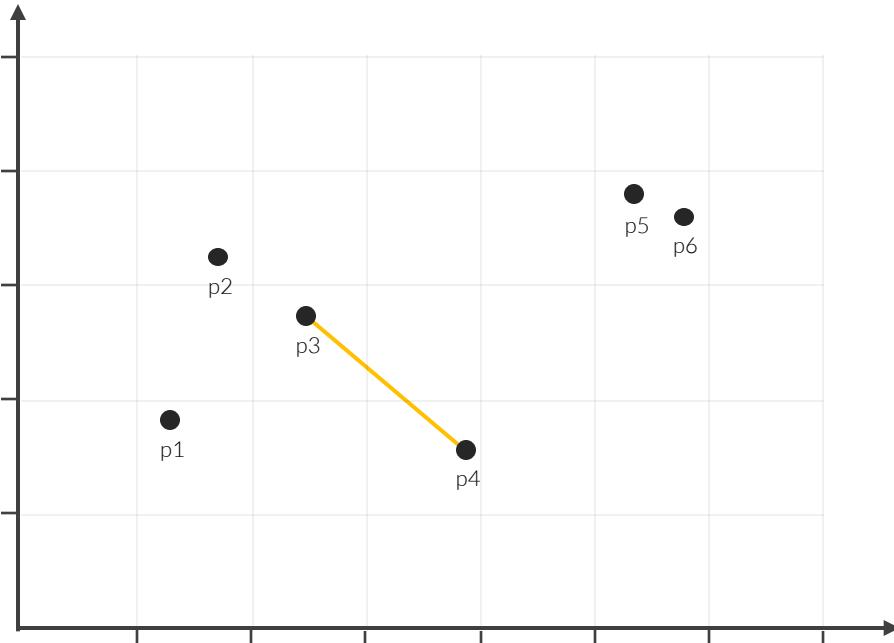
K-Means
Clustering

Hierarchical
Clustering

DBSCAN

Comparing
Models

STEP 1: Find the two closest points, and group them into a cluster



How do you define “closest”?

- Most commonly, the **Euclidean distance** is used



HIERARCHICAL CLUSTERING

Clustering Basics

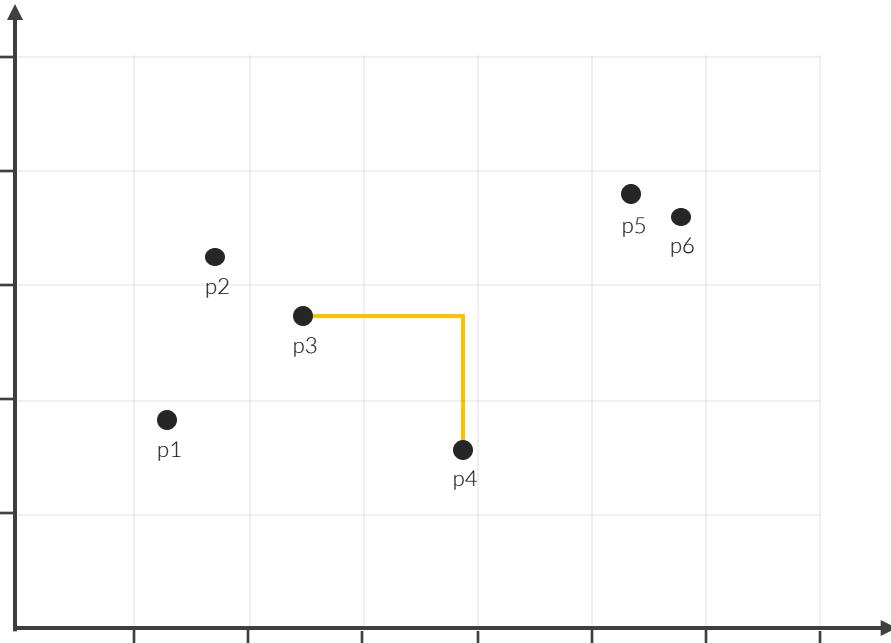
K-Means
Clustering

Hierarchical
Clustering

DBSCAN

Comparing
Models

STEP 1: Find the two closest points, and group them into a cluster



How do you define “closest”?

- Most commonly, the Euclidean distance is used
- Alternatively, there's **Manhattan distance**



HIERARCHICAL CLUSTERING

Clustering Basics

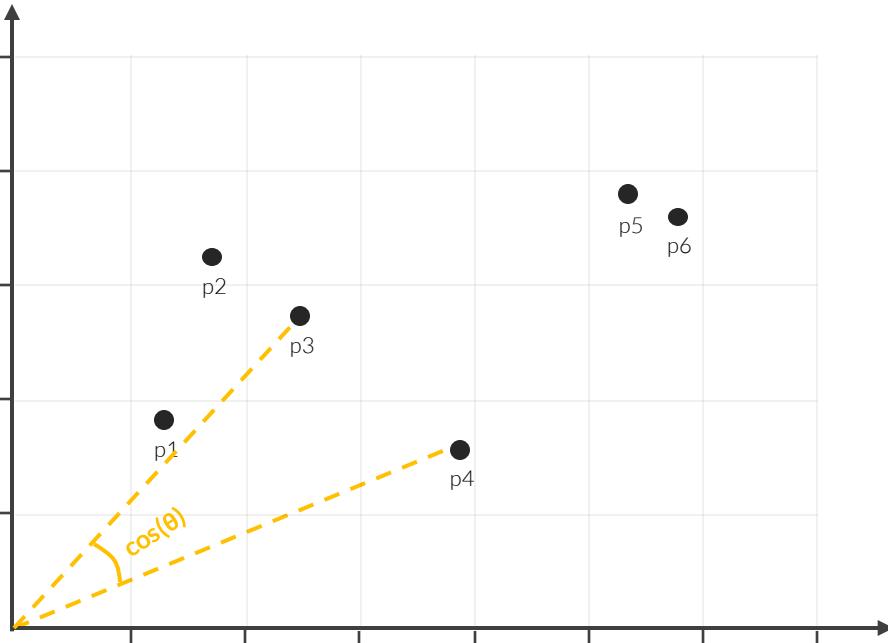
K-Means
Clustering

Hierarchical
Clustering

DBSCAN

Comparing
Models

STEP 1: Find the two closest points, and group them into a cluster



How do you define “closest”?

- Most commonly, the Euclidean distance is used
- Alternatively, there's Manhattan distance
- And **Cosine distance**



HIERARCHICAL CLUSTERING

Clustering Basics

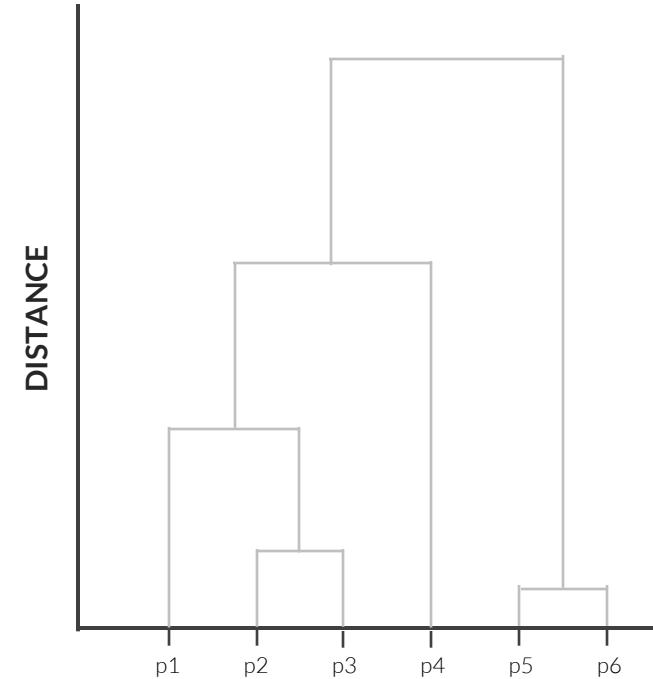
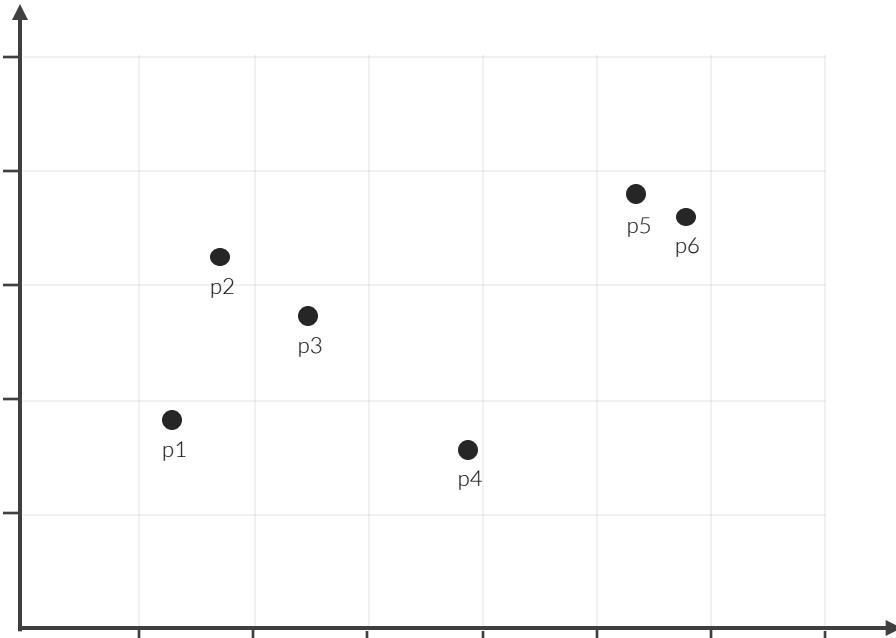
K-Means
Clustering

Hierarchical
Clustering

DBSCAN

Comparing
Models

STEP 1: Find the two closest points, and group them into a cluster





HIERARCHICAL CLUSTERING

Clustering Basics

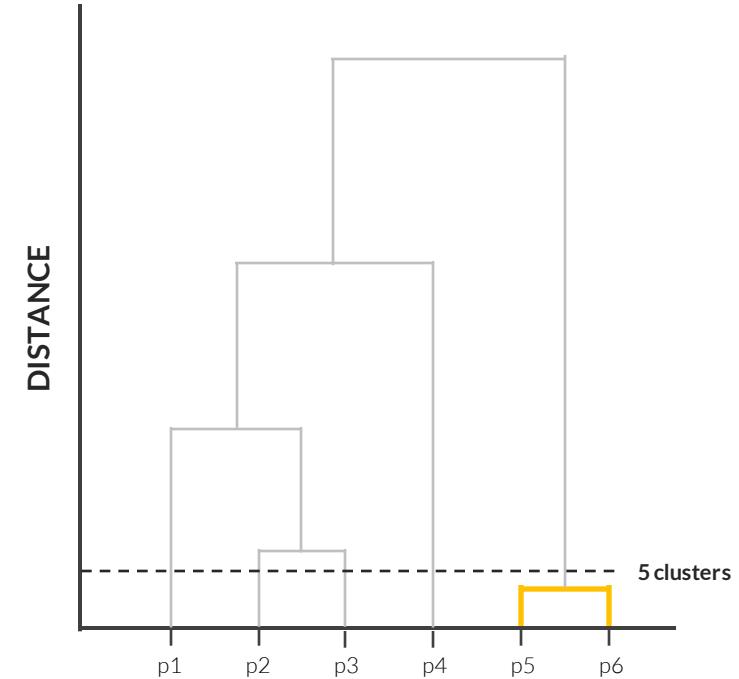
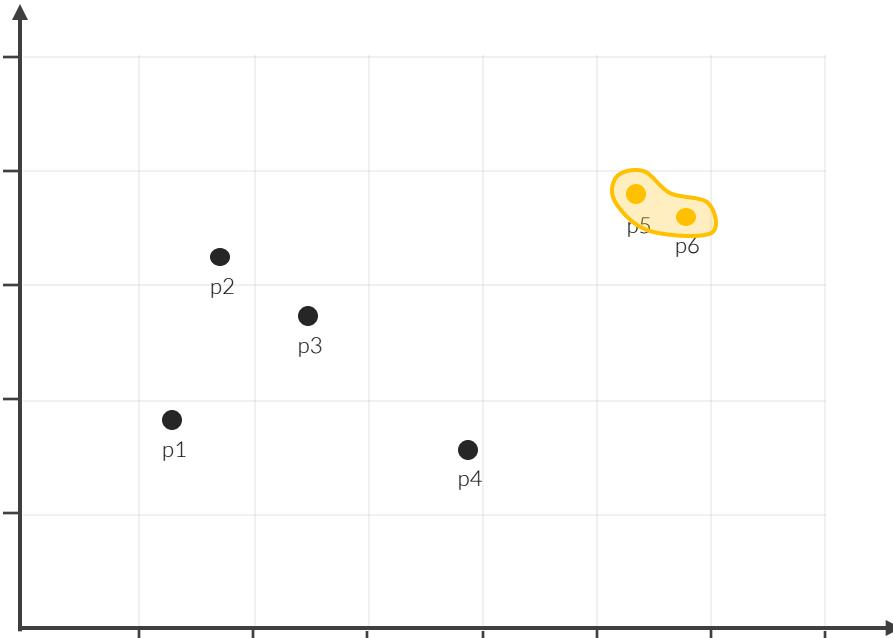
K-Means
Clustering

Hierarchical
Clustering

DBSCAN

Comparing
Models

STEP 1: Find the two closest points, and group them into a cluster





HIERARCHICAL CLUSTERING

Clustering Basics

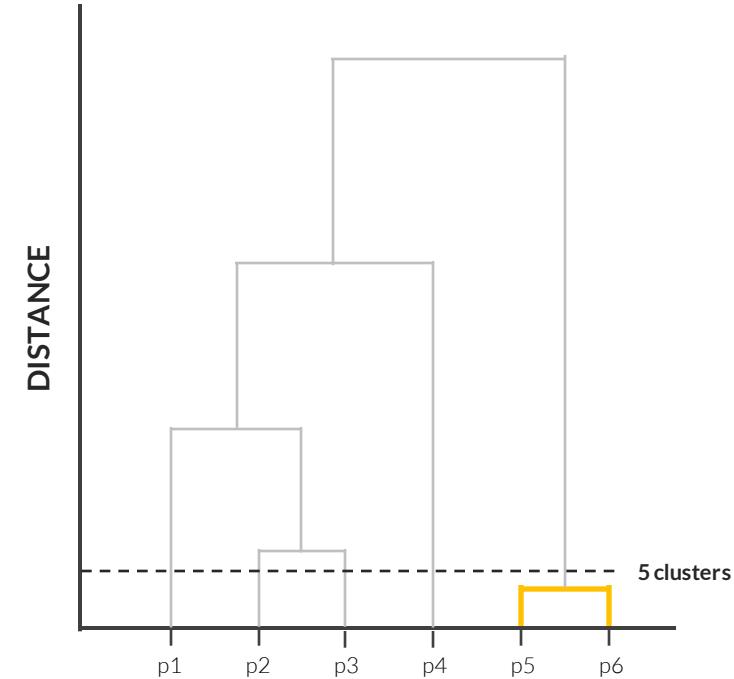
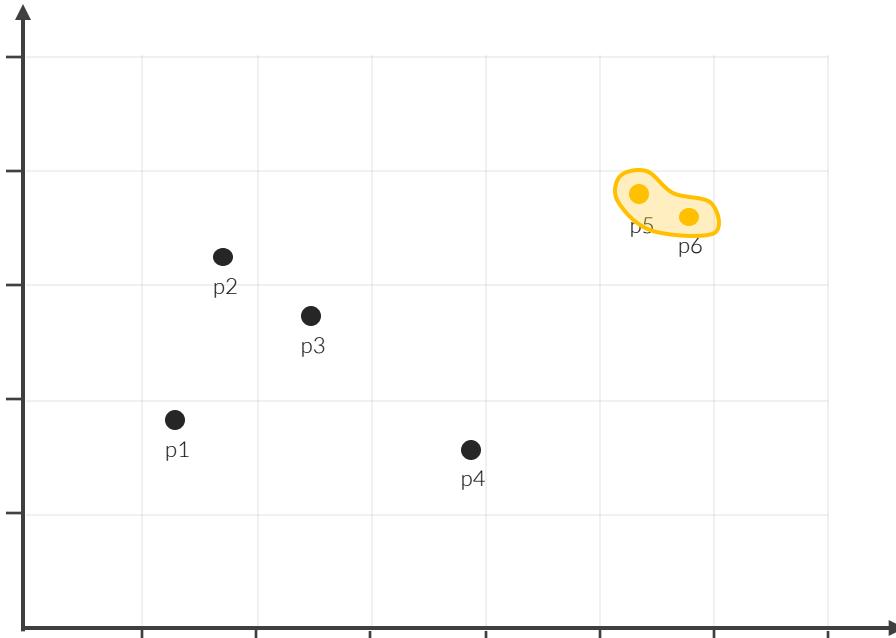
K-Means
Clustering

Hierarchical
Clustering

DBSCAN

Comparing
Models

STEP 2: Find the next two closest points/clusters, and group them together





HIERARCHICAL CLUSTERING

Clustering Basics

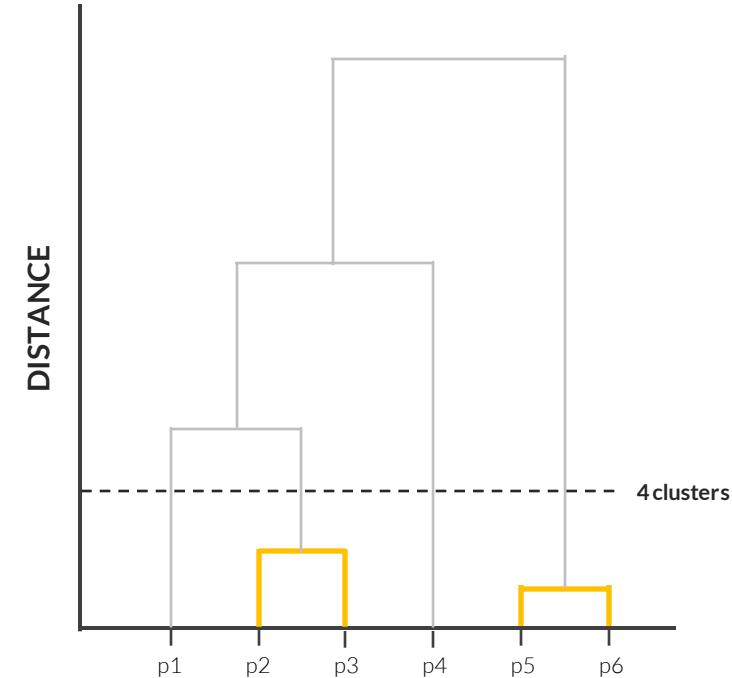
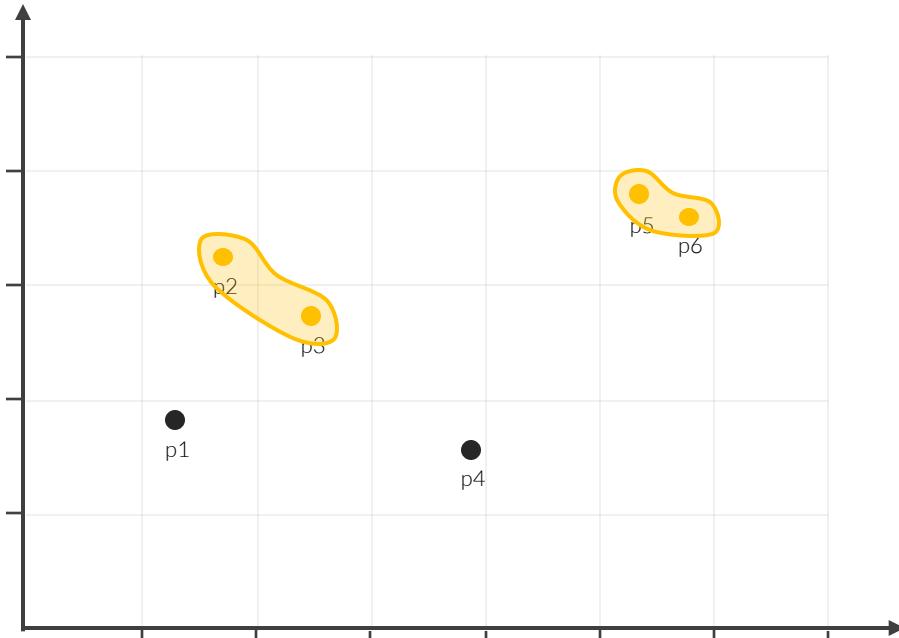
K-Means
Clustering

Hierarchical
Clustering

DBSCAN

Comparing
Models

STEP 2: Find the next two closest points/clusters, and group them together





HIERARCHICAL CLUSTERING

Clustering Basics

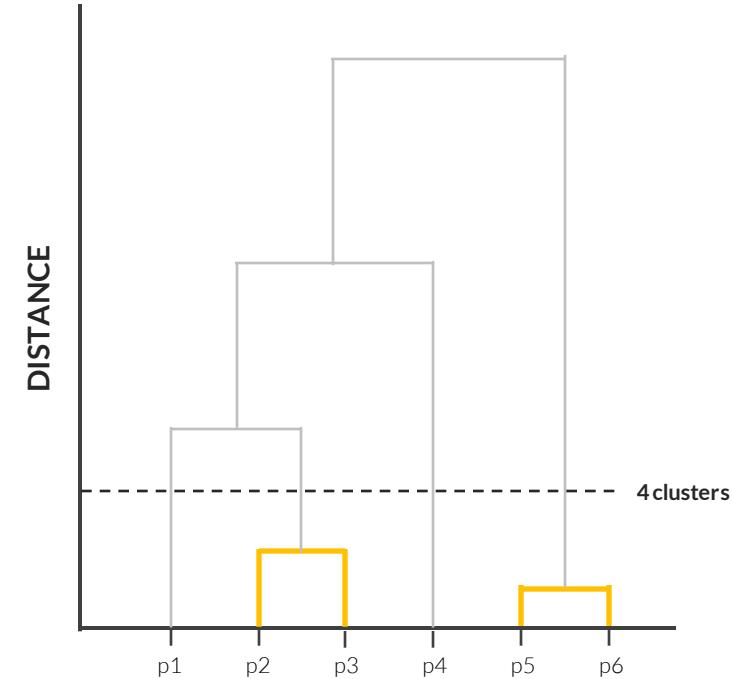
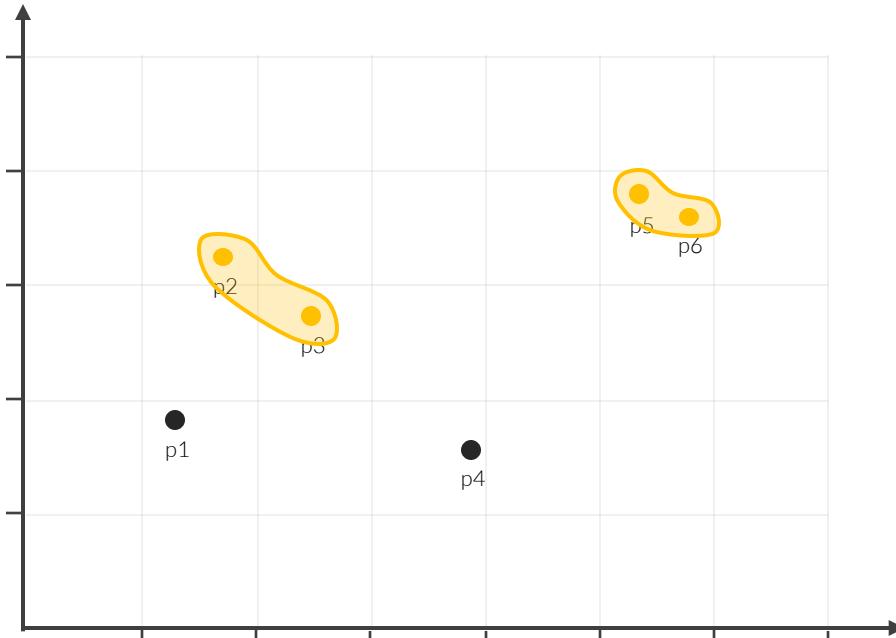
K-Means
Clustering

Hierarchical
Clustering

DBSCAN

Comparing
Models

STEP 3: Repeat the process until all points are part of the same cluster





HIERARCHICAL CLUSTERING

Clustering Basics

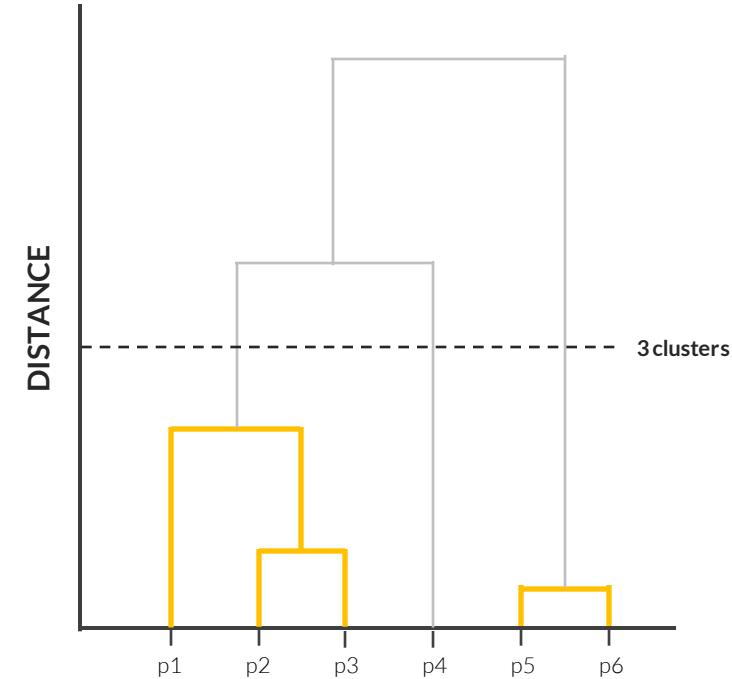
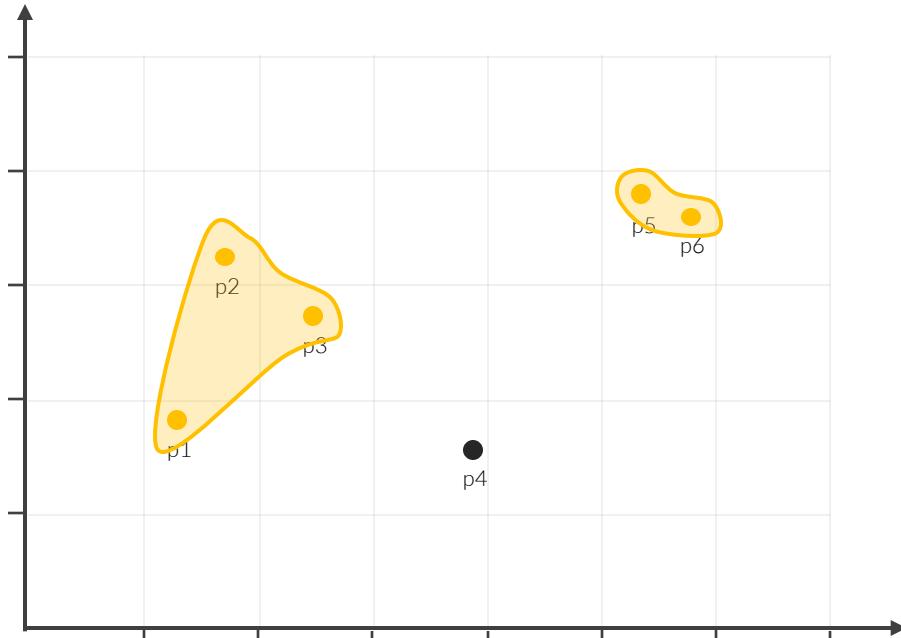
K-Means
Clustering

Hierarchical
Clustering

DBSCAN

Comparing
Models

STEP 3: Repeat the process until all points are part of the same cluster





HIERARCHICAL CLUSTERING

Clustering Basics

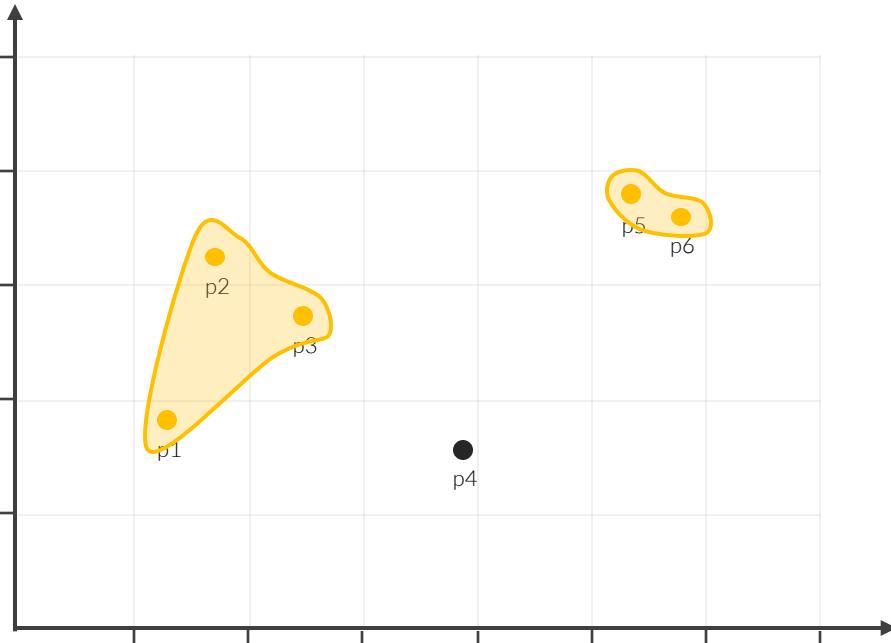
K-Means
Clustering

Hierarchical
Clustering

DBSCAN

Comparing
Models

STEP 3: Repeat the process until all points are part of the same cluster



How do you define distance between *clusters*?



HIERARCHICAL CLUSTERING

Clustering Basics

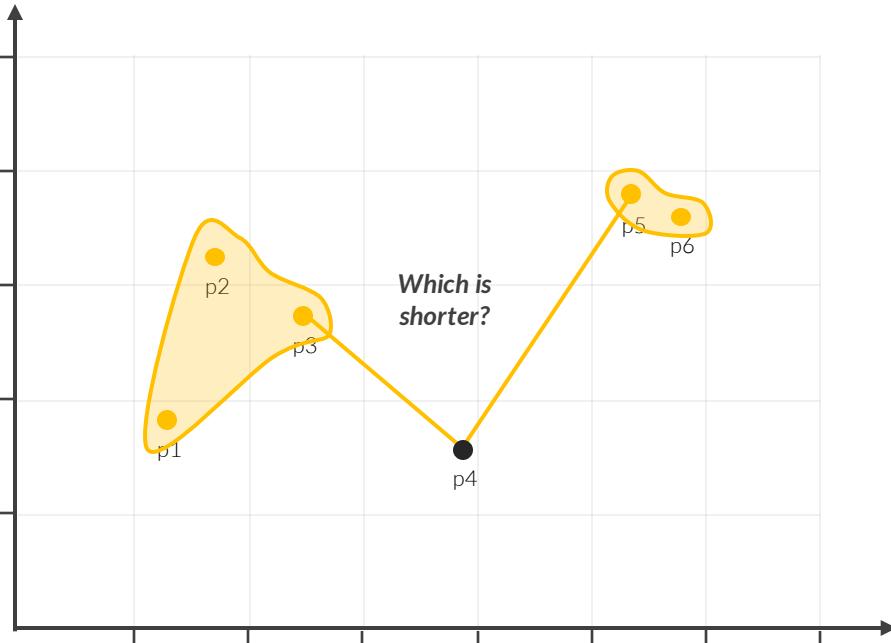
K-Means
Clustering

Hierarchical
Clustering

DBSCAN

Comparing
Models

STEP 3: Repeat the process until all points are part of the same cluster



How do you define distance between *clusters*?

- **Single** linkage (closest)



HIERARCHICAL CLUSTERING

Clustering Basics

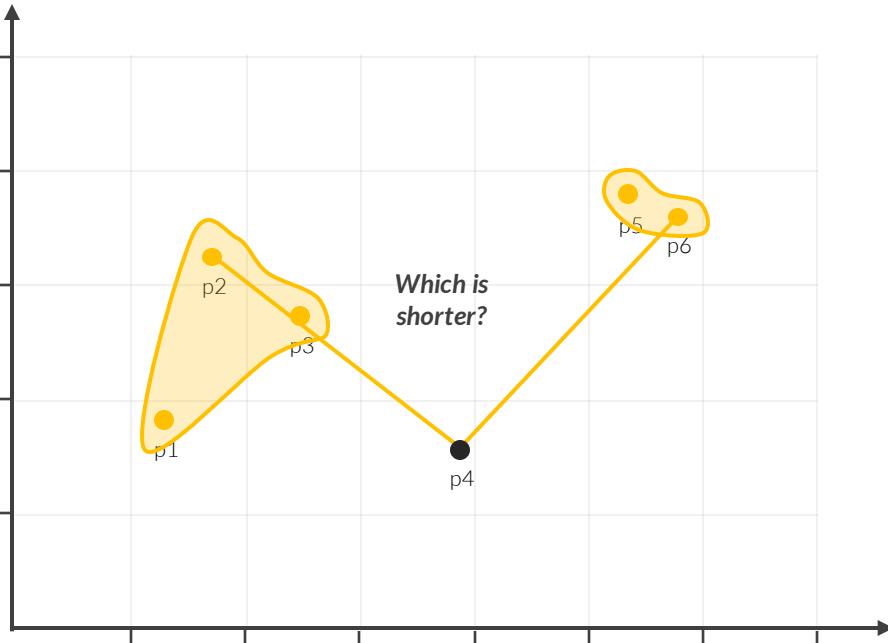
K-Means
Clustering

Hierarchical
Clustering

DBSCAN

Comparing
Models

STEP 3: Repeat the process until all points are part of the same cluster



How do you define distance between *clusters*?

- Single linkage (closest)
- **Complete** linkage (furthest)



HIERARCHICAL CLUSTERING

Clustering Basics

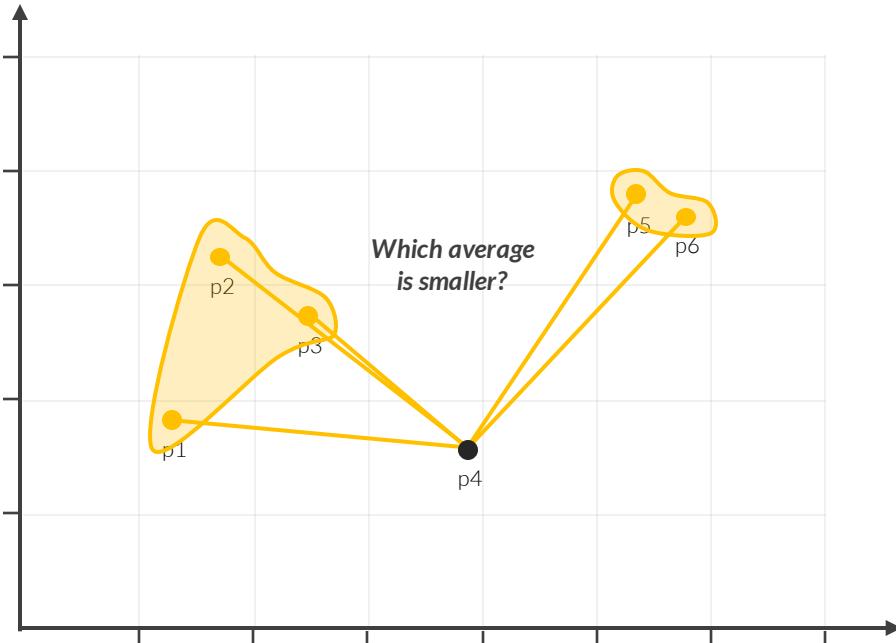
K-Means
Clustering

Hierarchical
Clustering

DBSCAN

Comparing
Models

STEP 3: Repeat the process until all points are part of the same cluster



How do you define distance between *clusters*?

- Single linkage (closest)
- Complete linkage (furthest)
- **Average** linkage (all pairs)



HIERARCHICAL CLUSTERING

Clustering Basics

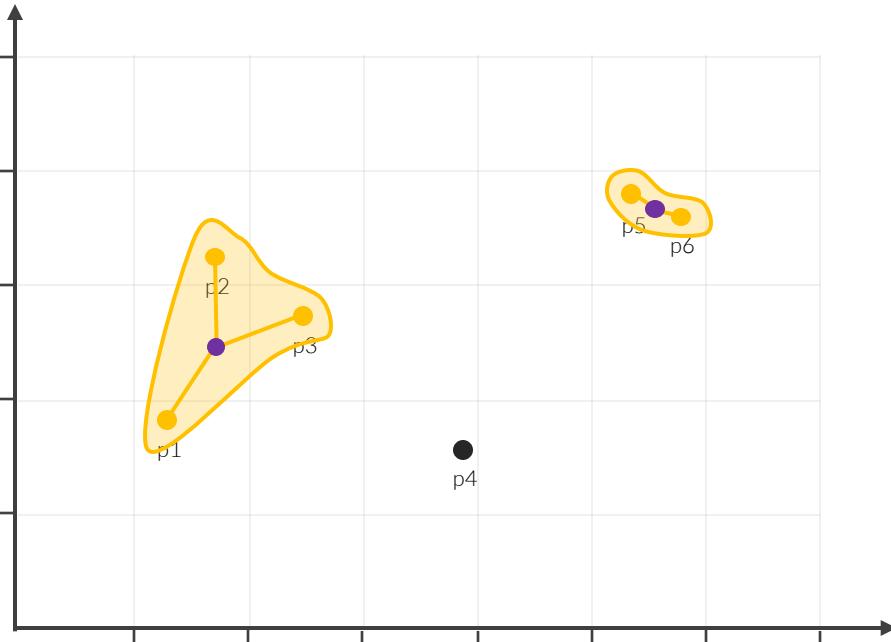
K-Means
Clustering

Hierarchical
Clustering

DBSCAN

Comparing
Models

STEP 3: Repeat the process until all points are part of the same cluster



How do you define distance between *clusters*?

- Single linkage (closest)
- Complete linkage (furthest)
- Average linkage (all pairs)
- **Ward's** method (variance)



HIERARCHICAL CLUSTERING

Clustering Basics

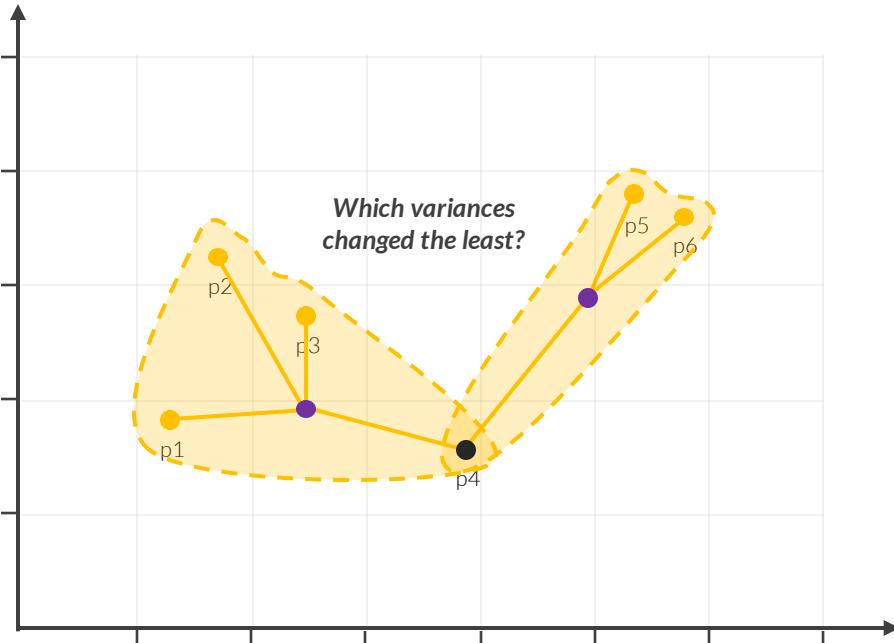
K-Means
Clustering

Hierarchical
Clustering

DBSCAN

Comparing
Models

STEP 3: Repeat the process until all points are part of the same cluster



How do you define distance between *clusters*?

- Single linkage (closest)
- Complete linkage (furthest)
- Average linkage (all pairs)
- **Ward's** method (variance)



HIERARCHICAL CLUSTERING

Clustering Basics

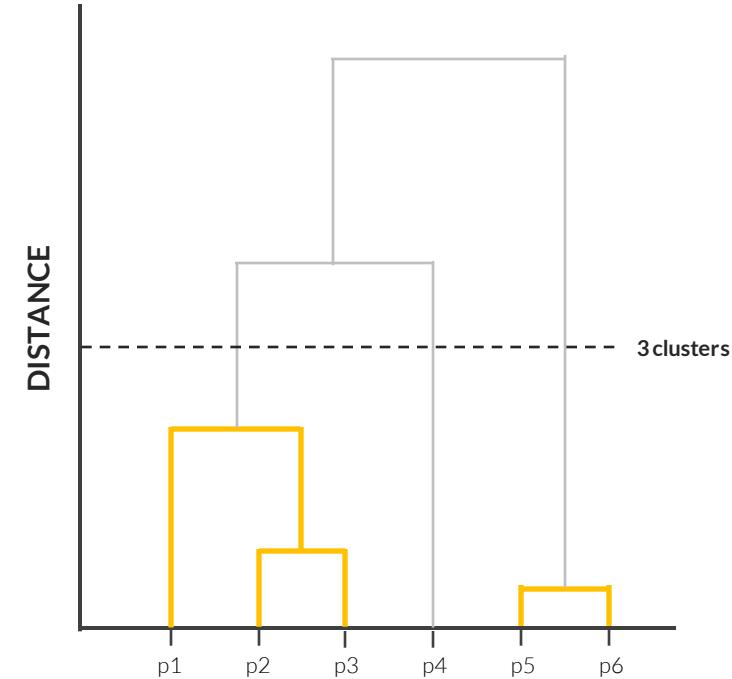
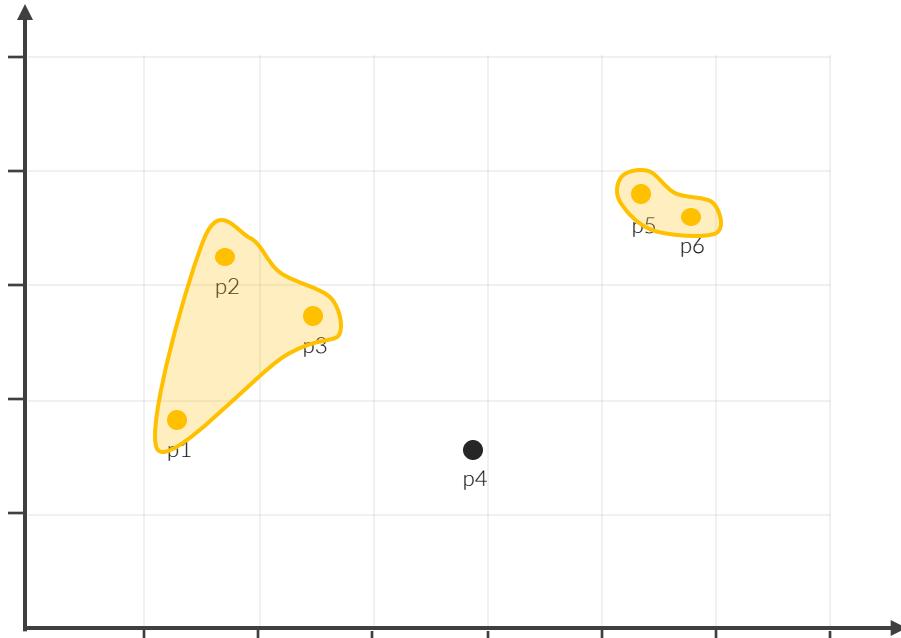
K-Means
Clustering

Hierarchical
Clustering

DBSCAN

Comparing
Models

STEP 3: Repeat the process until all points are part of the same cluster





HIERARCHICAL CLUSTERING

Clustering Basics

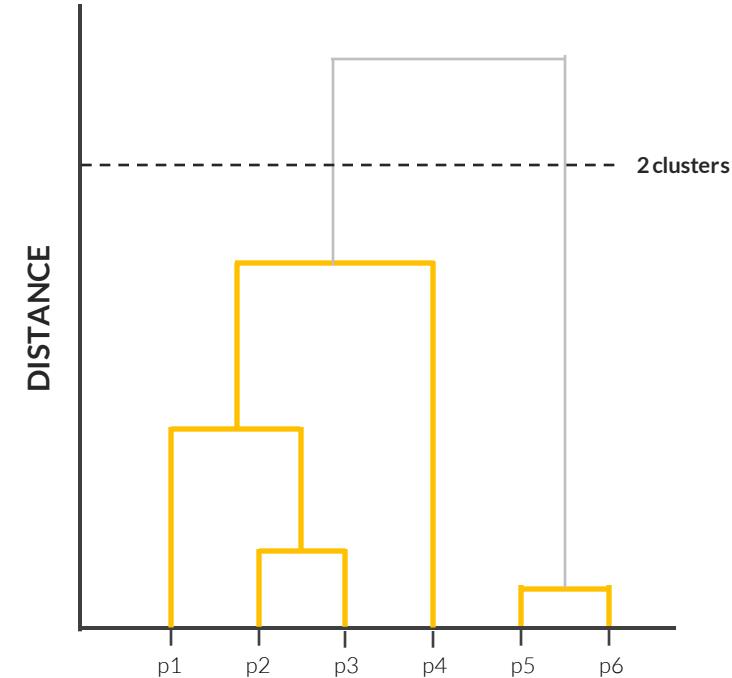
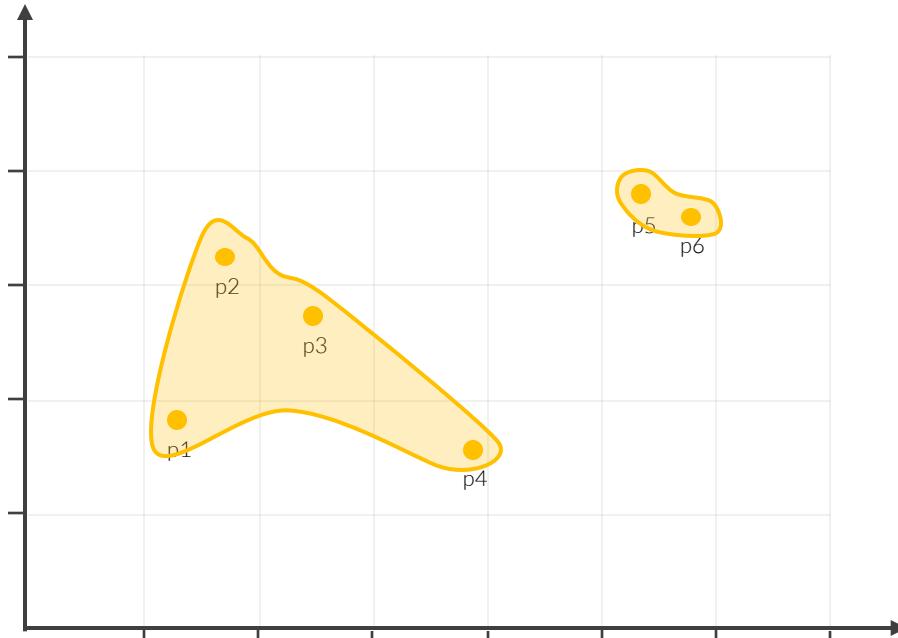
K-Means
Clustering

Hierarchical
Clustering

DBSCAN

Comparing
Models

STEP 3: Repeat the process until all points are part of the same cluster





HIERARCHICAL CLUSTERING

Clustering Basics

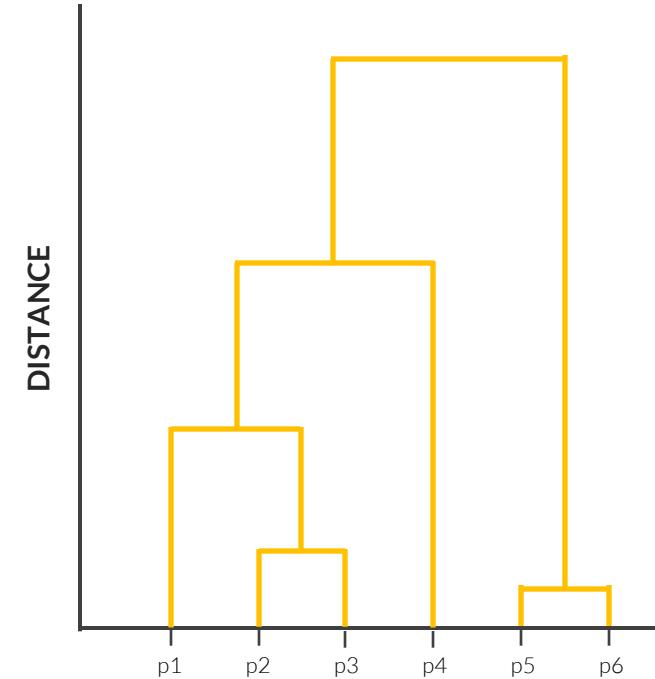
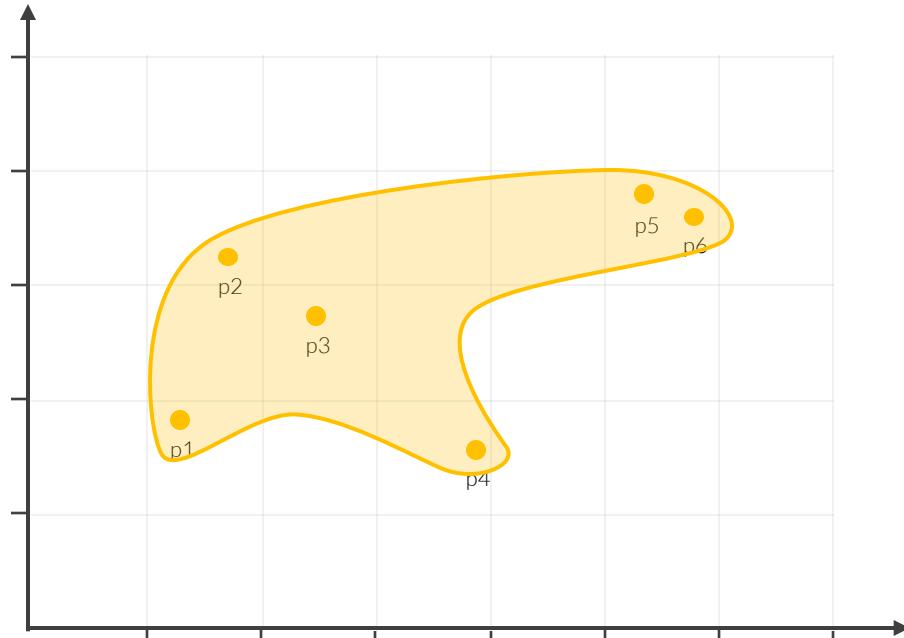
K-Means
Clustering

Hierarchical
Clustering

DBSCAN

Comparing
Models

STEP 3: Repeat the process until all points are part of the same cluster





DENDROGRAMS IN PYTHON

Clustering Basics

K-Means
Clustering

Hierarchical
Clustering

DBSCAN

Comparing
Models

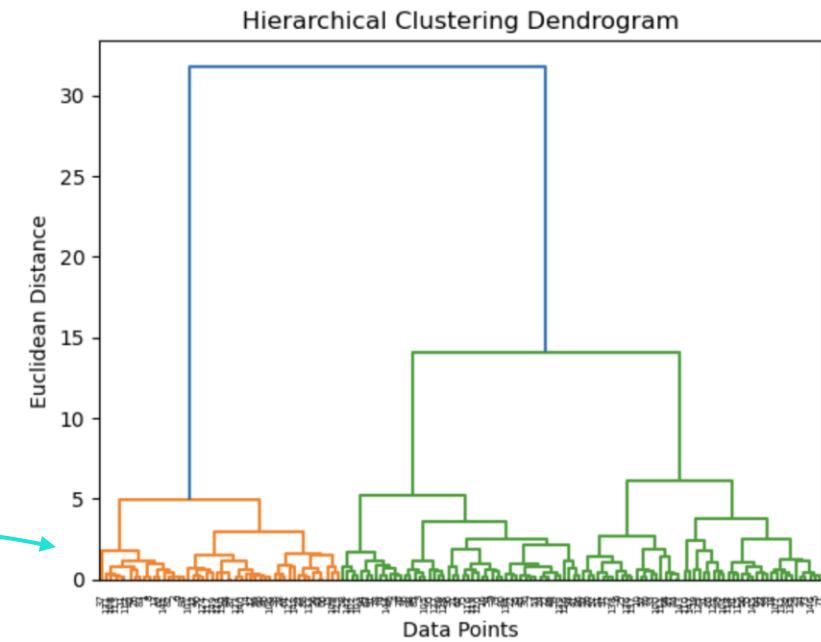
The **dendrogram()** function within the `scipy` library in Python allows you to visualize Hierarchical Clustering clusters

```
# visualize the clusters with a dendrogram
from scipy.cluster.hierarchy import linkage, dendrogram
import matplotlib.pyplot as plt

linkage_matrix = linkage(data, method='ward')
dendrogram_info = dendrogram(linkage_matrix)

plt.title("Hierarchical Clustering Dendrogram")
plt.xlabel("Data Points")
plt.ylabel("Euclidean Distance");
```

These colors are set by default, but you can set the `color_threshold` argument to update them for a specific number of clusters



There seem to be 3 clusters in the dendrogram, so let's update the `color_threshold` to 3



AGGLOMERATIVE CLUSTERING IN PYTHON

Clustering Basics

K-Means
Clustering

Hierarchical
Clustering

DBSCAN

Comparing
Models

The **AgglomerativeClustering()** function within sklearn's cluster module allows you to perform Hierarchical Clustering in Python

```
from sklearn.cluster import AgglomerativeClustering  
  
agg = AgglomerativeClustering(n_clusters=2, metric='euclidean', linkage='ward')
```

The number of clusters to identify (default is **2**)

The method used to measure the distance between points (default is "**euclidean**")

The method used to measure the distance between clusters (default is "**ward**")

Other distances:

- "manhattan"
- "cosine"
- "precomputed"

Other methods:

- "single"
- "complete"
- "average"



PRO TIP: While you can tune these parameters, the defaults are by far the most common



AGGLOMERATIVE CLUSTERING IN PYTHON

Clustering Basics

K-Means
Clustering

Hierarchical
Clustering

DBSCAN

Comparing
Models

The **AgglomerativeClustering()** function within sklearn's cluster module allows you to perform Hierarchical Clustering in Python

```
# import agglomerative clustering from sklearn
from sklearn.cluster import AgglomerativeClustering
```

```
# fit an agglomerative clustering model with 3 clusters
agg = AgglomerativeClustering(3)
agg.fit(data)
```

▼ AgglomerativeClustering

AgglomerativeClustering(n_clusters=3)

Once the model is fit, you can view the cluster assignments using the `.labels_` attribute

```
# view the cluster assignments
agg.labels_
```

```
array([1, 1, 1, 2, 2, 0, 0, 2, 1, 2, 0, 1, 1, 1, 1, 1, 1, 2, 1, 0, 0, 0, 2,
       2, 0, 2, 0, 1, 1, 1, 0, 2, 0, 0, 0, 0, 0, 2, 2, 2, 1, 1, 0, 2, 2, 1, 1,
       0, 2, 1, 0, 1, 0, 0, 0, 2, 2, 0, 1, 0, 2, 2, 2, 1, 1, 1, 2, 0, 0, 0,
       1, 2, 0, 2, 0, 0, 2, 2, 2, 2, 0, 2, 2, 2, 0, 1, 1, 1, 1, 2, 2, 1,
       1, 2, 1, 2, 0, 2, 2, 2, 2, 0, 2, 1, 0, 1, 2, 2, 0, 2, 1, 0, 1, 1, 1,
       0, 0, 1, 1, 1, 0, 2, 1, 0, 2, 2, 0, 1, 1, 1, 0, 2, 2, 2, 1, 1, 1,
       0, 1, 2, 0, 0, 0, 2, 1, 1, 2, 0, 2, 0, 0, 1, 1, 1, 0, 2, 2, 2, 1, 1, 1])
```



CLUSTER MAPS IN PYTHON

Clustering Basics

K-Means
Clustering

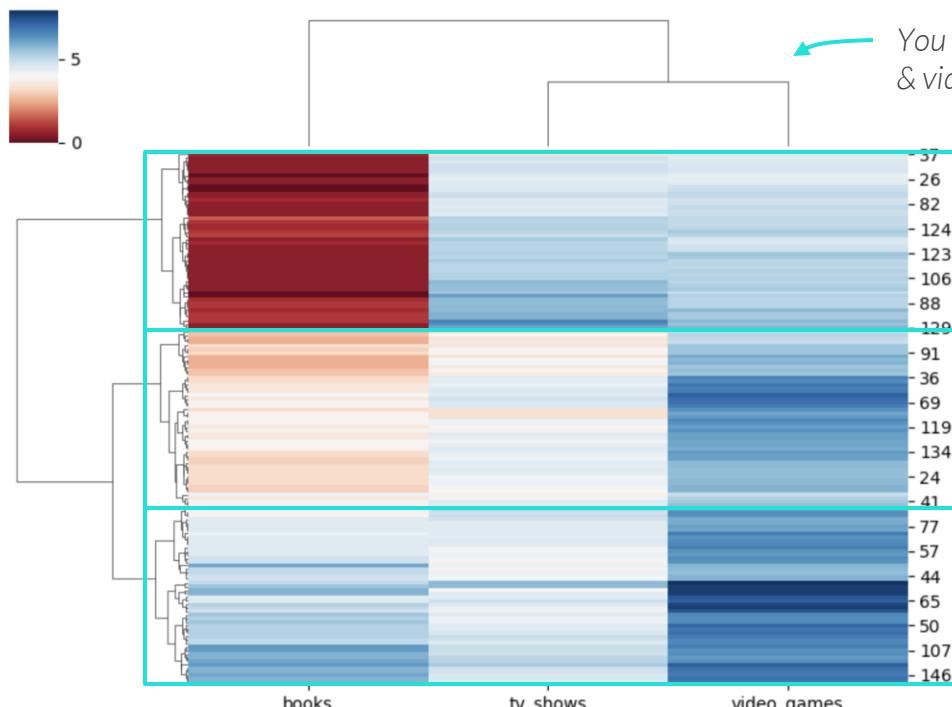
Hierarchical
Clustering

DBSCAN

Comparing
Models

Seaborn's **clustermap()** function adds a heatmap on top of the dendrogram, which helps visually interpret each cluster

```
# create a cluster map
sns.clustermap(data, method='ward', cmap='RdBu', figsize=(8, 6), xticklabels=data.columns)
plt.show()
```



You also get the relationships between features (tv & video game hours are more closely related!)

These students don't read many books

These students consume an average amount of each type of entertainment

These students play lots of video games

ASSIGNMENT: HIERARCHICAL CLUSTERING



NEW MESSAGE

March 18, 2024

From: **Clyde Clusters** (Sr. Data Scientist)
Subject: **Hierarchical Clustering Help**

Hi again!

Thanks for your help applying K-Means to the cereal data set.

To get another perspective, please create dendograms for both the original and standardized cereal data sets and let me know how many clusters you can visually detect in each!

Finally, fit a hierarchical model on the "best" results from the standardized data set, visualize the clusters using a cluster map and interpret the clusters.

Thanks!
Clyde

Reply

Forward

Key Objectives

1. Create a dendrogram using the 5 numeric fields of the cereal data set
2. Visually identify the "best" number of clusters and adjust the color_threshold
3. Create a dendrogram using the 4 standardized fields (excluding "Fat") of the cereal data set
4. Visually identify the "best" number of clusters and adjust the color_threshold
5. Fit a hierarchical clustering model on the "best" results from the last step and view the labels
6. Create a cluster map of the "best" results and interpret the clusters



DBSCAN

Clustering Basics

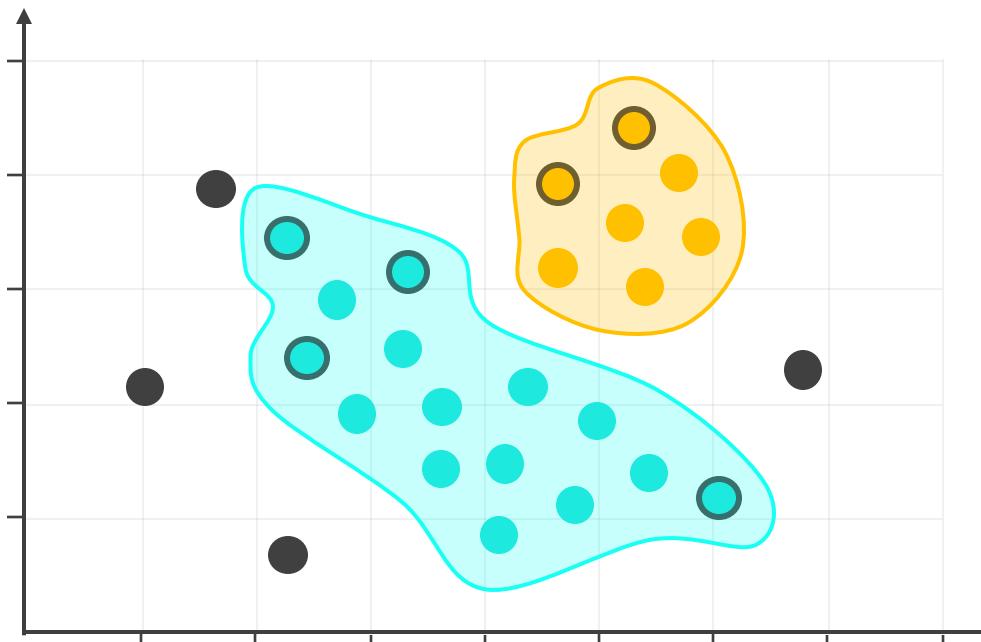
K-Means
Clustering

Hierarchical
Clustering

DBSCAN

Comparing
Models

DBSCAN (Density-Based Spatial Clustering of Applications with Noise) is a clustering technique that identifies clusters based on the density of data points





DBSCAN

Clustering Basics

K-Means
Clustering

Hierarchical
Clustering

DBSCAN

Comparing
Models

DBSCAN (Density-Based Spatial Clustering of Applications with Noise) is a clustering technique that identifies clusters based on the density of data points

Here's how it works*:

1. Select a radius (`eps`) and a minimum number of points (`min_samples`)
2. Within a scatter plot, label each point as one of the following:
 - **Core point** – has the minimum number of points within its radius (*in a dense region*)
 - **Border point** – does not have the minimum number of points within its radius, but has at least one core point within its radius (*on the outskirts of clusters*)
 - **Noise point** (outlier) – does not have a core point within its radius (*isolated points*)

DBSCAN allows for irregular-shaped clusters and can also identify outliers

*This is a rough summary of the steps – more detailed steps are included at the end of the DBSCAN section



DBSCAN

Clustering Basics

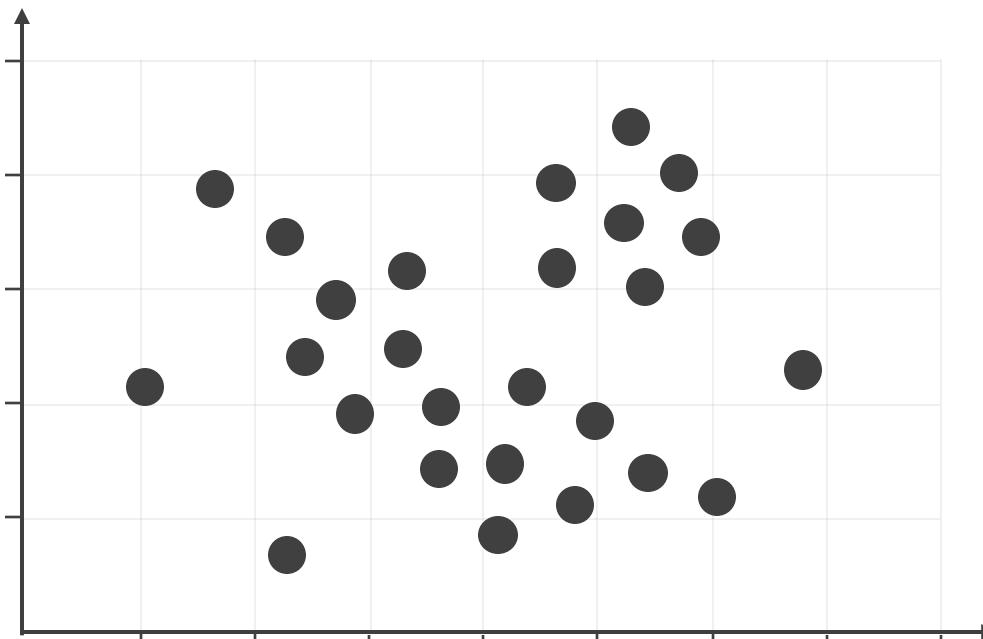
K-Means
Clustering

Hierarchical
Clustering

DBSCAN

Comparing
Models

STEP 1: Select a radius (eps) and minimum number of points (min_samples)



$\text{eps} = 0.75$
 $\text{min_samples} = 4$



DBSCAN

Clustering Basics

K-Means
Clustering

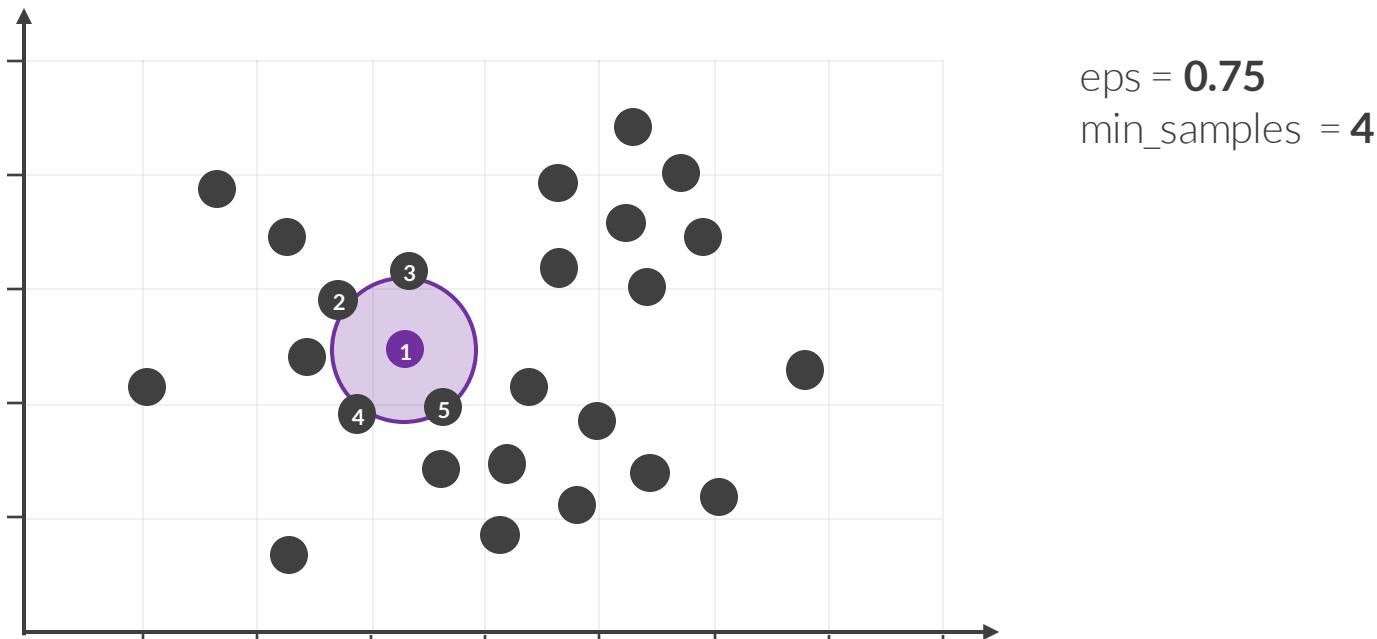
Hierarchical
Clustering

DBSCAN

Comparing
Models

STEP 2: Select a point at random and count the points within its radius

- If it's greater than or equal to the "min_samples", then start a cluster, label the point as a *core point*, and mark the points within its radius as a *neighbor*





DBSCAN

Clustering Basics

K-Means
Clustering

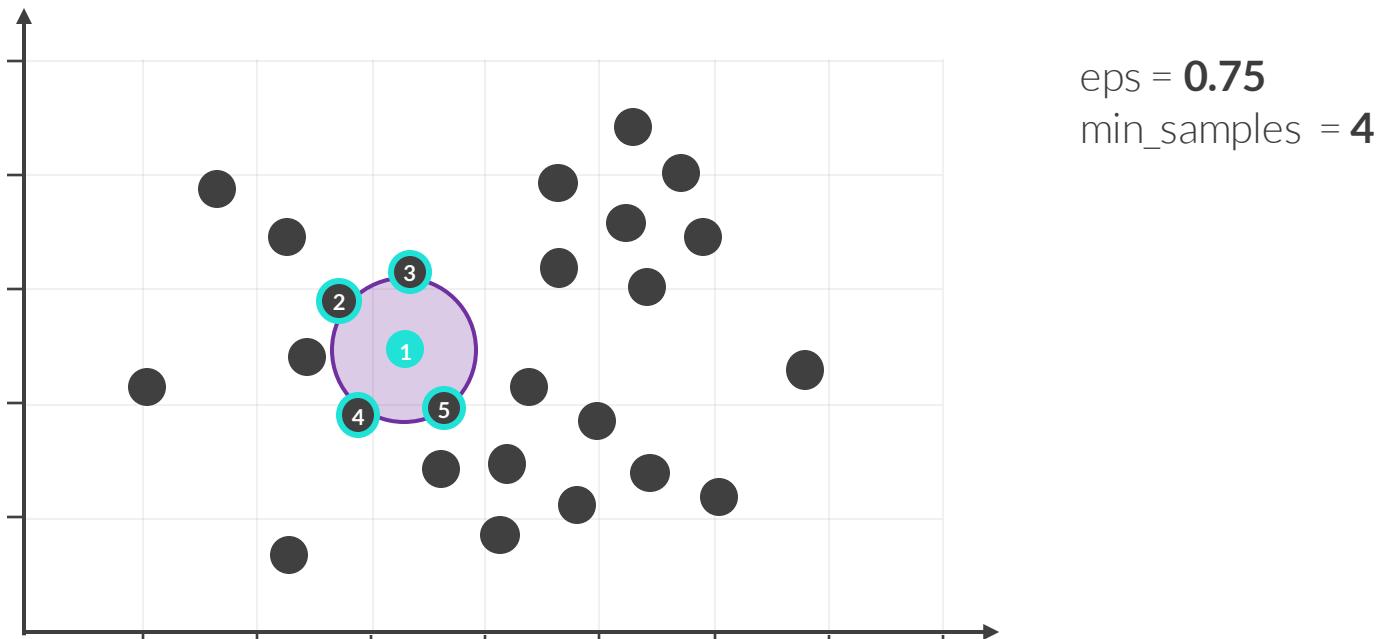
Hierarchical
Clustering

DBSCAN

Comparing
Models

STEP 2: Select a point at random and count the points within its radius

- If it's greater than or equal to the "min_samples", then start a cluster, label the point as a *core point*, and mark the points within its radius as a *neighbor*





DBSCAN

Clustering Basics

K-Means
Clustering

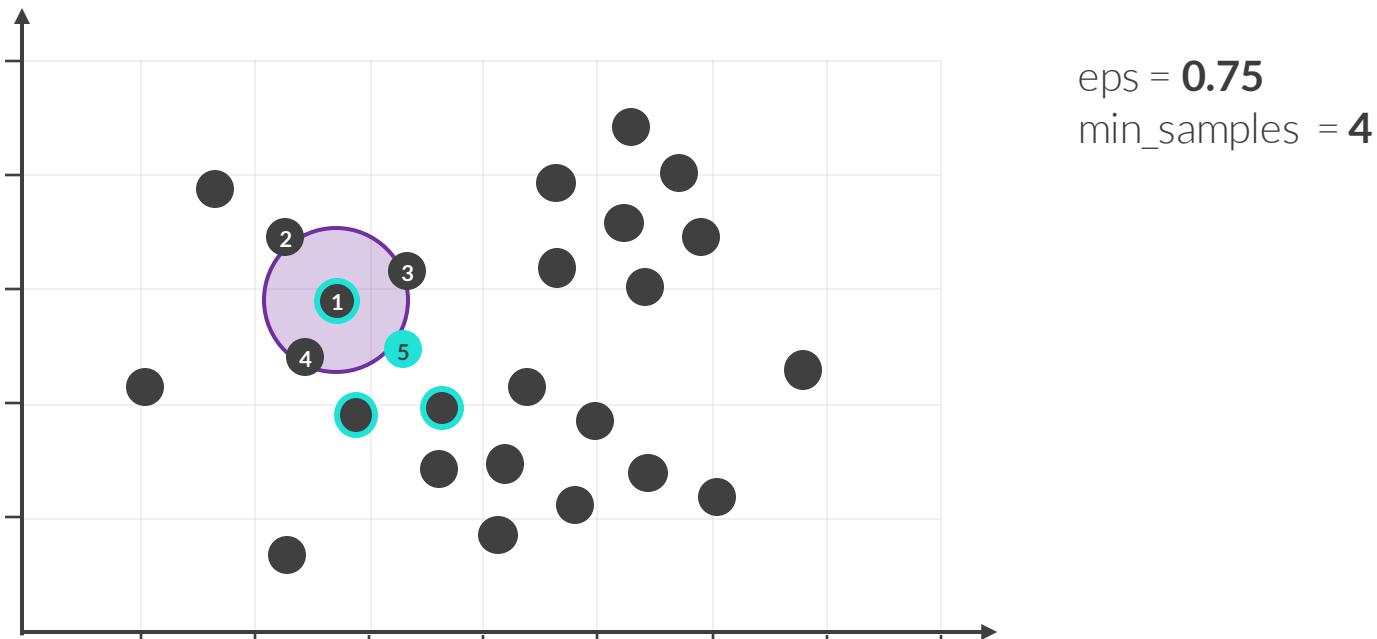
Hierarchical
Clustering

DBSCAN

Comparing
Models

STEP 3: Move to a neighbor and count the points within its radius

- If it's greater than or equal to the "min_samples", label it as a *core point*, and mark its *neighbors*





DBSCAN

Clustering Basics

K-Means
Clustering

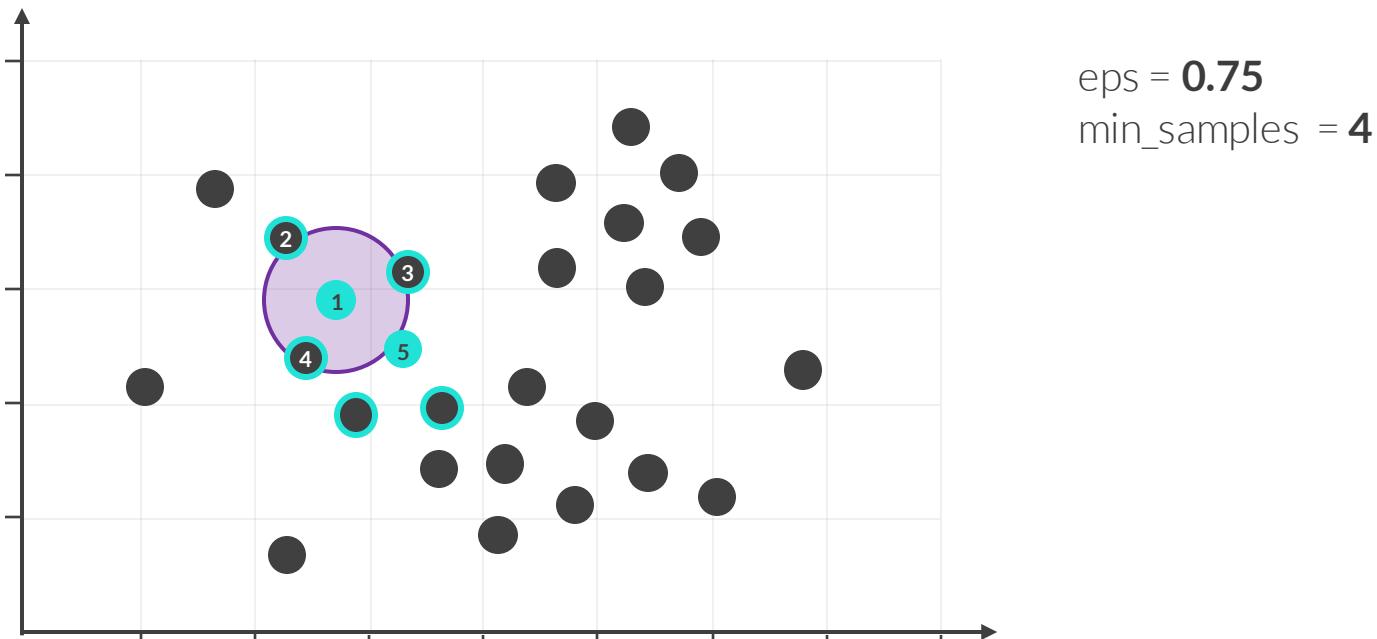
Hierarchical
Clustering

DBSCAN

Comparing
Models

STEP 3: Move to a neighbor and count the points within its radius

- If it's greater than or equal to the "min_samples", label it as a *core point*, and mark its *neighbors*





DBSCAN

Clustering Basics

K-Means
Clustering

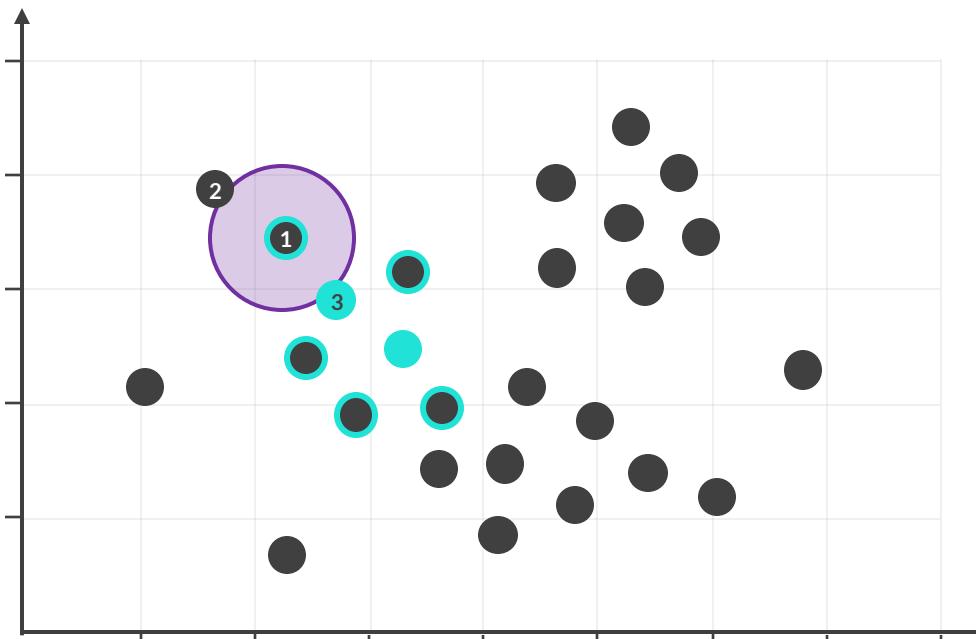
Hierarchical
Clustering

DBSCAN

Comparing
Models

STEP 3: Move to a neighbor and count the points within its radius

- If it's less than "min_samples", but at least one of them is a core point, label it as a *border point*



eps = **0.75**
min_samples = **4**



DBSCAN

Clustering Basics

K-Means
Clustering

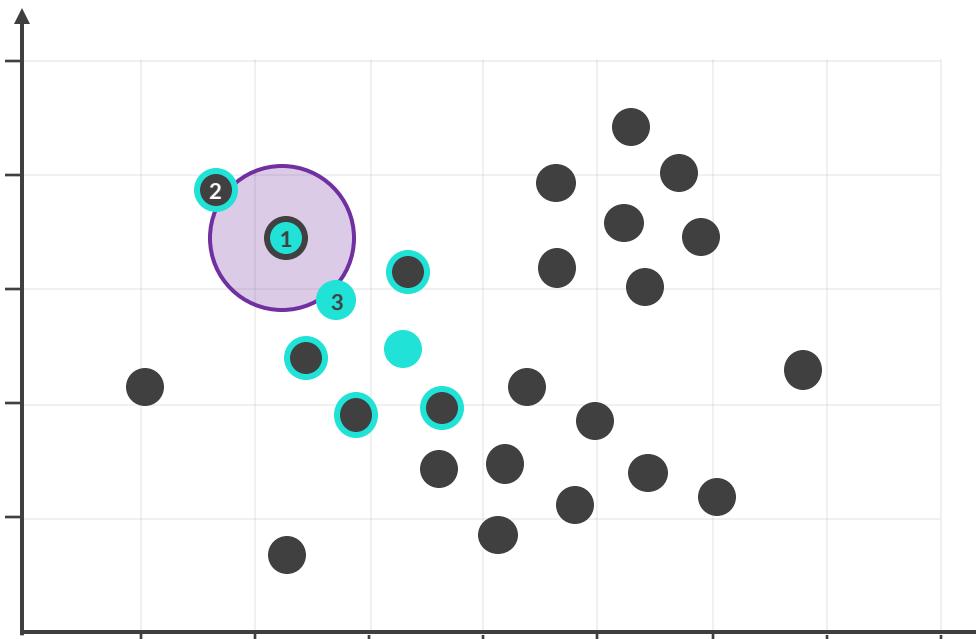
Hierarchical
Clustering

DBSCAN

Comparing
Models

STEP 3: Move to a neighbor and count the points within its radius

- If it's less than "min_samples", but at least one of them is a core point, label it as a *border point*



eps = **0.75**
min_samples = **4**



DBSCAN

Clustering Basics

K-Means
Clustering

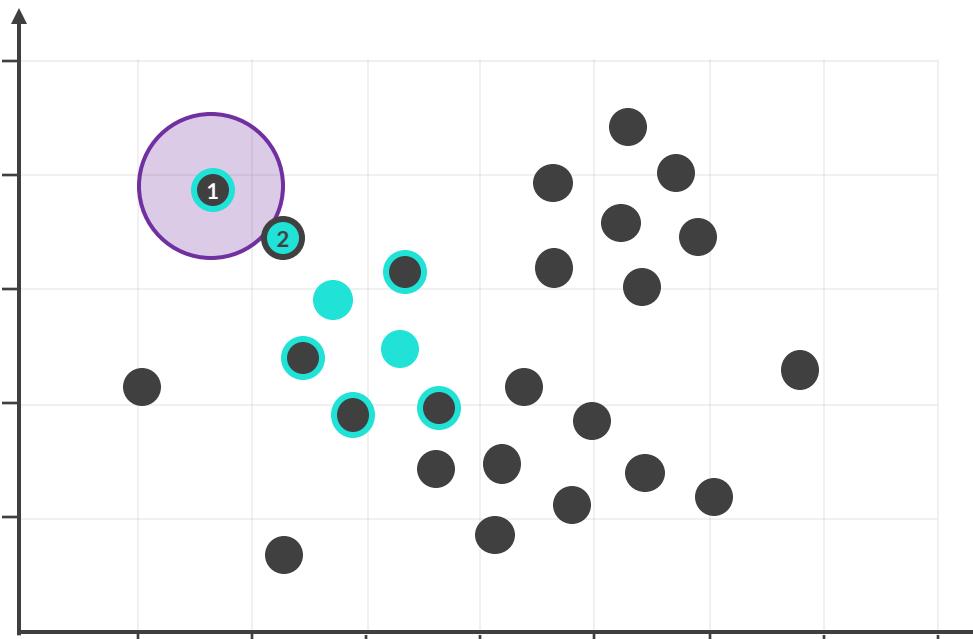
Hierarchical
Clustering

DBSCAN

Comparing
Models

STEP 3: Move to a neighbor and count the points within its radius

- If it's less than "min_samples", and none of them is a core point, label it as a *noise point*



eps = **0.75**
min_samples = **4**



DBSCAN

Clustering Basics

K-Means
Clustering

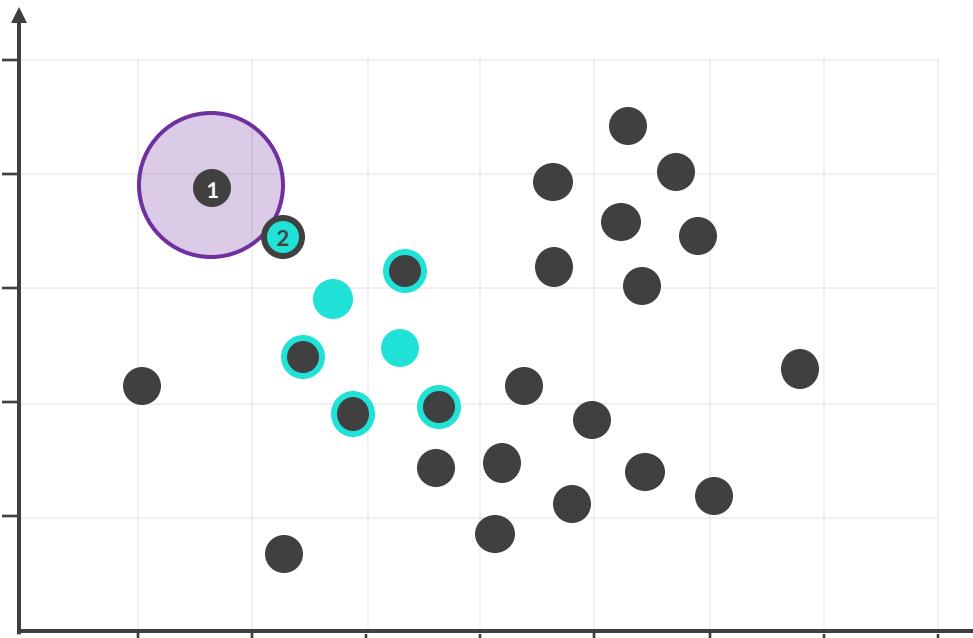
Hierarchical
Clustering

DBSCAN

Comparing
Models

STEP 3: Move to a neighbor and count the points within its radius

- If it's less than "min_samples", and none of them is a core point, label it as a *noise point*



eps = **0.75**
min_samples = **4**



DBSCAN

Clustering Basics

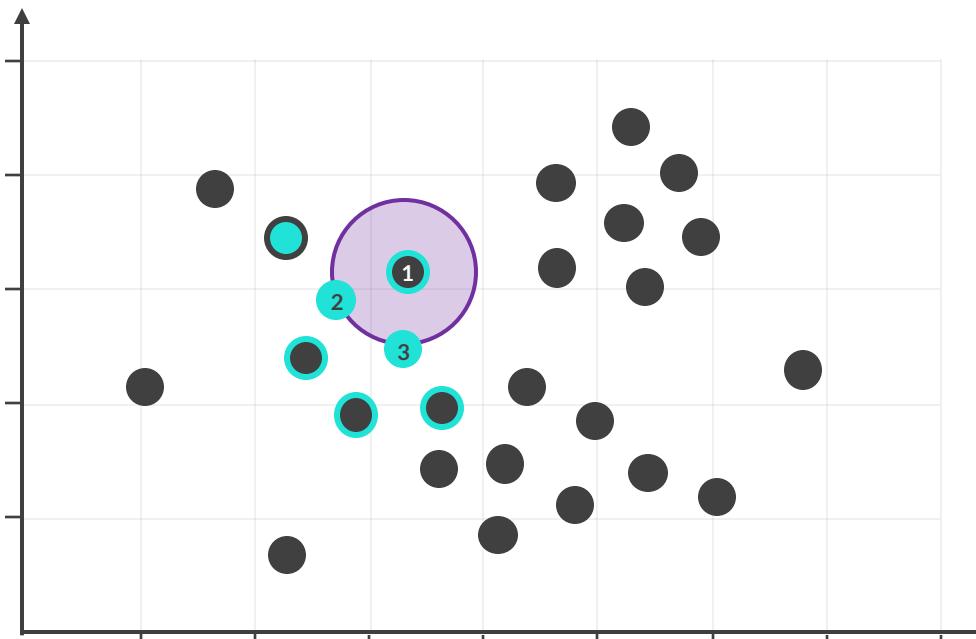
K-Means
Clustering

Hierarchical
Clustering

DBSCAN

Comparing
Models

STEP 4: Move on to another neighbor and continue this process until all the points are labeled within the cluster



eps = **0.75**
min_samples = **4**



DBSCAN

Clustering Basics

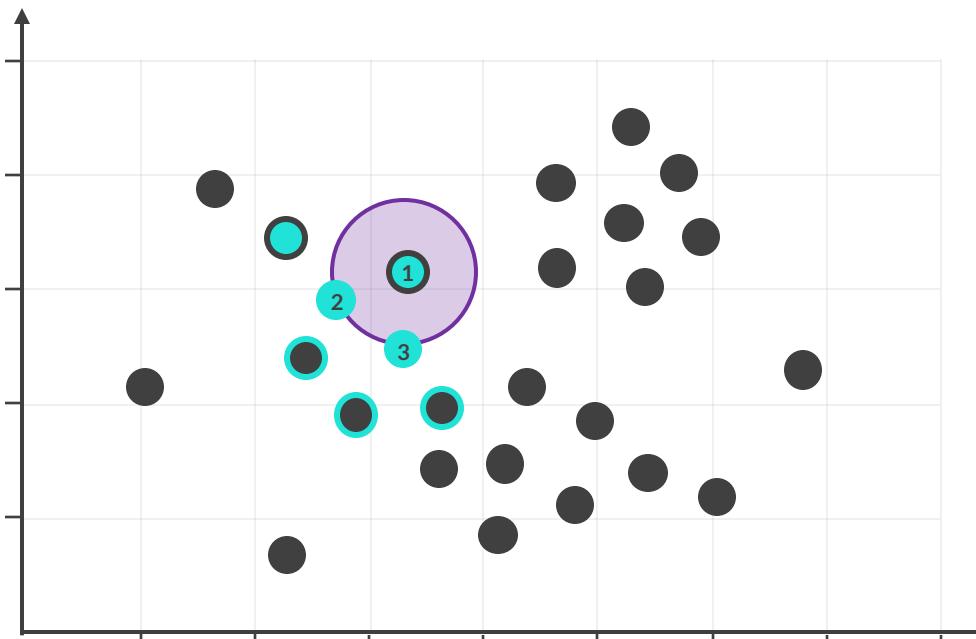
K-Means
Clustering

Hierarchical
Clustering

DBSCAN

Comparing
Models

STEP 4: Move on to another neighbor and continue this process until all the points are labeled within the cluster



$\text{eps} = 0.75$
 $\text{min_samples} = 4$



DBSCAN

Clustering Basics

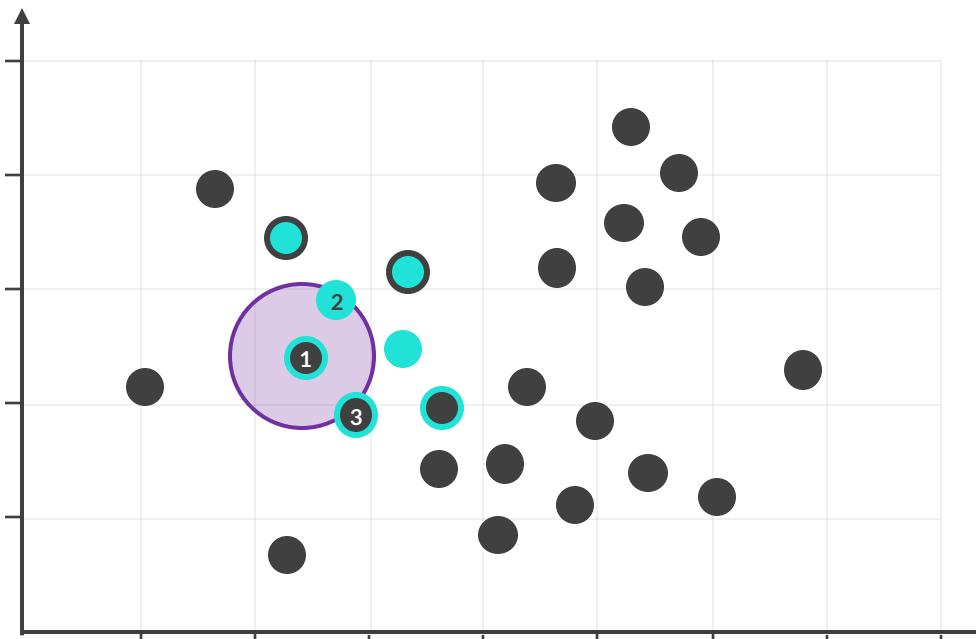
K-Means
Clustering

Hierarchical
Clustering

DBSCAN

Comparing
Models

STEP 4: Move on to another neighbor and continue this process until all the points are labeled within the cluster



$\text{eps} = 0.75$
 $\text{min_samples} = 4$



DBSCAN

Clustering Basics

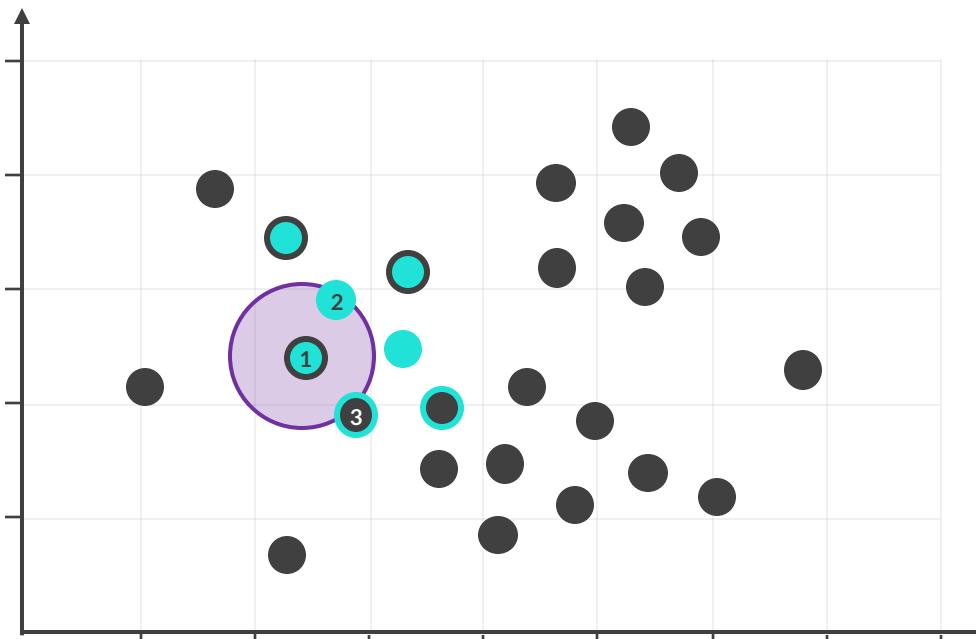
K-Means
Clustering

Hierarchical
Clustering

DBSCAN

Comparing
Models

STEP 4: Move on to another neighbor and continue this process until all the points are labeled within the cluster



$\text{eps} = 0.75$
 $\text{min_samples} = 4$



DBSCAN

Clustering Basics

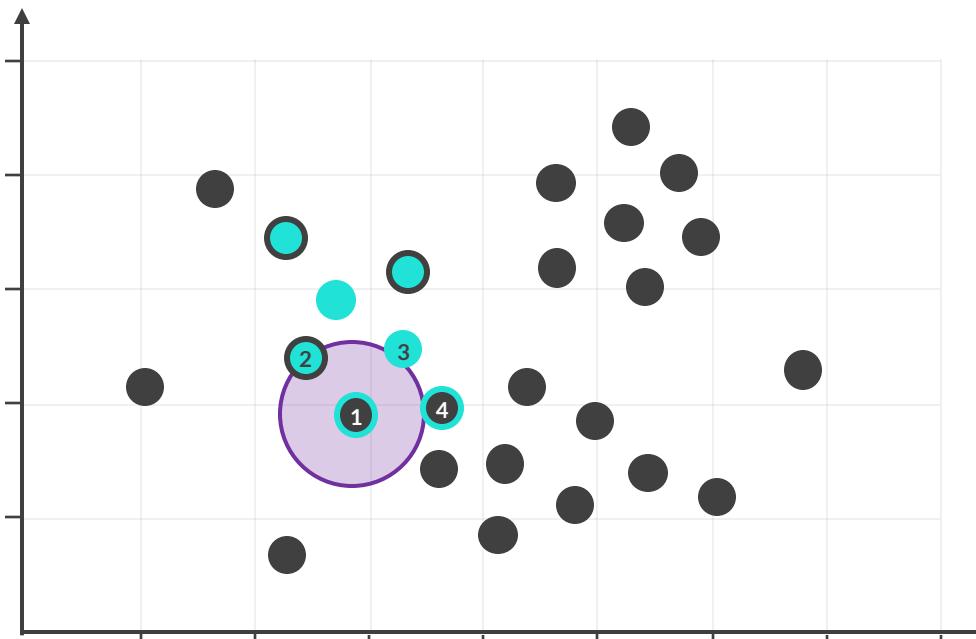
K-Means
Clustering

Hierarchical
Clustering

DBSCAN

Comparing
Models

STEP 4: Move on to another neighbor and continue this process until all the points are labeled within the cluster



$\text{eps} = 0.75$
 $\text{min_samples} = 4$



DBSCAN

Clustering Basics

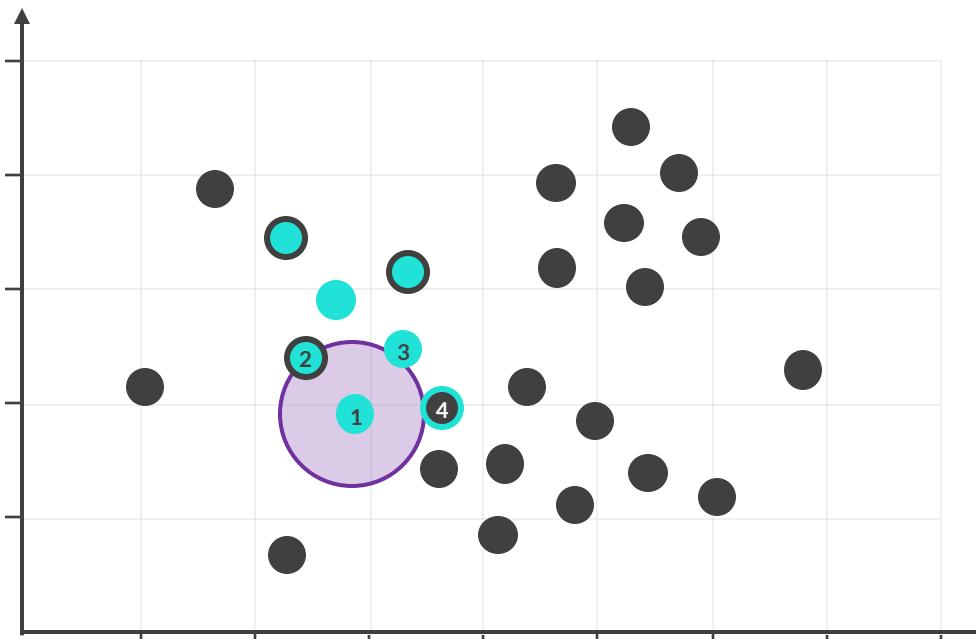
K-Means
Clustering

Hierarchical
Clustering

DBSCAN

Comparing
Models

STEP 4: Move on to another neighbor and continue this process until all the points are labeled within the cluster



eps = **0.75**
min_samples = **4**



DBSCAN

Clustering Basics

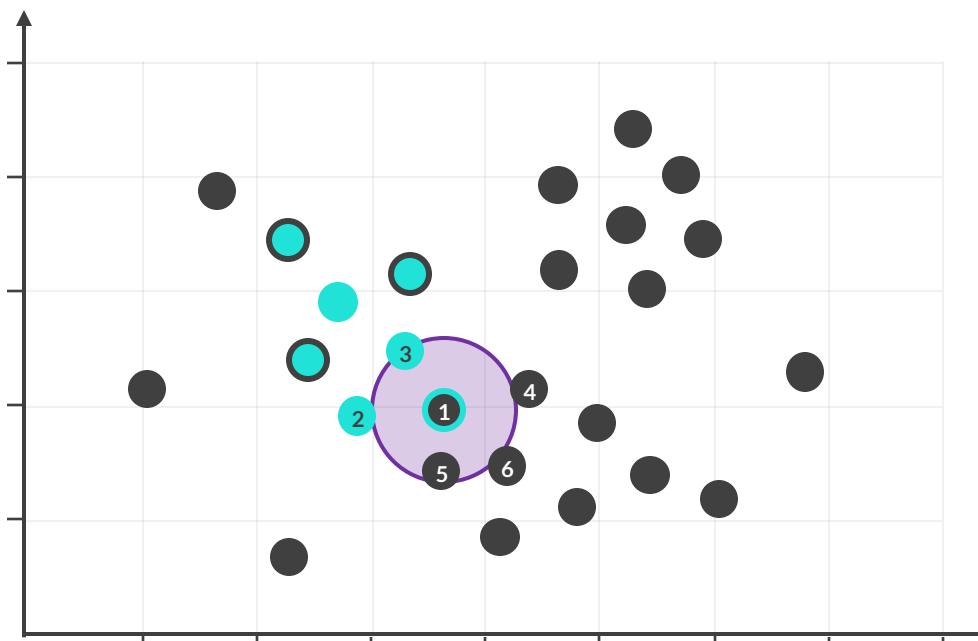
K-Means
Clustering

Hierarchical
Clustering

DBSCAN

Comparing
Models

STEP 4: Move on to another neighbor and continue this process until all the points are labeled within the cluster



$\text{eps} = 0.75$
 $\text{min_samples} = 4$



DBSCAN

Clustering Basics

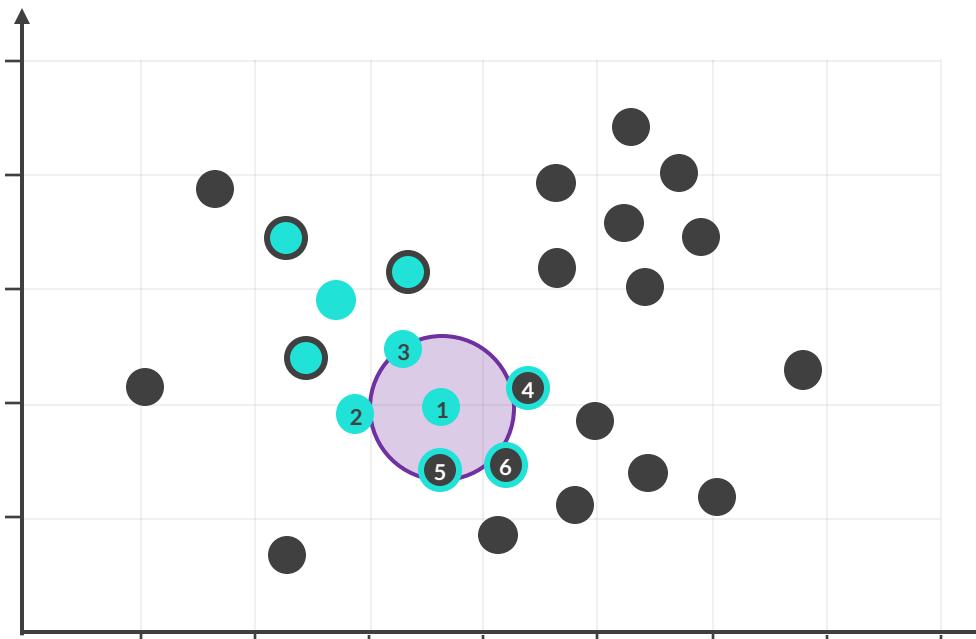
K-Means
Clustering

Hierarchical
Clustering

DBSCAN

Comparing
Models

STEP 4: Move on to another neighbor and continue this process until all the points are labeled within the cluster



$\text{eps} = 0.75$
 $\text{min_samples} = 6$



DBSCAN

Clustering Basics

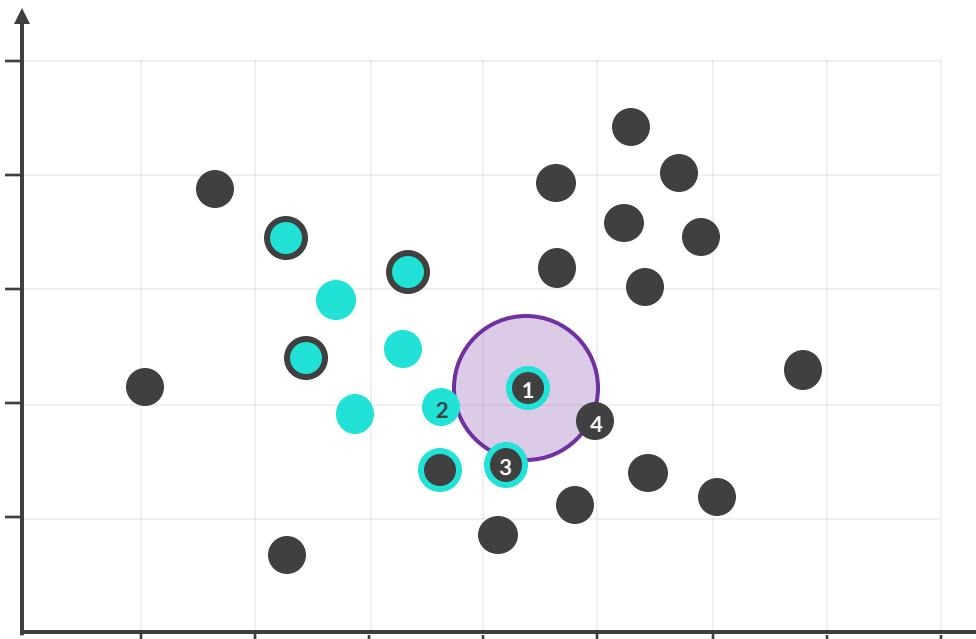
K-Means
Clustering

Hierarchical
Clustering

DBSCAN

Comparing
Models

STEP 4: Move on to another neighbor and continue this process until all the points are labeled within the cluster



$\text{eps} = 0.75$
 $\text{min_samples} = 4$



DBSCAN

Clustering Basics

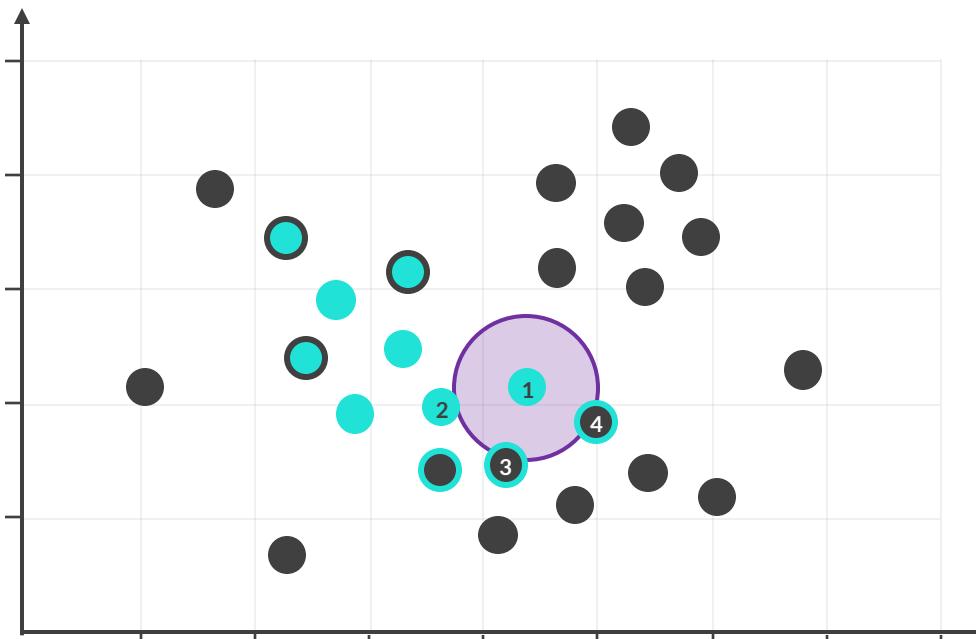
K-Means
Clustering

Hierarchical
Clustering

DBSCAN

Comparing
Models

STEP 4: Move on to another neighbor and continue this process until all the points are labeled within the cluster



eps = **0.75**
min_samples = **4**



DBSCAN

Clustering Basics

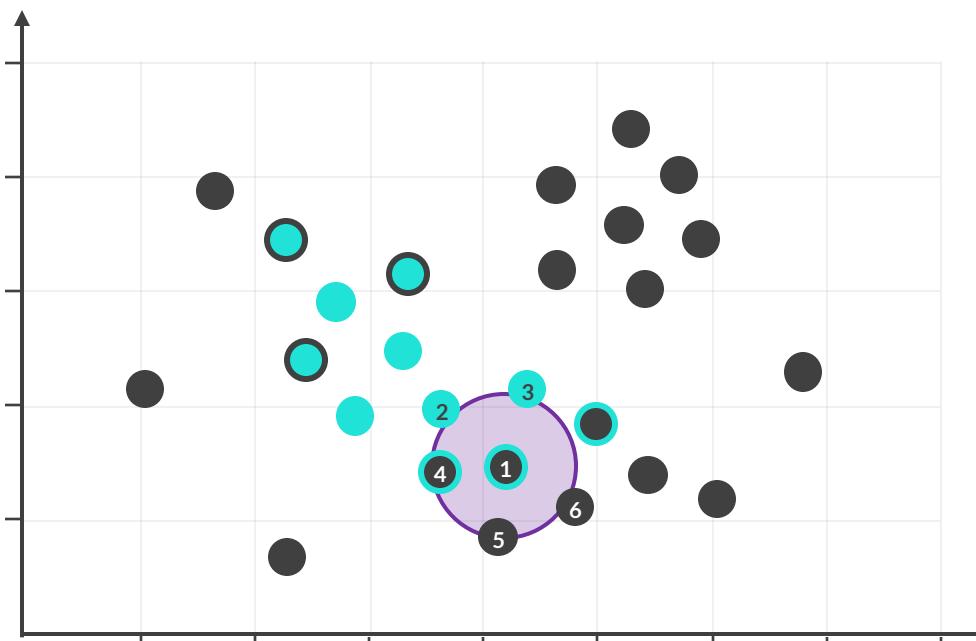
K-Means
Clustering

Hierarchical
Clustering

DBSCAN

Comparing
Models

STEP 4: Move on to another neighbor and continue this process until all the points are labeled within the cluster



eps = **0.75**
min_samples = **4**



DBSCAN

Clustering Basics

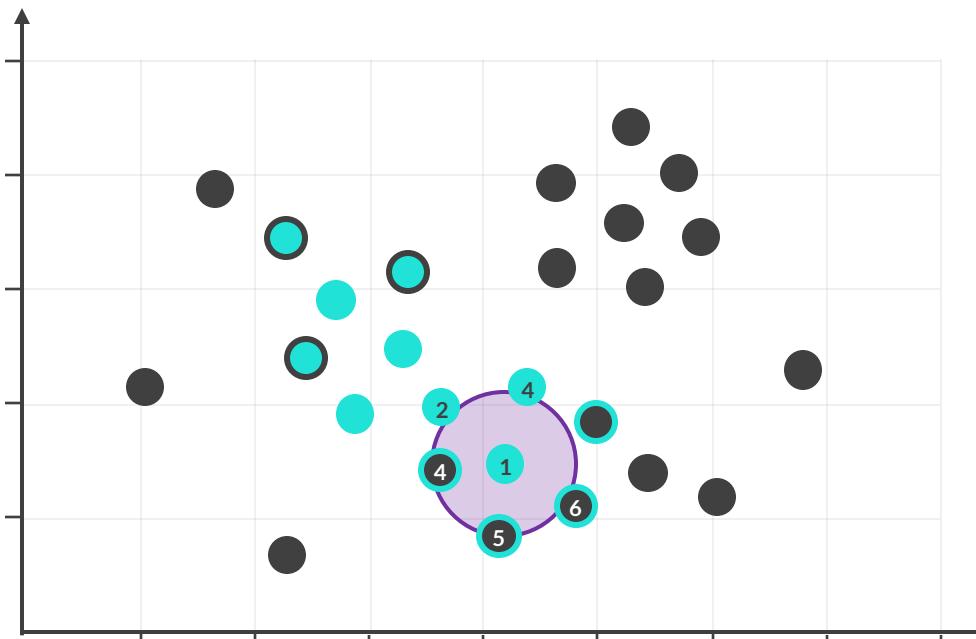
K-Means
Clustering

Hierarchical
Clustering

DBSCAN

Comparing
Models

STEP 4: Move on to another neighbor and continue this process until all the points are labeled within the cluster



$\text{eps} = 0.75$
 $\text{min_samples} = 4$



DBSCAN

Clustering Basics

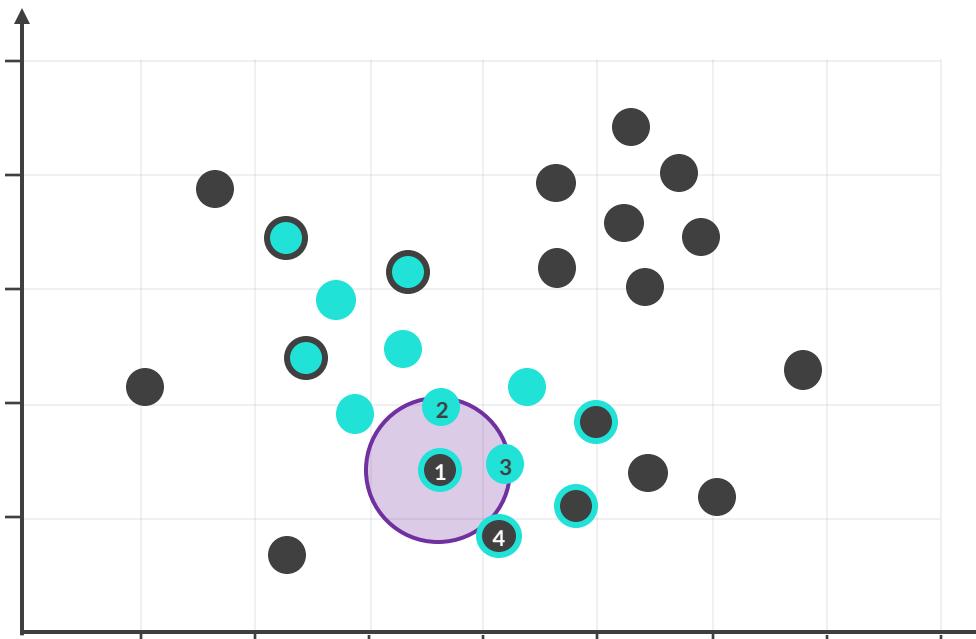
K-Means
Clustering

Hierarchical
Clustering

DBSCAN

Comparing
Models

STEP 4: Move on to another neighbor and continue this process until all the points are labeled within the cluster



$\text{eps} = 0.75$
 $\text{min_samples} = 4$



DBSCAN

Clustering Basics

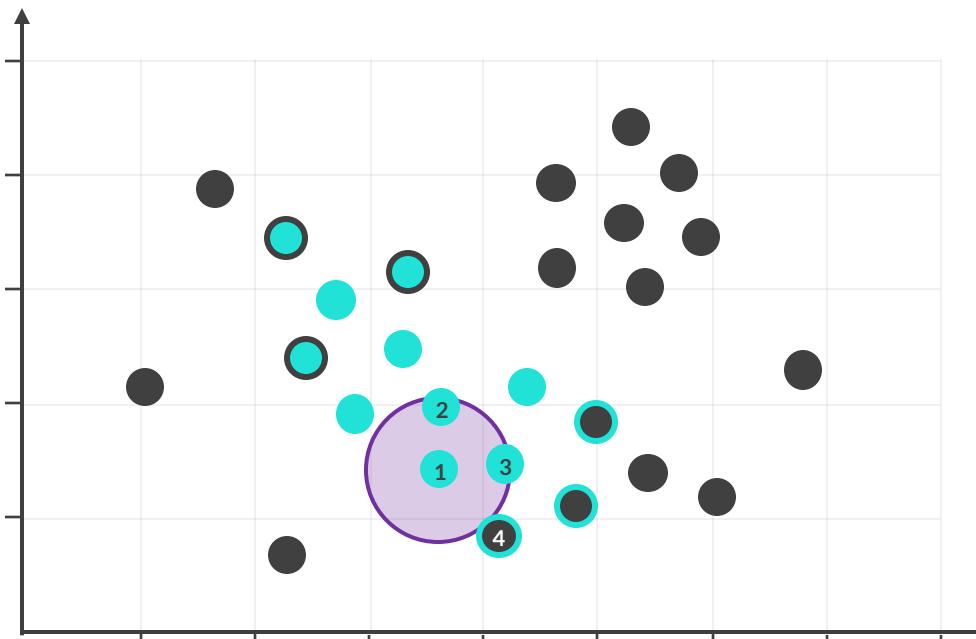
K-Means
Clustering

Hierarchical
Clustering

DBSCAN

Comparing
Models

STEP 4: Move on to another neighbor and continue this process until all the points are labeled within the cluster



$\text{eps} = 0.75$
 $\text{min_samples} = 4$



DBSCAN

Clustering Basics

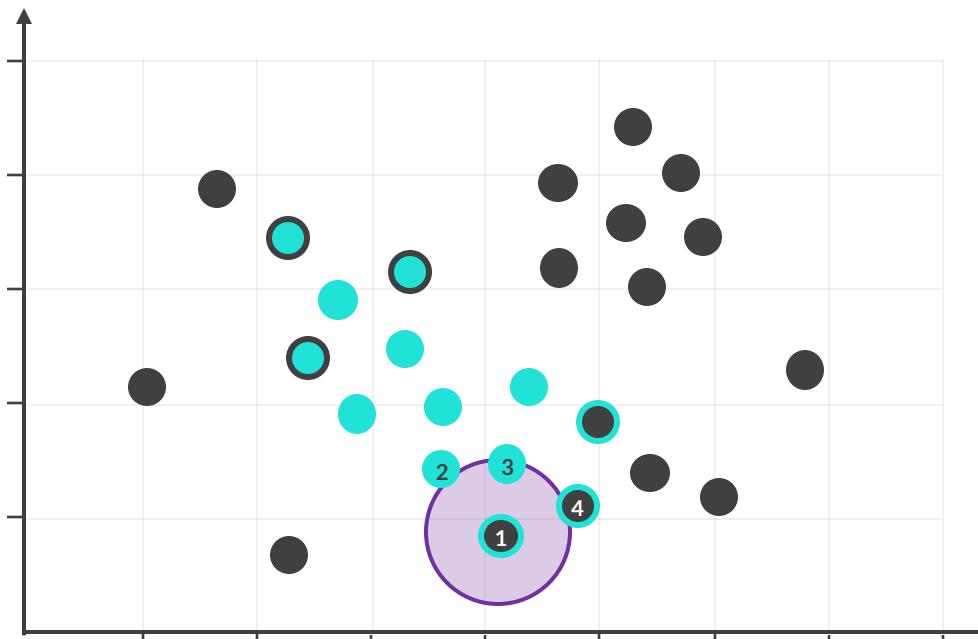
K-Means
Clustering

Hierarchical
Clustering

DBSCAN

Comparing
Models

STEP 4: Move on to another neighbor and continue this process until all the points are labeled within the cluster



eps = **0.75**
min_samples = **4**



DBSCAN

Clustering Basics

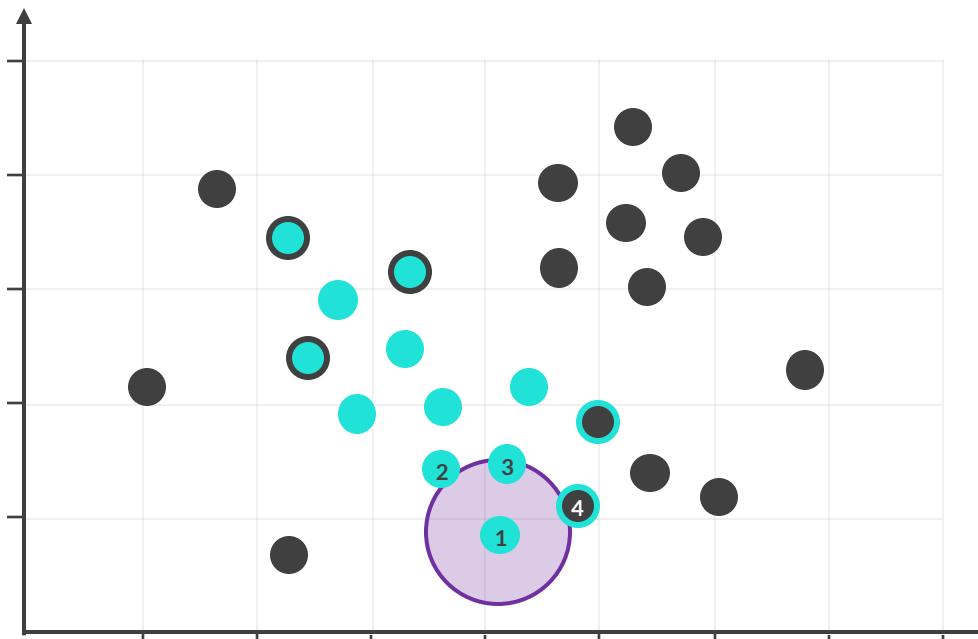
K-Means
Clustering

Hierarchical
Clustering

DBSCAN

Comparing
Models

STEP 4: Move on to another neighbor and continue this process until all the points are labeled within the cluster



eps = **0.75**
min_samples = **4**



DBSCAN

Clustering Basics

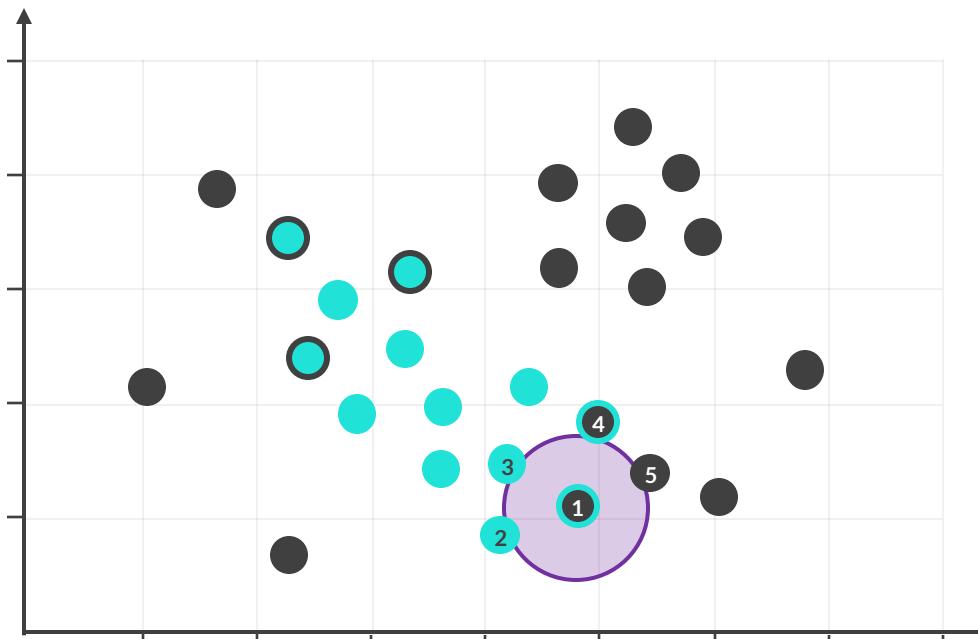
K-Means
Clustering

Hierarchical
Clustering

DBSCAN

Comparing
Models

STEP 4: Move on to another neighbor and continue this process until all the points are labeled within the cluster



eps = **0.75**
min_samples = **4**



DBSCAN

Clustering Basics

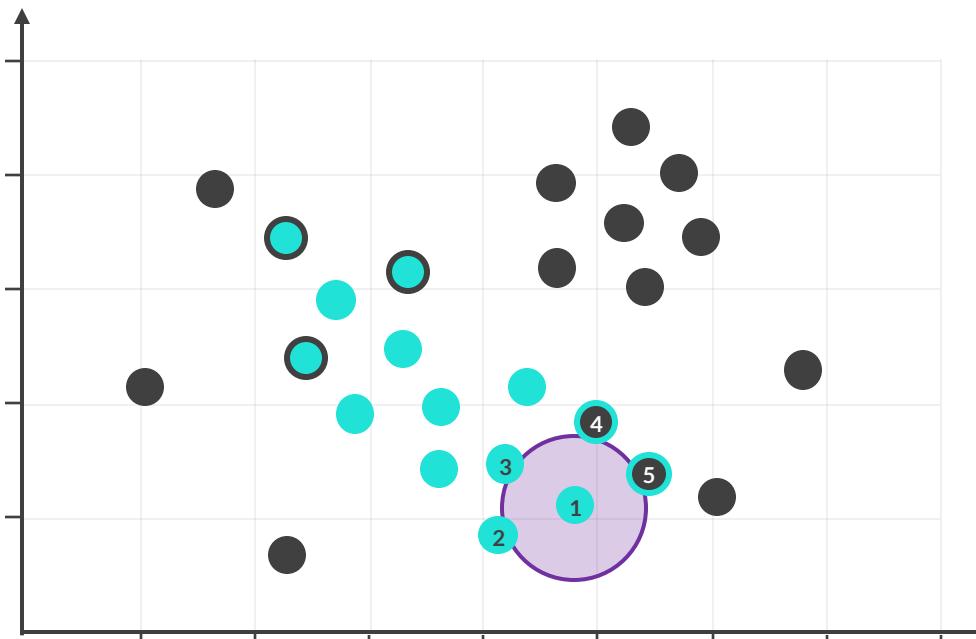
K-Means
Clustering

Hierarchical
Clustering

DBSCAN

Comparing
Models

STEP 4: Move on to another neighbor and continue this process until all the points are labeled within the cluster



eps = **0.75**
min_samples = **4**



DBSCAN

Clustering Basics

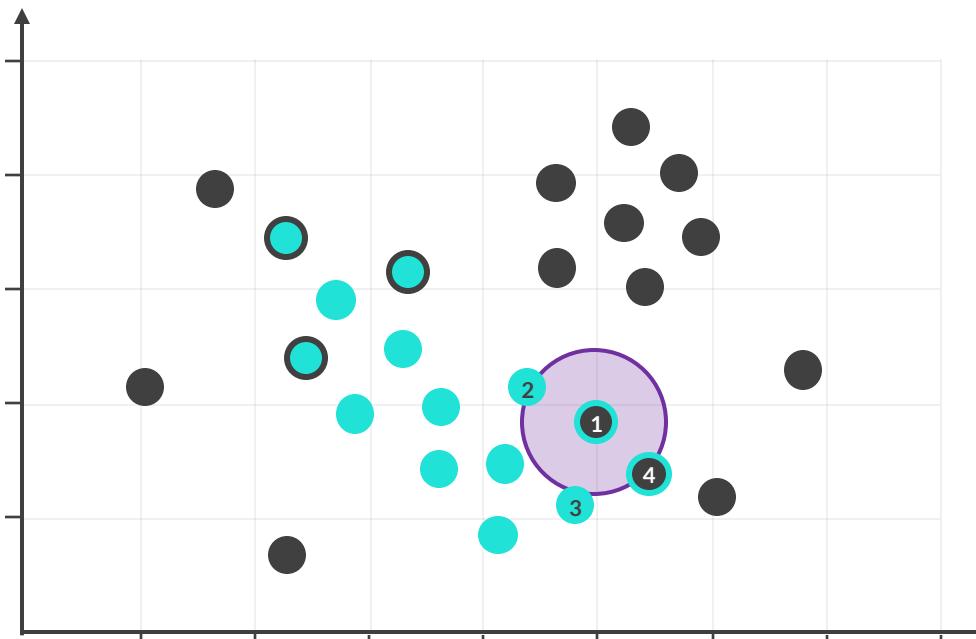
K-Means
Clustering

Hierarchical
Clustering

DBSCAN

Comparing
Models

STEP 4: Move on to another neighbor and continue this process until all the points are labeled within the cluster



eps = **0.75**
min_samples = **4**



DBSCAN

Clustering Basics

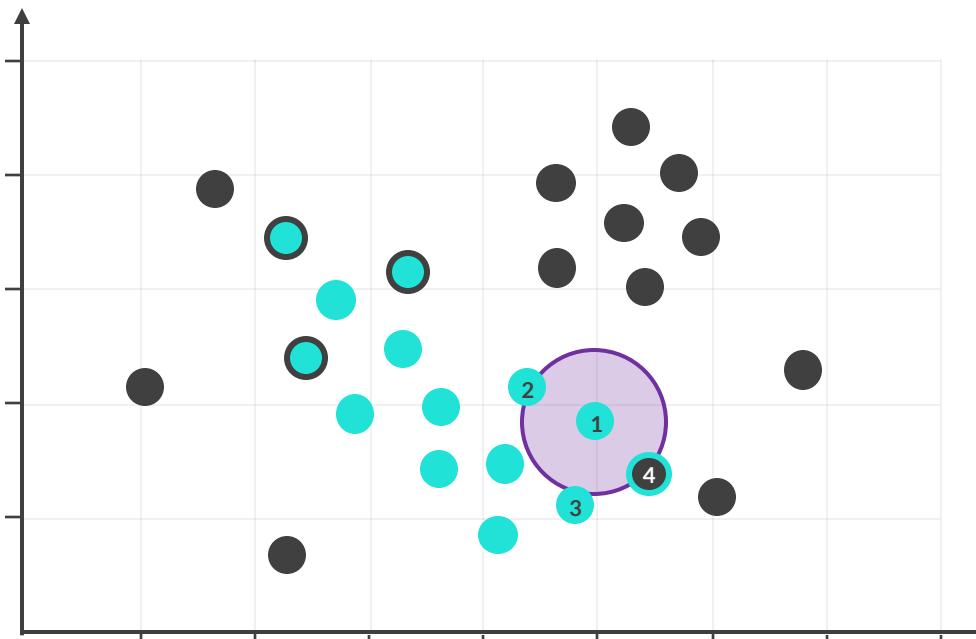
K-Means
Clustering

Hierarchical
Clustering

DBSCAN

Comparing
Models

STEP 4: Move on to another neighbor and continue this process until all the points are labeled within the cluster



$\text{eps} = 0.75$
 $\text{min_samples} = 4$



DBSCAN

Clustering Basics

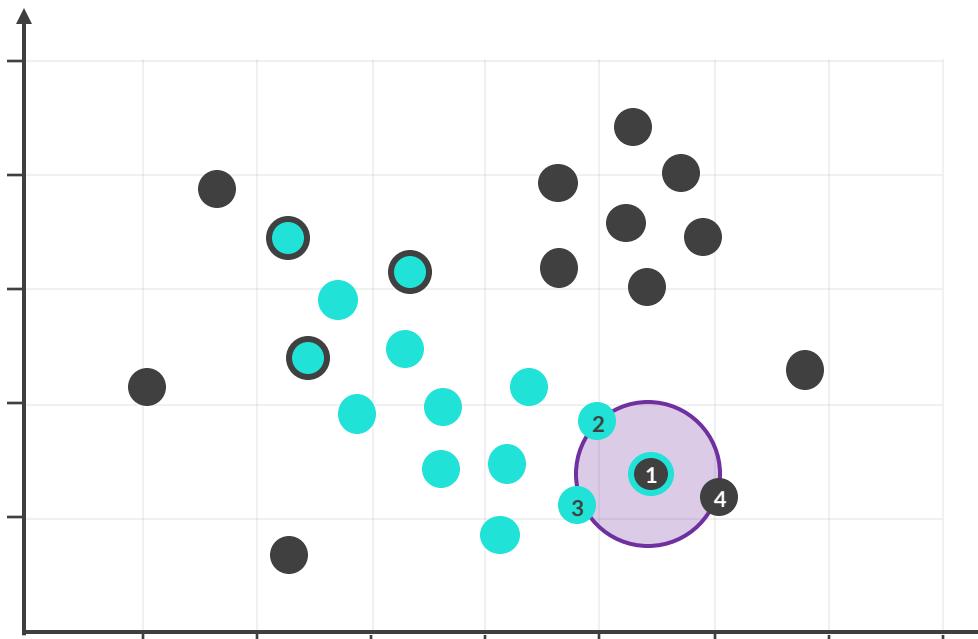
K-Means
Clustering

Hierarchical
Clustering

DBSCAN

Comparing
Models

STEP 4: Move on to another neighbor and continue this process until all the points are labeled within the cluster



$\text{eps} = 0.75$
 $\text{min_samples} = 4$



DBSCAN

Clustering Basics

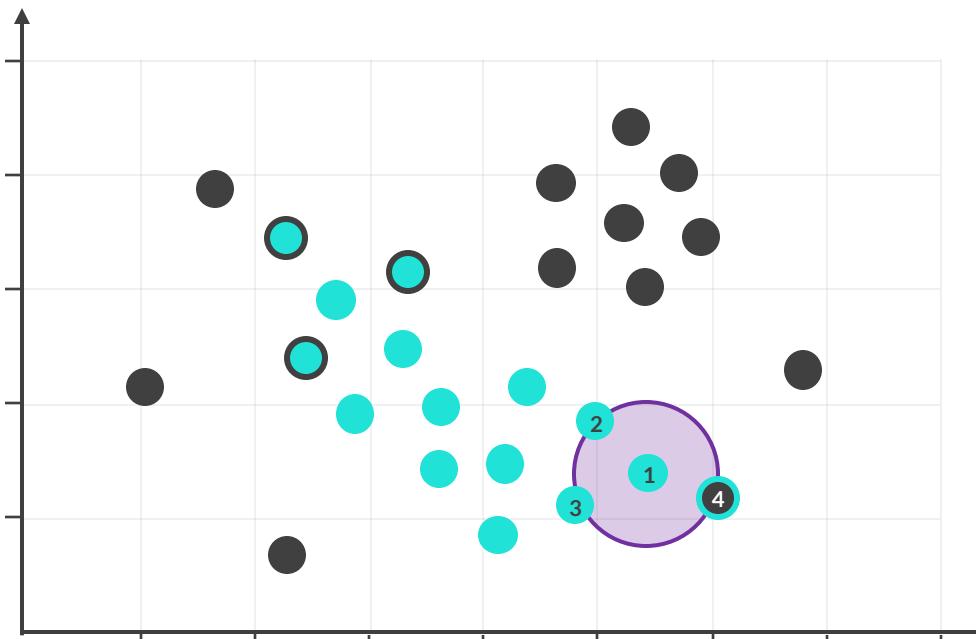
K-Means
Clustering

Hierarchical
Clustering

DBSCAN

Comparing
Models

STEP 4: Move on to another neighbor and continue this process until all the points are labeled within the cluster



$\text{eps} = 0.75$
 $\text{min_samples} = 4$



DBSCAN

Clustering Basics

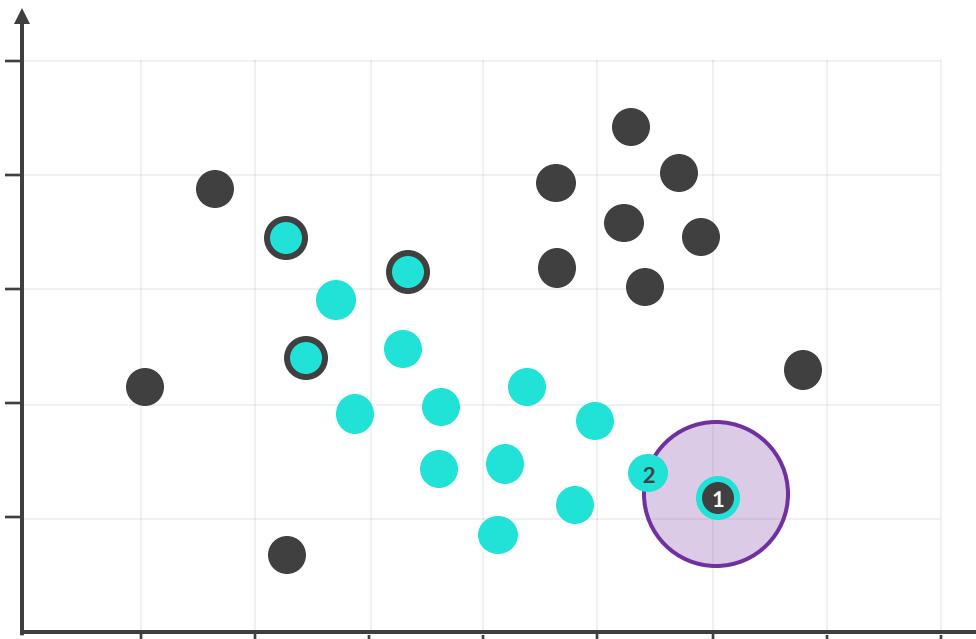
K-Means
Clustering

Hierarchical
Clustering

DBSCAN

Comparing
Models

STEP 4: Move on to another neighbor and continue this process until all the points are labeled within the cluster



$\text{eps} = 0.75$
 $\text{min_samples} = 4$



DBSCAN

Clustering Basics

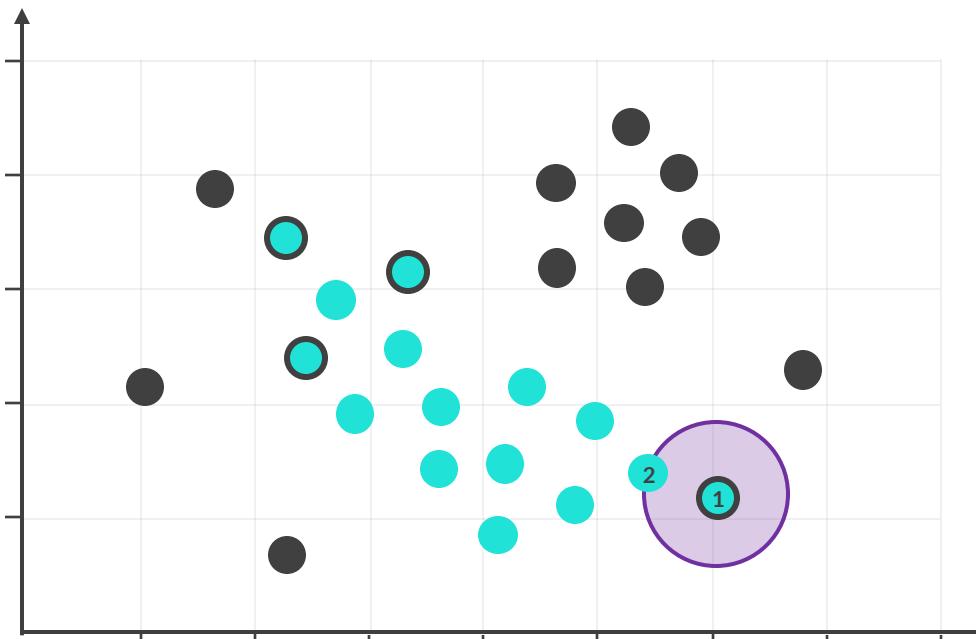
K-Means
Clustering

Hierarchical
Clustering

DBSCAN

Comparing
Models

STEP 4: Move on to another neighbor and continue this process until all the points are labeled within the cluster



$\text{eps} = 0.75$
 $\text{min_samples} = 4$



DBSCAN

Clustering Basics

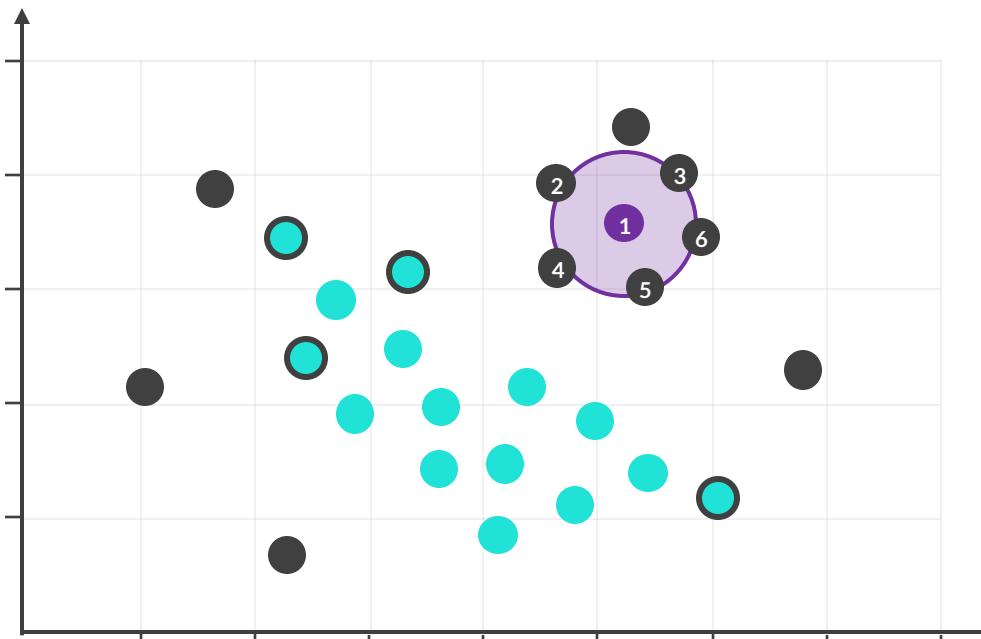
K-Means
Clustering

Hierarchical
Clustering

DBSCAN

Comparing
Models

STEP 5: Move on to another random point and repeat the same steps



eps = **0.75**
min_samples = **4**



DBSCAN

Clustering Basics

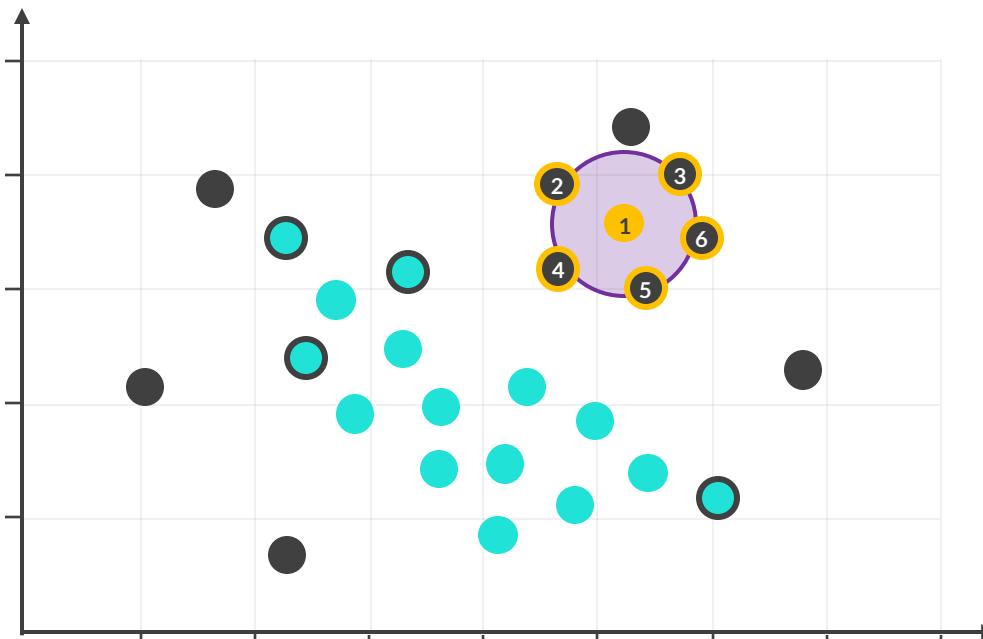
K-Means
Clustering

Hierarchical
Clustering

DBSCAN

Comparing
Models

STEP 5: Move on to another random point and repeat the same steps



eps = **0.75**
min_samples = **4**



DBSCAN

Clustering Basics

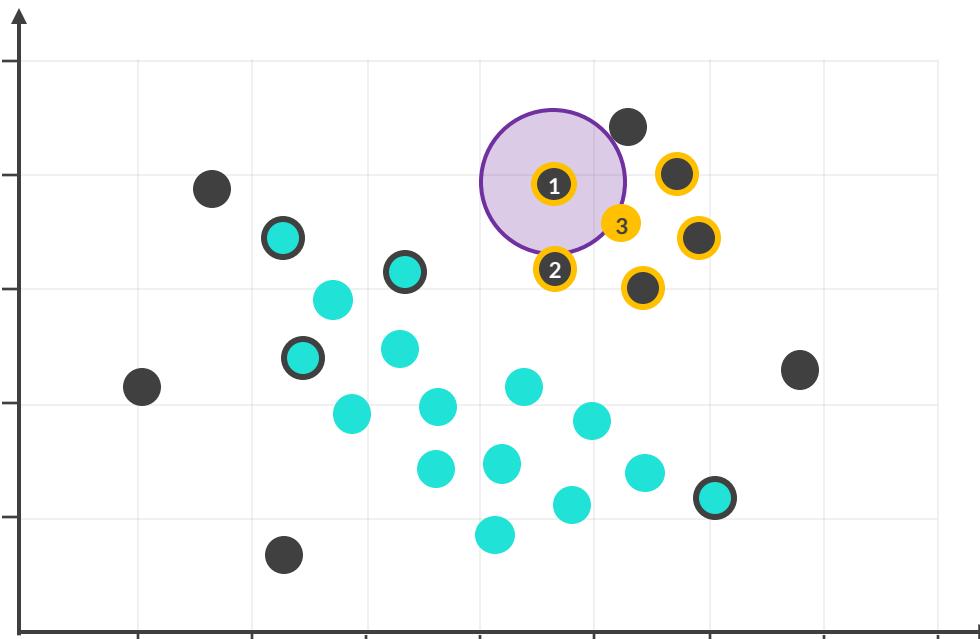
K-Means
Clustering

Hierarchical
Clustering

DBSCAN

Comparing
Models

STEP 5: Move on to another random point and repeat the same steps



eps = **0.75**
min_samples = **4**



DBSCAN

Clustering Basics

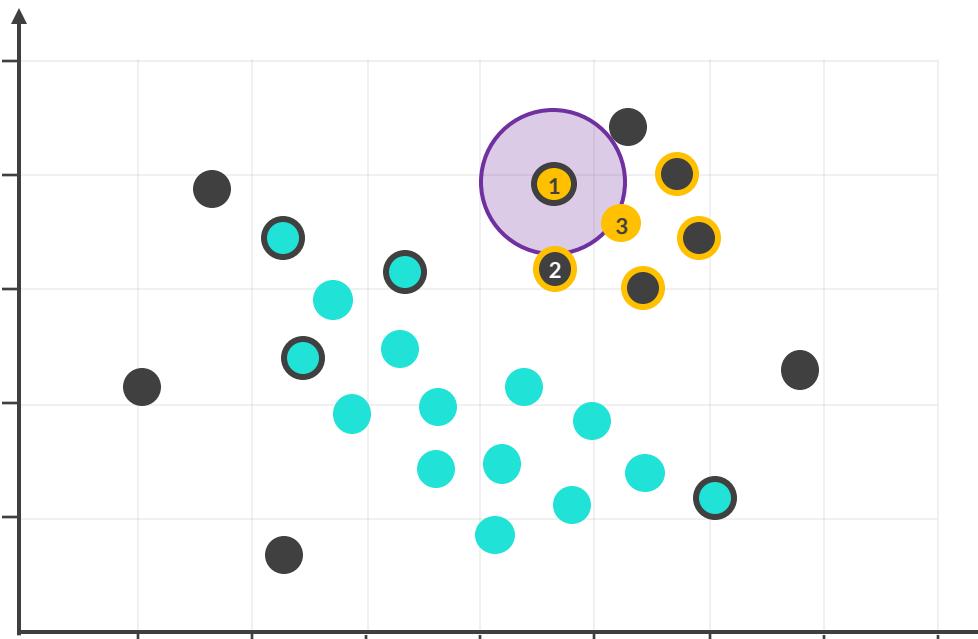
K-Means
Clustering

Hierarchical
Clustering

DBSCAN

Comparing
Models

STEP 5: Move on to another random point and repeat the same steps



eps = **0.75**
min_samples = **4**



DBSCAN

Clustering Basics

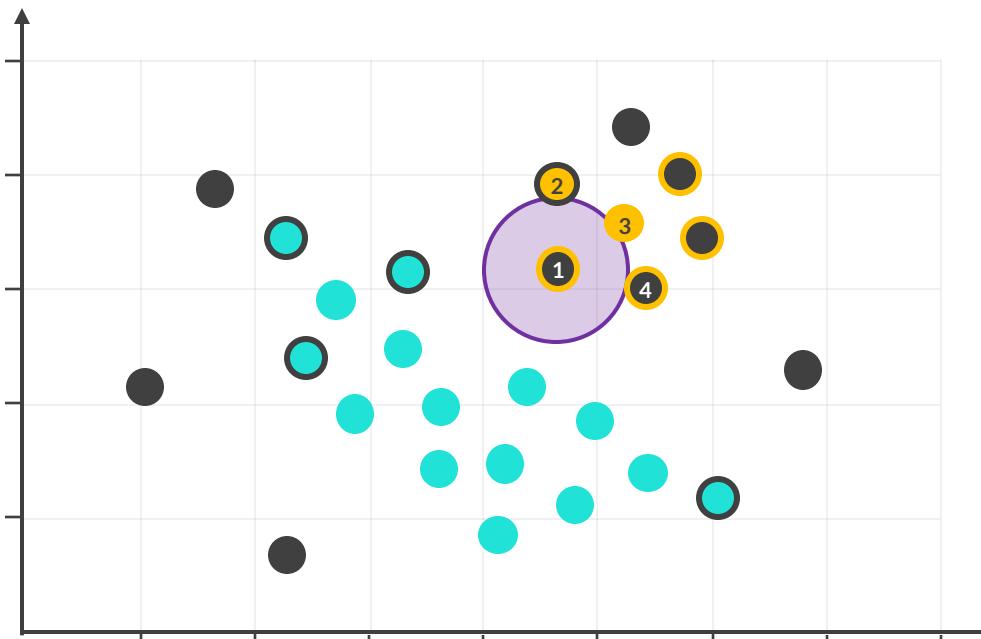
K-Means
Clustering

Hierarchical
Clustering

DBSCAN

Comparing
Models

STEP 5: Move on to another random point and repeat the same steps



`eps = 0.75`
`min_samples = 4`



DBSCAN

Clustering Basics

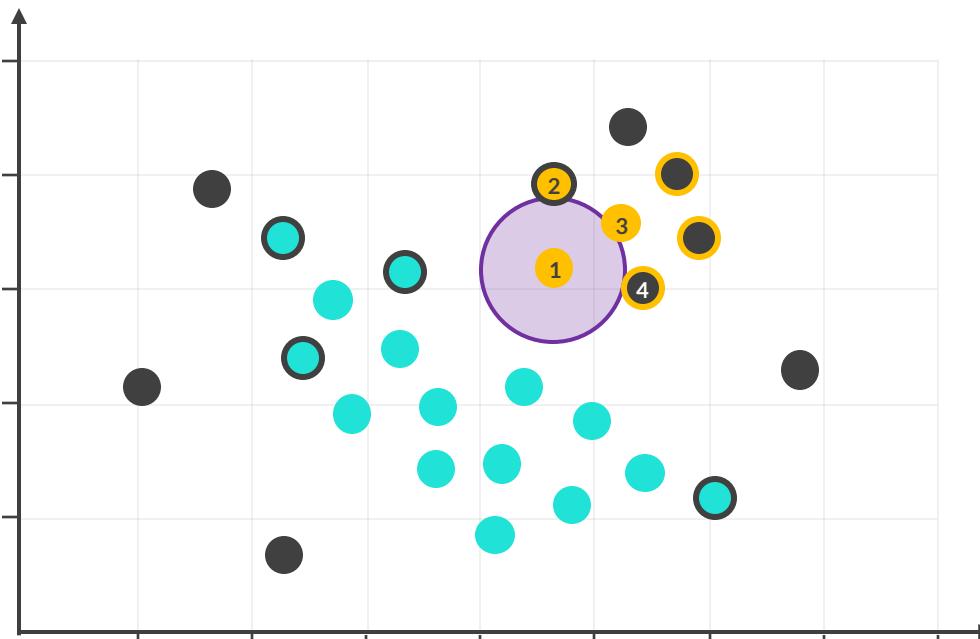
K-Means
Clustering

Hierarchical
Clustering

DBSCAN

Comparing
Models

STEP 5: Move on to another random point and repeat the same steps



$\text{eps} = 0.75$
 $\text{min_samples} = 4$



DBSCAN

Clustering Basics

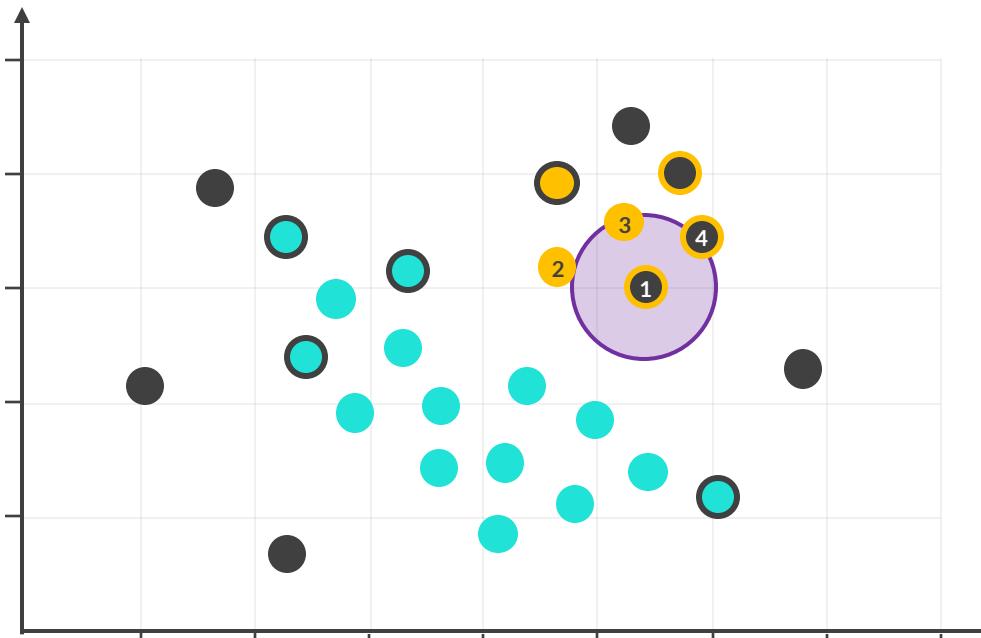
K-Means
Clustering

Hierarchical
Clustering

DBSCAN

Comparing
Models

STEP 5: Move on to another random point and repeat the same steps



`eps = 0.75`
`min_samples = 4`



DBSCAN

Clustering Basics

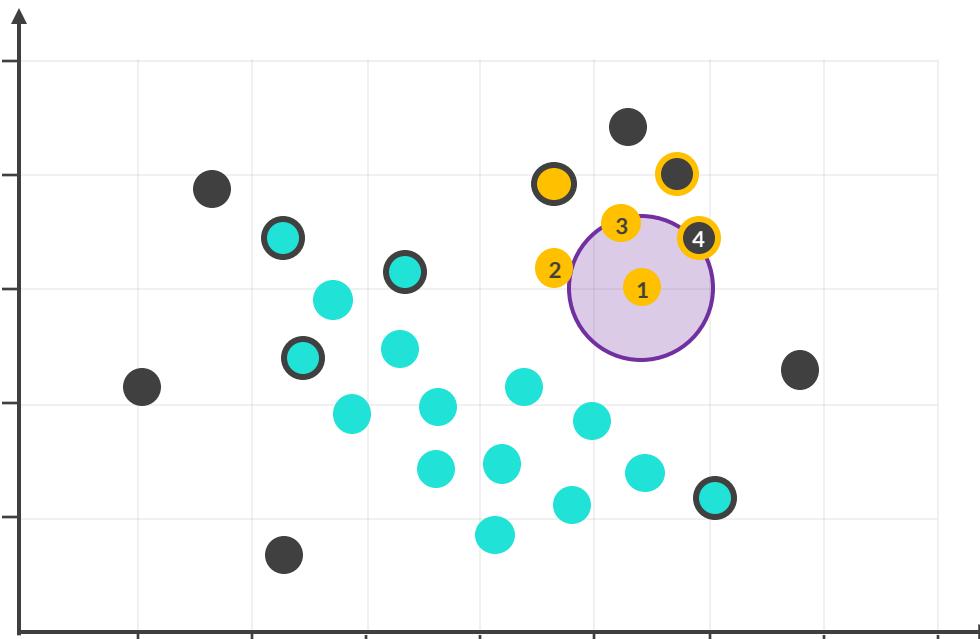
K-Means
Clustering

Hierarchical
Clustering

DBSCAN

Comparing
Models

STEP 5: Move on to another random point and repeat the same steps



eps = **0.75**
min_samples = **4**



DBSCAN

Clustering Basics

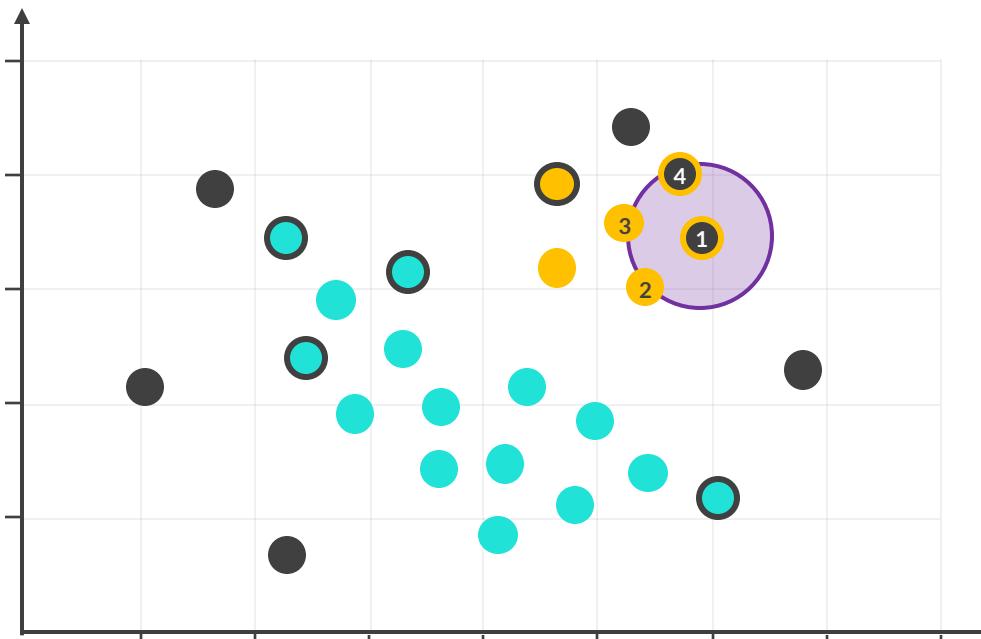
K-Means
Clustering

Hierarchical
Clustering

DBSCAN

Comparing
Models

STEP 5: Move on to another random point and repeat the same steps



eps = **0.75**
min_samples = **4**



DBSCAN

Clustering Basics

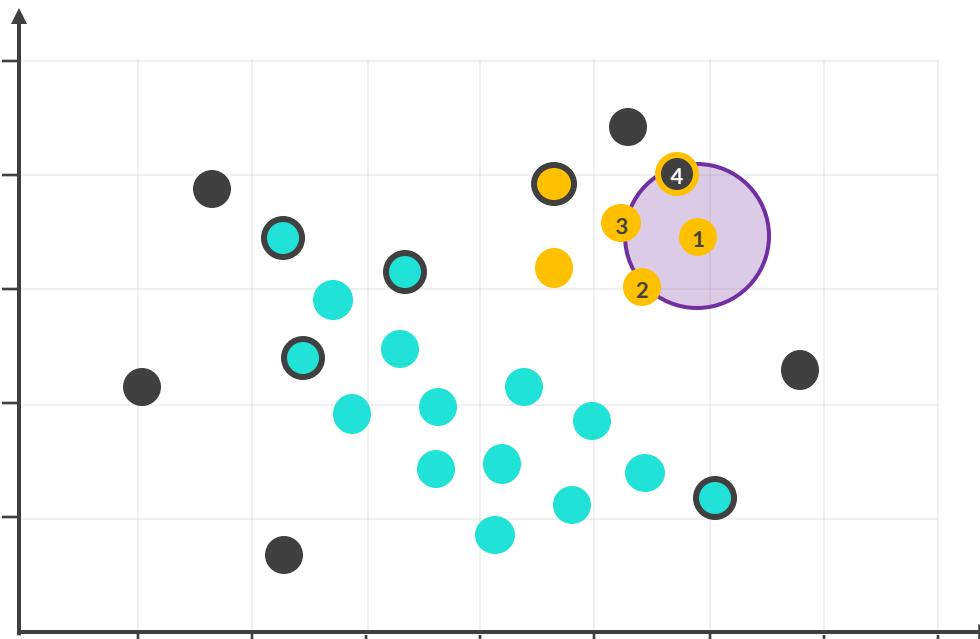
K-Means
Clustering

Hierarchical
Clustering

DBSCAN

Comparing
Models

STEP 5: Move on to another random point and repeat the same steps



$\text{eps} = 0.75$
 $\text{min_samples} = 4$



DBSCAN

Clustering Basics

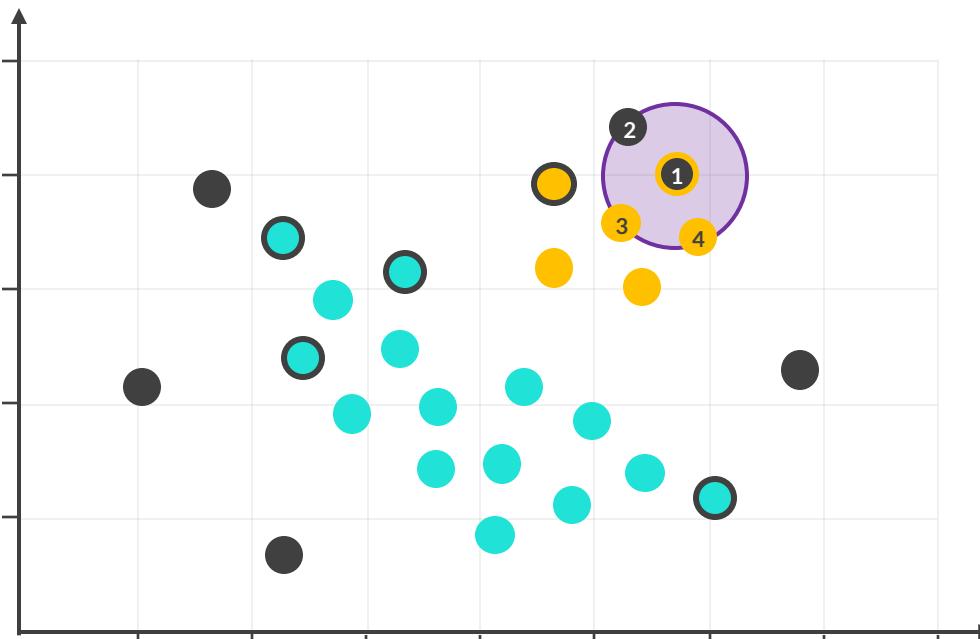
K-Means
Clustering

Hierarchical
Clustering

DBSCAN

Comparing
Models

STEP 5: Move on to another random point and repeat the same steps



$\text{eps} = 0.75$
 $\text{min_samples} = 4$



DBSCAN

Clustering Basics

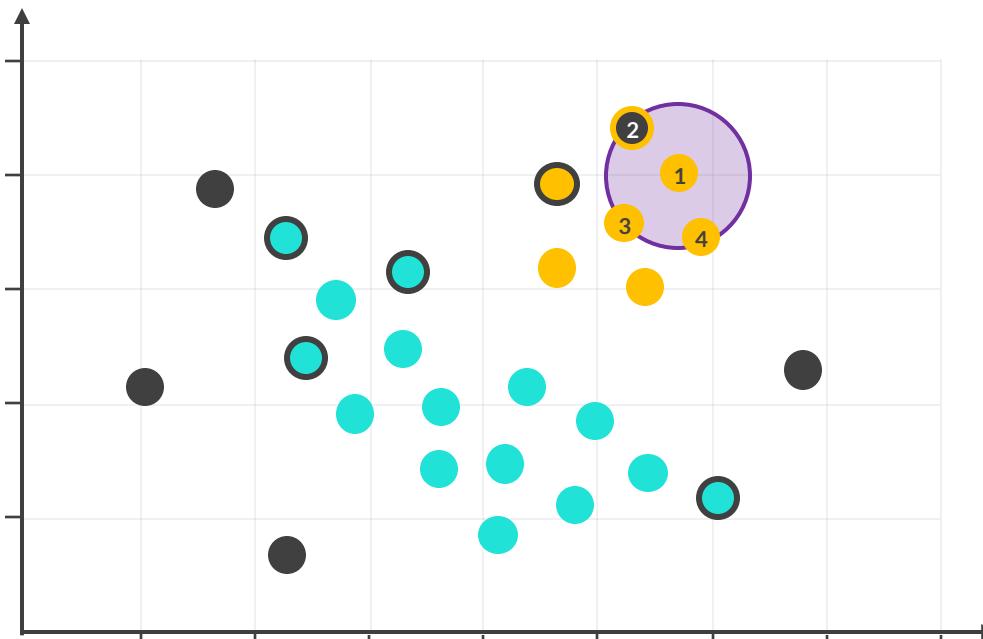
K-Means
Clustering

Hierarchical
Clustering

DBSCAN

Comparing
Models

STEP 5: Move on to another random point and repeat the same steps



`eps = 0.75`
`min_samples = 4`



DBSCAN

Clustering Basics

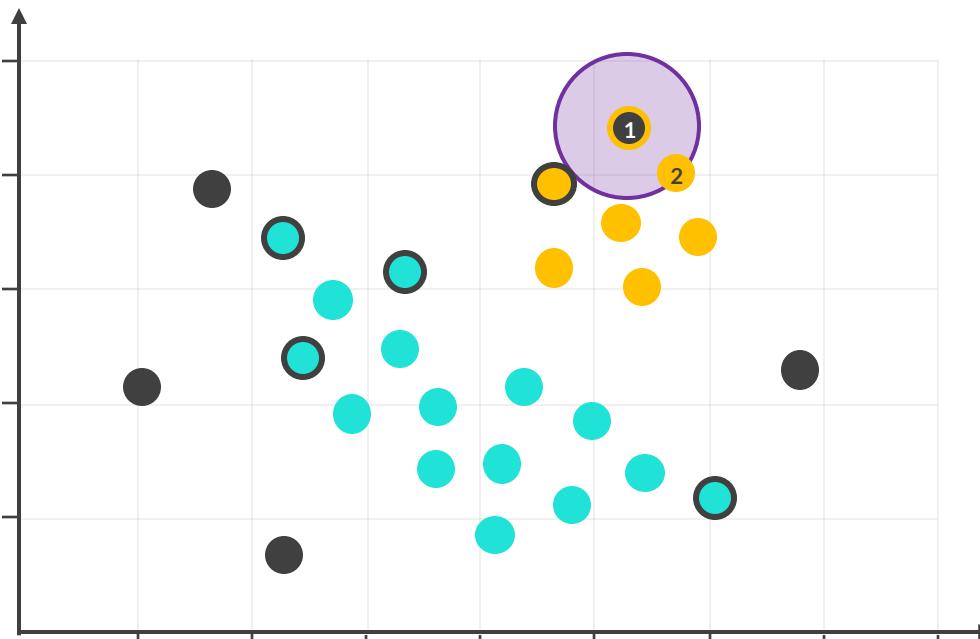
K-Means
Clustering

Hierarchical
Clustering

DBSCAN

Comparing
Models

STEP 5: Move on to another random point and repeat the same steps



$\text{eps} = 0.75$
 $\text{min_samples} = 4$



DBSCAN

Clustering Basics

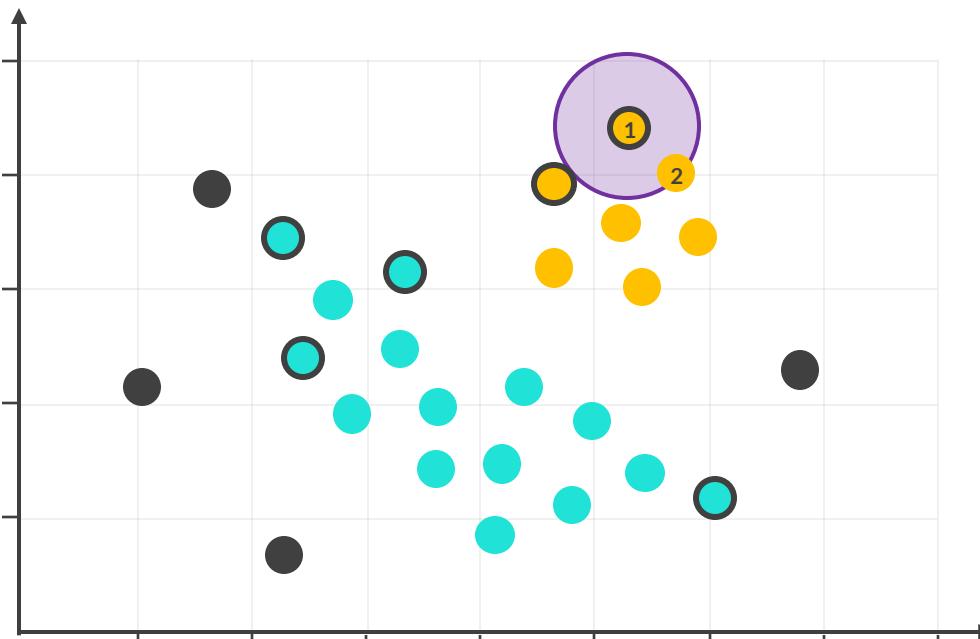
K-Means
Clustering

Hierarchical
Clustering

DBSCAN

Comparing
Models

STEP 5: Move on to another random point and repeat the same steps



eps = **0.75**
min_samples = **4**



DBSCAN

Clustering Basics

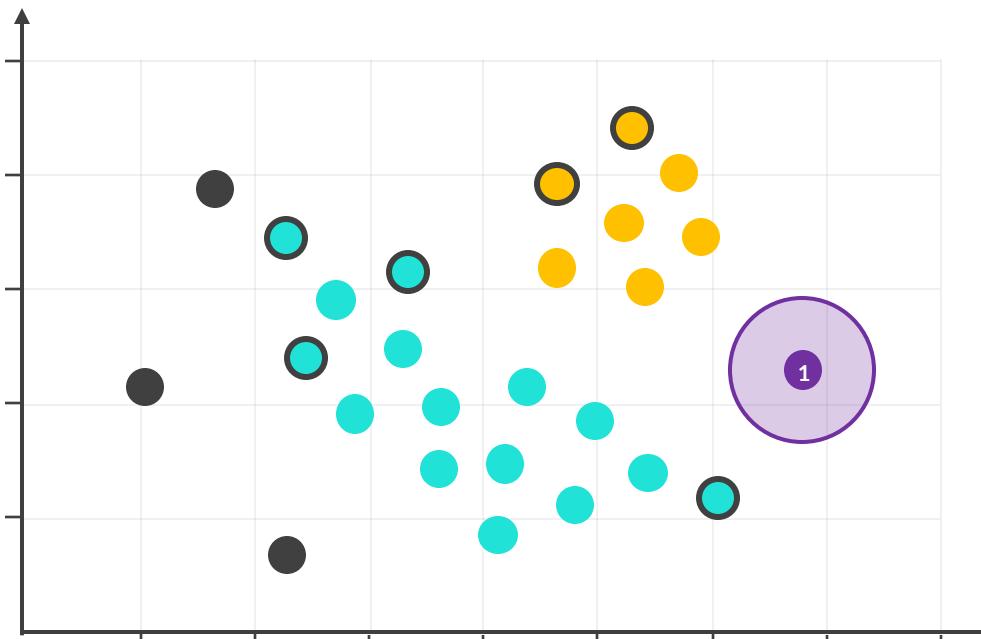
K-Means
Clustering

Hierarchical
Clustering

DBSCAN

Comparing
Models

STEP 5: Move on to another random point and repeat the same steps



eps = **0.75**
min_samples = **4**



DBSCAN

Clustering Basics

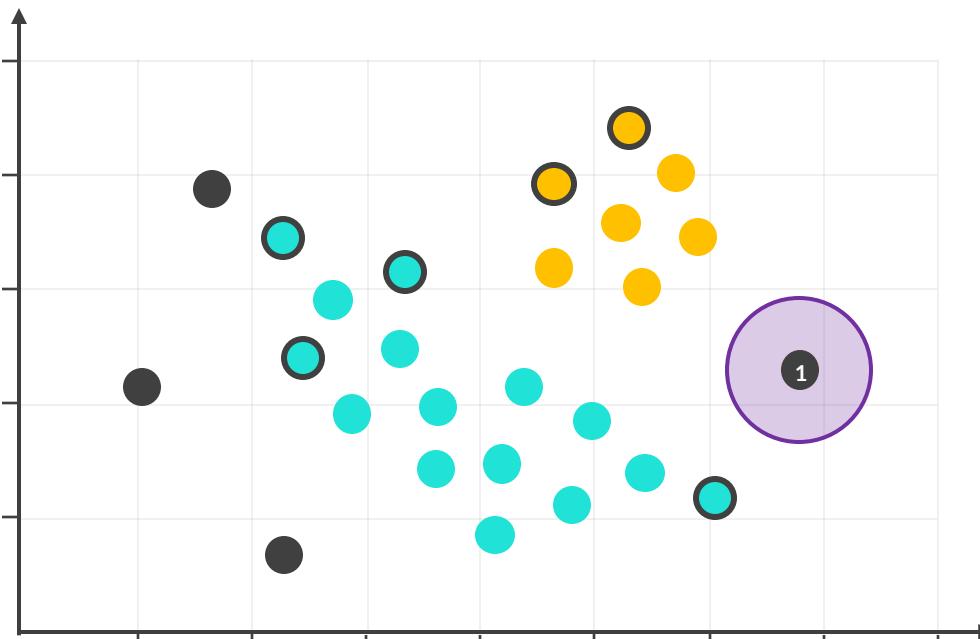
K-Means
Clustering

Hierarchical
Clustering

DBSCAN

Comparing
Models

STEP 5: Move on to another random point and repeat the same steps



eps = **0.75**
min_samples = **4**



DBSCAN

Clustering Basics

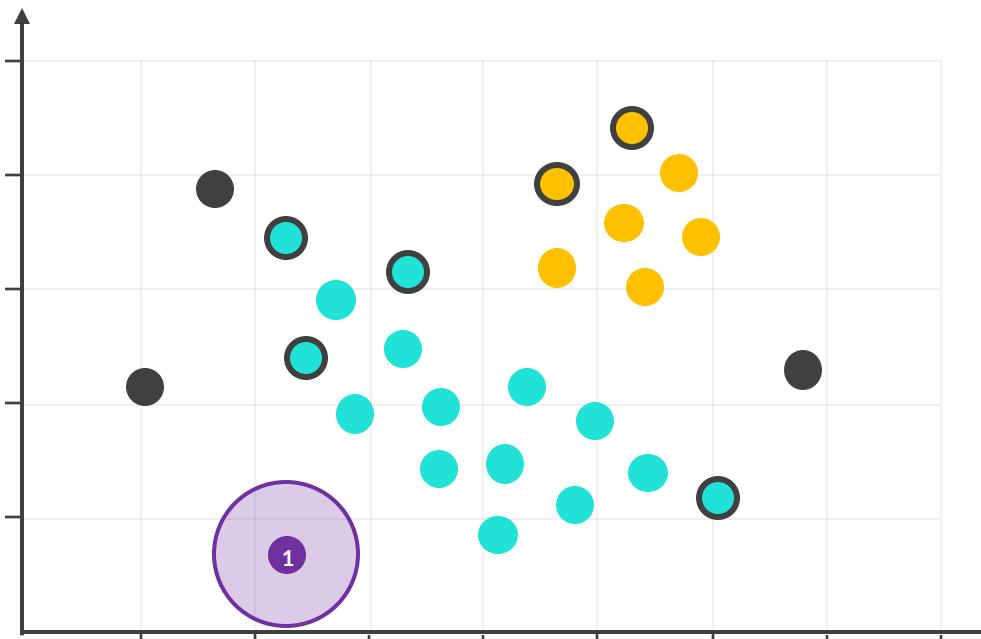
K-Means
Clustering

Hierarchical
Clustering

DBSCAN

Comparing
Models

STEP 5: Move on to another random point and repeat the same steps



eps = **0.75**
min_samples = **4**



DBSCAN

Clustering Basics

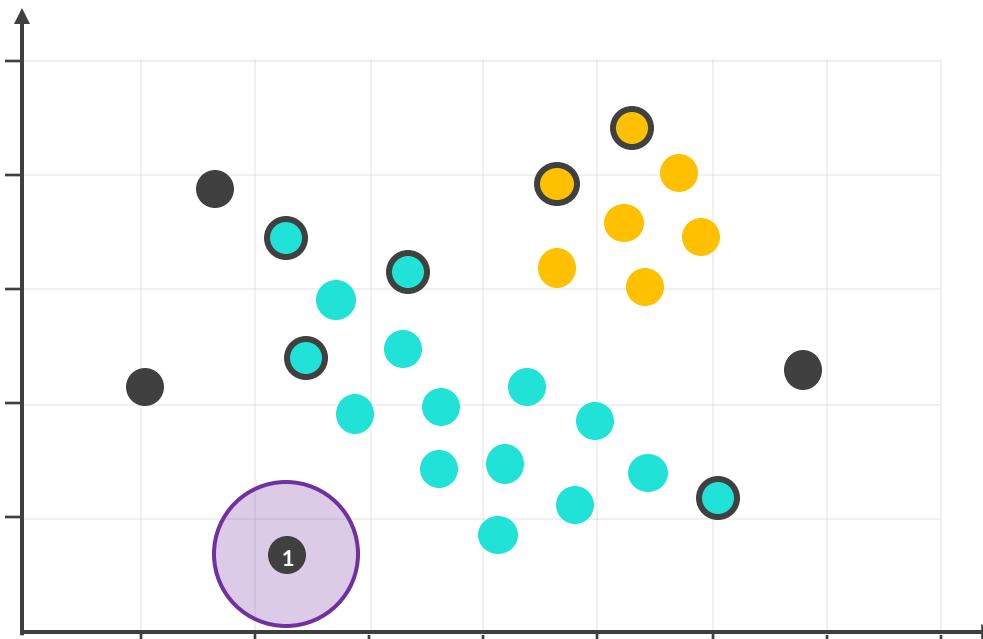
K-Means
Clustering

Hierarchical
Clustering

DBSCAN

Comparing
Models

STEP 5: Move on to another random point and repeat the same steps



eps = **0.75**
min_samples = **4**



DBSCAN

Clustering Basics

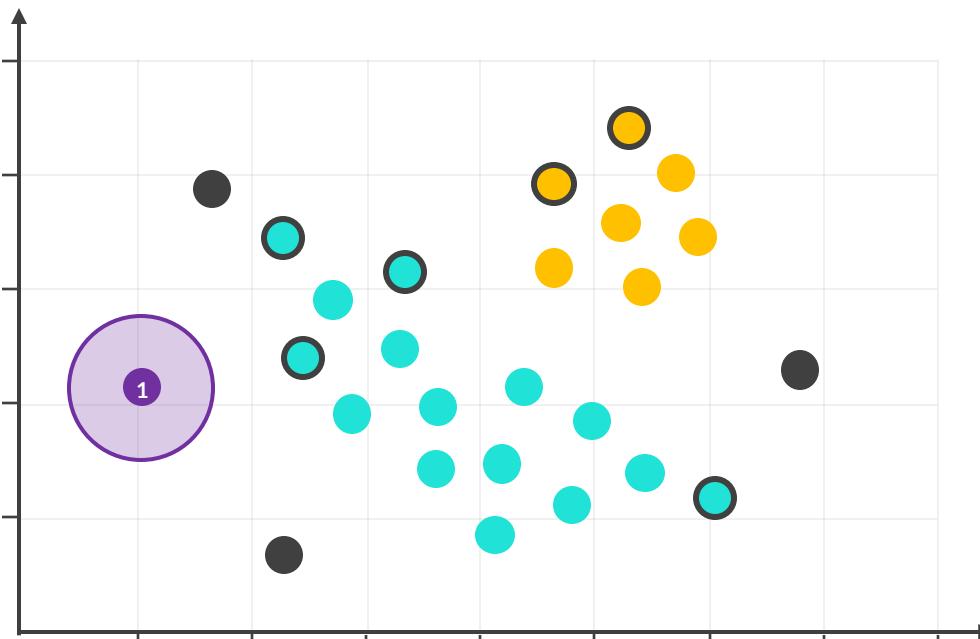
K-Means
Clustering

Hierarchical
Clustering

DBSCAN

Comparing
Models

STEP 5: Move on to another random point and repeat the same steps



eps = **0.75**
min_samples = **4**



DBSCAN

Clustering Basics

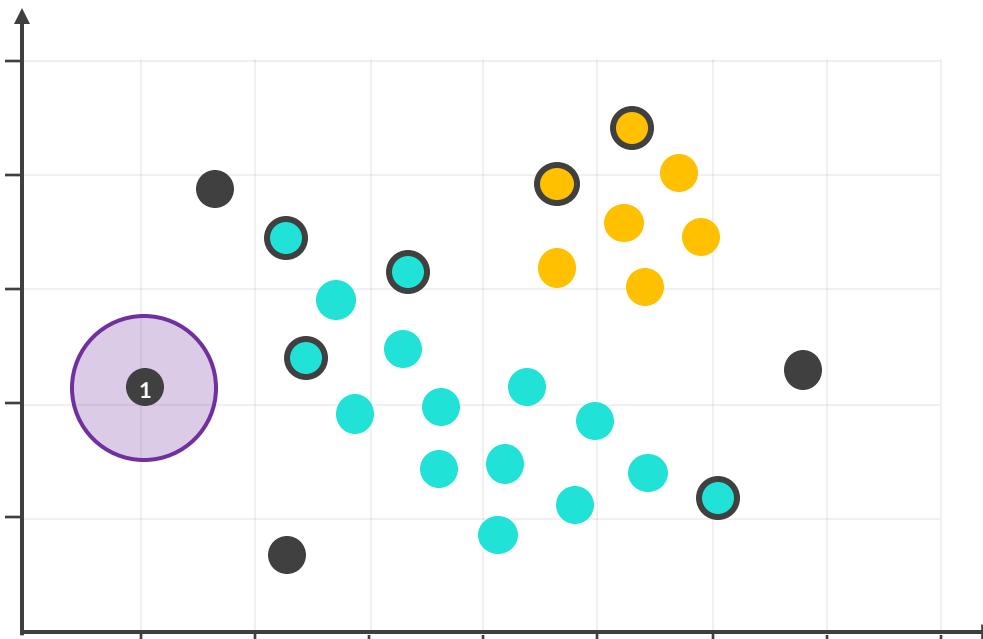
K-Means
Clustering

Hierarchical
Clustering

DBSCAN

Comparing
Models

STEP 5: Move on to another random point and repeat the same steps



eps = **0.75**
min_samples = **4**



DBSCAN

Clustering Basics

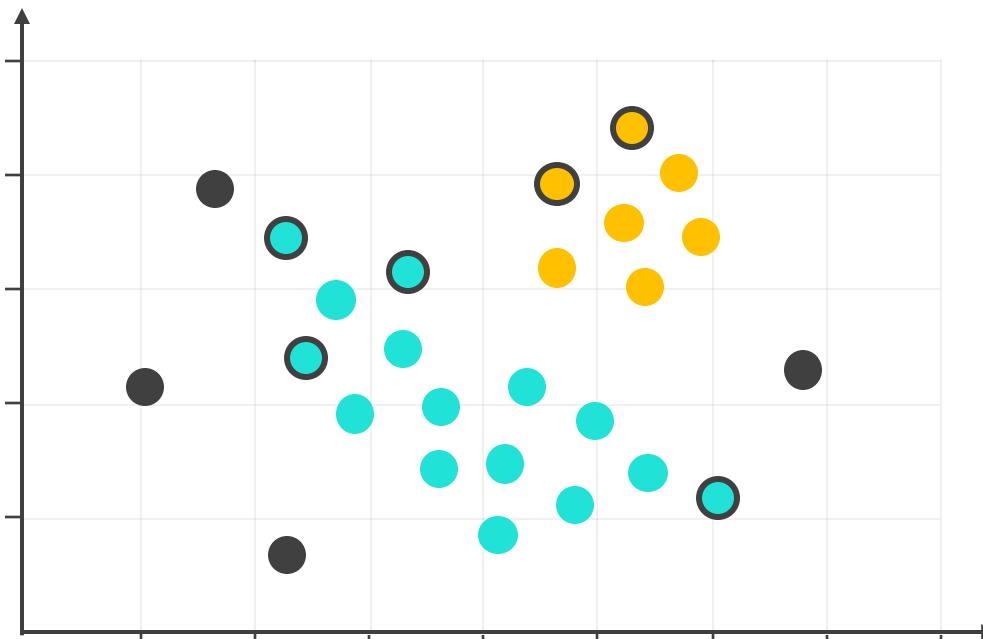
K-Means
Clustering

Hierarchical
Clustering

DBSCAN

Comparing
Models

STEP 5: Move on to another random point and repeat the same steps



eps = **0.75**
min_samples = **4**



DBSCAN

Clustering Basics

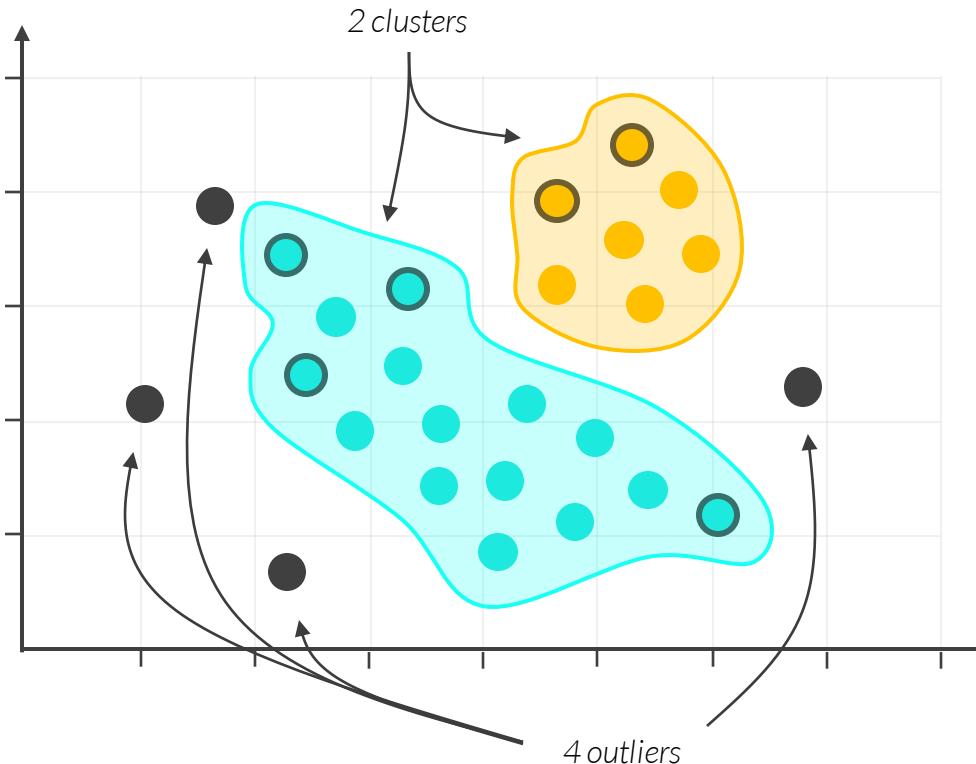
K-Means
Clustering

Hierarchical
Clustering

DBSCAN

Comparing
Models

STEP 5: Move on to another random point and repeat the same steps



eps = **0.75**
min_samples = **4**



DBSCAN DETAILED STEPS

Clustering Basics

K-Means
Clustering

Hierarchical
Clustering

DBSCAN

Comparing
Models

Here are some more detailed steps of how DBSCAN works:

1. Select a radius (eps) and a minimum number of points (min_samples)
2. Within a scatterplot, select a point at random and count the points within its radius
 - If it's greater than or equal to the " min_samples ", then start a cluster, label the point as a *core point*, and mark the points within its radius as a *neighbor*
 - If it's less than the " min_samples ", label the point as a *noise point* and move to step 5
3. Move to a neighbor and count the points within its radius
 - If it's greater than or equal to the " min_samples ", label it as a *core point*, and mark its *neighbors*
 - If it's less than the " min_samples ", but at least one of them is a core point, label it as a *border point*
 - If it's less than the " min_samples ", and none of them is a core point, label it as a *noise point*
4. Continue with another neighbor until all the points are labeled within the cluster
5. Move on to another random point and repeat the same steps



DBSCAN IN PYTHON

Clustering Basics

K-Means
Clustering

Hierarchical
Clustering

DBSCAN

Comparing
Models

You can use sklearn's **DBSCAN()** to perform DBSCAN clustering in Python

```
from sklearn.cluster import DBSCAN  
  
dbscan = DBSCAN(eps=0.5, min_samples=5)
```



The radius, or maximum distance
between neighboring points
(default is 0.5)

Minimum points in the radius
needed to become a core point,
including the point itself
(default is 5)



You may have noticed that there is **no random_state parameter**, even though DBSCAN starts at a random point. This is because sklearn's implementation of DBSCAN starts at the first point in the data set and moves on from there. You could attempt to get different labels by shuffling the data, but DBSCAN's labels don't vary as much as K-Mean Clustering's labels, so this isn't as big of an issue.



DBSCAN IN PYTHON

Clustering Basics

K-Means
Clustering

Hierarchical
Clustering

DBSCAN

Comparing
Models

You can use sklearn's **DBSCAN()** to perform DBSCAN clustering in Python

```
# import dbSCAN from sklearn
from sklearn.cluster import DBSCAN
```

```
# fit a dbSCAN model
dbSCAN = DBSCAN(eps=0.5, min_samples=5)
dbSCAN.fit(data)
```

▼ DBSCAN
DBSCAN()

```
# view the cluster assignments
dbSCAN.labels_
```

```
array([ 0, -1,  0,  1,  1,  1,  1,  1,  0,  1, -1,  0,  0,  0,  0,  0, -1,  1,
       0,  1, -1,  1, -1,  1,  1,  1,  1,  0,  0,  0, -1,  1, -1,  1,  1,
       1,  1,  1, -1,  0,  1,  1,  1, -1,  0, -1,  1,  0, -1,  0, -1,  1,
       1, -1,  1,  1,  0,  1, -1,  1,  0,  0,  0,  1,  1, -1,  0, -1,
      -1,  1,  1,  1,  1,  1,  1,  1,  1,  1, -1,  1,  1,  1,  0,  0,  0,
       1,  1,  0,  0,  1,  0,  1,  1,  1,  1,  1, -1,  1,  1,  0,  1,  0,
      -1,  1, -1,  1,  0,  1,  0, -1, -1, -1,  0,  0,  0,  1,  1,  1,  0,
       1,  1, -1, -1,  0,  0, -1, -1,  1,  1,  0,  0,  0,  1,  0,  1,  1,
       1,  1,  1,  1,  0,  0,  1, -1,  1, -1,  0])
```

You can view the cluster assignments using the `.labels_` attribute
(-1 values represent noise points)



There are too many noise
points (-1) here, let's tweak
`eps` and `min_samples` to
mainly see clusters instead



SILHOUETTE SCORE

Clustering Basics

K-Means Clustering

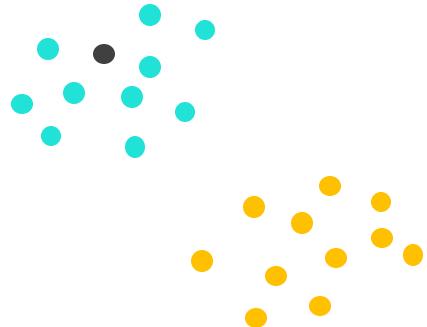
Hierarchical Clustering

DBSCAN

Comparing Models

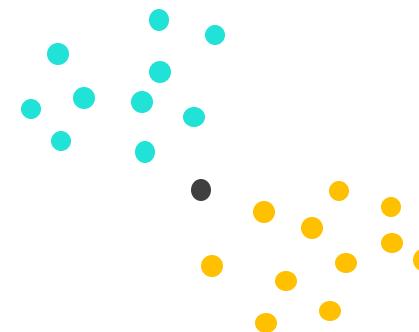
While intuition is a large part of comparing clustering models, you can also use metrics like **silhouette score** to help compare models

- Scores range from -1 to +1, with higher values meaning data points are highly matched to their own cluster and poorly matched to other clusters (*which is a good thing!*)



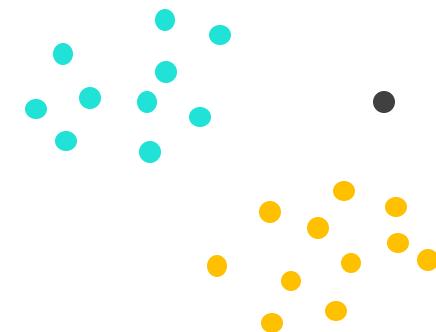
Silhouette score = positive

The data point fits well in its cluster and is distant from other clusters



Silhouette score ~ zero

The data point is near the boundary between two clusters



Silhouette score = negative

The data point doesn't fit well in its own cluster and might belong to a neighboring cluster



SILHOUETTE SCORE

Clustering Basics

K-Means Clustering

Hierarchical Clustering

DBSCAN

Comparing Models

The following formula is used to calculate the **silhouette score** for each data point:

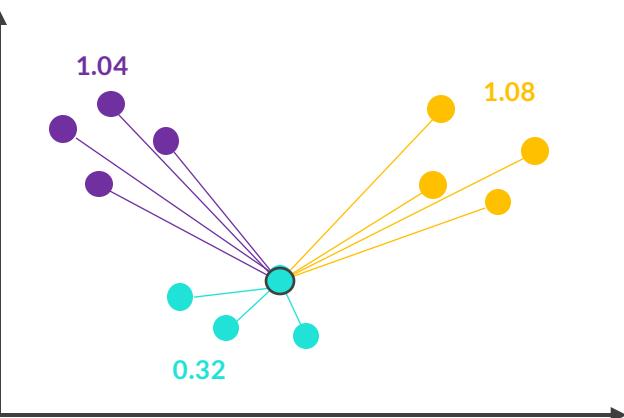
Smallest average distance to points in a different cluster

Average distance to other points in the same cluster

$$S(i) = \frac{b(i) - a(i)}{\max\{a(i), b(i)\}}$$

Silhouette score for the i^{th} data point

The denominator normalizes the difference so the silhouette score ends up between -1 and 1



$$\frac{1.04 - 0.32}{1.04} = 0.69$$



The data point fits well in its cluster!



To calculate the **overall silhouette score** for a particular model, average all the silhouette scores



SILHOUETTE SCORE IN PYTHON

Clustering Basics

K-Means
Clustering

Hierarchical
Clustering

DBSCAN

Comparing
Models

You can use sklearn's **silhouette_score()** function to calculate the silhouette score for the results of a clustering model in Python

```
from sklearn.metrics import silhouette_score  
  
silhouette_score(data, labels, metric='euclidean', sample_size=None)
```

0.3419620

The data you fit
the model on
(required)

The labels your
model generated
(required)

Distance calculation used to
calculate the silhouette score
(optional, default is "euclidean")

Size of the random subset of data
used to calculate the score
(optional, default is None)

ASSIGNMENT: DBSCAN

 **1 NEW MESSAGE**
March 19, 2024

From: **Clyde Clusters** (Sr. Data Scientist)
Subject: **DBSCAN Help**

Hi again!

To get one more perspective, please create DBSCAN models on the original and standardized data sets using these ranges:

- eps: 0.1 to 2, with steps of 0.1
- min_samples: 2 to 10, with steps of 1

Which combination of these values ends up with the highest silhouette score?

Thanks again for all your help this week!
Clyde

Reply **Forward**

Key Objectives

1. Loop through multiple “eps” and “min_samples” values to fit multiple DBSCAN models
2. Apply the function on both the original and standardized datasets
3. Find the highest silhouette score and note down the “eps” and “min_samples” values
4. Fit a final DBSCAN model with those “eps” and “min_samples” values and review the labels



CLUSTERING ALGORITHMS RECAP

Clustering Basics

K-Means Clustering

Hierarchical Clustering

DBSCAN

Comparing Models

Each clustering algorithm has **pros and cons** to keep in mind:

Model	Pros	Cons	In Practice
K-Means Clustering	<ul style="list-style-type: none">✓ Easy to understand & interpret✓ Scales well with large datasets	<ul style="list-style-type: none">✗ Must specify the number of clusters✗ Different initial centroids lead to different results✗ Assumes clusters are roughly spherical	Popular clustering model Typically the first choice when it comes to clustering
Hierarchical Clustering	<ul style="list-style-type: none">✓ No need to predefine "k" upfront✓ Can work with complex datasets	<ul style="list-style-type: none">✗ Does not scale well with large datasets✗ Data points can connect to outliers, which distorts distance calculations	Good for visualization Dendrogram allows you to explore the clusters visually
DBSCAN	<ul style="list-style-type: none">✓ No need to predefine "k" upfront✓ Can work with complex datasets	<ul style="list-style-type: none">✗ Does not scale well with large datasets✗ Hyperparameter tuning is challenging	Good for outliers Effectively identifies & handles noise points in the data



CLUSTERING ALGORITHMS RECAP

Clustering Basics

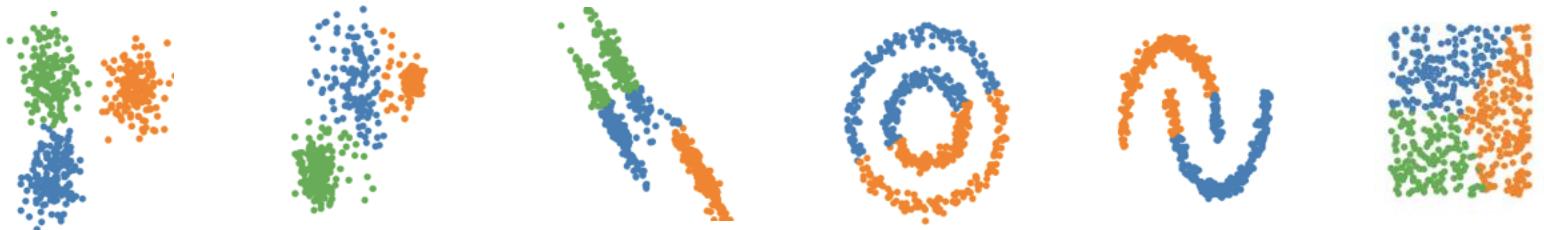
K-Means Clustering

Hierarchical Clustering

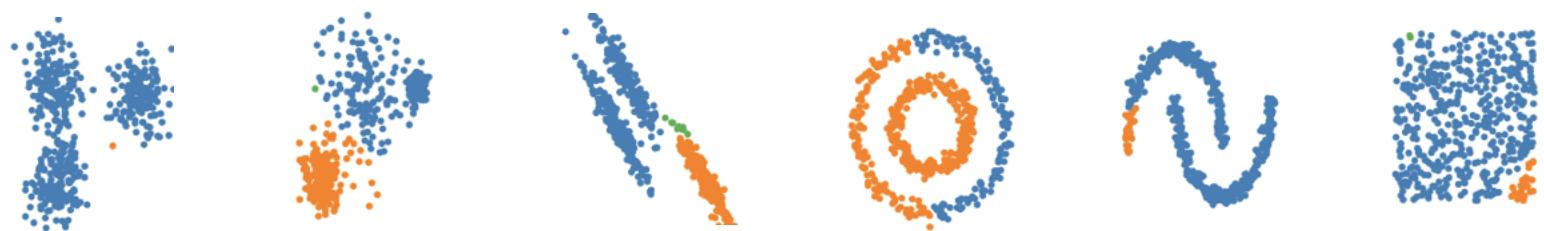
DBSCAN

Comparing Models

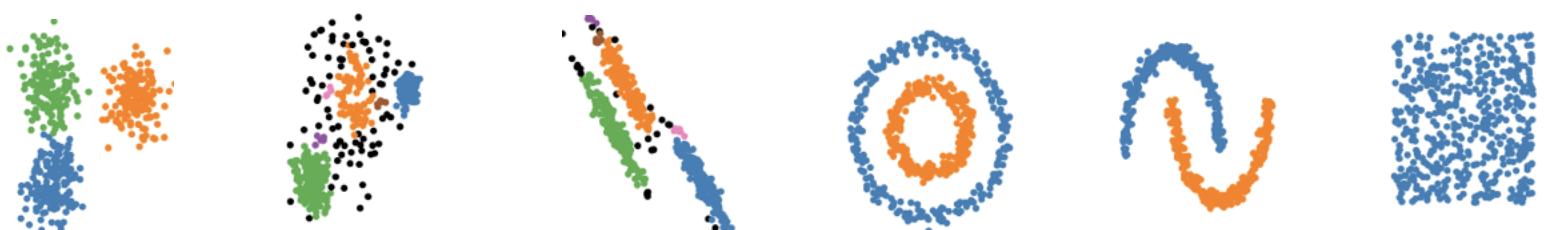
K-Means Clustering



Hierarchical Clustering



DBSCAN





CLUSTERING NEXT STEPS

Clustering Basics

K-Means
Clustering

Hierarchical
Clustering

DBSCAN

Comparing
Models

Once you've fit & interpreted your initial models, the potential **next steps** are to:

1 Compare clustering models

- **Silhouette Score:** Higher scores closer to 1 mean that clusters are better defined
- **Intuition:** At the end of the day, the main objective is to answer your business question, so you want to pick the clusters that make the most sense for you to make decisions

2 Label unseen data

- **Data Prep:** Before labeling unseen data, you must apply the same transformations you applied to the original data set (*feature scaling, etc.*) when creating the clusters
- **K-Means Clustering:** The `.predict` method lets you assign unseen data points to a cluster
- **Hierarchical Clustering & DBSCAN:** While there is no `.predict` method for these, you can label unseen data points by re-fitting the models (*you may need to update the parameters*)

KEY TAKEAWAYS



Clustering lets you find **groups of observations** similar to one another

- Common clustering techniques include K-Means Clustering, Hierarchical Clustering, and DBSCAN



In practice, follow the **clustering workflow**

- 1) Prepare your data for clustering (correct row granularity, non-null & numeric values, feature scaling, etc.)
- 2) Start with K-Means Clustering, then tune and select a “best” model
- 3) If you want better results, try Hierarchical Clustering or DBSCAN, then tune and select a “best” model
- 4) Compare models using silhouette scores & intuition, and select the one that best answers your business question

CLUSTERING PROJECT

PROJECT: CLUSTERING CLIENTS



THE SITUATION

You work as a Data Scientist for **Northwind Traders**, a wholesale distributor that supplies gourmet food products to restaurants, cafes, and specialty food retailers



THE ASSIGNMENT

You have access to their **yearly client report**, which includes the channel & region for each client, as well as their total spend by product categories in the last year
Your task is to **use clustering techniques** to understand the different client segments so the company can better cater to and support each type



THE OBJECTIVES

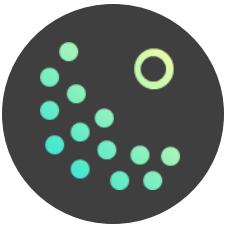
1. **Scale** the data
2. Apply 3 different **clustering techniques**
3. Compare the segments using silhouette score & intuition and **select the “best” segments**
4. **Predict** the cluster of a new client



NORTHWIND
TRADERS

ANOMALY DETECTION

ANOMALY DETECTION



In this section we'll introduce the concept of **anomaly detection** and cover two popular unsupervised learning techniques for finding anomalies: Isolation Forests and DBSCAN

TOPICS WE'LL COVER:

Anomaly Detection
Basics

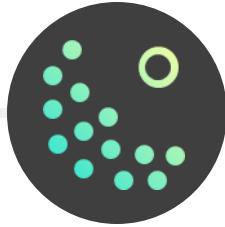
Isolation Forests

DBSCAN

Comparing Models

GOALS FOR THIS SECTION:

- Review where anomaly detection sits within the data science workflow
- Learn how popular unsupervised techniques for anomaly detection fundamentally work
- Apply and interpret the results of Isolation Forests and DBSCAN models for anomaly detection
- Compare & contrast unsupervised techniques for anomaly detection



ANOMALY DETECTION BASICS

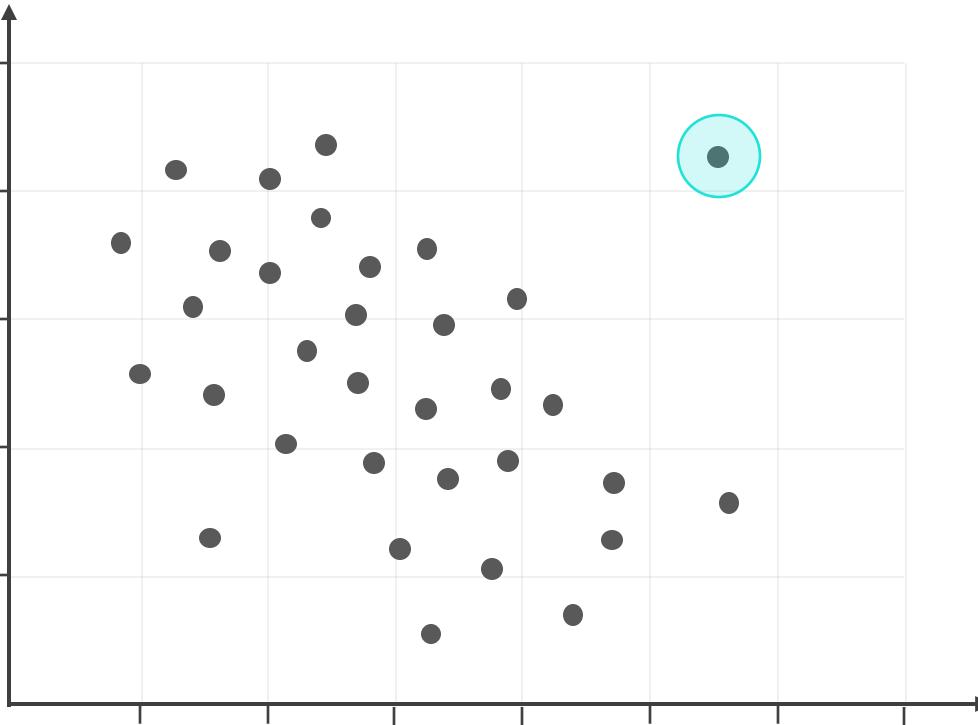
Anomaly
Detection Basics

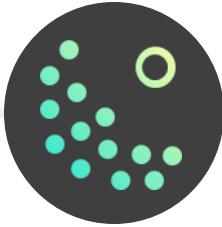
Isolation Forests

DBSCAN

Comparing
Models

Anomaly detection, also known as *outlier detection*, is used to identify observations in a data set that are significantly different from the others





ANOMALY DETECTION BASICS

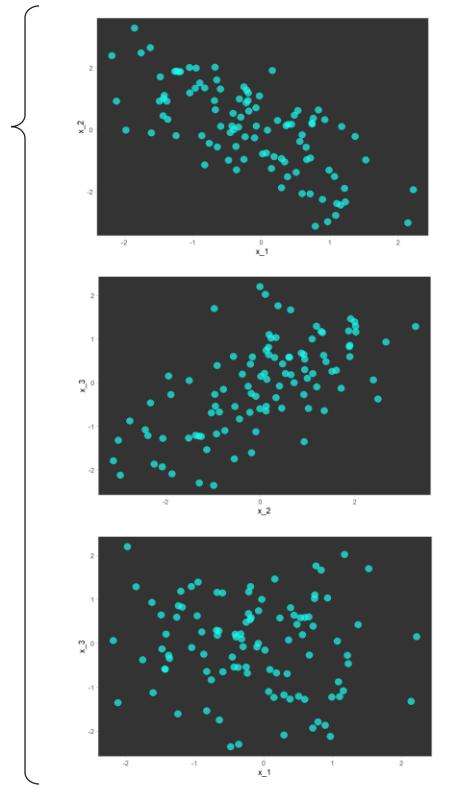
Anomaly
Detection Basics

Isolation Forests

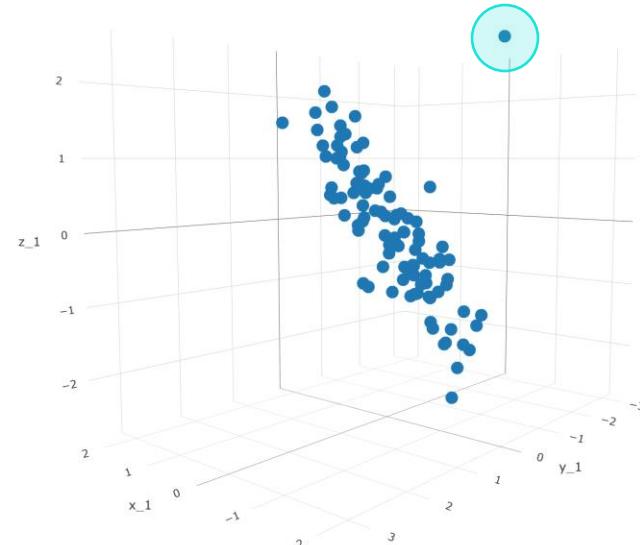
DBSCAN

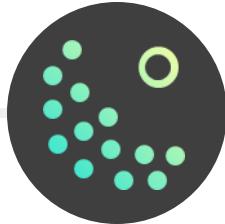
Comparing
Models

	A	B	C	D
1	id	x_1	x_2	x_3
2	1	0.0187	-0.7806	-0.1540
3	2	-0.1843	0.6036	0.5848
4	3	-1.3713	0.3314	-0.3448
5	4	-0.5992	1.3107	1.1476
6	5	0.2945	-0.9278	-1.1736
7	6	0.3898	0.1734	0.8111
8	7	-1.2081	1.8691	1.1873
9	8	-0.3637	-1.2944	-2.2954
10	9	-1.6267	2.6548	0.9313
11	10	-0.2565	1.3844	0.4804
12	11	1.1018	-2.3819	-1.2092
13	12	0.7558	0.3731	1.7597
14	13	-0.2382	-0.2259	-0.6799
15	14	0.9874	-1.3032	-1.2271
16	15	0.7414	0.1829	1.1006
17	16	0.0893	-0.0122	-0.5971
18	17	-0.9549	1.9949	1.3938
19	18	-0.1952	0.9370	0.5364
20	19	0.9255	0.3300	1.0325
21	20	0.4830	0.4679	0.4303
22	21	-0.5963	0.1149	-0.6460
23	22	-2.1853	2.3882	0.0615



This anomaly could not be detected via 2D scatter plots alone, but is clearly an anomaly in 3D





ANOMALY DETECTION IN THE DS WORKFLOW

Anomaly
Detection Basics

Isolation Forests

DBSCAN

Comparing
Models



Data Issues

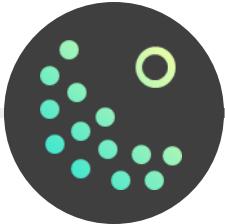
Identify data points to exclude to avoid skewing future analysis

- Typically called outlier detection
- Done using statistics or plots
- Can use models to detect outliers that are difficult or impossible to detect with statistics or plots alone

Insights

Identify data points to look into to provide meaningful information

- Typically called anomaly detection
- Done using either supervised or unsupervised learning models
- Can use unsupervised learning models if you don't have any information on past anomalies (*data is unlabeled*)



ANOMALY DETECTION APPROACHES

Anomaly
Detection Basics

Isolation Forests

DBSCAN

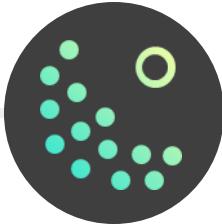
Comparing
Models

Popular **approaches to anomaly detection** include:

- **Statistical analysis:** flagging data points more than 3 standard deviations away from the mean
- **Data visualization:** visually identifying distant points by creating histograms, scatterplots, or boxplots
- **Supervised learning:** if you have information on past anomalies, you can apply classification algorithms like Random Forests to identify new ones
- **Unsupervised learning:** if you have unlabeled data, you need algorithms like Isolation Forests and DBSCAN



Many machine learning algorithms can be **modified for anomaly detection** including K-Nearest Neighbors, One-Class Support Vector Machines, Time Series Analysis, etc.



UNSUPERVISED LEARNING TECHNIQUES

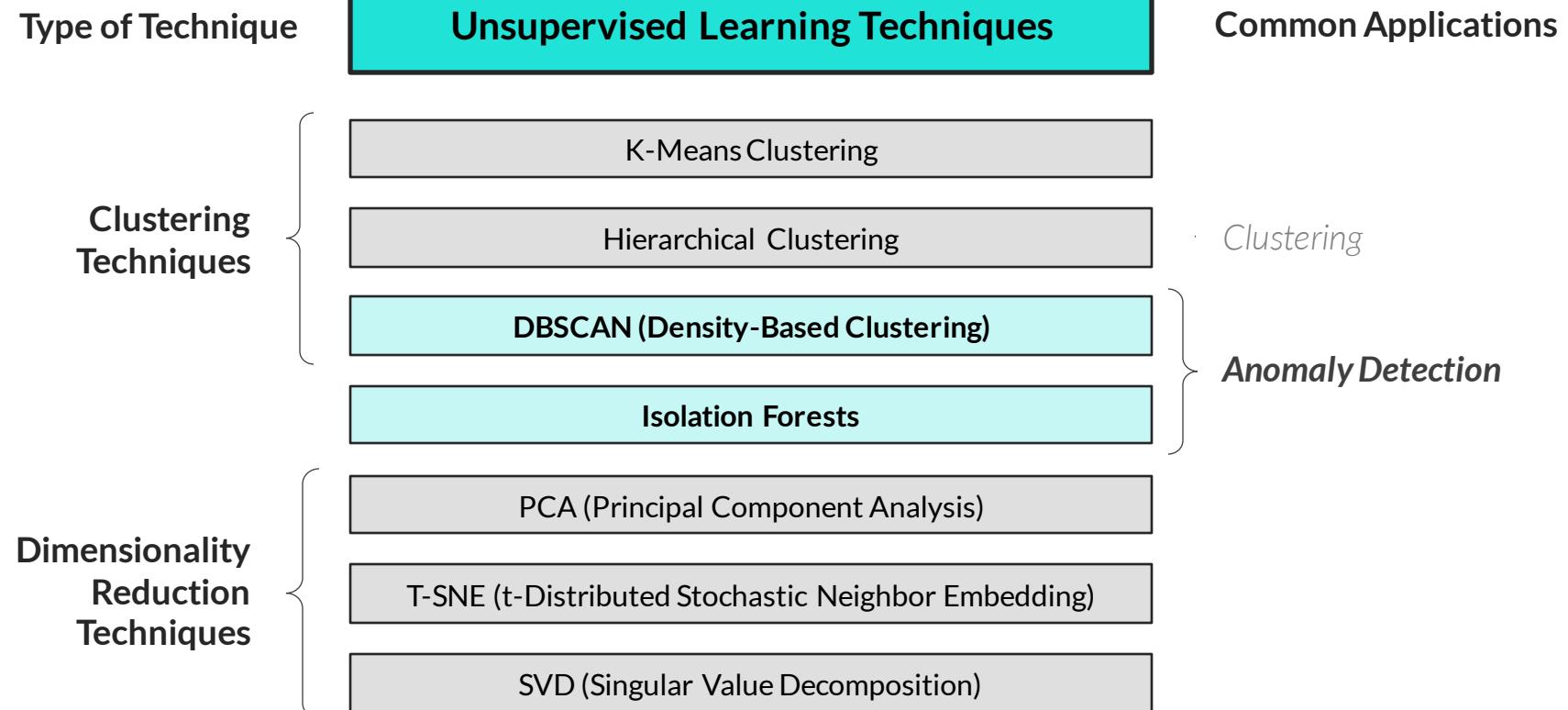
You can perform anomaly detection using **unsupervised learning techniques**

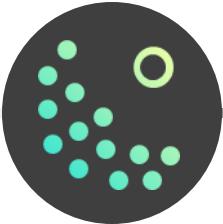
Anomaly
Detection Basics

Isolation Forests

DBSCAN

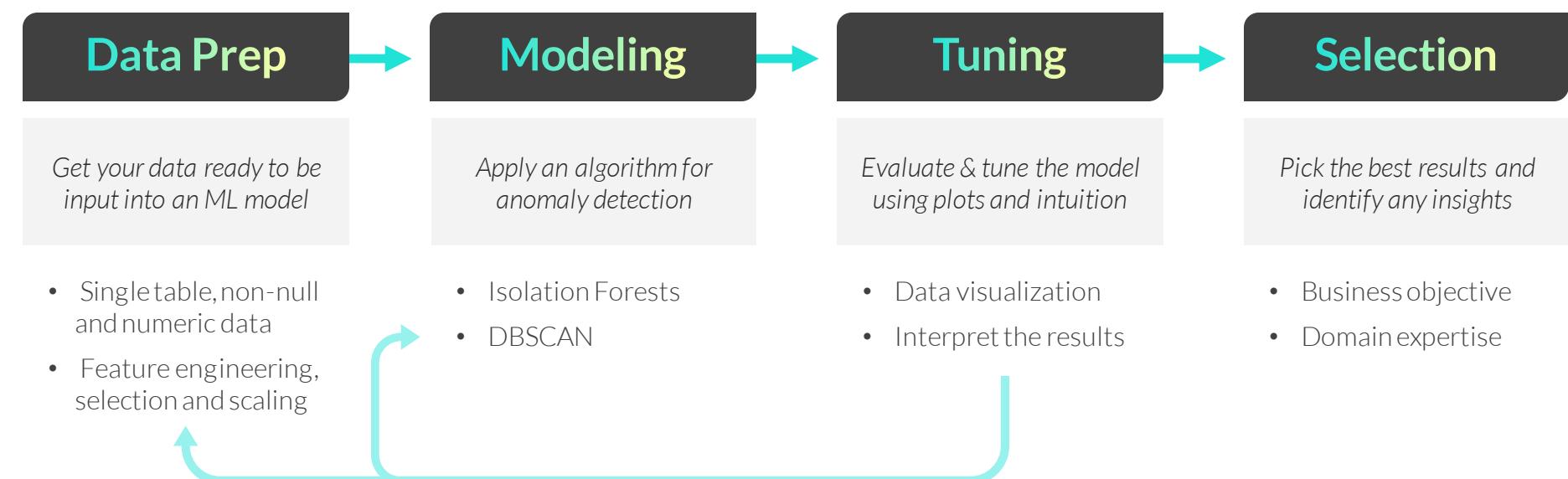
Comparing
Models



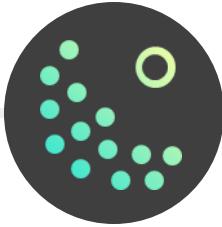


ANOMALY DETECTION WORKFLOW

The general **anomaly detection workflow** using unsupervised machine learning techniques consists of the following steps:



Remember, there's no "right" answer or single optimization metric when it comes to anomaly detection; the best outputs are the ones which help you **answer the question at hand** and make practical, data-driven business decisions



ISOLATION FORESTS

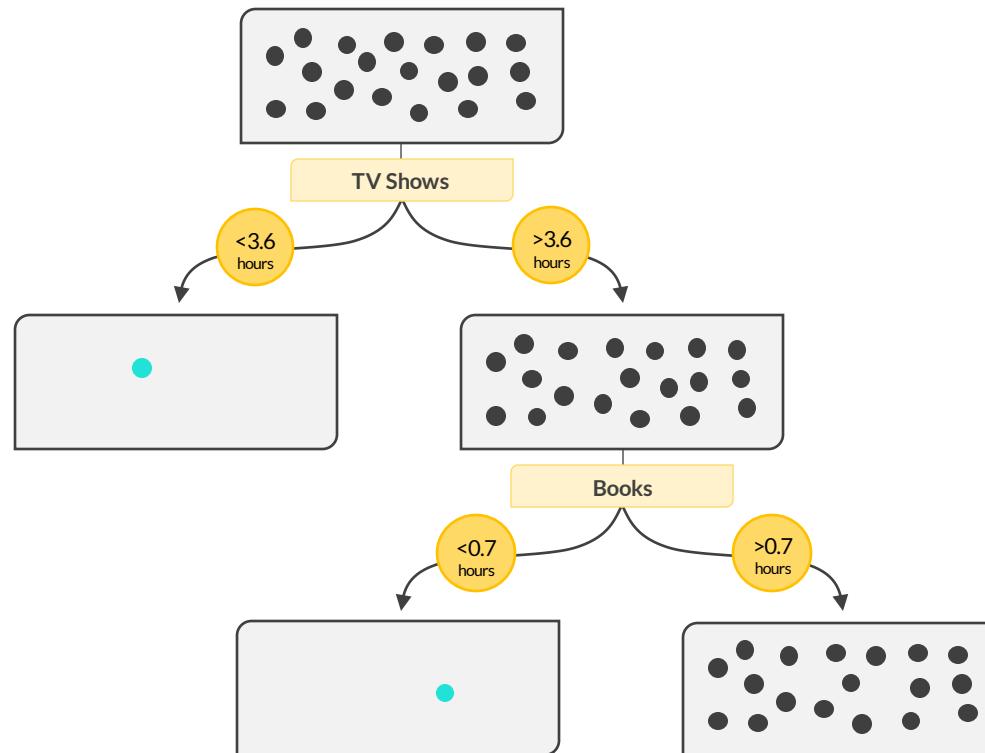
An **Isolation Forest** detects anomalies by building decision trees that split the data randomly and measure how quickly specific data points can be isolated

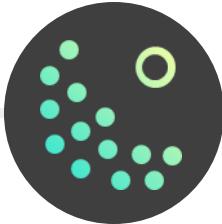
Anomaly
Detection Basics

Isolation Forests

DBSCAN

Comparing
Models





ISOLATION FORESTS

Anomaly
Detection Basics

Isolation Forests

DBSCAN

Comparing
Models

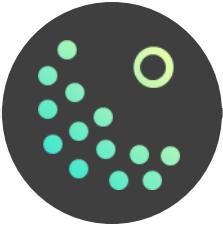
An **Isolation Forest** detects anomalies by building decision trees that split the data randomly and measure how quickly specific data points can be isolated

Here's how it works:

1. Randomly select a feature from the data and split the observations using a random threshold
 - If a value is under the threshold, it goes to one **branch**; otherwise, it goes to the other
2. Continue splitting the branches using random features and thresholds, creating a **tree** structure, until every data point is **isolated** or a maximum depth is reached
3. Repeat these steps using different features and splits to create multiple trees (aka a **forest**)
4. Calculate the **anomaly score** for each observation by averaging the number of splits it took to isolate (**path length**) in each tree – the lowest scores are anomalies!

Example use cases:

- Detect fraudulent activities in financial data or credit card transactions
- Detect unexpected malfunctions in sensor data or unusual test results in patient data



ISOLATION FORESTS

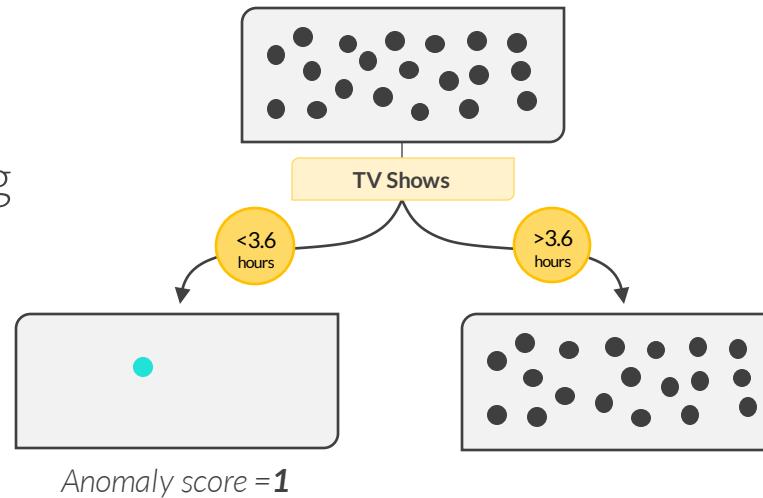
Anomaly
Detection Basics

Isolation Forests

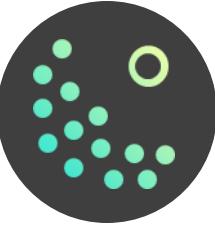
DBSCAN

Comparing
Models

STEP 1: Randomly select a feature from the data and split the observations using a random threshold



Features
Books
TV Shows
Video Games



ISOLATION FORESTS

Anomaly
Detection Basics

Isolation Forests

DBSCAN

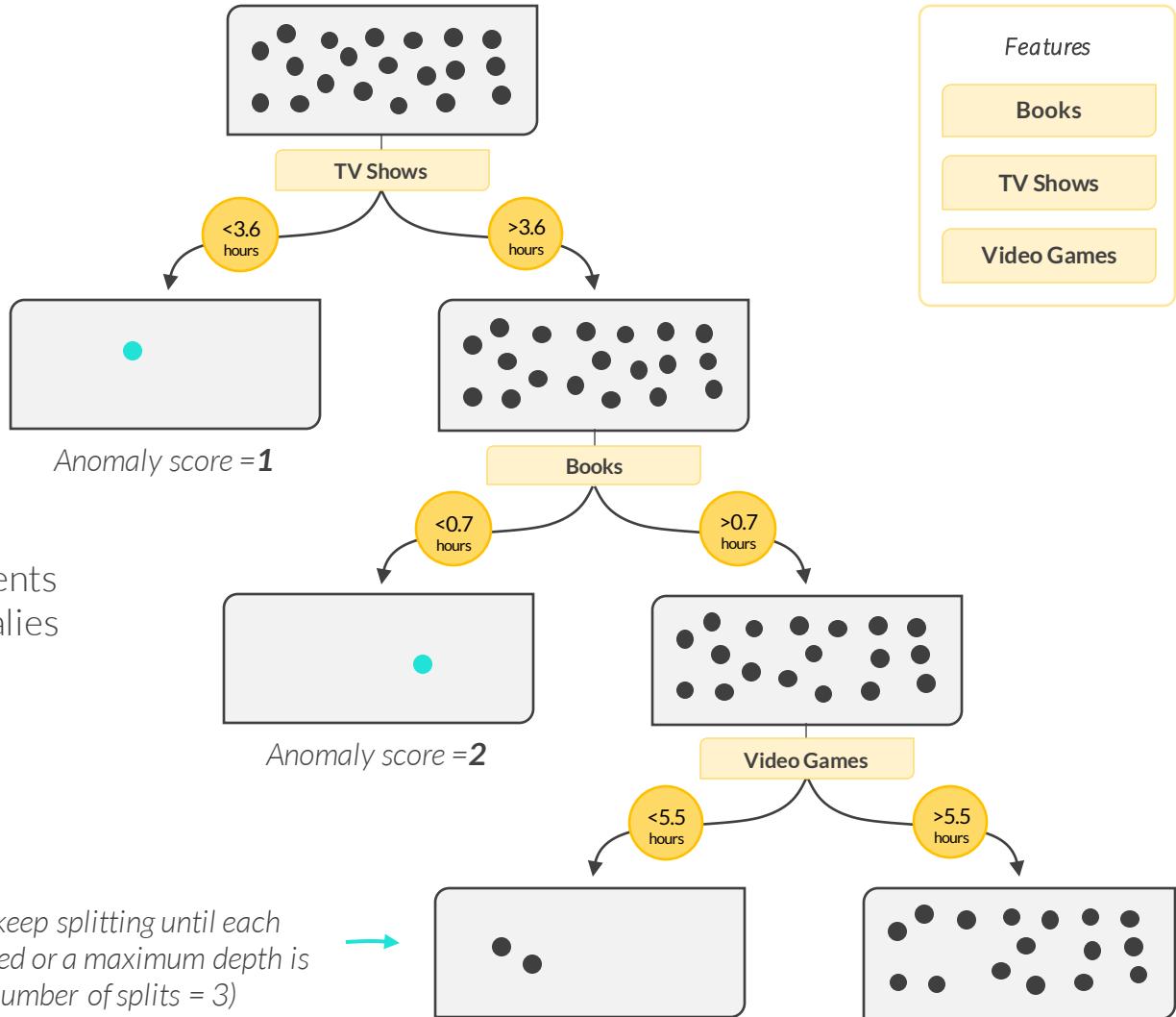
Comparing
Models

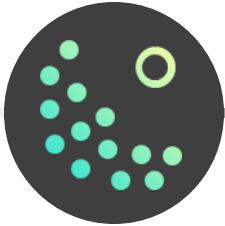
STEP 2: Continue splitting using random features and thresholds until every data point is isolated or a max depth is reached



These two students are likely anomalies

These would keep splitting until each point is isolated or a maximum depth is reached (i.e. number of splits = 3)





ISOLATION FORESTS

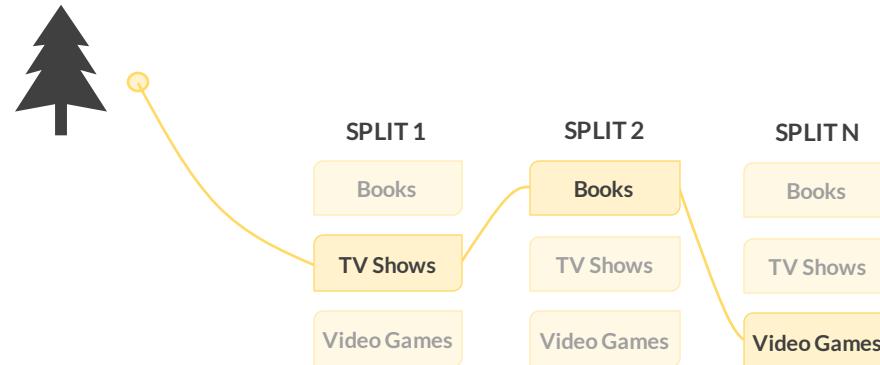
STEP 3: Repeat using different features and splits to create multiple trees

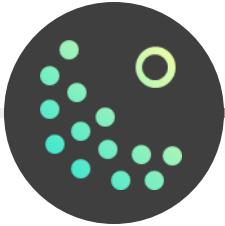
Anomaly
Detection Basics

Isolation Forests

DBSCAN

Comparing
Models





ISOLATION FORESTS

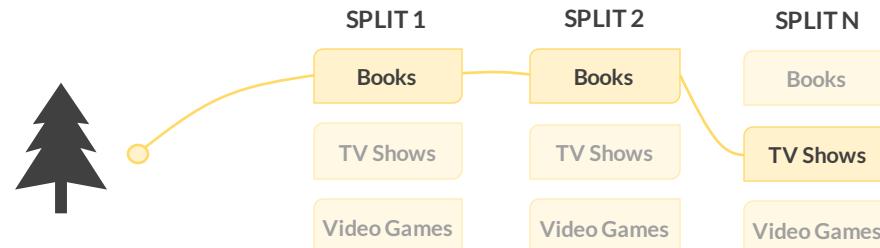
STEP 3: Repeat using different features and splits to create multiple trees

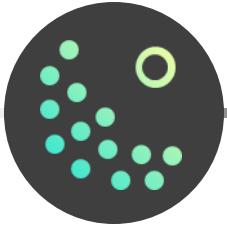
Anomaly
Detection Basics

Isolation Forests

DBSCAN

Comparing
Models





ISOLATION FORESTS

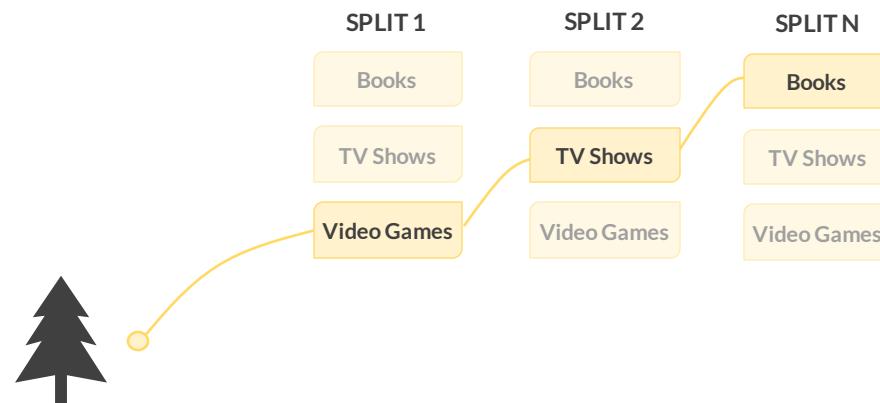
STEP 3: Repeat using different features and splits to create multiple trees

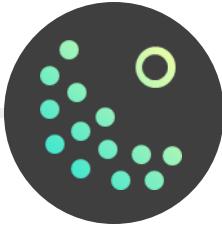
Anomaly
Detection Basics

Isolation Forests

DBSCAN

Comparing
Models





ISOLATION FORESTS

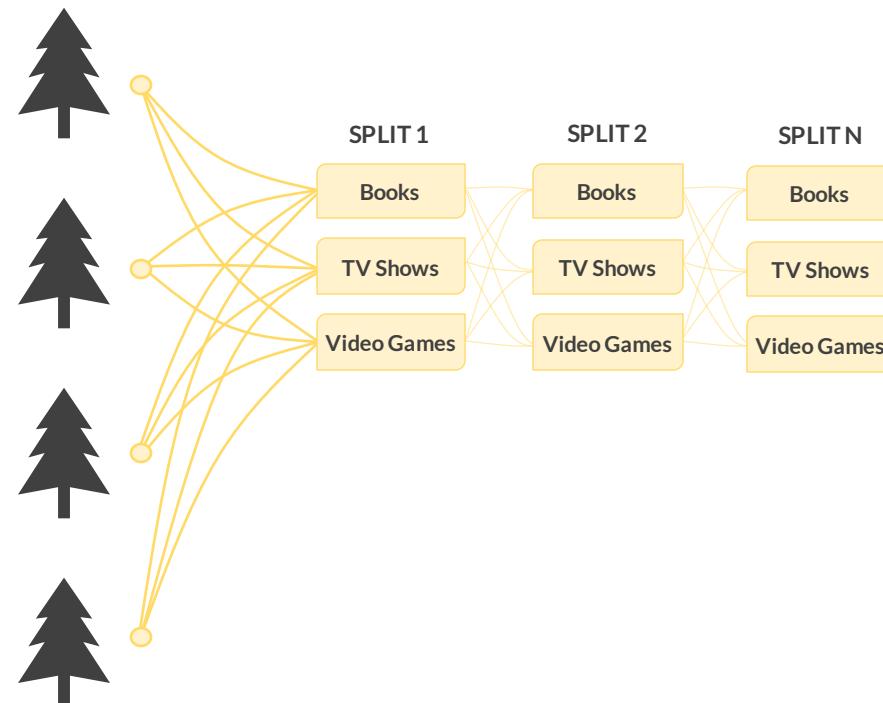
STEP 3: Repeat using different features and splits to create multiple trees

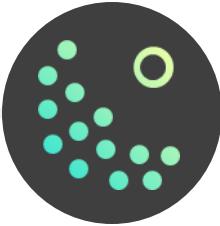
Anomaly
Detection Basics

Isolation Forests

DBSCAN

Comparing
Models





ISOLATION FORESTS

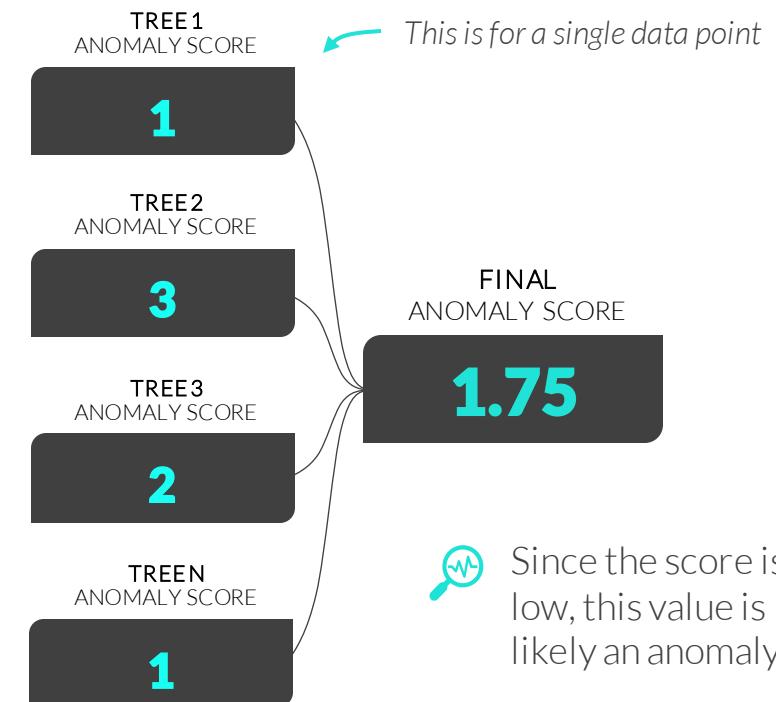
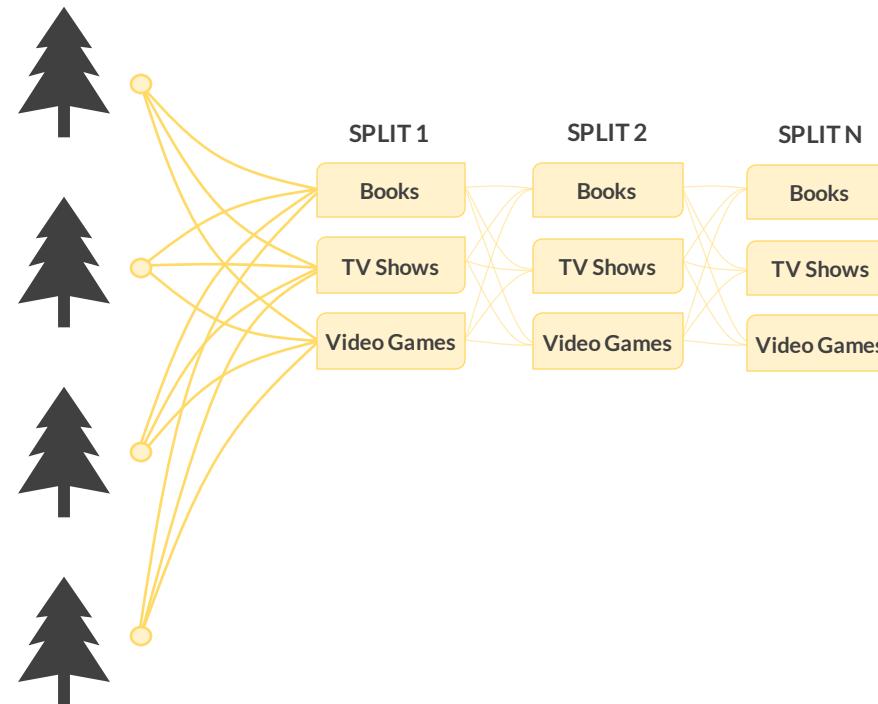
Anomaly
Detection Basics

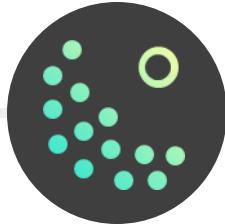
Isolation Forests

DBSCAN

Comparing
Models

STEP 4: Calculate the anomaly score for each observation by averaging the number of splits it took to isolate (path length) in each tree





ISOLATION FORESTS IN PYTHON

Anomaly
Detection Basics

Isolation Forests

DBSCAN

Comparing
Models

You can use sklearn's **IsolationForest()** to apply Isolation Forests in Python

```
from sklearn.ensemble import IsolationForest  
  
model = IsolationForest(contamination=0.02, random_state=42)
```



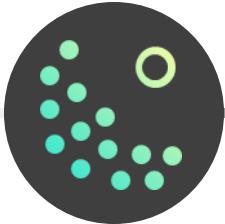
The percentage of data points
you'd like to flag as an anomaly
(optional, between 0 - 0.5)



Setting a random_state value
guarantees the same results
each time the model is fit



PRO TIP: Even though there is an option to set the contamination percentage
upfront, you can also decide the number of anomalies after fitting the model



ISOLATION FORESTS IN PYTHON

Anomaly
Detection Basics

Isolation Forests

DBSCAN

Comparing
Models

You can use sklearn's **IsolationForest()** to apply Isolation Forests in Python

```
# fit an isolation forest model with 2% of the data set as anomalies
from sklearn.ensemble import IsolationForest
```

```
model = IsolationForest(contamination=0.02)
model.fit(X)
```

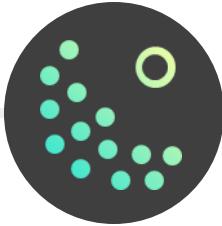
```
▼ IsolationForest
IsolationForest(contamination=0.02)
```

Once the model is fit, you can view the scaled anomaly scores (-0.5 to 0.5) using the **.decision_function()**

```
# view the anomaly scores and anomaly flags
df['anomaly_scores'] = model.decision_function(X)
df['anomaly'] = model.predict(X)
df.sort_values('anomaly_scores').head()
```

	name	books	tv_shows	video_games	anomaly_scores	anomaly
110	Octavia	5.0	5.7	7.9	-0.025049	-1
15	Avery	1.0	6.6	5.7	-0.002304	-1
147	Zara	5.5	5.7	7.7	-0.000180	-1
29	Clementine	6.2	5.4	7.2	0.000004	1
58	Hailey	2.5	3.0	5.0	0.001814	1

Use **.predict()** to flag anomalies (-1) as determined by the contamination you set



VISUALIZING ANOMALIES

You can **visualize anomalies** by using the anomaly flags as the “hue” in a pair plot

Anomaly
Detection Basics

Isolation Forests

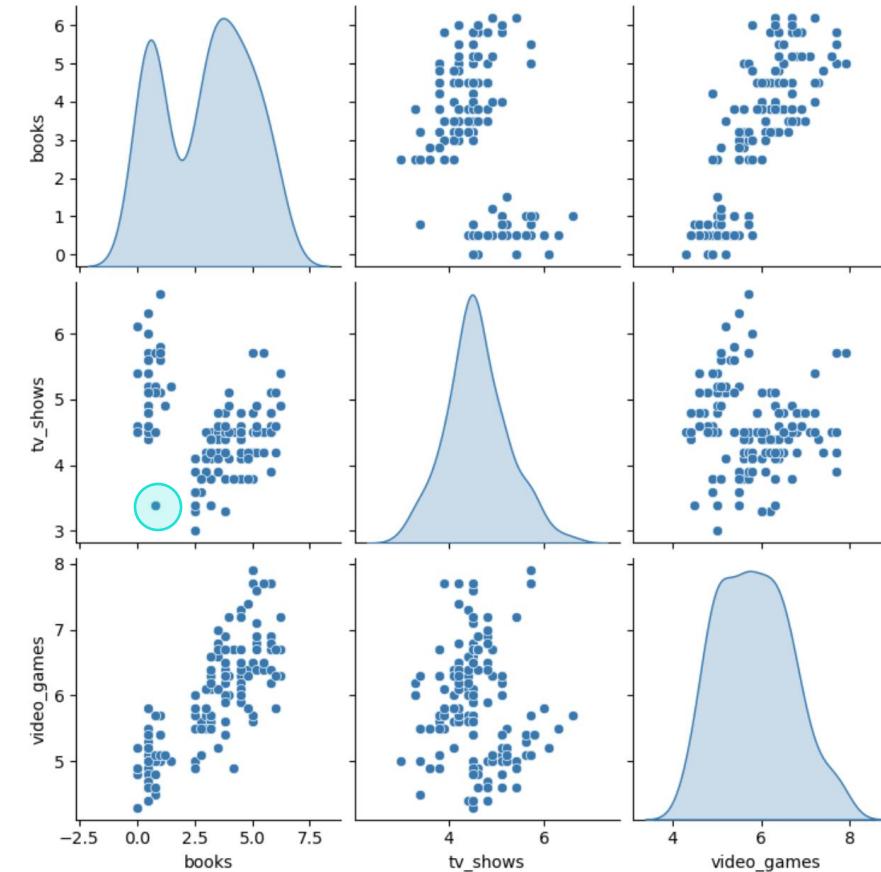
DBSCAN

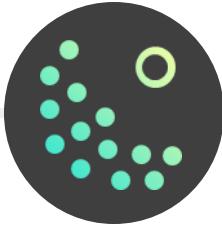
Comparing
Models

```
import seaborn as sns  
  
sns.pairplot(df);
```



Visually, this student watches less TV
and reads fewer books than the rest





VISUALIZING ANOMALIES

You can **visualize anomalies** by using the anomaly flags as the “hue” in a pair plot

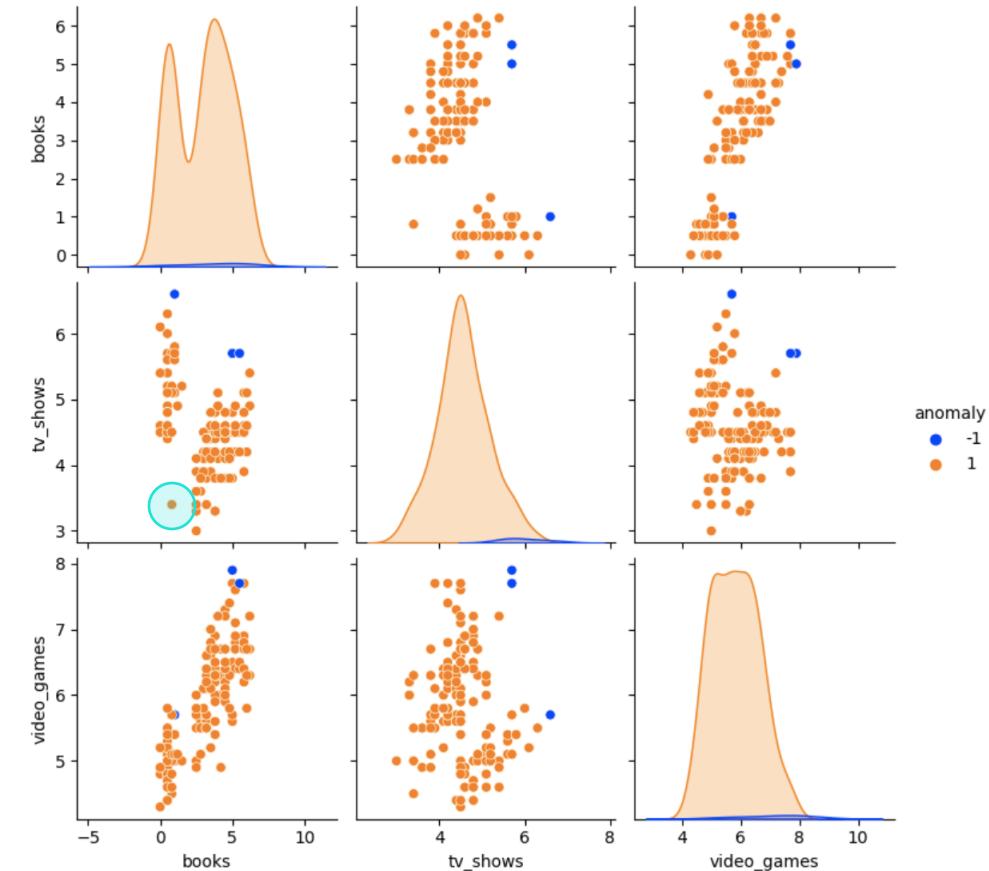
Anomaly
Detection Basics

Isolation Forests

DBSCAN

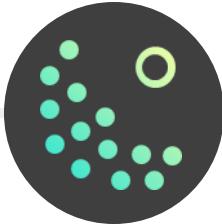
Comparing
Models

```
sns.pairplot(df,  
             hue='anomaly',  
             palette='bright');
```



The student that looked like an anomaly in two dimensions was not flagged using Isolation Forests

Instead, the anomalies were the students that watched a lot of TV and played a lot of video games



TUNING & INTERPRETING ISOLATION FORESTS

Anomaly
Detection Basics

Isolation Forests

DBSCAN

Comparing
Models

After you view the results of your first Isolation Forest model, you can:

- **Tune the model** by updating the contamination percent to flag more or fewer anomalies
- **Interpret the results** by filtering on the anomaly labels or sorting the anomaly scores

```
# update the model to flag 5% of rows as anomalies
model5 = IsolationForest(contamination=0.05)
model5.fit(X)
```

```
▼      IsolationForest
IsolationForest(contamination=0.05)
```

```
# view the anomaly flags
df['anomaly_5'] = model5.predict(X)
df.sort_values('anomaly_5').head(10)
```



	name	books	tv_shows	video_games	anomaly_scores_5	anomaly_5
19	Bianca	5.8	3.9	7.7	-0.014885	-1
147	Zara	5.5	5.7	7.7	-0.022450	-1
148	Zoe	0.0	6.1	5.2	-0.006511	-1
37	Elena	0.8	3.4	4.5	-0.042216	-1
110	Octavia	5.0	5.7	7.9	-0.051103	-1
15	Avery	1.0	6.6	5.7	-0.037016	-1
42	Elizabeth	0.0	4.5	4.3	-0.003259	-1
29	Clementine	6.2	5.4	7.2	-0.035113	-1
98	Melody	3.2	4.4	6.4	0.163342	1
99	Mia	0.5	5.1	5.0	0.194472	1



Based on your objective, you may decide to either exclude the anomalies before doing any further analysis or dig into them to discover insights



Some “anomaly” students consumed a ton of entertainment, while others read almost no books

ASSIGNMENT: ISOLATION FORESTS

 **NEW MESSAGE**
March 20, 2024

From: **Molly Anomaly** (Researcher)
Subject: Identify Tourist Anomalies

Hi there!

We just received the results of our quarterly tourist survey, where we have tourists rate their experiences at our town's museums, parks, restaurants, and nightlife locations.

There are always a few tourists whose feedback really skews our results. Could you identify them using that anomaly detection technique you mentioned earlier (Isolation Forests)?

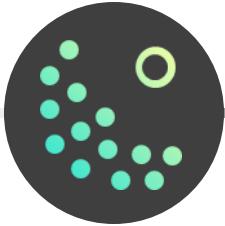
Thanks!
Molly

 tripadvisor_reviews.csv

Reply Forward

Key Objectives

1. Open the tripadvisor_reviews.csv file, remove the user_id column and view the range of each rating
2. Visualize the data using a seaborn pair plot
3. Fit an Isolation Forest model using a contamination of 0.01
4. Visualize the anomalies on the seaborn pair plot
5. Notice where there are anomalies in the pair plot
6. Modify the contamination to 0.005, visualize the anomalies and note the differences



DBSCAN FOR ANOMALY DETECTION

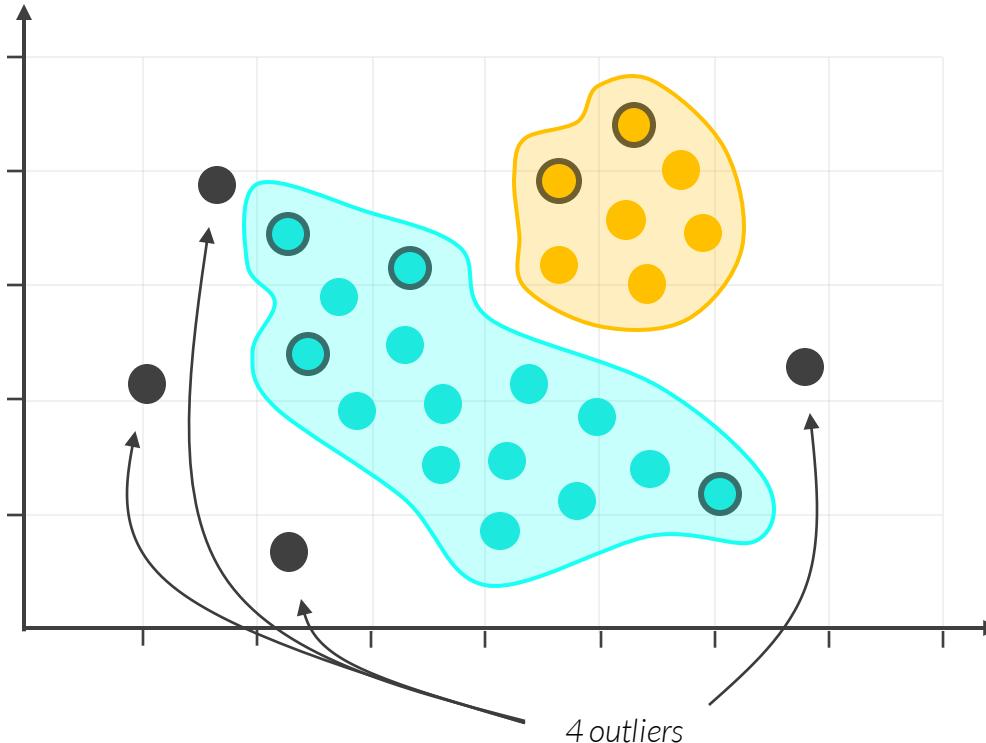
Anomaly
Detection Basics

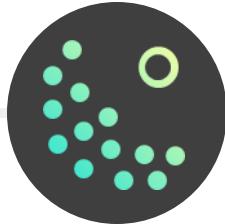
Isolation Forests

DBSCAN

Comparing
Models

DBSCAN (Density-Based Spatial Clustering of Applications with Noise), which is a clustering technique we covered, can also be used for anomaly detection





DBSCAN FOR ANOMALY DETECTION IN PYTHON

Anomaly
Detection Basics

Isolation Forests

DBSCAN

Comparing
Models

You can use sklearn's **DBSCAN()** to apply the DBSCAN algorithm in Python

```
from sklearn.cluster import DBSCAN  
  
dbscan = DBSCAN(eps=0.5, min_samples=5)
```

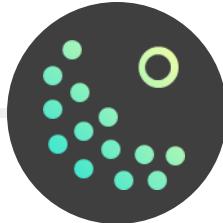


The radius, or maximum distance
between neighboring points
(default is 0.5)

Minimum points in the radius
needed to become a core point,
including the point itself
(default is 5)



Since DBSCAN is a distance-based algorithm, the **data should be scaled** before being input into the model.
This is different than Isolation Forests, which is NOT distance-based, so no scaling is needed



DBSCAN FOR ANOMALY DETECTION IN PYTHON

Anomaly
Detection Basics

Isolation Forests

DBSCAN

Comparing
Models

You can use sklearn's **DBSCAN()** to apply the DBSCAN algorithm in Python

```
# import dbSCAN from sklearn
from sklearn.cluster import DBSCAN
```

```
# fit a dbSCAN model
dbSCAN = DBSCAN(eps=0.5, min_samples=5)
dbSCAN.fit(data)
```

▼ DBSCAN
DBSCAN()

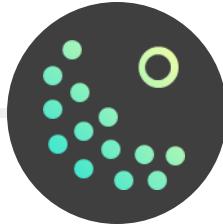
```
# view the cluster assignments
dbSCAN.labels_
```

```
array([ 0, -1,  0,  1,  1,  1,  1,  1,  0,  1, -1,  0,  0,  0,  0, -1,  1,
       0,  1, -1,  1, -1,  1,  1,  1,  1,  0,  0,  0, -1,  1, -1,  1,  1,
       1,  1,  1, -1,  0,  1,  1,  1, -1,  0, -1,  1,  0, -1,  0, -1,  1,
       1, -1,  1,  1,  0,  1, -1,  1,  0,  0,  0,  1,  1, -1,  0, -1,
      -1,  1,  1,  1,  1,  1,  1,  1,  1,  1, -1,  1,  1,  1,  0,  0,  0,
       1,  1,  0,  0,  1,  0,  1,  1,  1,  1,  1, -1,  1,  1,  0,  1,  0,
      -1,  1, -1,  1,  0,  1,  0, -1, -1, -1,  0,  0,  0,  1,  1,  1,  0,
       1,  1, -1, -1,  0,  0, -1, -1,  1,  1,  0,  0,  0,  1,  0,  1,  1,
       1,  1,  1,  1,  0,  0,  1, -1,  1,  1, -1, -1,  0])
```

You can view the cluster assignments using the `.labels_` attribute
(-1 values represent noise points)



For anomaly detection, the cluster labels are irrelevant, but **the noise points are the anomalies we're looking for!**



VISUALIZING DBSCAN ANOMALIES

You can **visualize anomalies** by using the anomaly labels as the “hue” in a pair plot

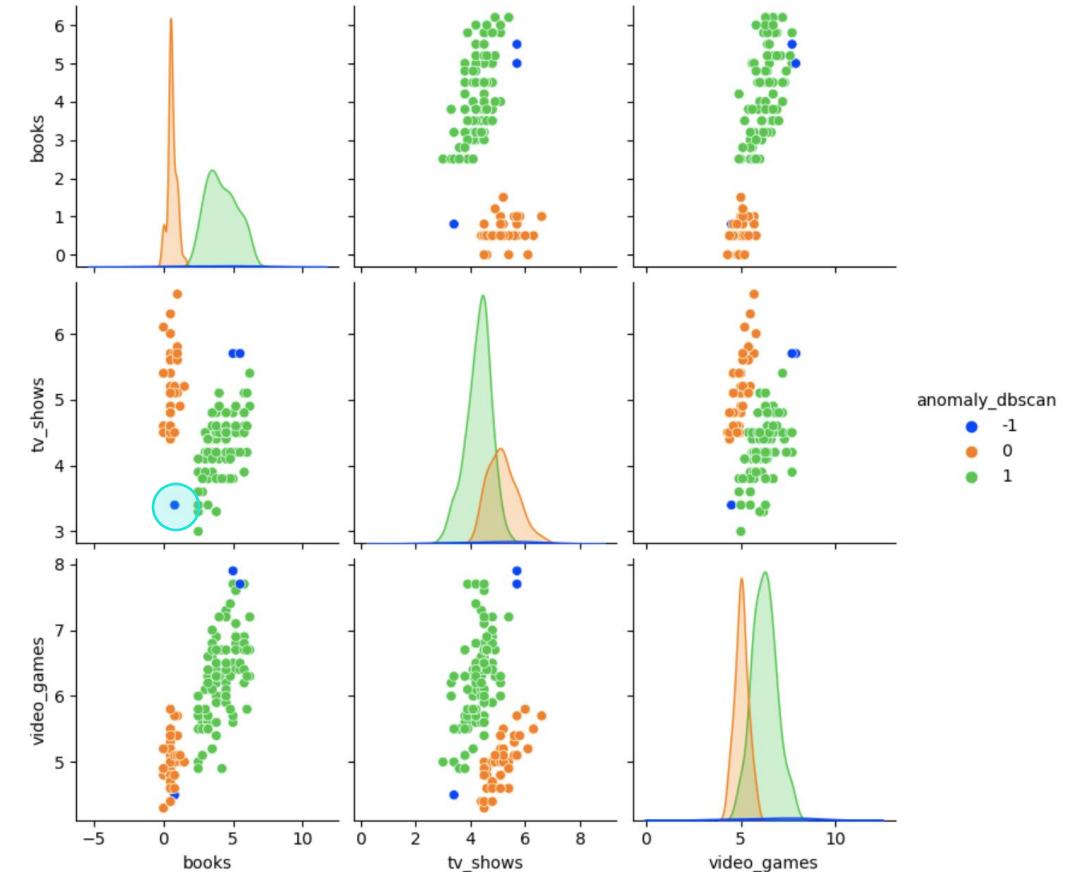
Anomaly
Detection Basics

Isolation Forests

DBSCAN

Comparing
Models

```
sns.pairplot(df,  
             hue='anomaly_dbSCAN',  
             palette='bright');
```



The student that looked like an anomaly in two dimensions WAS flagged using DBSCAN

Additionally, the anomalies were the students that consumed the most entertainment

ASSIGNMENT: DBSCAN FOR ANOMALY DETECTION

 **NEW MESSAGE**
March 21, 2024

From: **Molly Anomaly** (Researcher)
Subject: **RE: Identify Tourist Anomalies**

Hi again!

Thanks for the Isolation Forest results earlier - those anomalies were very interesting.

Can you try using another technique on the data set to confirm our results?

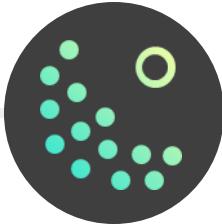
This time, please give DBSCAN a try and visualize the results.

Thanks!
Molly

[Reply](#) [Forward](#)

Key Objectives

1. Paste the DBSCAN function from the anomaly detection demo notebook
2. Apply the function on the tourist rating data set
3. Find the highest silhouette score and note down the eps and min_samples values
4. Fit a single DBSCAN model using those eps and min_sample values
5. Note the anomalies (-1) and visualize them on a pair plot



RECAP: ANOMALY DETECTION

Each anomaly detection technique has **pros and cons** to keep in mind:

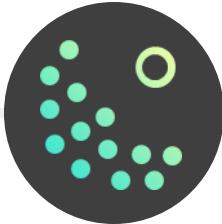
Anomaly
Detection Basics

Isolation Forests

DBSCAN

Comparing
Models

Model	Pros	Cons	In Practice
Isolation Forests	<ul style="list-style-type: none">✓ Great for high-dimensional data✓ Computationally efficient✓ Great for global anomalies	<ul style="list-style-type: none">✗ Not as good for low-dimensional data✗ Not as good for local anomalies	Popular anomaly detection choice Typically the first choice when it comes to anomaly detection
DBSCAN	<ul style="list-style-type: none">✓ Handles complex cluster shapes✓ Great for local anomalies	<ul style="list-style-type: none">✗ Computationally intensive✗ Hyperparameter tuning is challenging	Best for local anomaly detection Lets you identify local anomalies based on the internal clusters in the data



RECAP: CLUSTERING & ANOMALY DETECTION

So far, we've covered two unsupervised learning concepts:

Anomaly
Detection Basics

Isolation Forests

DBSCAN

Comparing
Models



Clustering

Used to identify data points that are **similar** to one another

Unsupervised learning techniques:

- K-Means Clustering
- Hierarchical Clustering
- DBSCAN



Anomaly Detection

Used to identify data points that are **different** from the rest

Unsupervised learning techniques:

- Isolation Forests
- DBSCAN



Although they are different concepts, the common thread is that you're comparing data points to determine how similar and different they are from one another.

Unsupervised learning is all about **finding relationships in data**, and these are two ways to capture that

KEY TAKEAWAYS



Anomaly detection aims to find observations **different than the rest**

- Common unsupervised techniques for anomaly detection are Isolation Forests and DBSCAN
- There are other ways to detect anomalies, including statistics, data visualization, and supervised learning

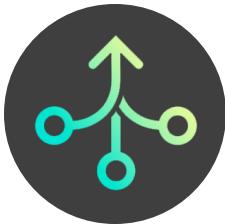


For unsupervised learning, follow the **anomaly detection workflow**

- 1) Prep your data for anomaly detection (scaling is not required for Isolation Forests, but it is for DBSCAN)
- 2) Start with Isolation Forests, then visualize, tune, and select a model
- 3) Try DBSCAN for different results, then visualize, tune and select a model
- 4) Compare the anomalies from all the models using data visualization, exploratory analysis, and your intuition
- 5) Select the model that best addresses your business question
- 6) You can choose to further explore or exclude anomalies detected by one or more models

DIMENSIONALITY REDUCTION

DIMENSIONALITY REDUCTION



In this section we'll introduce the fundamentals of **dimensionality reduction** and cover two popular techniques: Principal Component Analysis (PCA) and t-SNE

TOPICS WE'LL COVER:

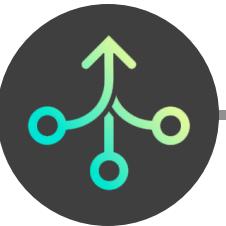
Dimensionality Reduction

PCA

t-SNE

GOALS FOR THIS SECTION:

- Review where dimensionality reduction sits within the data science workflow
- Learn how popular dimensionality reduction techniques fundamentally work
- Apply and interpret the results of PCA and t-SNE models in Python



DIMENSIONALITY REDUCTION BASICS

Dimensionality Reduction

PCA

t-SNE



	books	tv_shows	video_games		pc1	pc2
0	0.5	4.6	4.9	0	-2.649291	-0.276055
1	0.0	4.5	4.8	1	-3.137860	-0.384986
2	0.5	4.5	5.0	2	-2.603177	-0.322388
3	3.5	4.5	6.6	3	0.742853	0.227407
4	2.8	3.8	5.6	4	-0.172578	-0.802994
5	5.8	4.6	6.9	5	2.976964	0.335225
6	4.2	4.5	6.7	6	1.429338	0.236652
7	3.2	4.5	5.6	7	0.118587	-0.193404
8	0.0	4.6	4.9	8	-3.115006	-0.251550
9	4.0	4.1	6.0	9	1.048184	-0.417945

3 dimensions

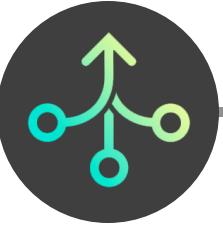
2 dimensions

The “pc1” and “pc2” columns each hold a combination of information from the “books”, “tv_shows” and “video_games” columns



How is this different from feature selection?

- Feature selection involves dropping entire columns
- Dimensionality reduction involves transforming the original columns

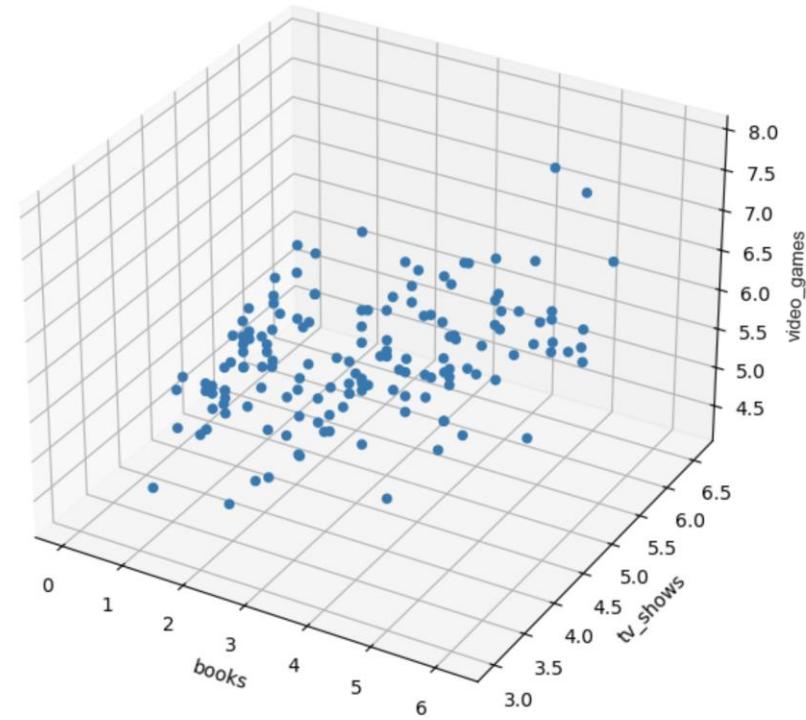


DIMENSIONALITY REDUCTION BASICS

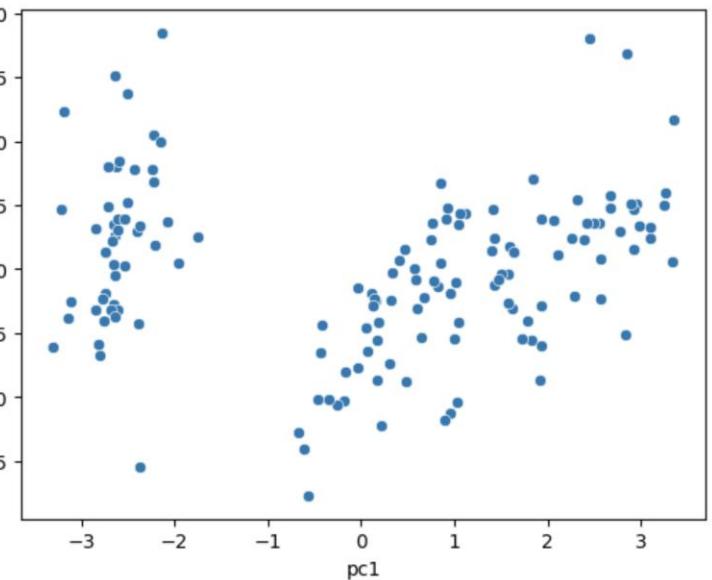
Dimensionality Reduction

PCA

t-SNE



3 dimensions



2 dimensions



Notice how with fewer dimensions it's easier to see clusters in the data!



WHY REDUCE DIMENSIONS?

Dimensionality Reduction

PCA

t-SNE

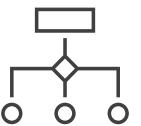


EDA

It's difficult for humans to visualize data with more than 2 or 3 dimensions clearly

Solution:

- Using dimensionality reduction, 4+ columns can be reduced to 2 or 3



Supervised Learning

Predictive models fit on “wide” data don’t perform as well as those fit on “long” data

Solution:

- By reducing dimensions in feature engineering, the data gets “narrower”



Unsupervised Learning

Data points spread out with more dimensions, and their distances become more similar
(curse of dimensionality)

Solution:

- By reducing dimensions, distance calculations become more meaningful

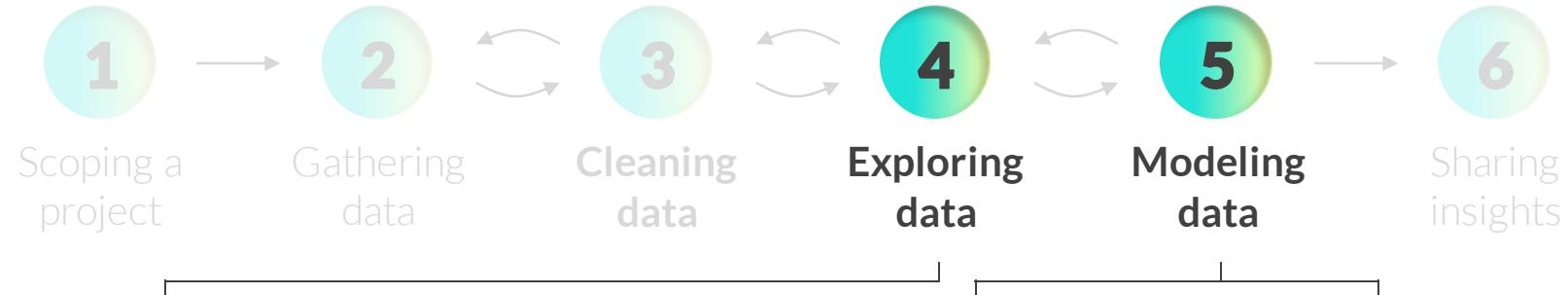


DIMENSIONALITY REDUCTION IN DS WORKFLOW

Dimensionality Reduction

PCA

t-SNE



PCA / t-SNE

Can be used as a machine learning alternative to **Data Visualization**

- Data is typically visualized using two dimensions (x and y-axes)
- By using PCA or t-SNE, high-dimensional data can also be visualized in two dimensions

PCA

Can be used for **Feature Engineering** when preparing supervised learning

- Feature selection or engineering is typically done manually by removing features or applying transformations to create them
- By using PCA, multiple fields can be mathematically reduced to fewer fields, which is called feature extraction

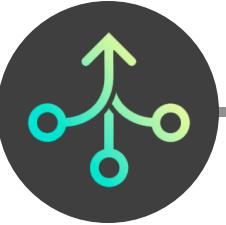
SVD

Can be used as an unsupervised learning technique for **Recommenders**

- SVD is a general matrix factorization technique that encompasses PCA
- SVD reduces the dimensions of a user's behavior to the most important so that they can be better compared with others

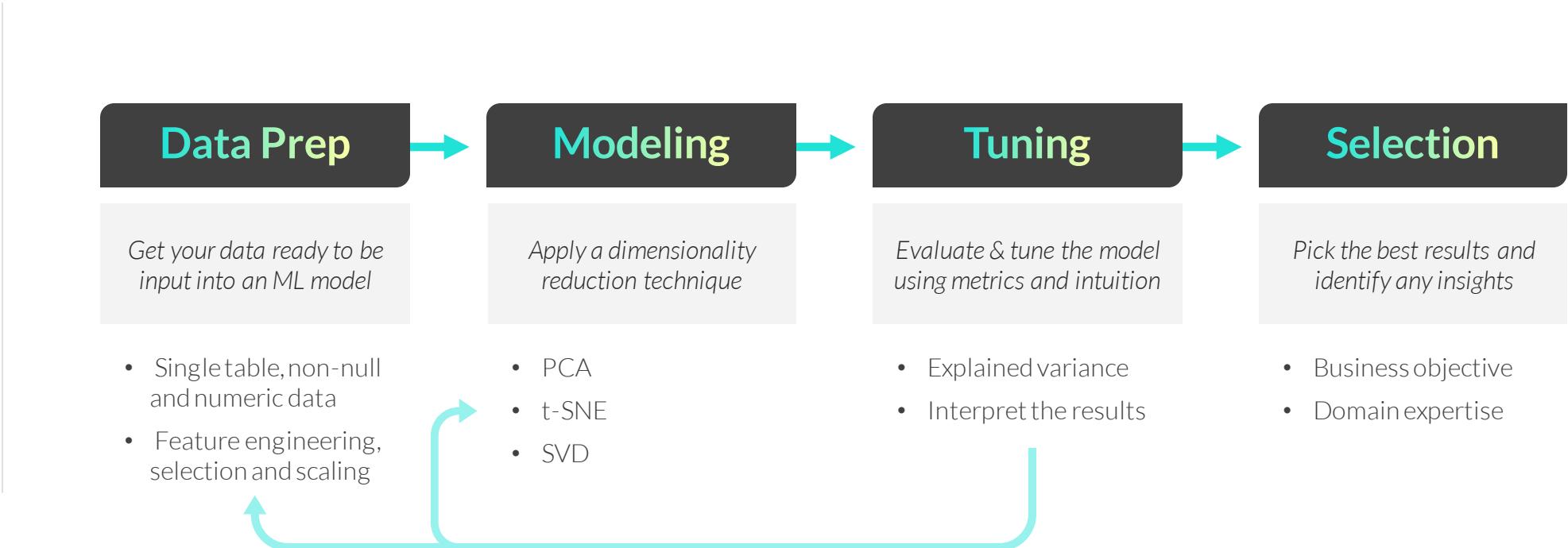
We'll cover SVD in the Recommenders section





DIMENSIONALITY REDUCTION WORKFLOW

The general **dimensionality reduction workflow** consists of the following steps:



Remember that dimensionality reduction is often a **step taken before other modeling steps**, so in addition to going through the workflow above, you may need to follow it up with other supervised or unsupervised learning workflows



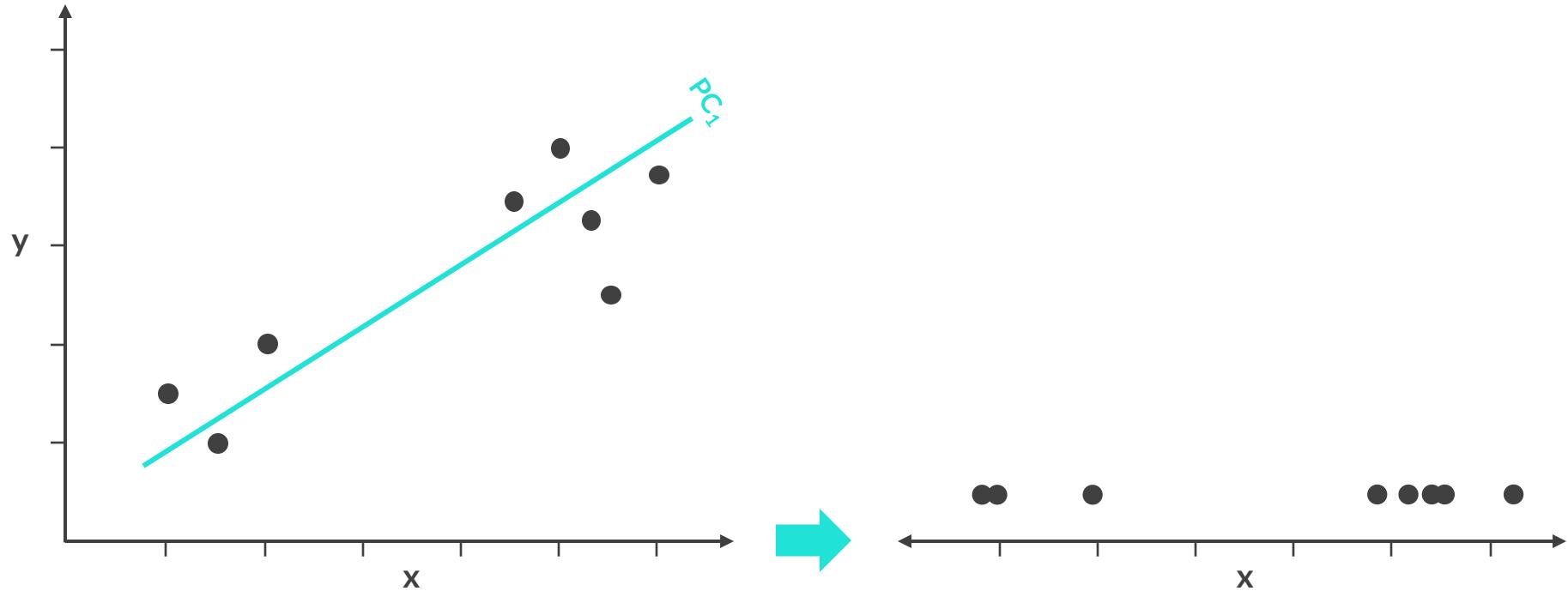
PRINCIPAL COMPONENT ANALYSIS

Dimensionality Reduction

PCA

t-SNE

Principal Component Analysis (PCA) is a dimensionality reduction technique that finds *linear combinations* of features that explain the most variation in the data





PRINCIPAL COMPONENT ANALYSIS

Dimensionality Reduction

PCA

t-SNE

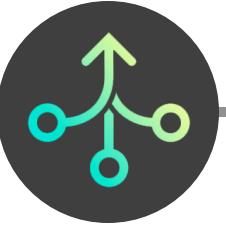
Principal Component Analysis (PCA) is a dimensionality reduction technique that finds *linear combinations* of features that explain the most variation in the data

Here's how it works:

1. Create a scatter plot and find the center of the data
2. Trace a line through the center that captures the most spread, or variation, in the data – this is the **first principal component (PC_1)**
3. Create another line perpendicular to the first one that captures the most spread, or variation, in the data – this is the **second principal component (PC_2)**
4. Repeat Step 3 until you have as many principal components as original columns
5. To reduce dimensions, keep a subset of principal components (i.e. PC_1 only)

Example use cases:

- Visualizing clusters in fewer dimensions to spot-check clustering/segmentation results
- Improving a model's predictive power by reducing redundant or unnecessary dimensions

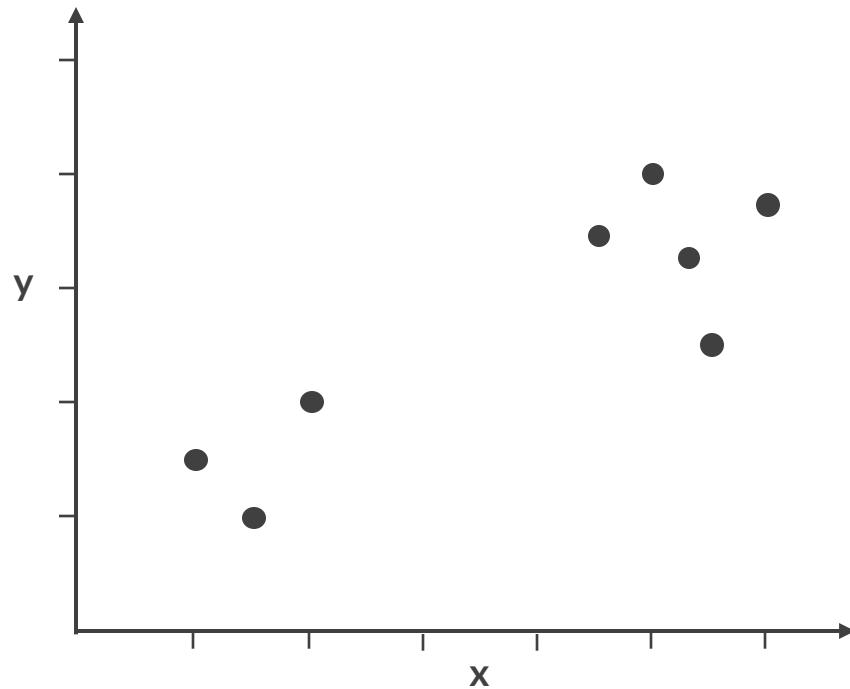


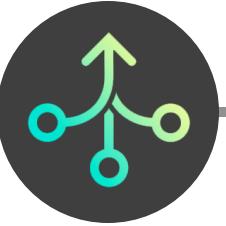
PRINCIPAL COMPONENT ANALYSIS

Dimensionality Reduction

PCA

t-SNE



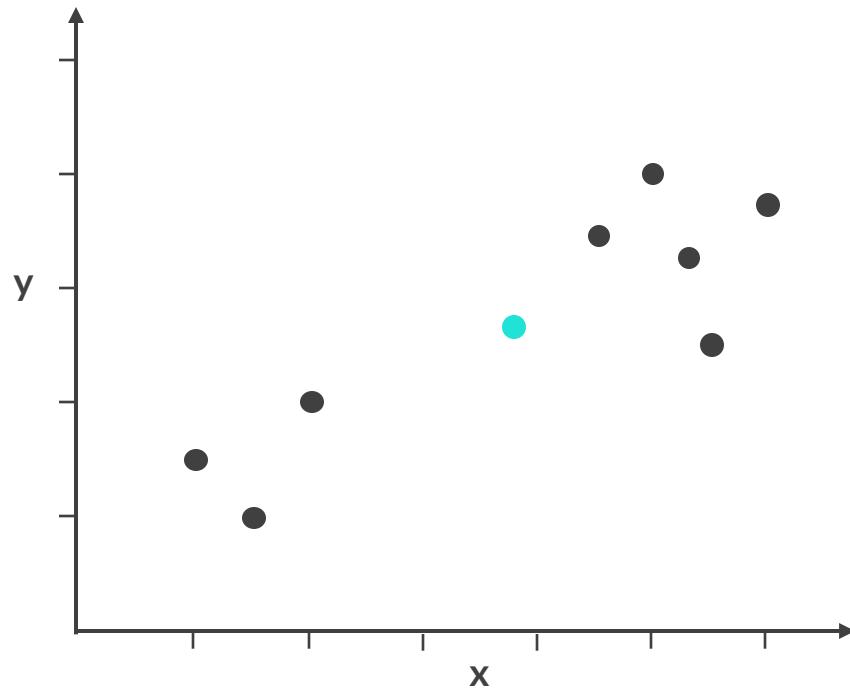


PRINCIPAL COMPONENT ANALYSIS

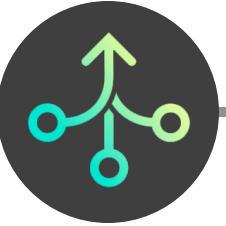
Dimensionality Reduction

PCA

t-SNE



STEP 1: Create a scatter plot and find the center of the data



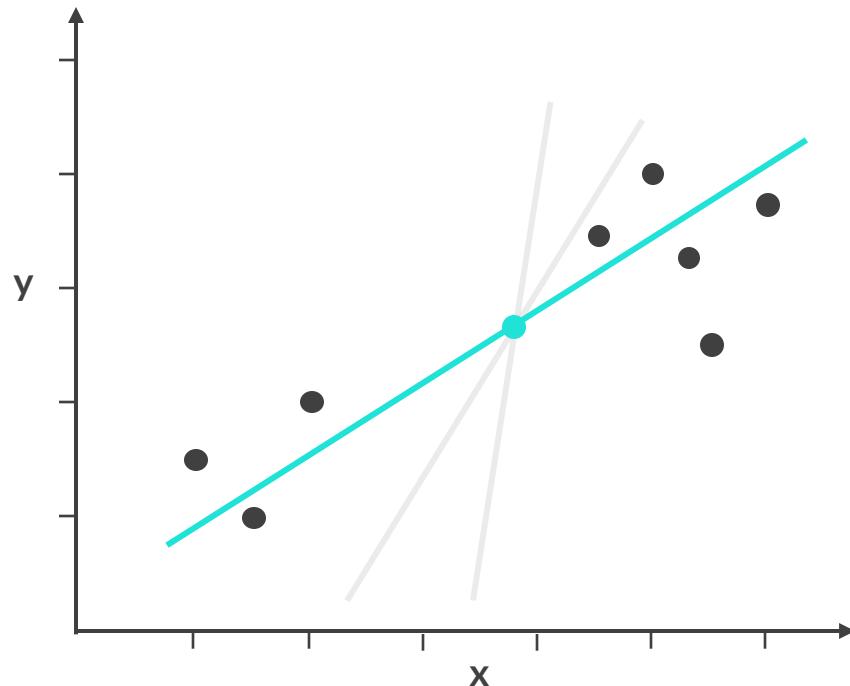
PRINCIPAL COMPONENT ANALYSIS

Dimensionality Reduction

PCA

t-SNE

STEP 2: Trace a line through the center that captures the most spread, or variation, in the data – this is the first principal component (PC_1)





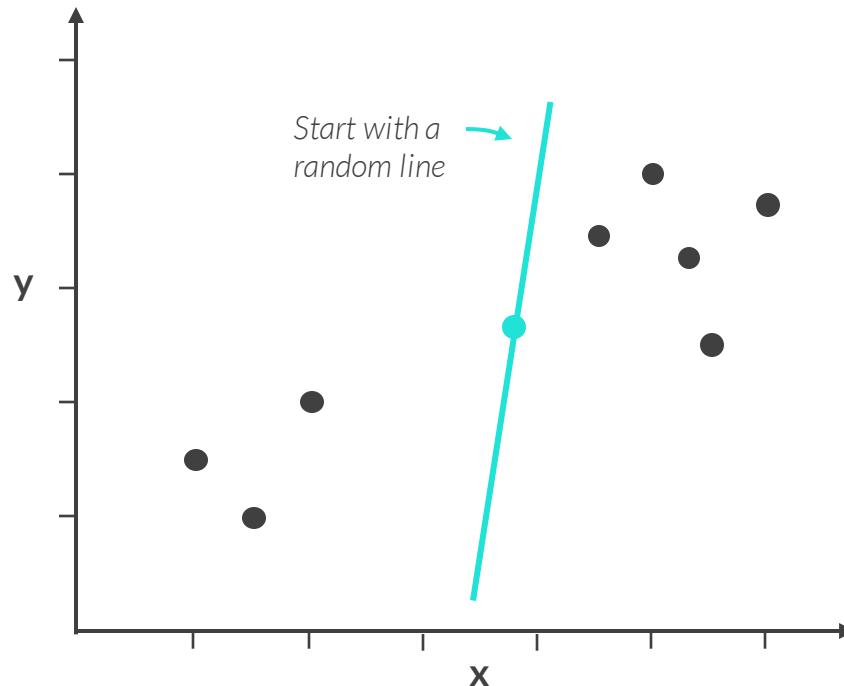
PRINCIPAL COMPONENT ANALYSIS

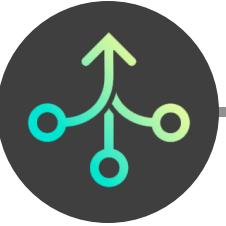
Dimensionality Reduction

PCA

t-SNE

STEP 2: Trace a line through the center that captures the most spread, or variation, in the data – this is the first principal component (PC_1)





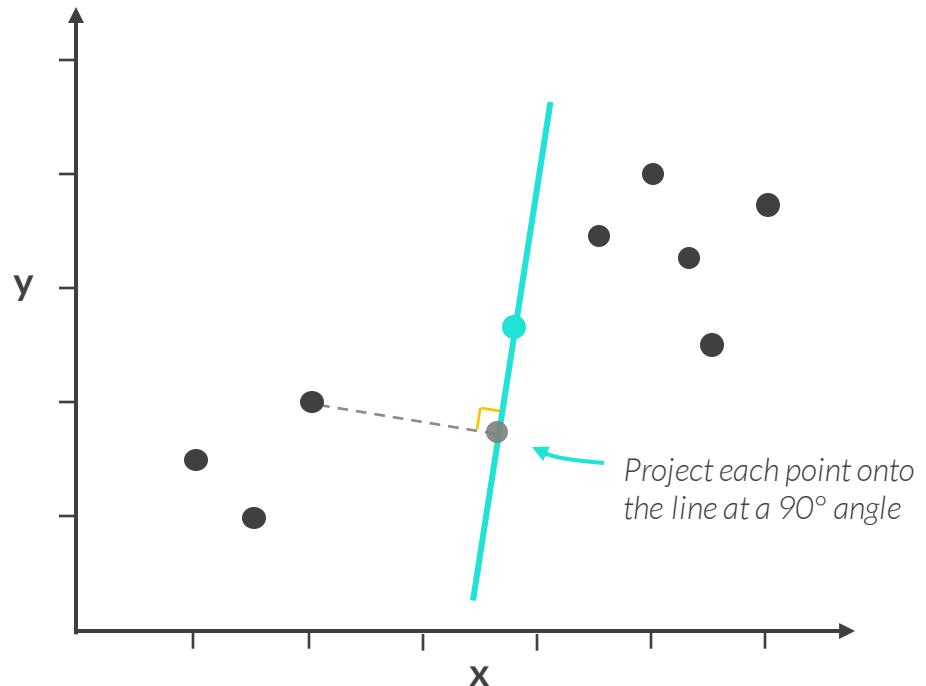
PRINCIPAL COMPONENT ANALYSIS

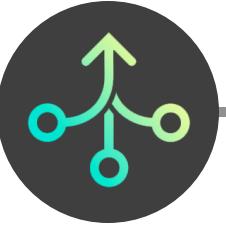
Dimensionality Reduction

PCA

t-SNE

STEP 2: Trace a line through the center that captures the most spread, or variation, in the data – this is the first principal component (PC_1)





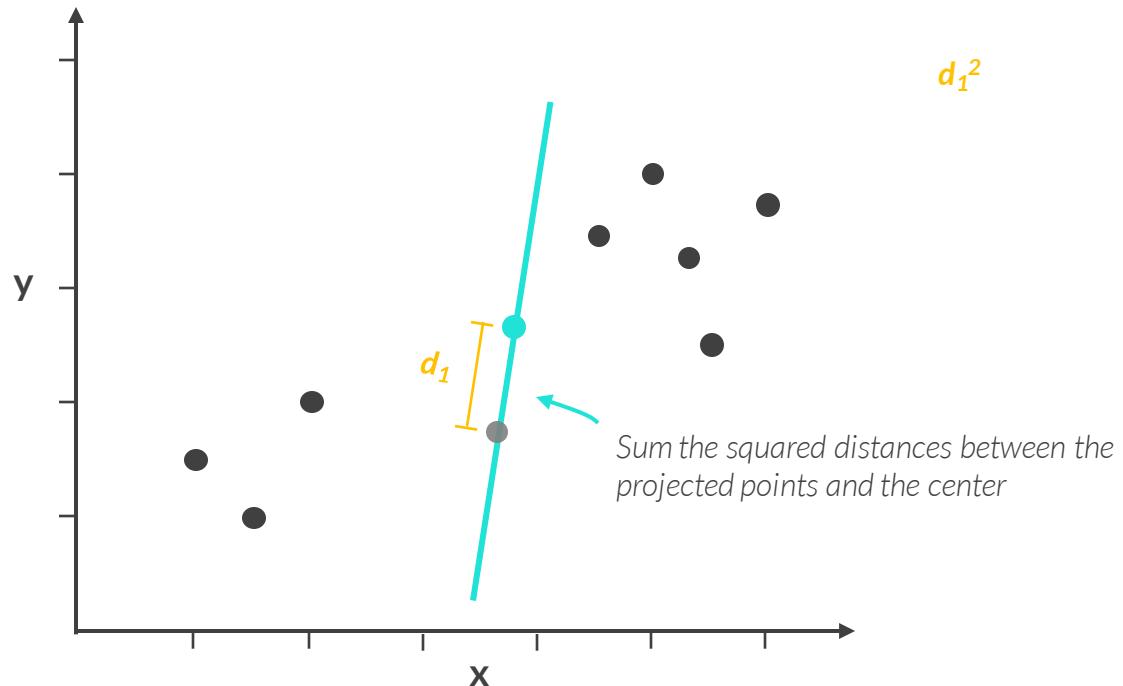
PRINCIPAL COMPONENT ANALYSIS

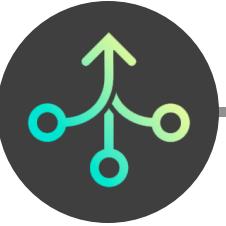
Dimensionality Reduction

PCA

t-SNE

STEP 2: Trace a line through the center that captures the most spread, or variation, in the data – this is the first principal component (PC_1)





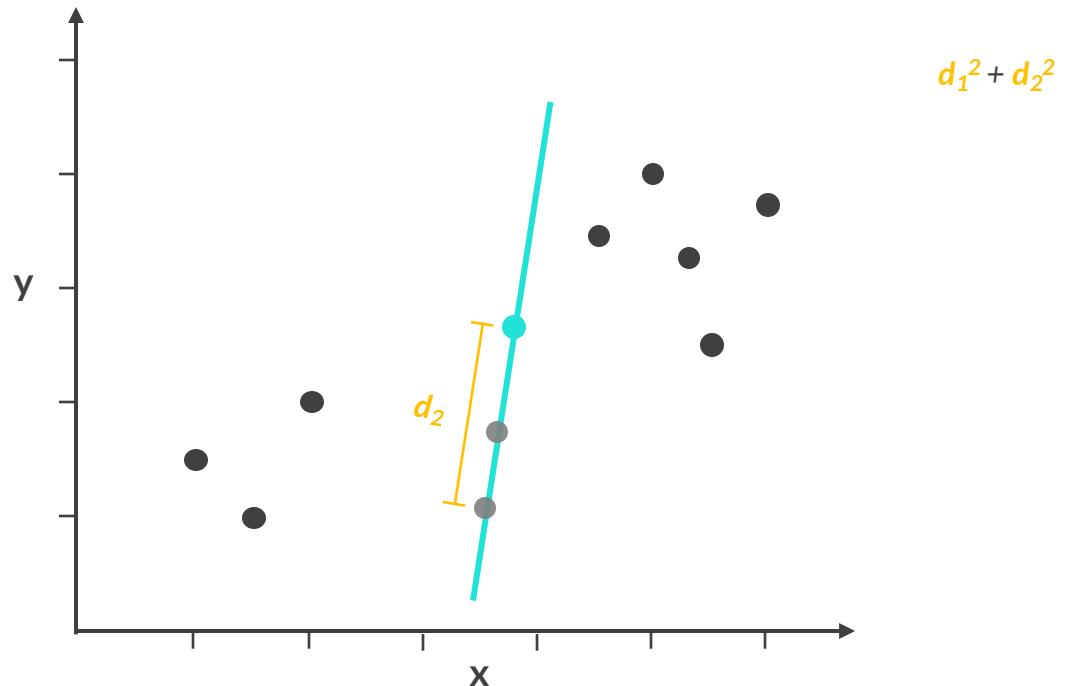
PRINCIPAL COMPONENT ANALYSIS

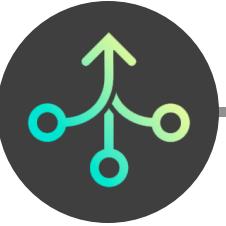
Dimensionality Reduction

PCA

t-SNE

STEP 2: Trace a line through the center that captures the most spread, or variation, in the data – this is the first principal component (PC_1)





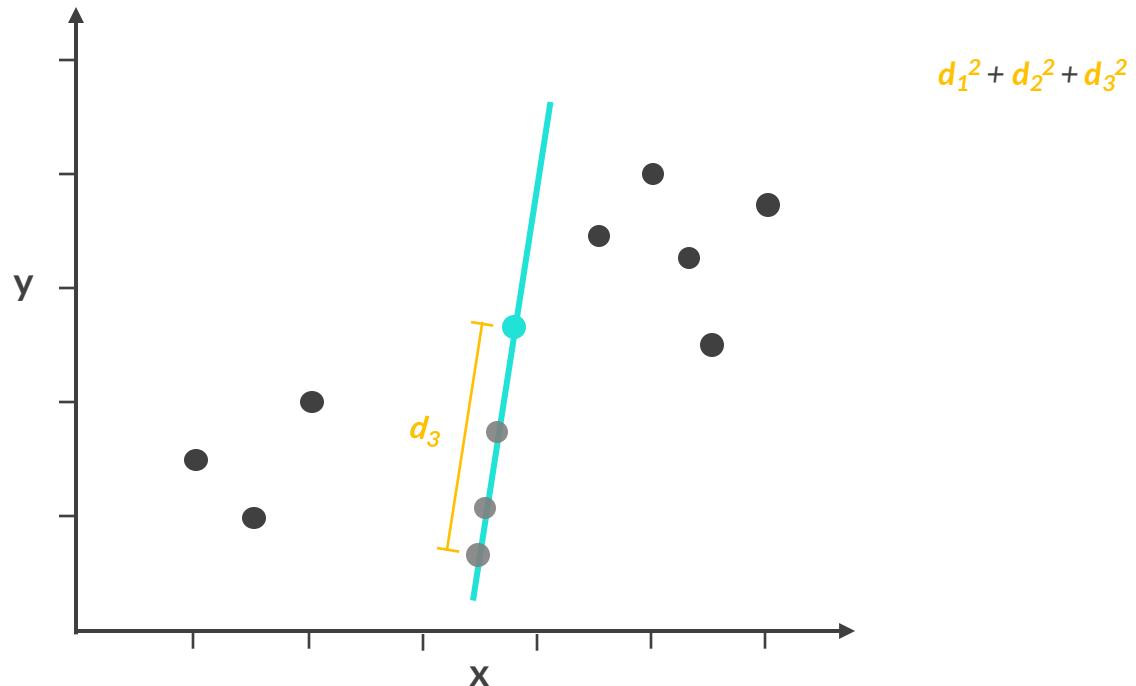
PRINCIPAL COMPONENT ANALYSIS

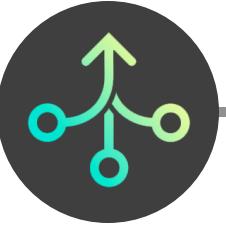
Dimensionality Reduction

PCA

t-SNE

STEP 2: Trace a line through the center that captures the most spread, or variation, in the data – this is the first principal component (PC_1)





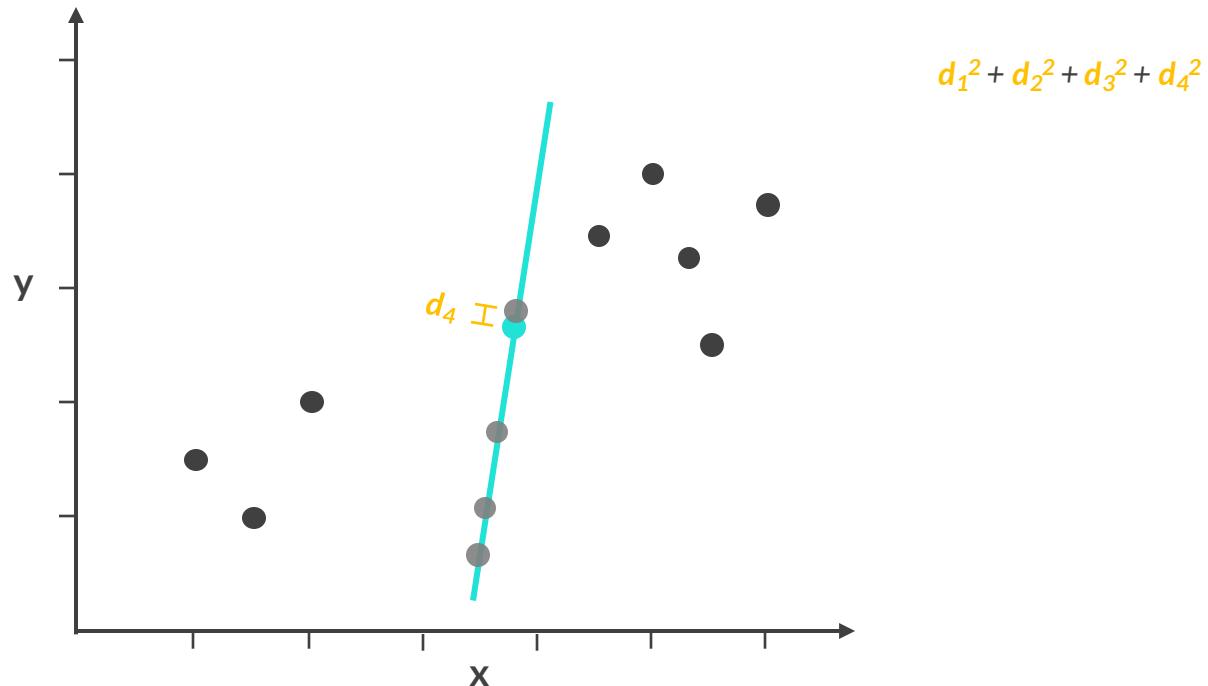
PRINCIPAL COMPONENT ANALYSIS

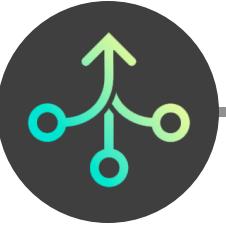
Dimensionality Reduction

PCA

t-SNE

STEP 2: Trace a line through the center that captures the most spread, or variation, in the data – this is the first principal component (PC_1)





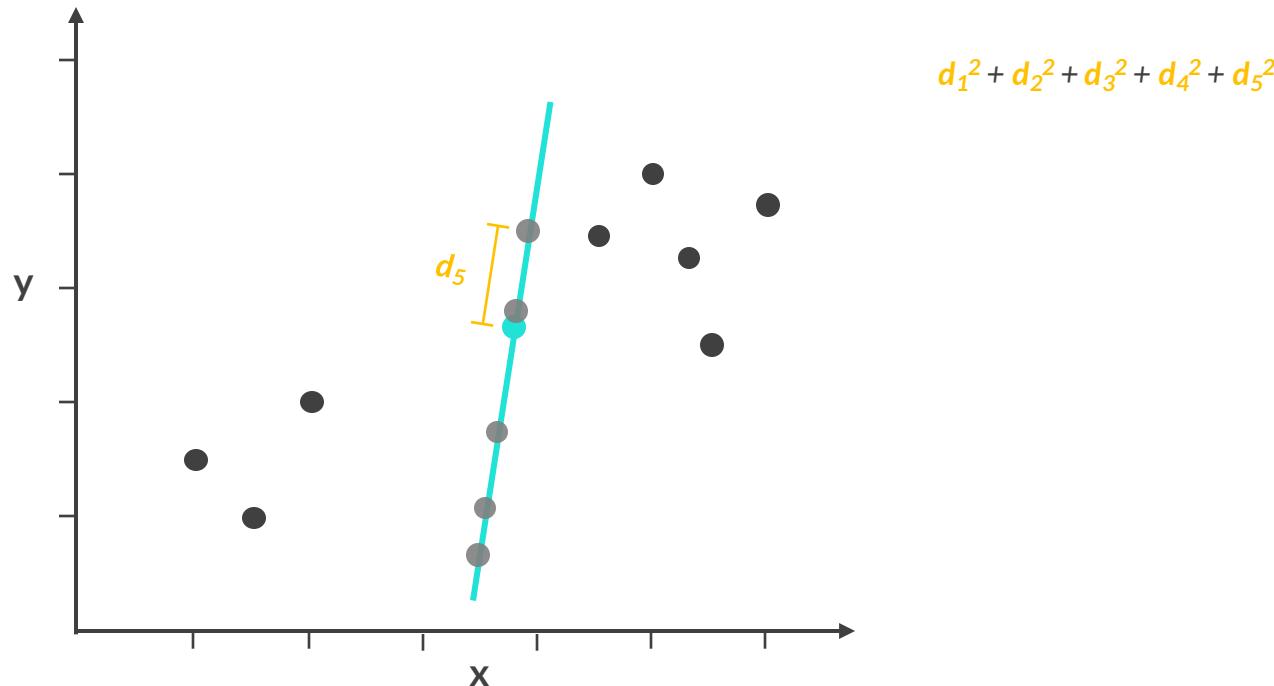
PRINCIPAL COMPONENT ANALYSIS

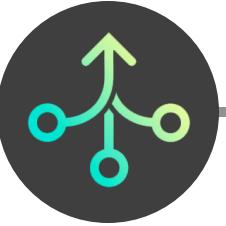
Dimensionality Reduction

PCA

t-SNE

STEP 2: Trace a line through the center that captures the most spread, or variation, in the data – this is the first principal component (PC_1)





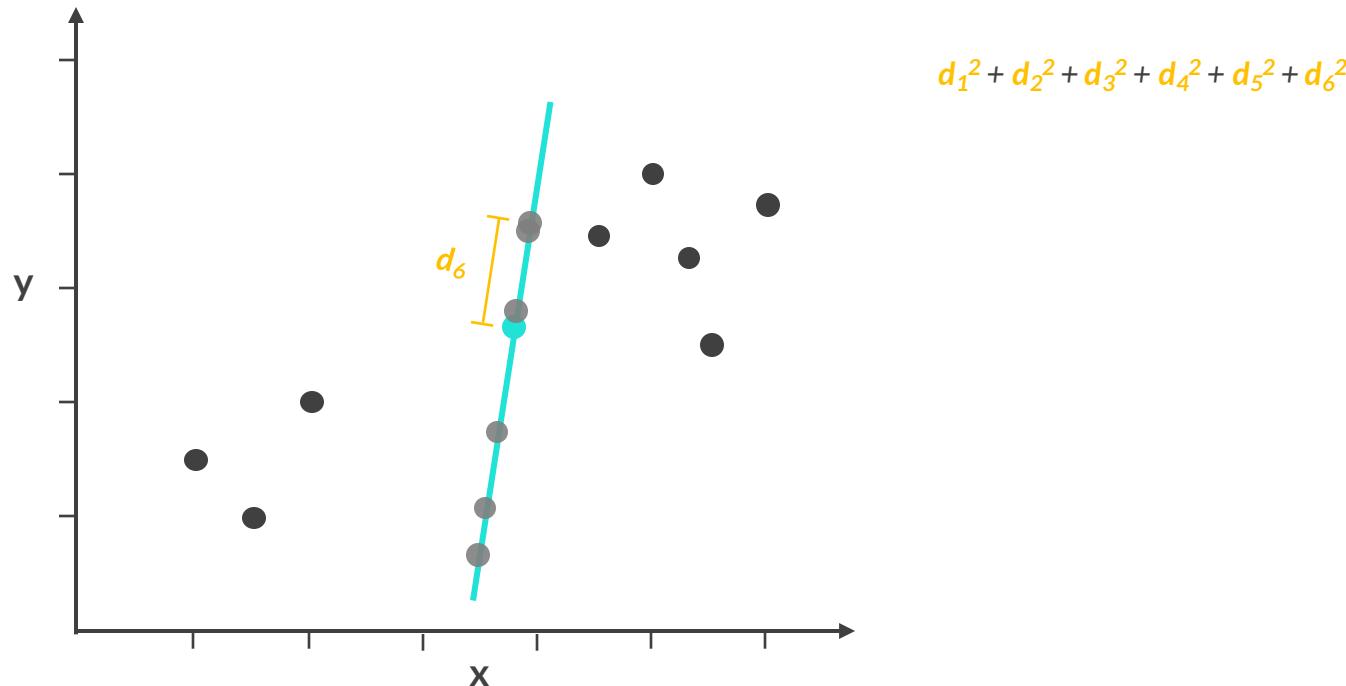
PRINCIPAL COMPONENT ANALYSIS

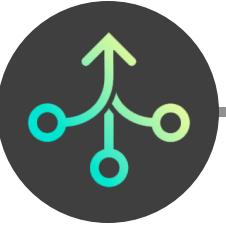
Dimensionality Reduction

PCA

t-SNE

STEP 2: Trace a line through the center that captures the most spread, or variation, in the data – this is the first principal component (PC_1)





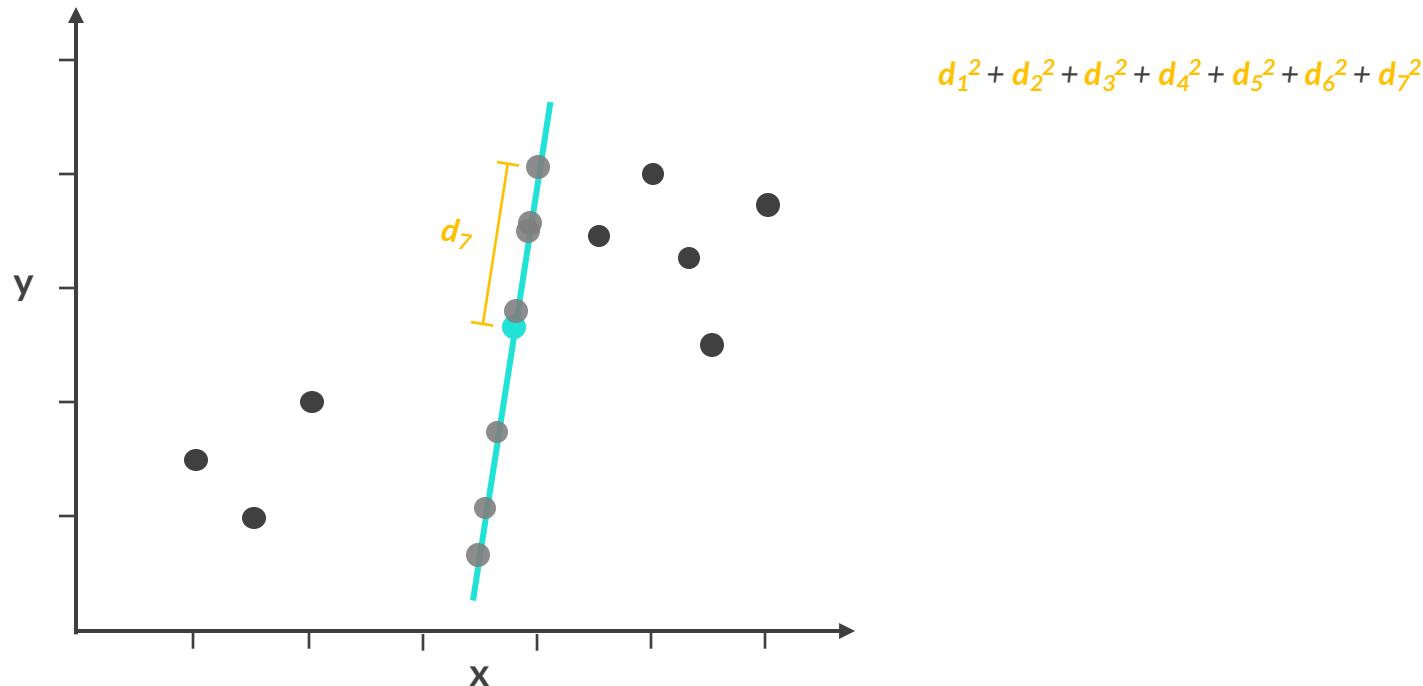
PRINCIPAL COMPONENT ANALYSIS

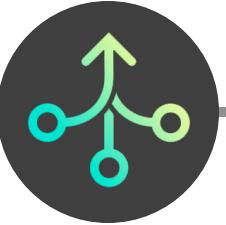
Dimensionality Reduction

PCA

t-SNE

STEP 2: Trace a line through the center that captures the most spread, or variation, in the data – this is the first principal component (PC_1)





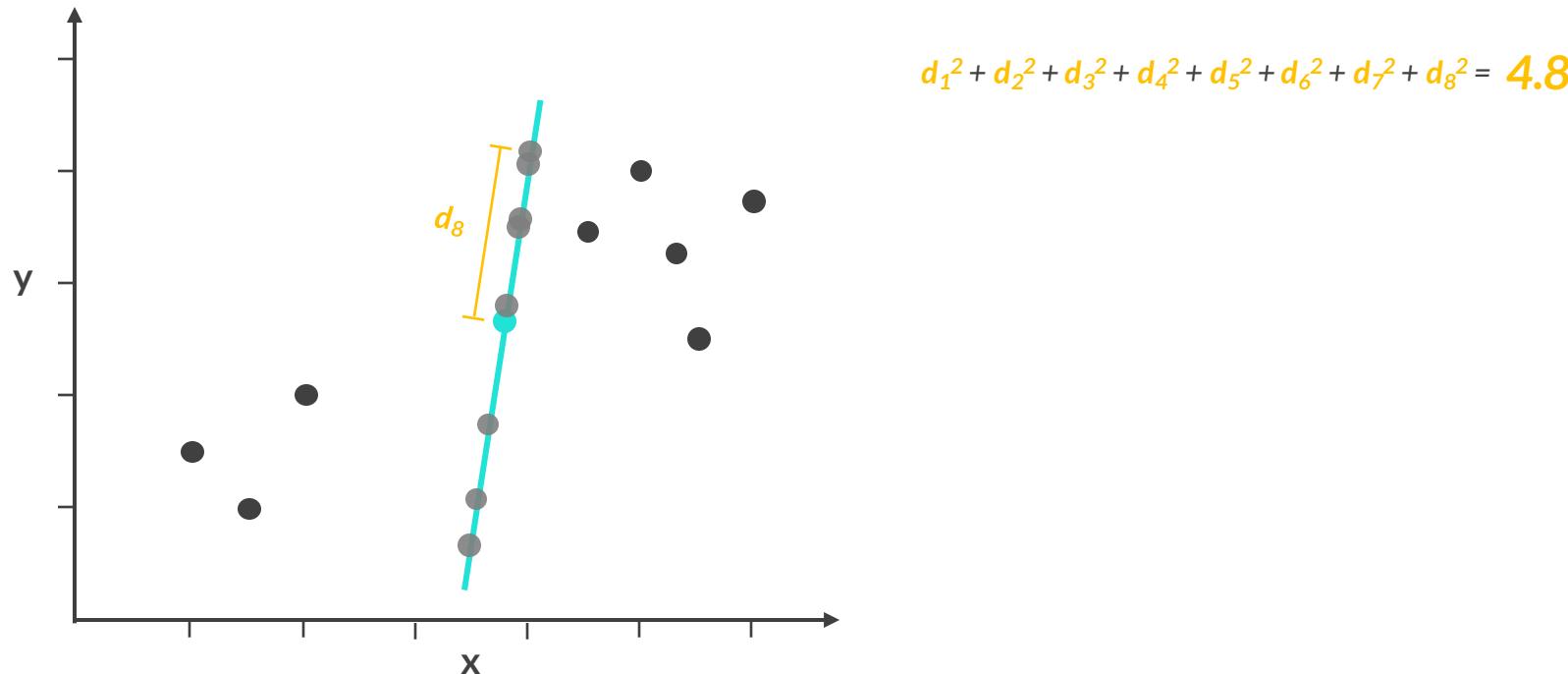
PRINCIPAL COMPONENT ANALYSIS

Dimensionality Reduction

PCA

t-SNE

STEP 2: Trace a line through the center that captures the most spread, or variation, in the data – this is the first principal component (PC_1)



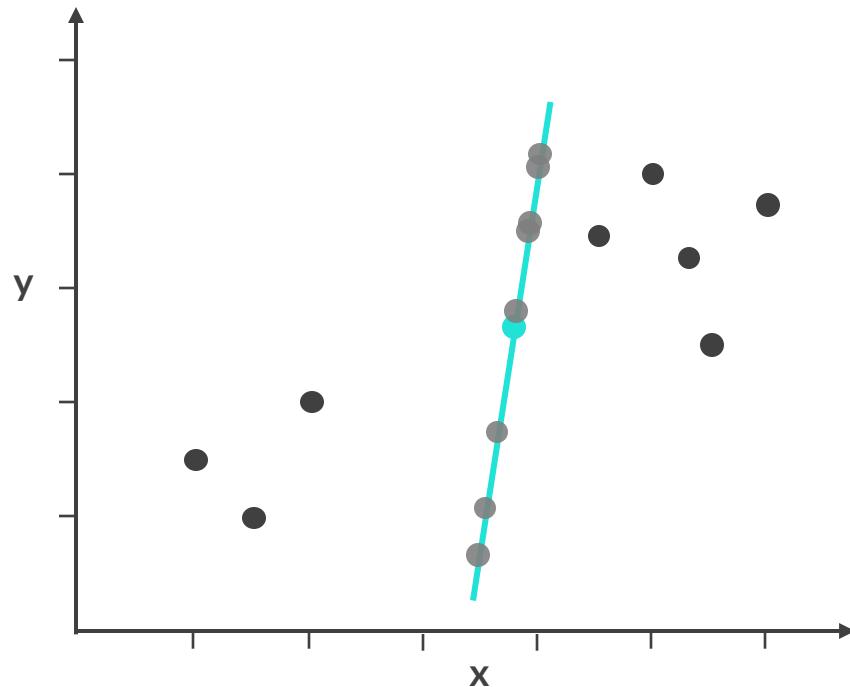


PRINCIPAL COMPONENT ANALYSIS

Dimensionality Reduction

PCA

t-SNE



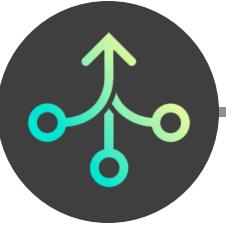
$$d_1^2 + d_2^2 + d_3^2 + d_4^2 + d_5^2 + d_6^2 + d_7^2 + d_8^2 = 4.8$$

Find the line that maximizes
the sum of squared distances



Why maximize the distance?

- This guarantees that the principal component line captures the most variation in the data



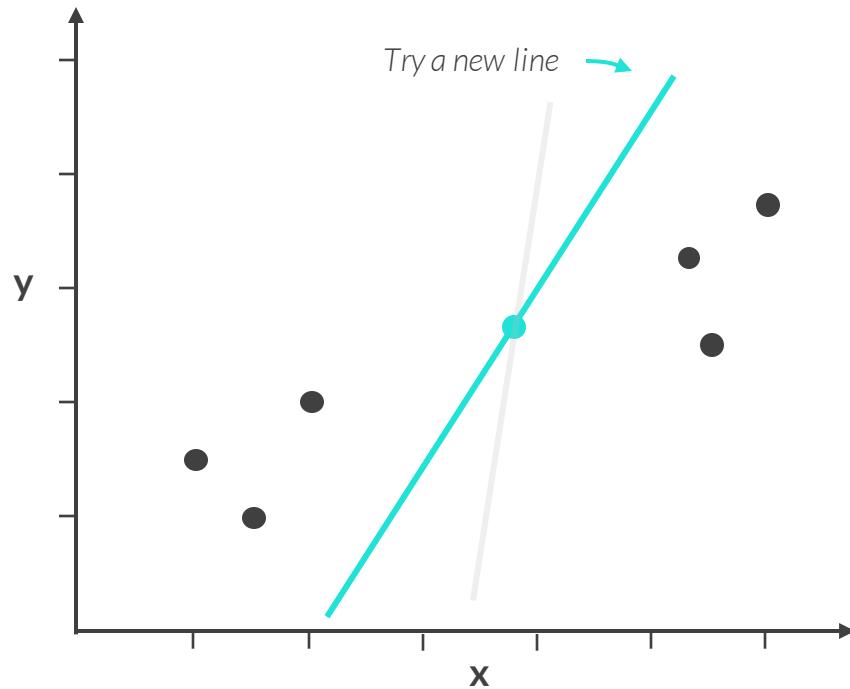
PRINCIPAL COMPONENT ANALYSIS

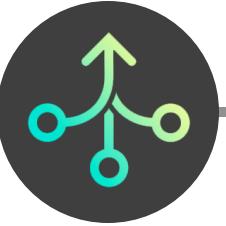
Dimensionality Reduction

PCA

t-SNE

STEP 2: Trace a line through the center that captures the most spread, or variation, in the data – this is the first principal component (PC_1)





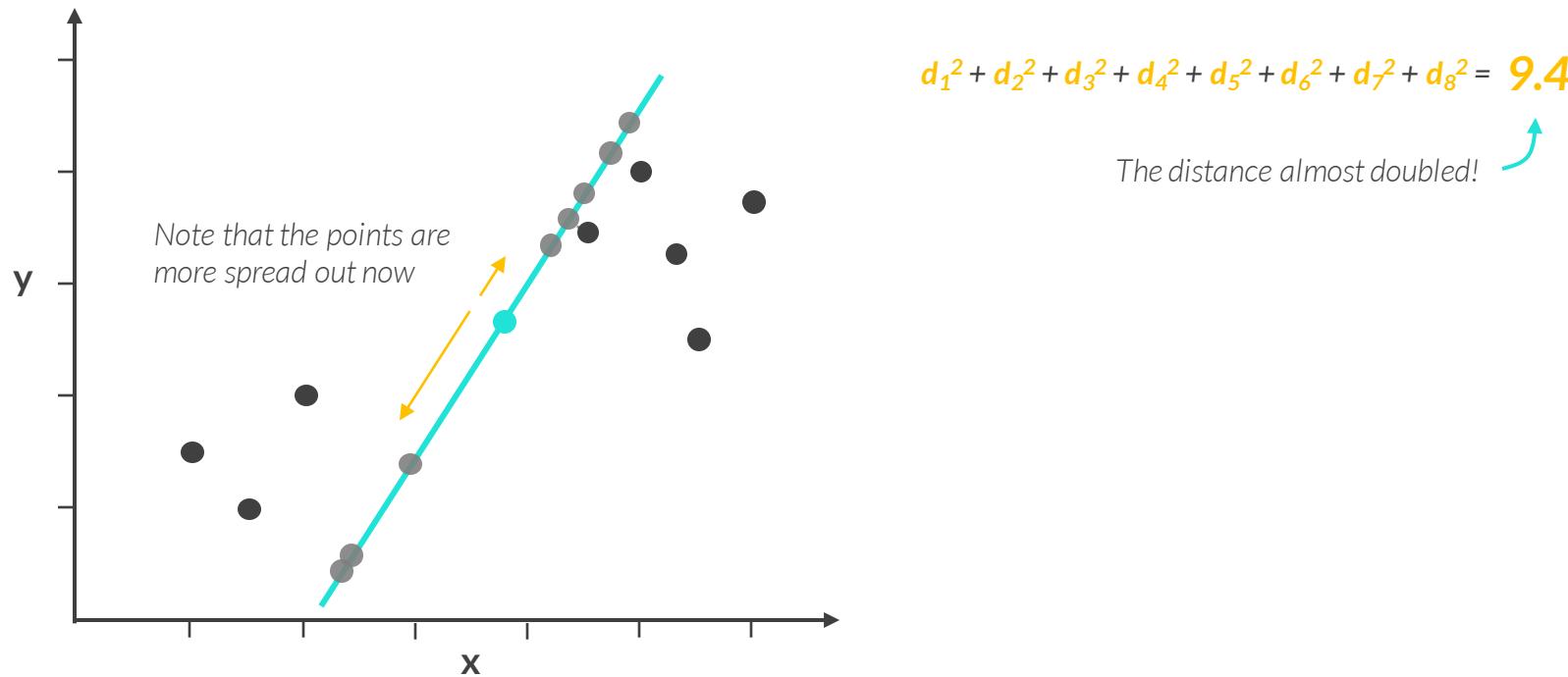
PRINCIPAL COMPONENT ANALYSIS

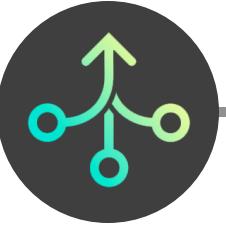
Dimensionality Reduction

PCA

t-SNE

STEP 2: Trace a line through the center that captures the most spread, or variation, in the data – this is the first principal component (PC_1)





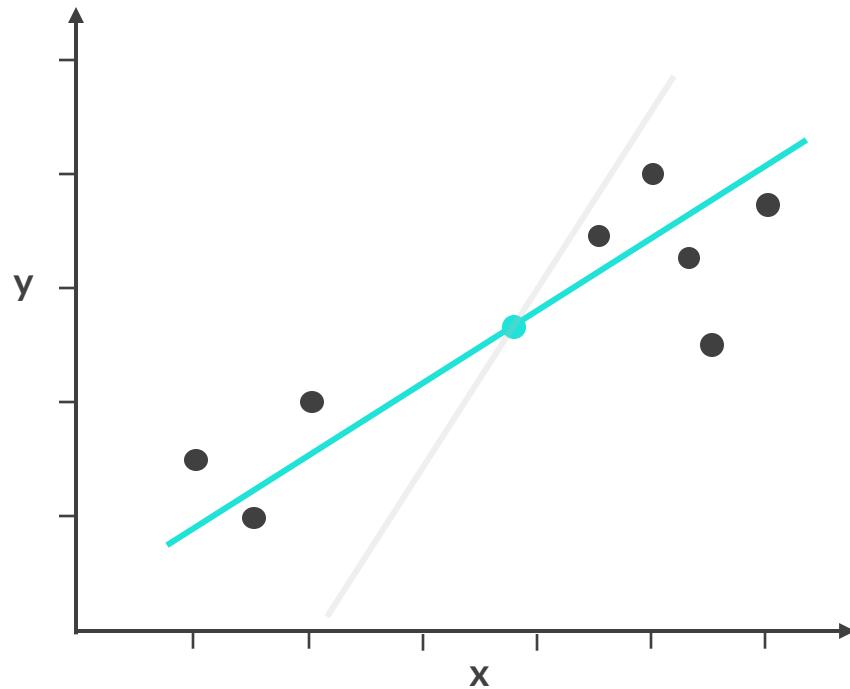
PRINCIPAL COMPONENT ANALYSIS

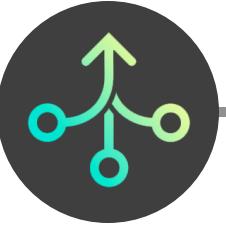
Dimensionality Reduction

PCA

t-SNE

STEP 2: Trace a line through the center that captures the most spread, or variation, in the data – this is the first principal component (PC_1)





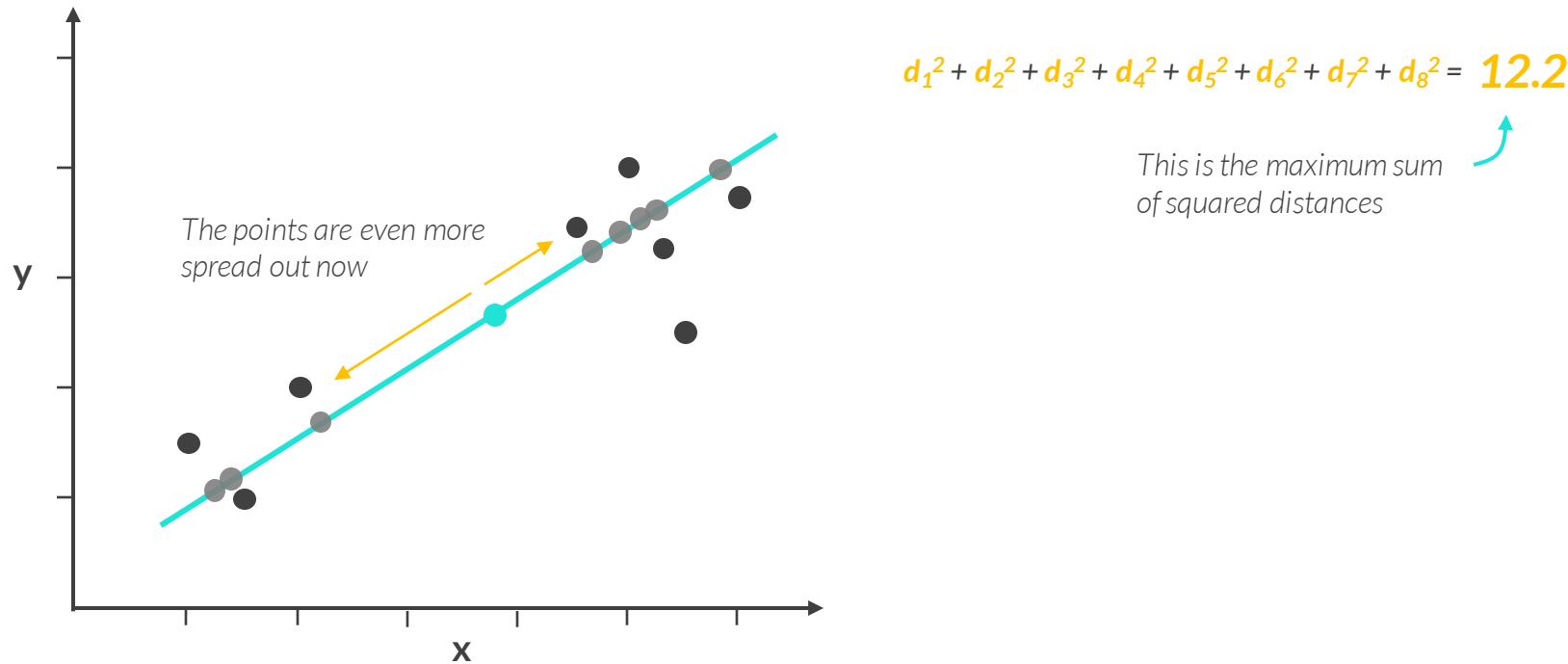
PRINCIPAL COMPONENT ANALYSIS

Dimensionality Reduction

PCA

t-SNE

STEP 2: Trace a line through the center that captures the most spread, or variation, in the data – this is the first principal component (PC_1)





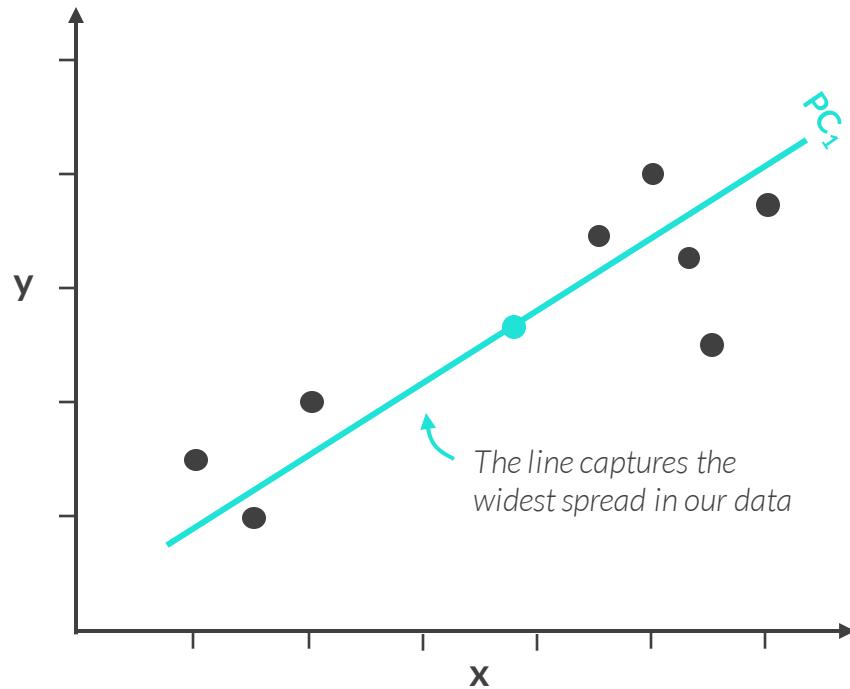
PRINCIPAL COMPONENT ANALYSIS

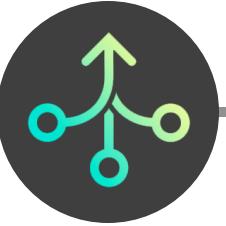
Dimensionality Reduction

PCA

t-SNE

STEP 2: Trace a line through the center that captures the most spread, or variation, in the data – this is the first principal component (PC_1)





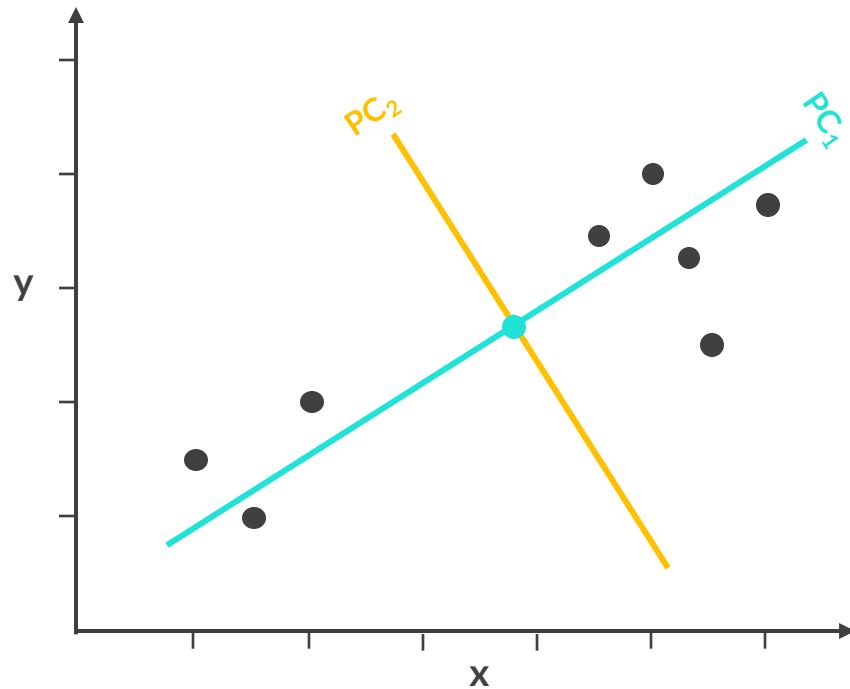
PRINCIPAL COMPONENT ANALYSIS

Dimensionality Reduction

PCA

t-SNE

STEP 3: Create another line perpendicular to the first one that captures the most spread, or variation, in the data – this is PC_2



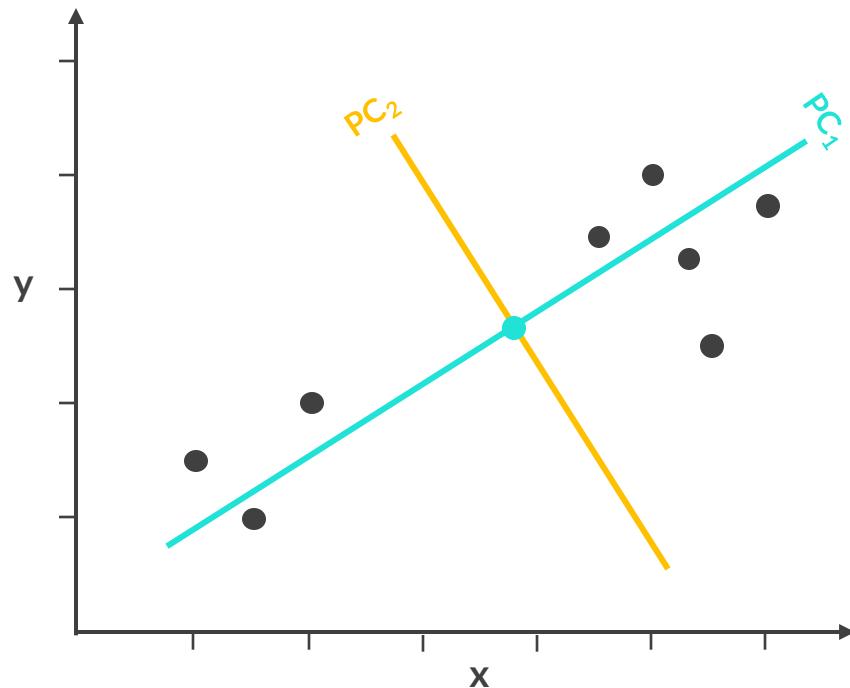


PRINCIPAL COMPONENT ANALYSIS

Dimensionality Reduction

PCA

t-SNE



We can't draw another perpendicular line in a 2-dimensional space, so we're all set!



In practice, these steps are completed using a linear algebra technique called **eigendecomposition**



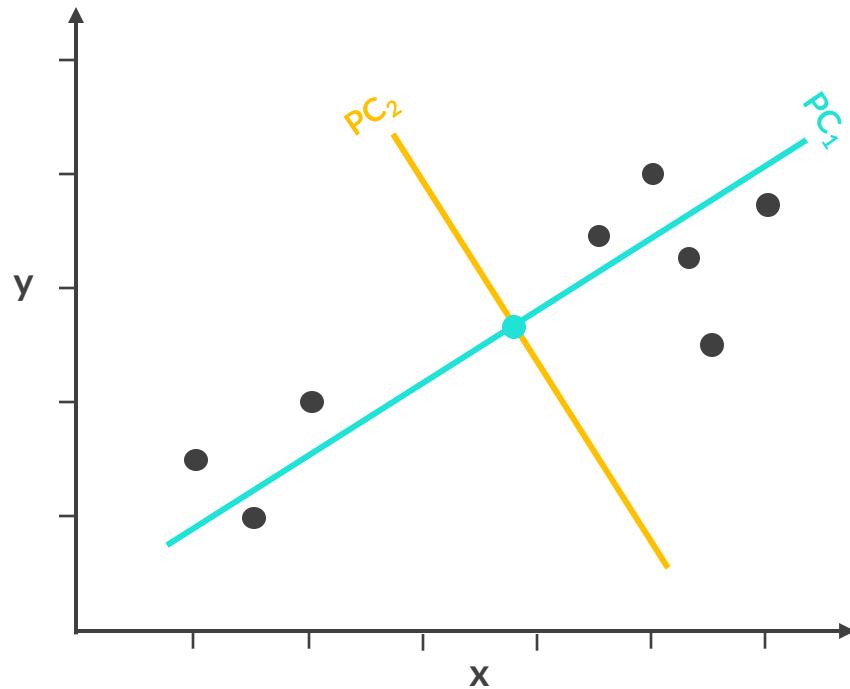
PRINCIPAL COMPONENT ANALYSIS

Dimensionality Reduction

PCA

t-SNE

STEP 5: Transform the data into a new space with the PC_1 line as the x-axis, the PC_2 line as the y-axis, and so on





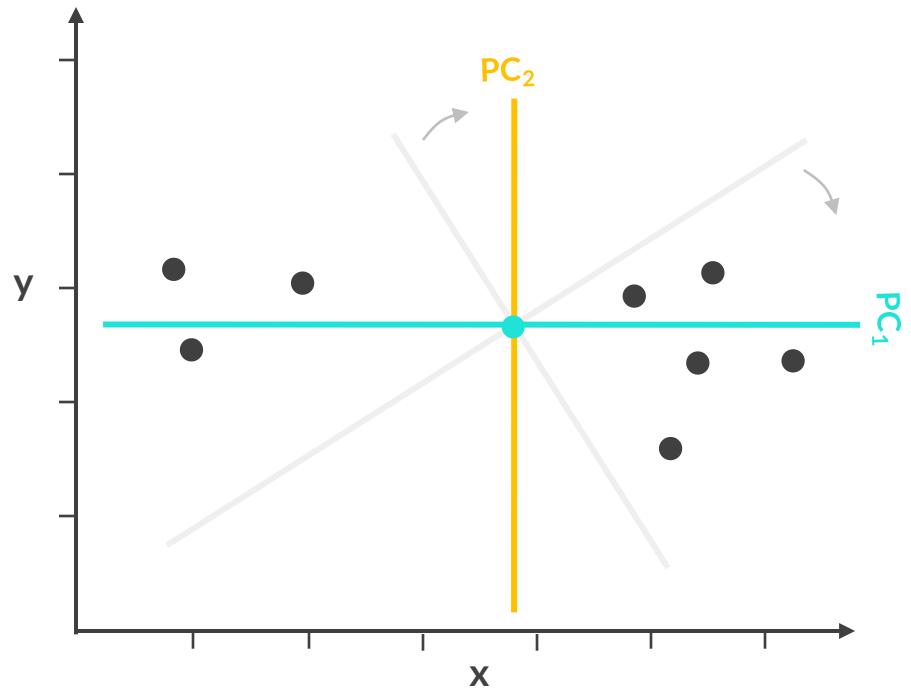
PRINCIPAL COMPONENT ANALYSIS

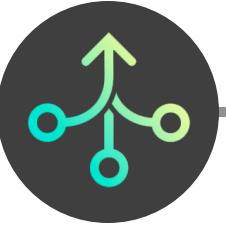
Dimensionality Reduction

PCA

t-SNE

STEP 5: Transform the data into a new space with the PC_1 line as the x-axis, the PC_2 line as the y-axis, and so on





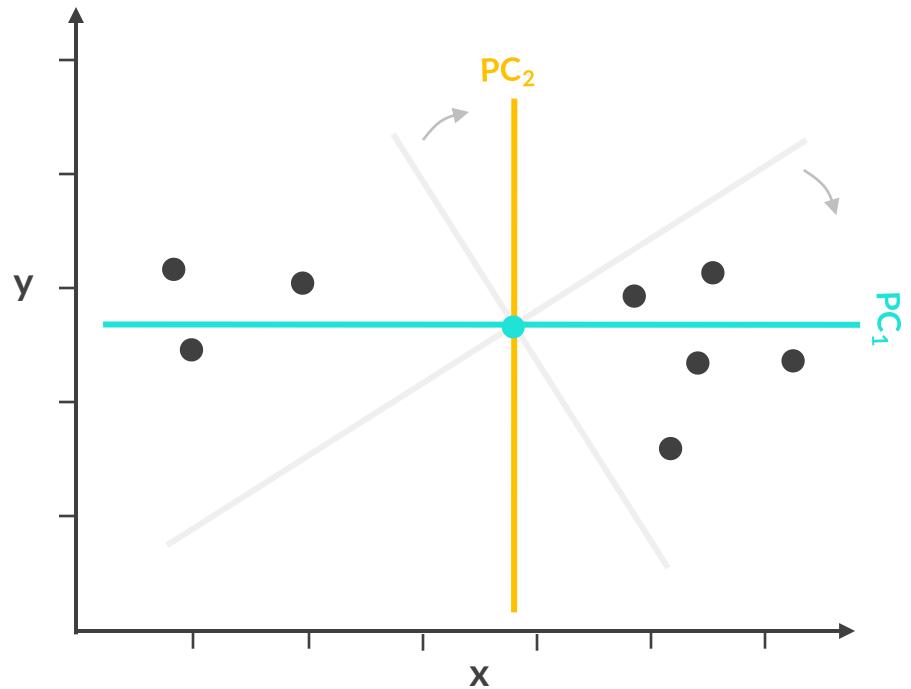
PRINCIPAL COMPONENT ANALYSIS

Dimensionality Reduction

PCA

t-SNE

STEP 6: To reduce dimensions, keep a subset of principal components – for example, remove PC_2 and project the points onto PC_1 , which is your new x-axis





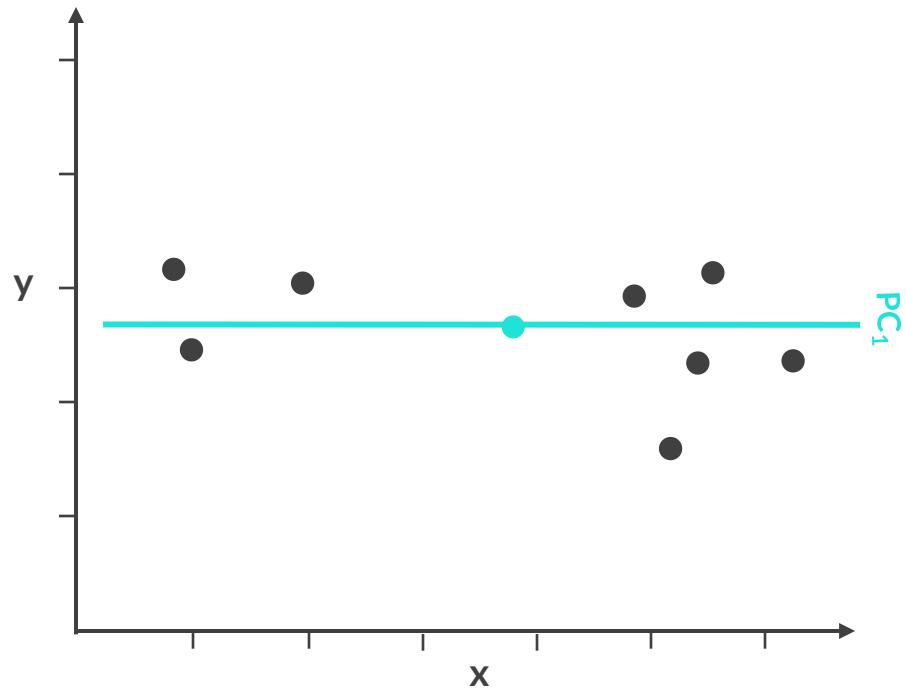
PRINCIPAL COMPONENT ANALYSIS

Dimensionality Reduction

PCA

t-SNE

STEP 6: To reduce dimensions, keep a subset of principal components – for example, remove PC_2 and project the points onto PC_1 , which is your new x-axis





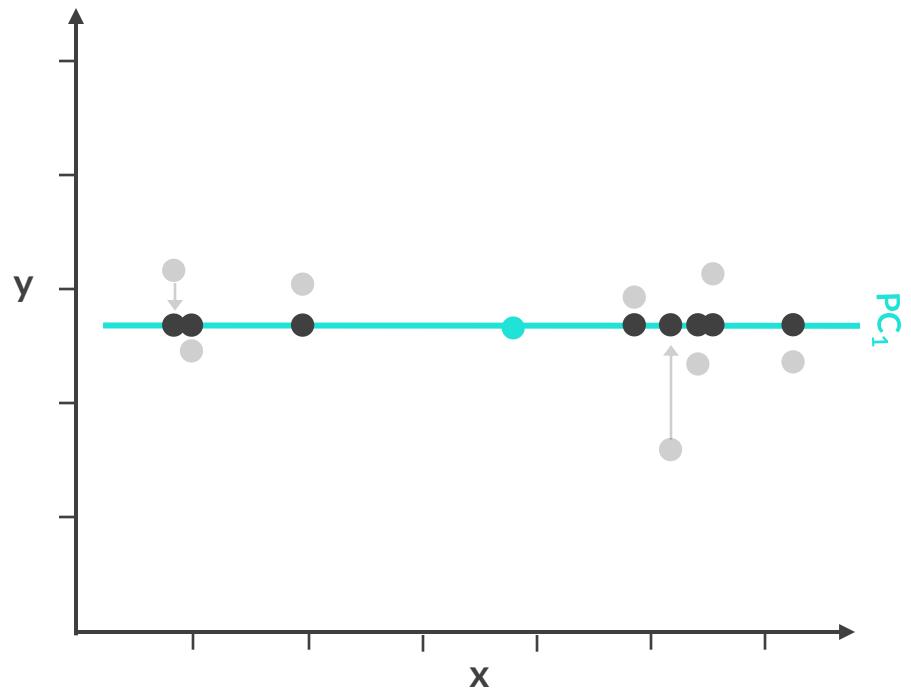
PRINCIPAL COMPONENT ANALYSIS

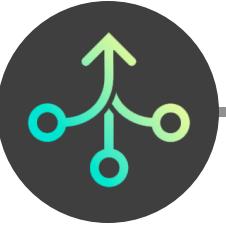
Dimensionality Reduction

PCA

t-SNE

STEP 6: To reduce dimensions, keep a subset of principal components – for example, remove PC_2 and project the points onto PC_1 , which is your new x-axis



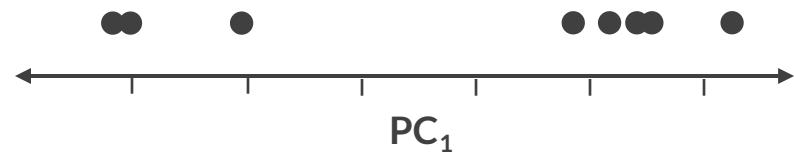


PRINCIPAL COMPONENT ANALYSIS

Dimensionality Reduction

PCA

t-SNE



STEP 6: To reduce dimensions, keep a subset of principal components – for example, remove PC₂ and project the points onto PC₁, which is your new x-axis



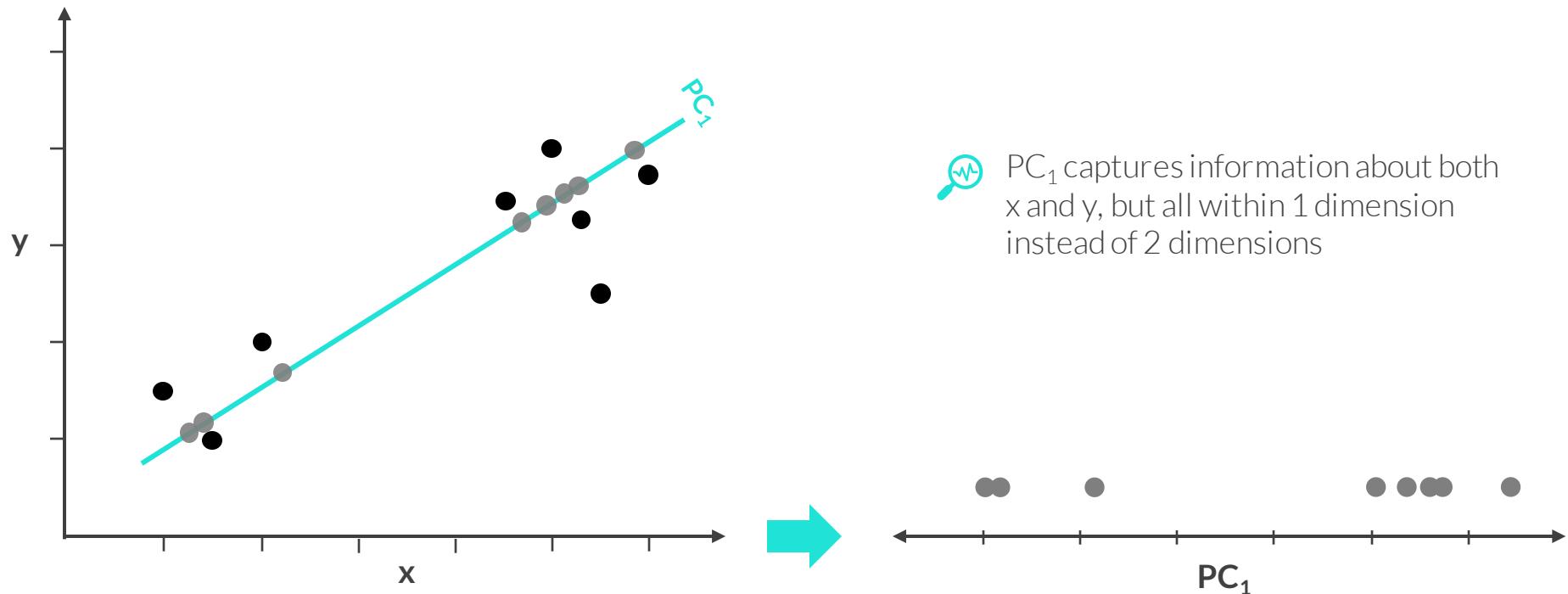
PRINCIPAL COMPONENT ANALYSIS

Dimensionality Reduction

PCA

t-SNE

By using **principal component analysis**, we've reduced the number of columns (dimensions) in the data while losing as little information as possible





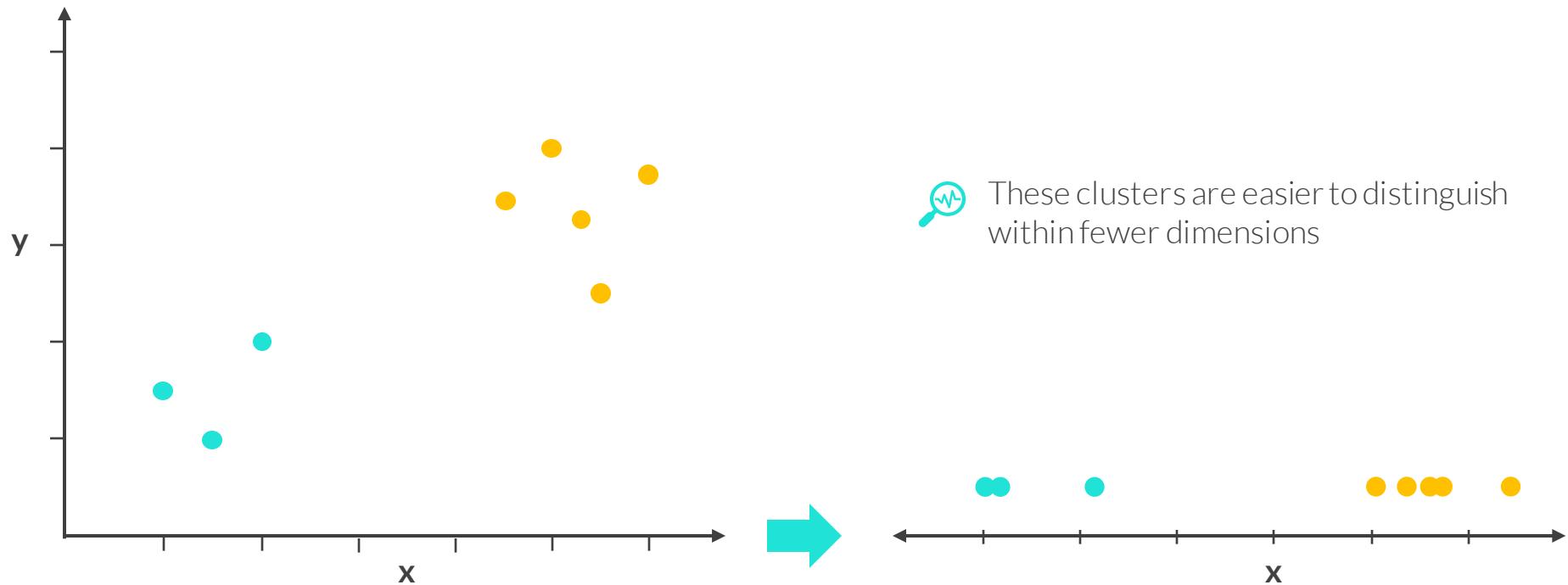
PRINCIPAL COMPONENT ANALYSIS

Dimensionality Reduction

PCA

t-SNE

By using **principal component analysis**, we've reduced the number of columns (dimensions) in the data while losing as little information as possible





PRINCIPAL COMPONENT ANALYSIS IN PYTHON

Dimensionality Reduction

PCA

t-SNE

You can use sklearn's **PCA()** to apply the Principal Component Analysis algorithm

```
from sklearn.decomposition import PCA  
  
pca = PCA(n_components=2)
```



The number of columns you want to reduce the dataset into



PRO TIP: For visualization, set n_components to 2 or 3; for feature extraction, use the number of columns in your original data set



PRINCIPAL COMPONENT ANALYSIS IN PYTHON

Dimensionality Reduction

PCA

t-SNE

```
# import pca from sklearn
from sklearn.decomposition import PCA
```

```
# center the data
data_centered = data - data.mean()
data_centered.head(2)
```

	books	tv_shows	video_games
0	-2.493333	0.014	-0.943333
1	-2.993333	-0.086	-1.043333

```
# fit a pca model on the centered data
pca = PCA(n_components=2)
pca.fit(data_centered)
```

```
▼      PCA
PCA(n_components=2)
```

You need to center each column around zero before applying a PCA model by either:

- Subtracting the mean from each value
- Standardizing the columns



PRO TIP: If the columns are on very different scales, it is recommended that you scale the data as well, which is something that standardization takes care of

Principal components 1 and 2 have been determined!



EXPLAINED VARIANCE RATIO

Dimensionality Reduction

PCA

t-SNE

```
# view the explained variance  
pca.explained_variance_ratio_  
  
array([0.88175186, 0.08603611])
```



PRO TIP: Typically you want your components to capture 80%+ of the variance in your data



You can represent 88% of the information captured in the 3 original columns by only keeping **one principal component**



You can represent 97% of the information captured in the 3 original columns by keeping **two principal components**

ASSIGNMENT: PRINCIPAL COMPONENT ANALYSIS

 **1 NEW MESSAGE**
March 25, 2024

From: Tim Menschen (Assistant Principal)
Subject: Student Analysis

Hello!

Our guidance counselor would like to understand the types of students at our school to recommend appropriate colleges.

I was a data scientist in my past life, and I remember that PCA helps visualize many dimensions in two dimensional charts.

Could you apply PCA on the attached student grades data set and let me know if two dimensions are able to capture a decent amount of variance in the data?

Thanks!
Tim

 student_grades.csv Reply Forward

Key Objectives

1. Read in the student grades data set
2. Drop the first column with student_id
3. Center the data
4. Fit a PCA model with 2 components
5. View and interpret the explained variance ratios



INTERPRETING PCA

Dimensionality Reduction

PCA

t-SNE

The `.components_` attribute lets you interpret the principal components

```
# view the components  
pca.components_
```

```
array([[ 0.93143032, -0.11629655,  0.34483717],  
       [-0.04900906,  0.89884697,  0.43551376]])
```

Each row represents a principal component, and each column represents a column from the original data set

```
# view the columns again  
data.columns
```

```
Index(['books', 'tv_shows', 'video_games'], dtype='object')
```



PC 1: Higher values = more books (0.93) and some more video games (0.34)



PC 2: Higher values = more TV shows (0.89) and some more video games (0.43)



VISUALIZING PCA

Dimensionality Reduction

PCA

t-SNE

```
# view the transformed data  
data_transformed = pd.DataFrame(pca.transform(data_centered),  
                                 columns=['pc1', 'pc2'])  
  
data_transformed.head()
```

	pc1	pc2
0	-2.649291	-0.276055
1	-3.137860	-0.384986
2	-2.603177	-0.322388
3	0.742853	0.227407
4	-0.172578	-0.802994

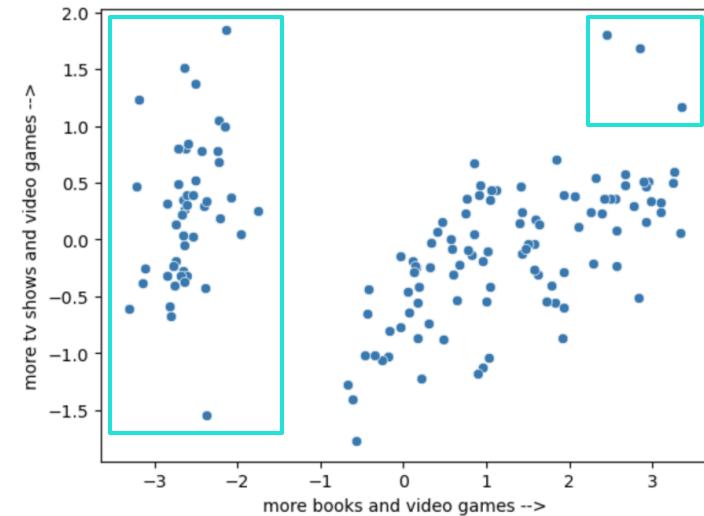


Students in the left cluster read very few books



Students in the top right corner love all 3 types of entertainment

```
# view the data on a scatter plot  
import matplotlib.pyplot as plt  
import seaborn as sns  
  
sns.scatterplot(x='pc1', y='pc2', data=data_transformed)  
plt.xlabel('more books and video games -->')  
plt.ylabel('more tv shows and video games -->');
```



ASSIGNMENT: INTERPRETING & VISUALIZING PCA

 **NEW MESSAGE**
March 26, 2024

From: Tim Menschen (Assistant Principal)
Subject: Student Analysis Visualization

Hi again!

Thanks for kicking off the PCA analysis earlier. I'm excited to hear that the explained variance ratio is high enough so that we can get a good idea of the students in just two dimensions!

Can you plot the students in the two dimensions, explain what the axes represent, and what you observe visually in the plot?

I'll pass that info along to the guidance counselor.

Thanks for your help!
Tim

[Reply](#) [Forward](#)

Key Objectives

1. Interpret the components of the PCA model
2. Plot the students on a scatter plot with the x-axis as PC_1 and the y-axis as PC_2
3. Interpret the clusters of students that you see on the plot and make recommendations for the guidance counselor



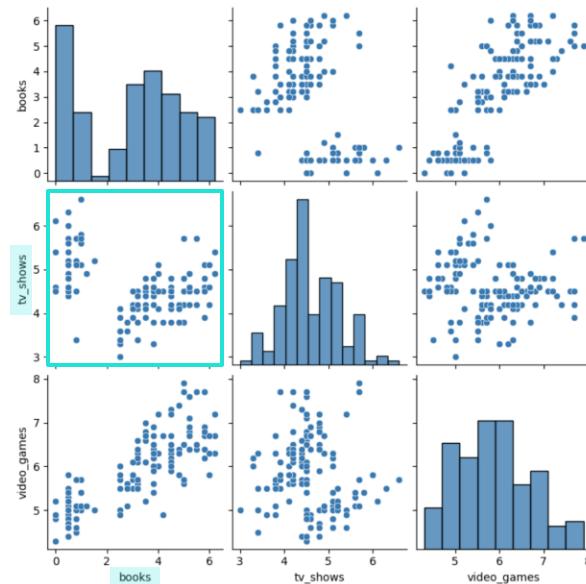
FEATURE SELECTION VS FEATURE EXTRACTION

Dimensionality Reduction

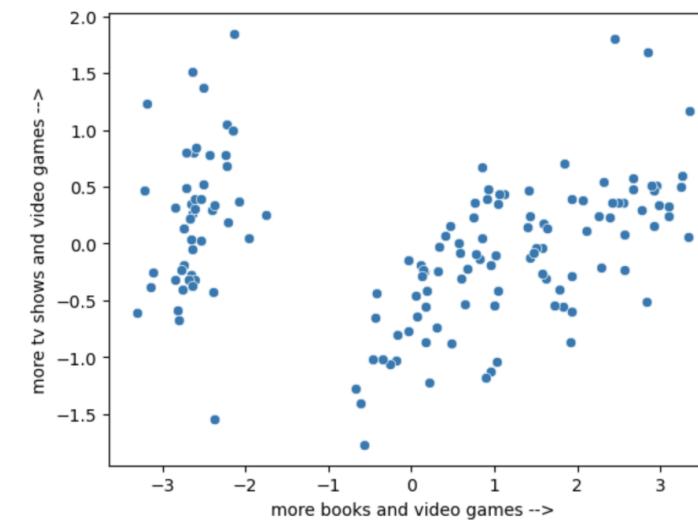
PCA

t-SNE

Feature Selection



Feature Extraction



For this particular data set, PCA didn't perform much better than simple feature selection, since the "books" and "tv_shows" variables highlight the same clusters



FEATURE SELECTION VS FEATURE EXTRACTION

Dimensionality Reduction

PCA

t-SNE

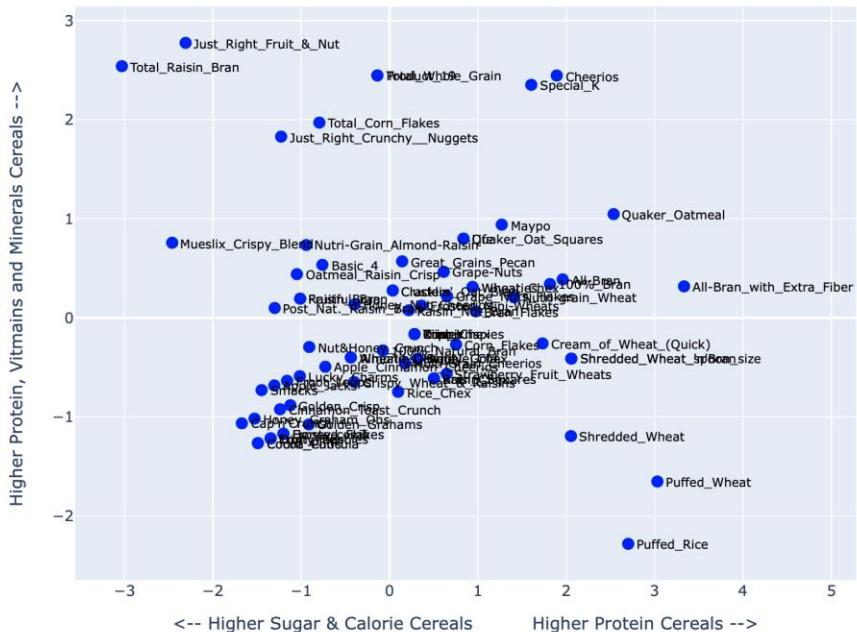
cereal_data.head()

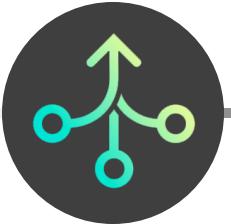
	Cereal Name	Manufacturer	Calories	Protein (g)	Sugars	Vitamins and Minerals
0	100%_Bran	Nabisco	70	4	6	25
1	100%_Natural_Bran	Quaker Oats	120	3	8	0
2	All-Bran	Kelloggs	70	4	5	25
3	All-Bran_with_Extra_Fiber	Kelloggs	50	4	0	25
4	Almond_Delight	Ralston Purina	110	2	8	25

In a data set like this one, reducing the 4 columns of data using PCA to just 2 lets you convey a lot of information in a single scatter plot

Raisin Bran is healthy but sugary, and Cheerios are probably the best cereal you can eat

Comparing Cereals by Nutritional Facts





PCA NEXT STEPS

Dimensionality Reduction

PCA

t-SNE

Data Visualization

- If you're able to distinguish data points visually, you're all set!
- If it's difficult to visually distinguish data points:
 - You can vary the inputs, modify the number of dimensions, etc.
 - You can try other dimensionality reduction techniques, such as t-SNE (*coming up next!*)

Modeling

- Once you've identified the principal components, they can be used as inputs into a machine learning model
- You'll need to apply the same PCA transformations to the test data



While PCA is a good technique for reducing dimensions to potentially increase predictive model performance, one of the main downsides is that **the transformed features are difficult to interpret**



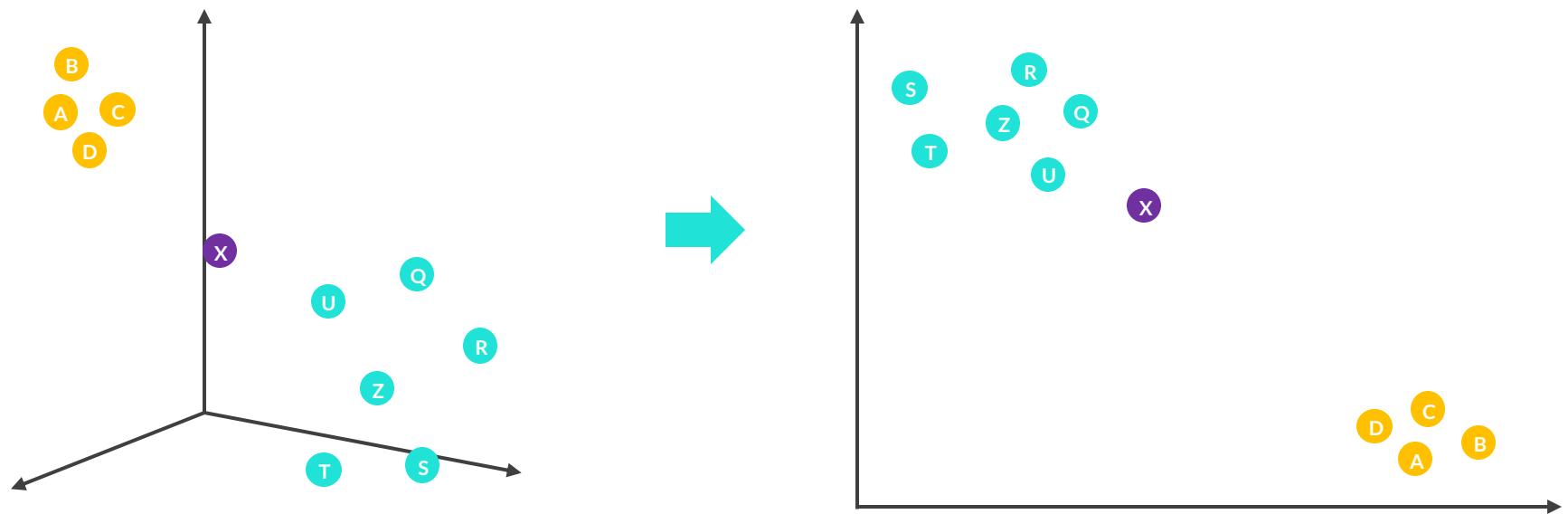
T-SNE

Dimensionality Reduction

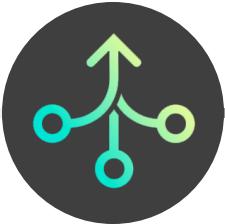
PCA

t-SNE

t-Distributed Stochastic Neighbor Embedding (t-SNE) is a non-linear dimensionality reduction technique that emphasizes relationships between points



t-SNE isn't concerned with maintaining the absolute distances between the points accurate, instead it focuses on offering a useful **estimate of the relationships** between them



T-SNE

t-Distributed Stochastic Neighbor Embedding (t-SNE) is a non-linear dimensionality reduction technique that emphasizes relationships between points

Dimensionality Reduction

PCA

t-SNE

Here's how it works:

1. Calculate the distance between points in a high-dimensional space
2. Use the distances to calculate the affinity score (0-1) for each point with the rest; in other words, how likely it is that they are “neighbors” (*high affinity score = high probability of being neighbors*)
3. Calculate the average affinity score for each pair of points
4. Randomly place the data points in a low dimensional space
5. Use the affinity scores between pairs to attract (*high affinity*) or repulse (*low affinity*) the points until they have “stabilized” into their final position

Example use cases:

- Visualizing clusters in limited dimensions to spot-check cluster/segmentation outputs



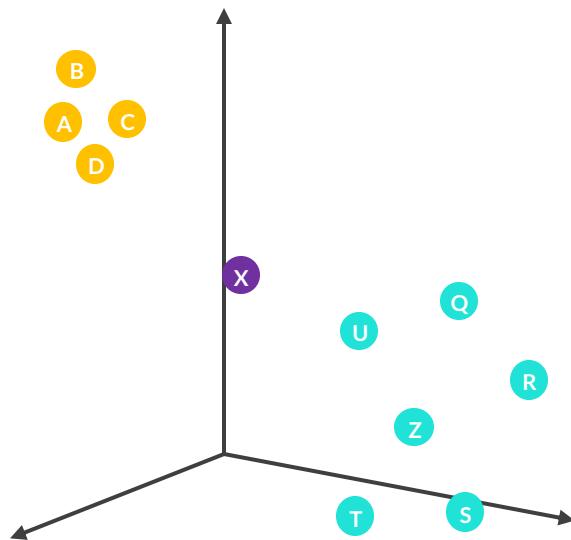
T-SNE

Dimensionality Reduction

PCA

t-SNE

STEP 1: Calculate the distance between points in a high-dimensional space





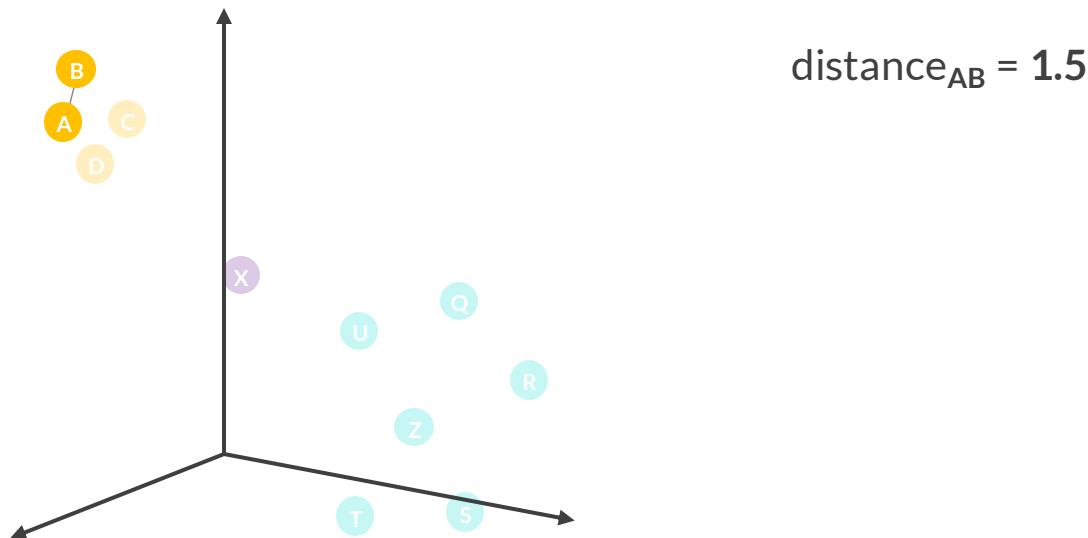
T-SNE

Dimensionality Reduction

PCA

t-SNE

STEP 1: Calculate the distance between points in a high-dimensional space





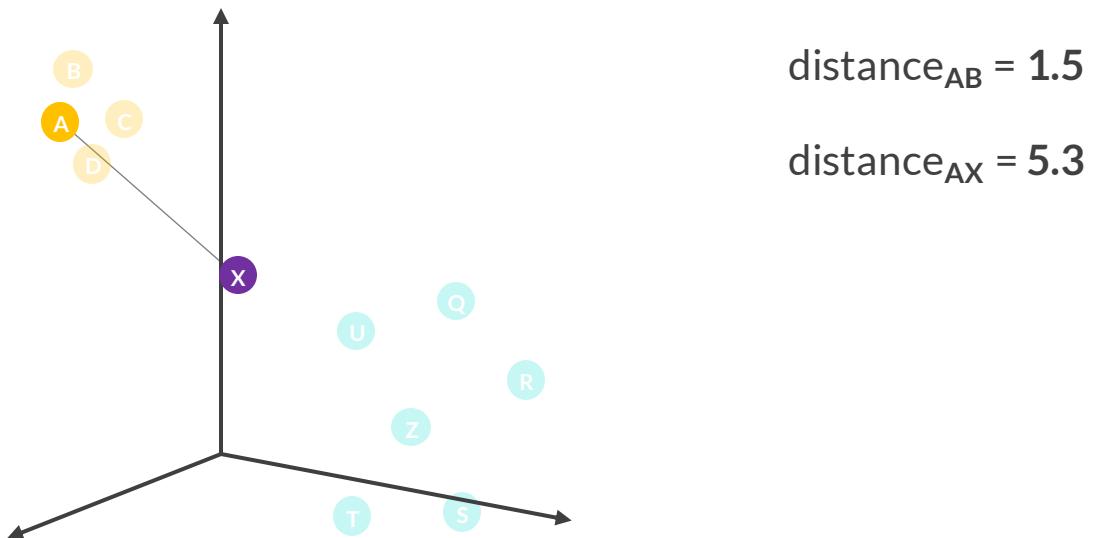
T-SNE

Dimensionality Reduction

PCA

t-SNE

STEP 1: Calculate the distance between points in a high-dimensional space





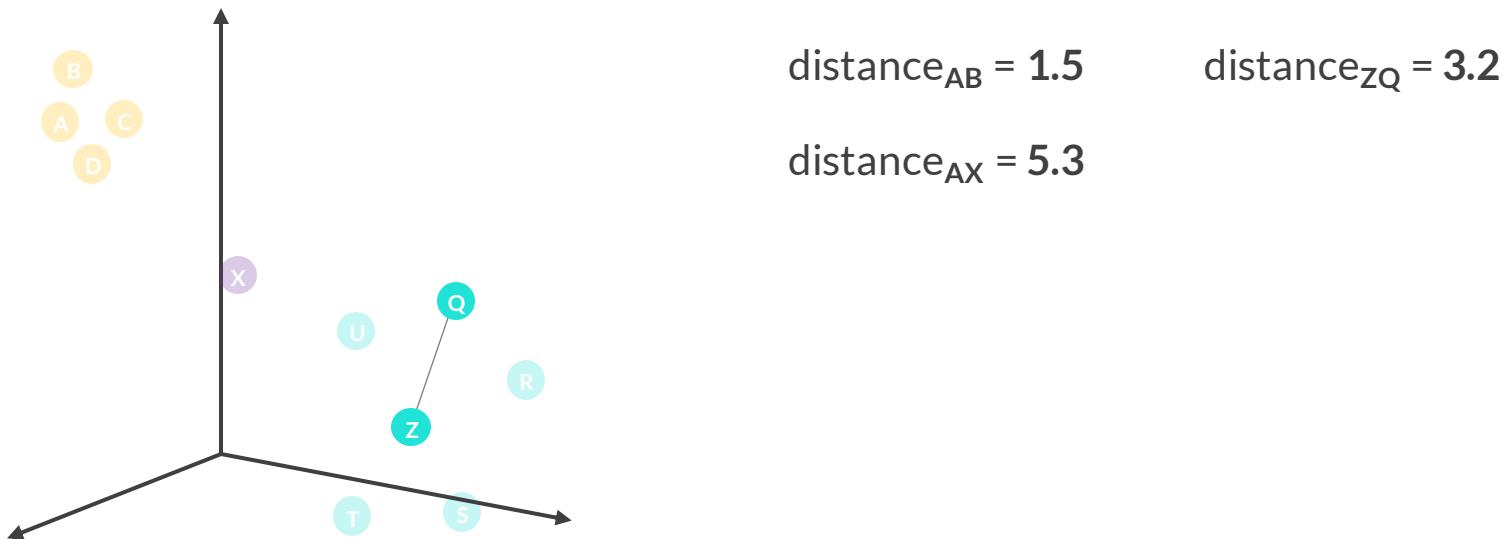
T-SNE

Dimensionality Reduction

PCA

t-SNE

STEP 1: Calculate the distance between points in a high-dimensional space





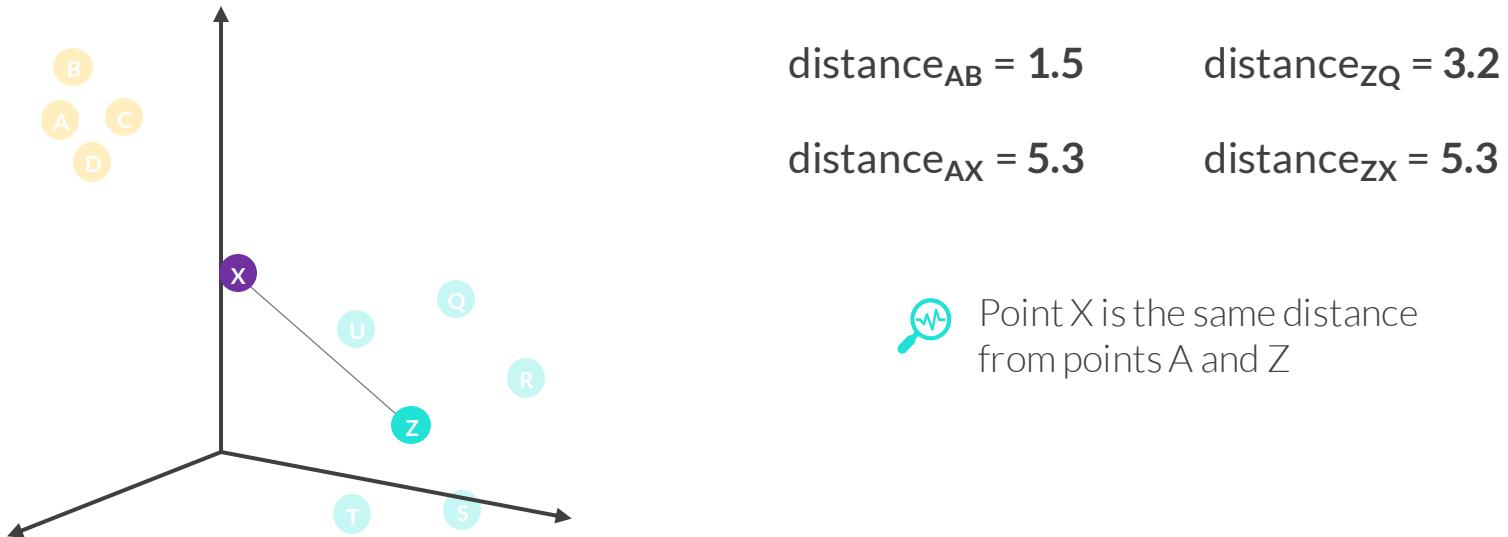
T-SNE

Dimensionality Reduction

PCA

t-SNE

STEP 1: Calculate the distance between points in a high-dimensional space



T-SNE

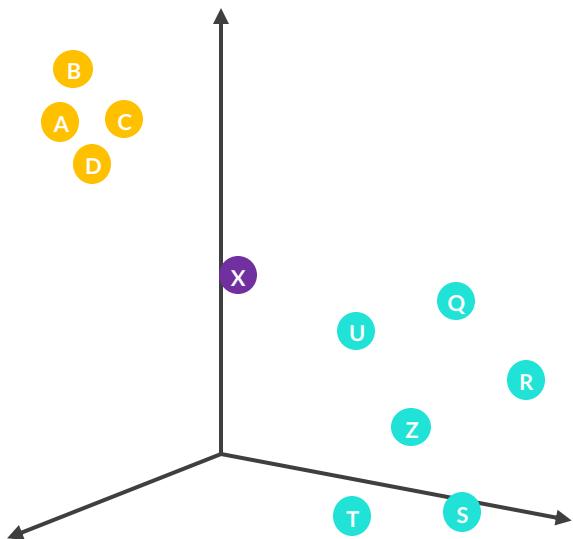


Dimensionality Reduction

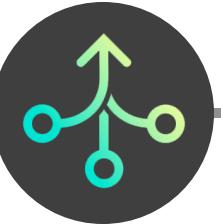
PCA

t-SNE

STEP 2: Use the distances to calculate the affinity score (0-1) for each point with the rest (*high affinity score = high probability of being neighbors*)



T-SNE

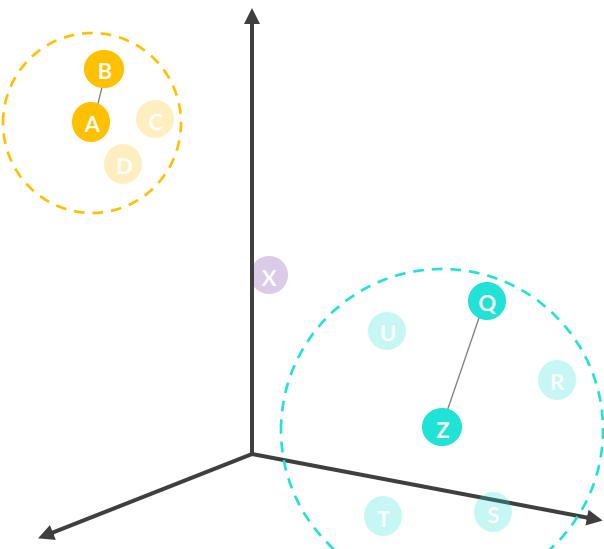


Dimensionality Reduction

PCA

t-SNE

STEP 2: Use the distances to calculate the affinity score (0-1) for each point with the rest (*high affinity score = high probability of being neighbors*)



$$\text{distance}_{AB} = 1.5$$

$$\text{distance}_{ZQ} = 3.2$$

affinity_{AB} = high

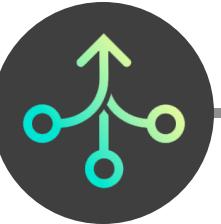
affinity_{ZQ} = high



How does this work?

- Based on the density of points around A and the distance between A and B, it's highly likely that B is A's neighbor
- Even though Q is further from Z than B is from A, since Z is part of a less dense region there is still a high chance that Q is Z's neighbor

T-SNE

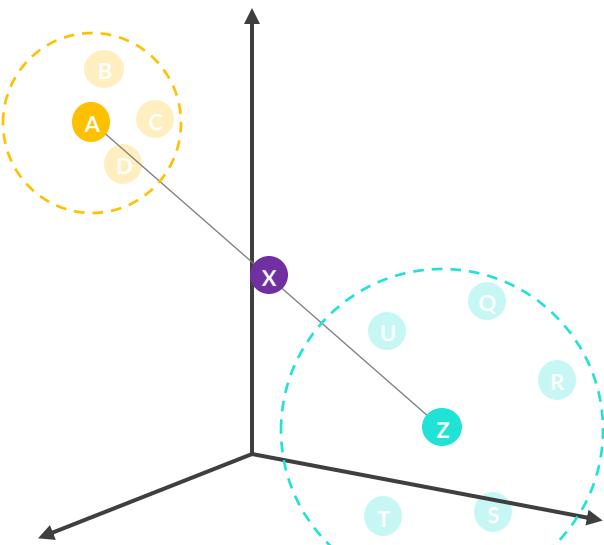


Dimensionality Reduction

PCA

t-SNE

STEP 2: Use the distances to calculate the affinity score (0-1) for each point with the rest (*high affinity score = high probability of being neighbors*)



$\text{distance}_{AX} = 5.3$

$\text{distance}_{ZX} = 5.3$

$\text{affinity}_{AX} = \text{low}$

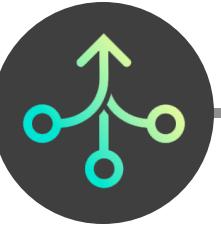
$\text{affinity}_{ZX} = \text{medium}$



How does this work?

- Even though X is the same distance from A and Z, it's more likely that X is a neighbor of Z due to the sparse density around Z

T-SNE

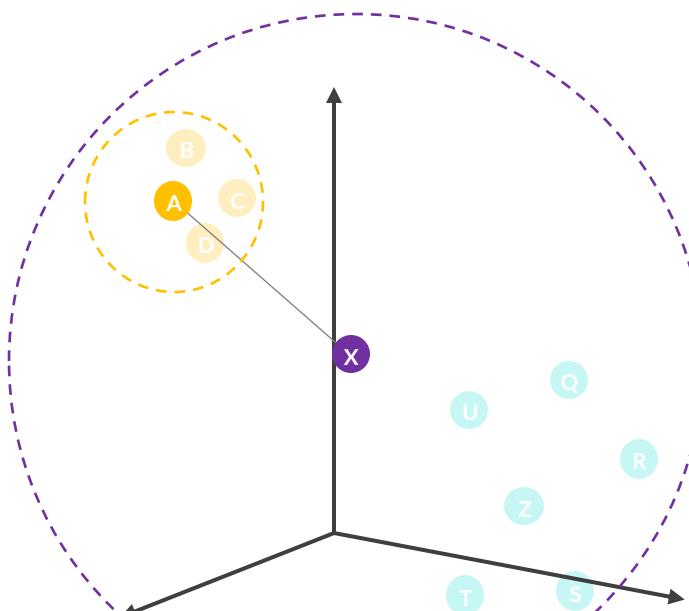


Dimensionality Reduction

PCA

t-SNE

STEP 2: Use the distances to calculate the affinity score (0-1) for each point with the rest (*high affinity score = high probability of being neighbors*)



$$\text{distance}_{AX} = 5.3$$

$$\text{distance}_{XA} = 5.3$$

$\text{affinity}_{AX} = \text{low}$

$\text{affinity}_{XA} = \text{medium}$



How does this work?

- The density of points around X is much sparser, so it's much more likely that A is a neighbor of X than X being a neighbor of A



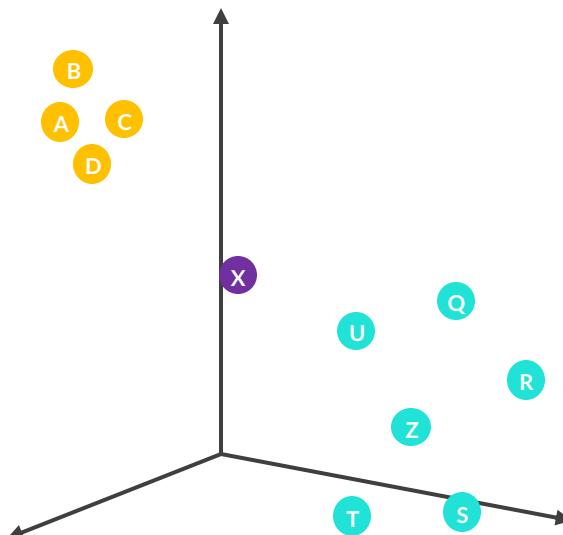
T-SNE

Dimensionality Reduction

PCA

t-SNE

STEP 2: Use the distances to calculate the affinity score (0-1) for each point with the rest (*high affinity score = high probability of being neighbors*)



$\text{affinity}_{AB} = \text{high}$

$\text{affinity}_{AX} = \text{low}$

$\text{affinity}_{XA} = \text{medium}$

$\text{affinity}_{zQ} = \text{high}$

$\text{affinity}_{zx} = \text{medium}$

$\text{affinity}_{xz} = \text{medium}$



If you're curious, the affinity score is calculated by putting the distances through a Gaussian transformation and calculating probabilities – but it's **not something you should be worried about!**



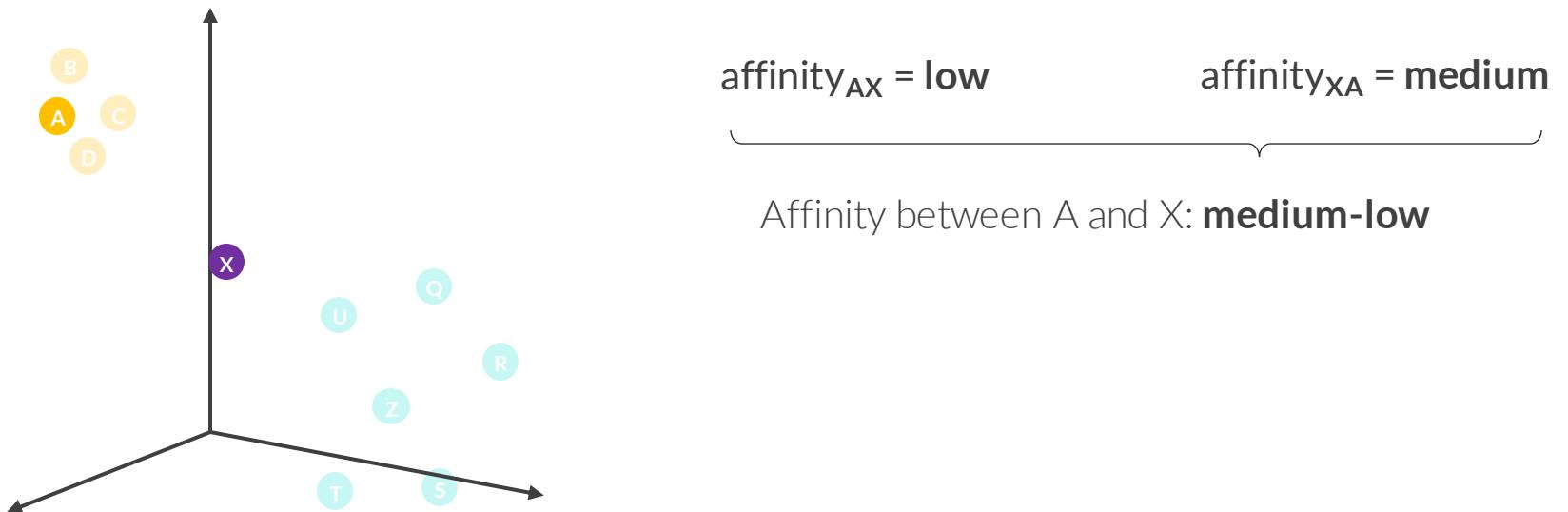
T-SNE

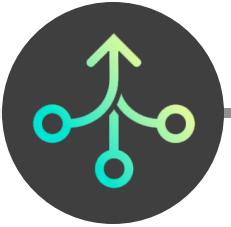
Dimensionality Reduction

PCA

t-SNE

STEP 3: Calculate the average affinity score for each pair of points





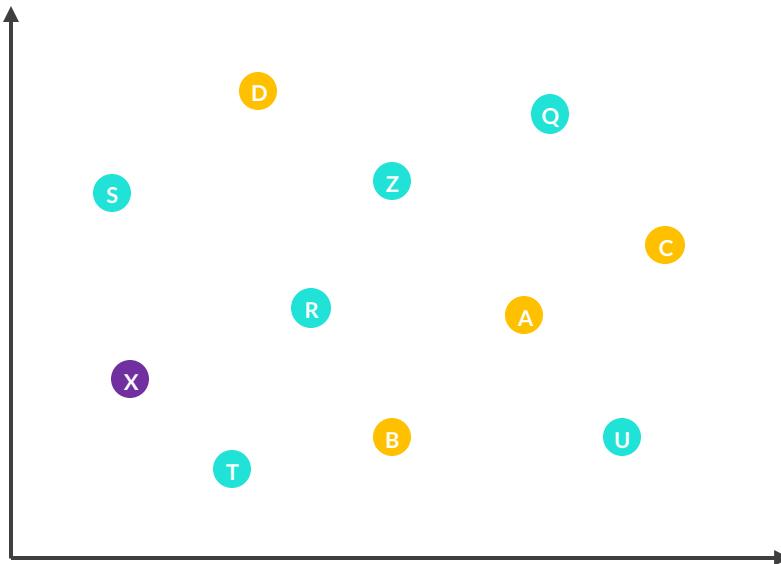
T-SNE

Dimensionality Reduction

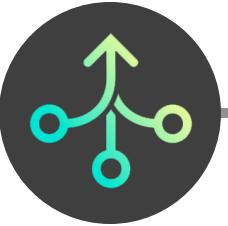
PCA

t-SNE

STEP 4: Randomly place the data points in a low dimensional space



T-SNE

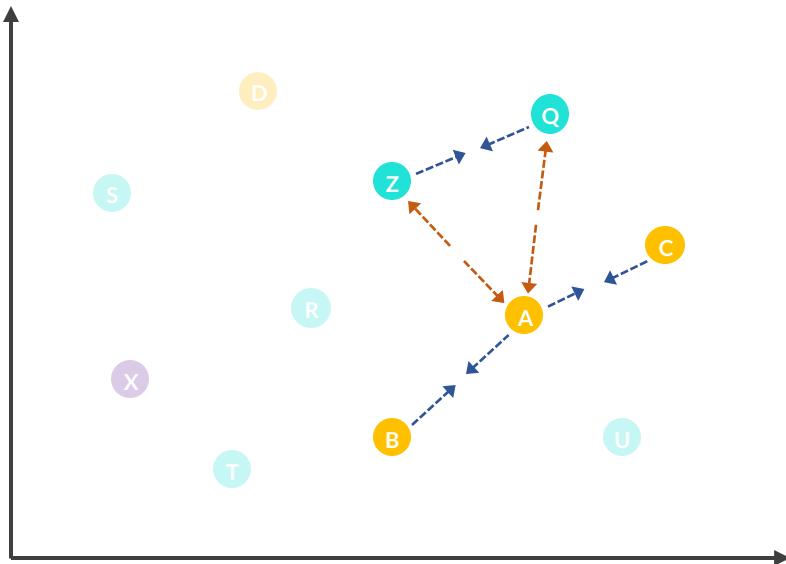


Dimensionality Reduction

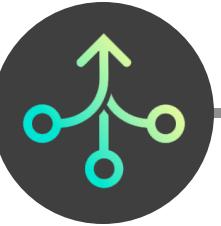
PCA

t-SNE

STEP 5: Use the affinity scores between pairs to attract (*high affinity*) or repulse (*low affinity*) the points until they have “stabilized” into their final position



T-SNE

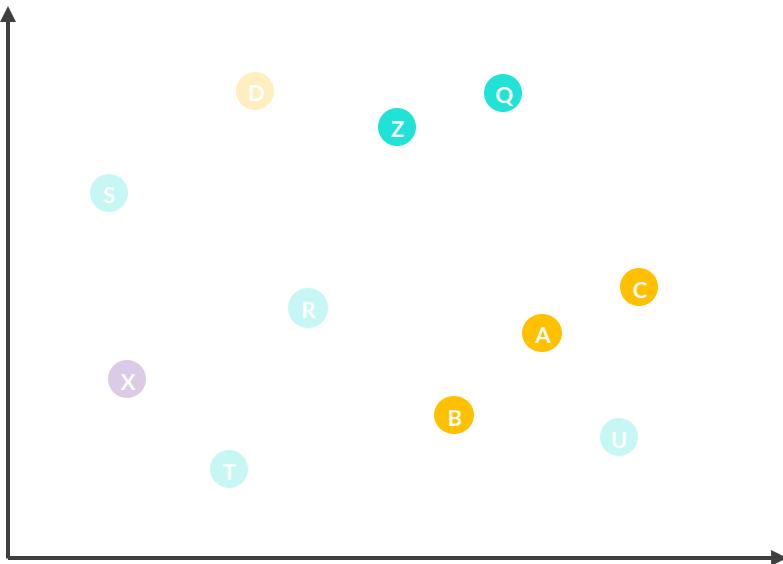


Dimensionality Reduction

PCA

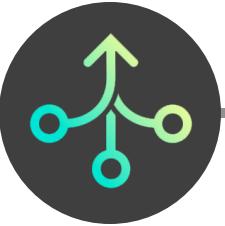
t-SNE

STEP 5: Use the affinity scores between pairs to attract (*high affinity*) or repulse (*low affinity*) the points until they have “stabilized” into their final position



This attraction and repulsion is done by **gradient descent** on a KL loss function

T-SNE

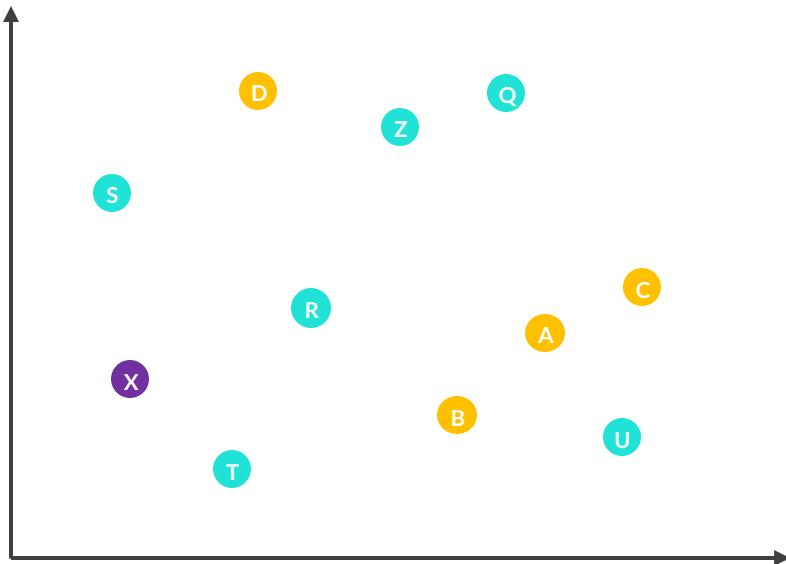


Dimensionality Reduction

PCA

t-SNE

STEP 5: Use the affinity scores between pairs to attract (*high affinity*) or repulse (*low affinity*) the points until they have “stabilized” into their final position





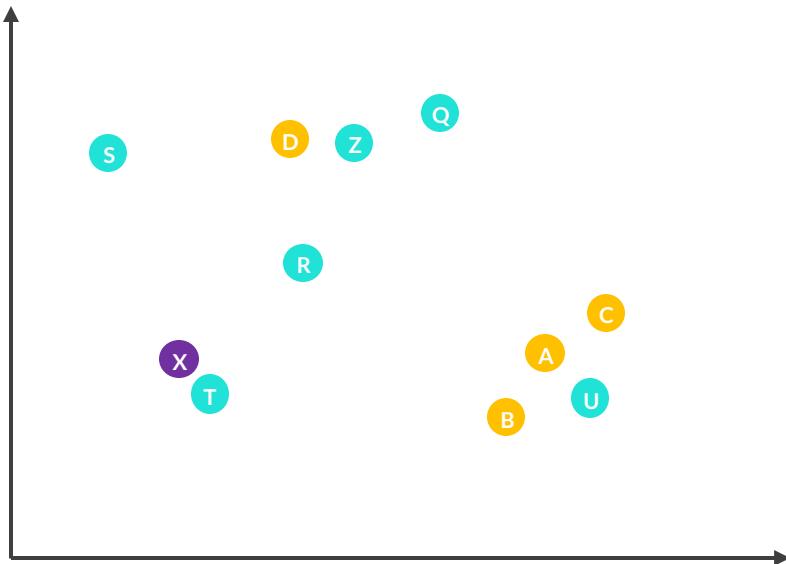
T-SNE

Dimensionality Reduction

PCA

t-SNE

STEP 5: Use the affinity scores between pairs to attract (*high affinity*) or repulse (*low affinity*) the points until they have “stabilized” into their final position



T-SNE

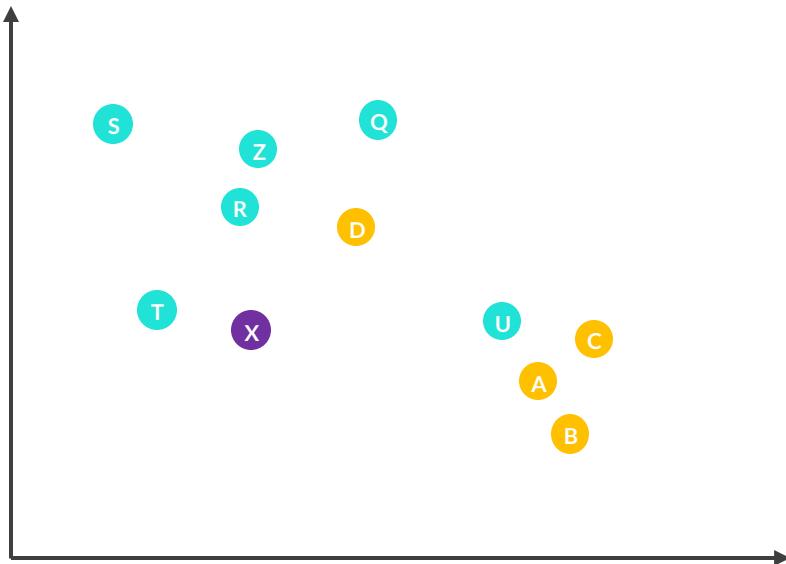


Dimensionality Reduction

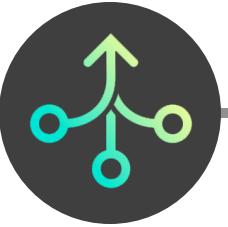
PCA

t-SNE

STEP 5: Use the affinity scores between pairs to attract (*high affinity*) or repulse (*low affinity*) the points until they have “stabilized” into their final position



T-SNE

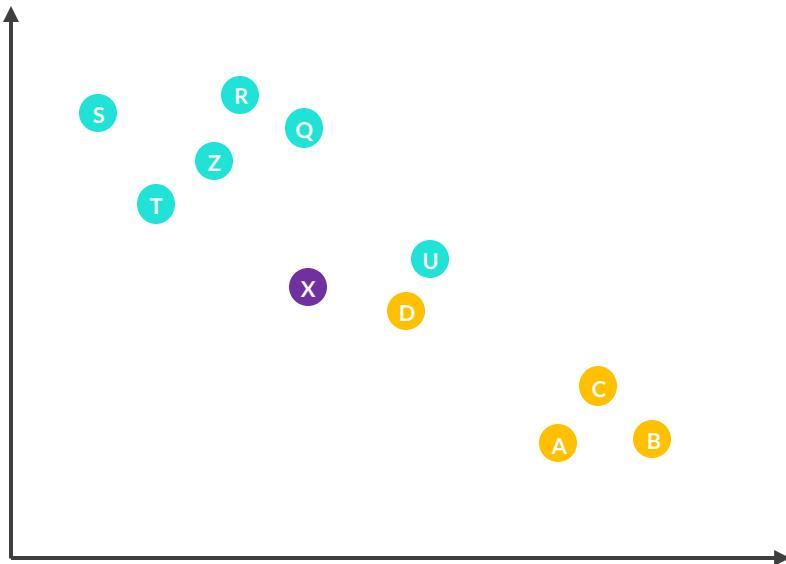


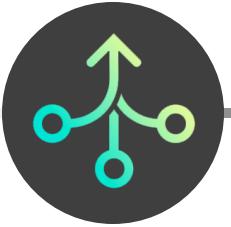
Dimensionality Reduction

PCA

t-SNE

STEP 5: Use the affinity scores between pairs to attract (*high affinity*) or repulse (*low affinity*) the points until they have “stabilized” into their final position





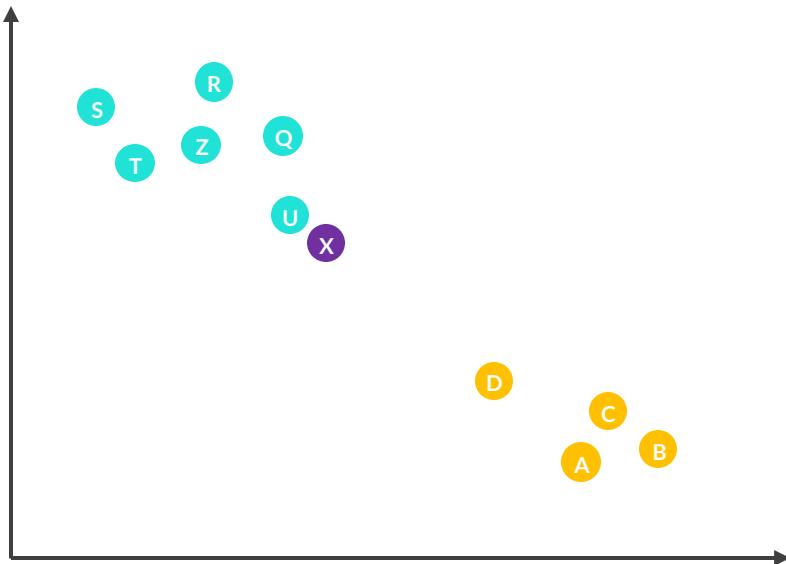
T-SNE

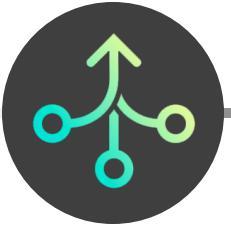
Dimensionality Reduction

PCA

t-SNE

STEP 5: Use the affinity scores between pairs to attract (*high affinity*) or repulse (*low affinity*) the points until they have “stabilized” into their final position





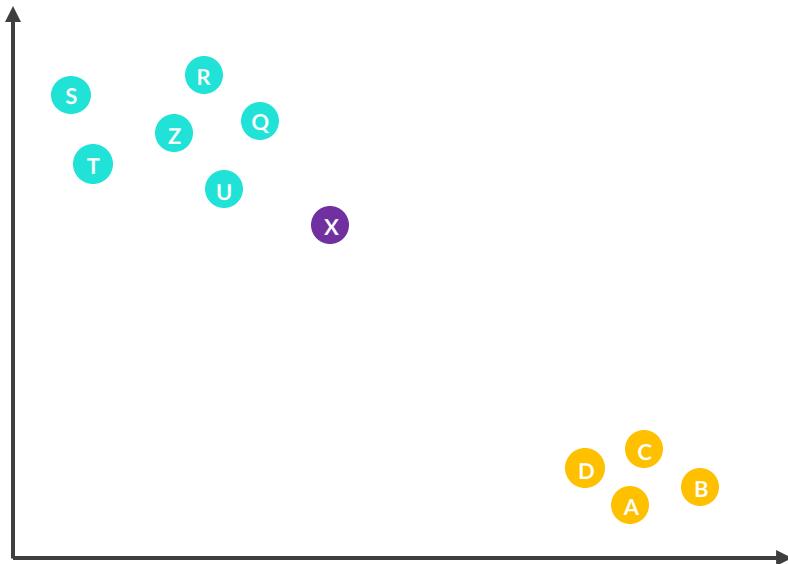
T-SNE

Dimensionality Reduction

PCA

t-SNE

STEP 5: Use the affinity scores between pairs to attract (*high affinity*) or repulse (*low affinity*) the points until they have “stabilized” into their final position



T-SNE

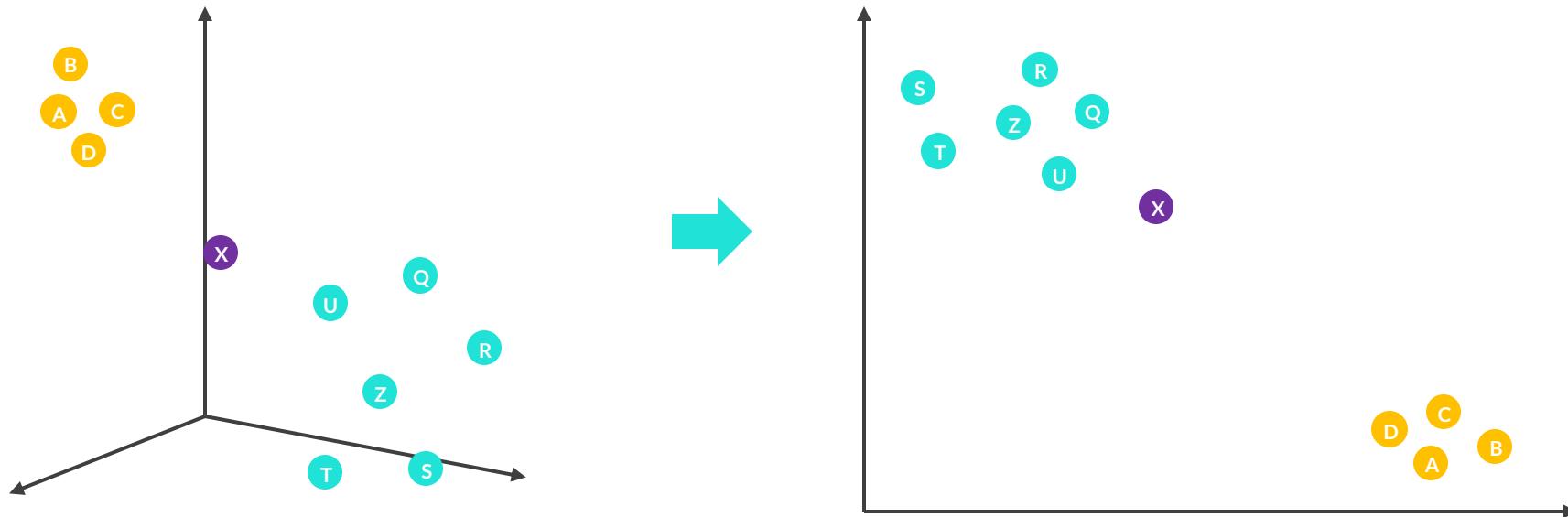


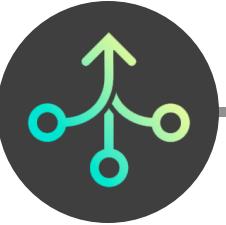
Dimensionality Reduction

PCA

t-SNE

STEP 5: Use the affinity scores between pairs to attract (*high affinity*) or repulse (*low affinity*) the points until they have “stabilized” into their final position





T-SNE IN PYTHON

Dimensionality Reduction

PCA

t-SNE

You can use sklearn's **TSNE()** to apply the t-SNE algorithm in Python

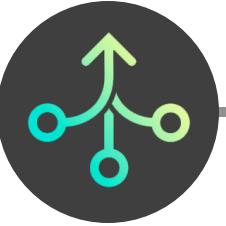
```
from sklearn.manifold import TSNE  
  
tsne = TSNE(n_components=2, random_state=42)
```

The number of columns you want to reduce the dataset into (ideally 2 or 3 for visualization)

Setting a random_state value guarantees the same output after each run



PRO TIP: Unlike PCA, you don't need to center the data for t-SNE since the algorithm is based on pairwise similarities, but like PCA, scaling is recommended



T-SNE IN PYTHON

Dimensionality Reduction

PCA

t-SNE

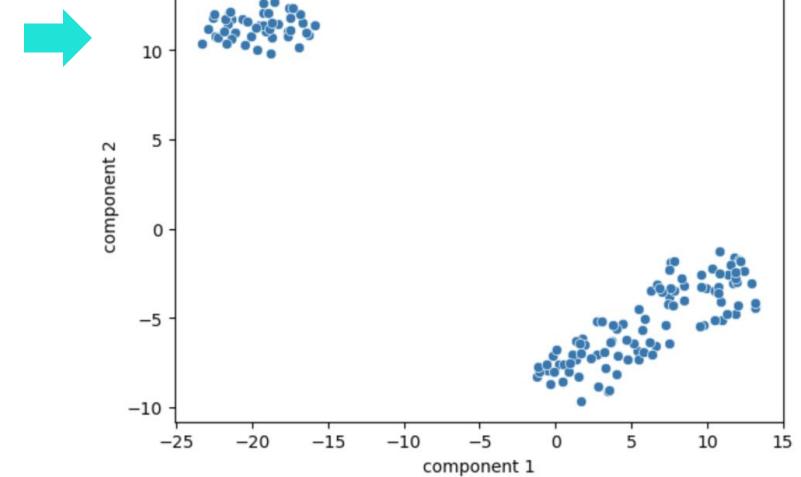
```
# import tsne from sklearn
from sklearn.manifold import TSNE

# create a tsne object
tsne = TSNE(n_components=2, random_state=100)

# fit a model and transform the data into a lower dimensionality space
data_tsne = tsne.fit_transform(data)

# plot the data
df_tsne = pd.DataFrame(data_tsne,
                        columns=['component 1',
                                  'component 2'])

sns.scatterplot(x='component 1',
                 y='component 2',
                 data=df_tsne);
```



Using t-SNE, it's much easier to identify the two clusters in the data

ASSIGNMENT: T-SNE

 **NEW MESSAGE**
March 28, 2024

From: Tim Menschen (Assistant Principal)
Subject: RE: Student Analysis Visualization

Hello again!

Thanks for plotting the students using PCA earlier.

Can you do the same using t-SNE?

I'm hoping we can better distinguish them visually.

Thanks,
Tim

Reply **Forward**

Key Objectives

1. Fit a t-SNE model with 2 components
2. Plot the students on a scatter plot with the x-axis as component 1 and the y-axis as component 2
3. Interpret the data you see on the plot

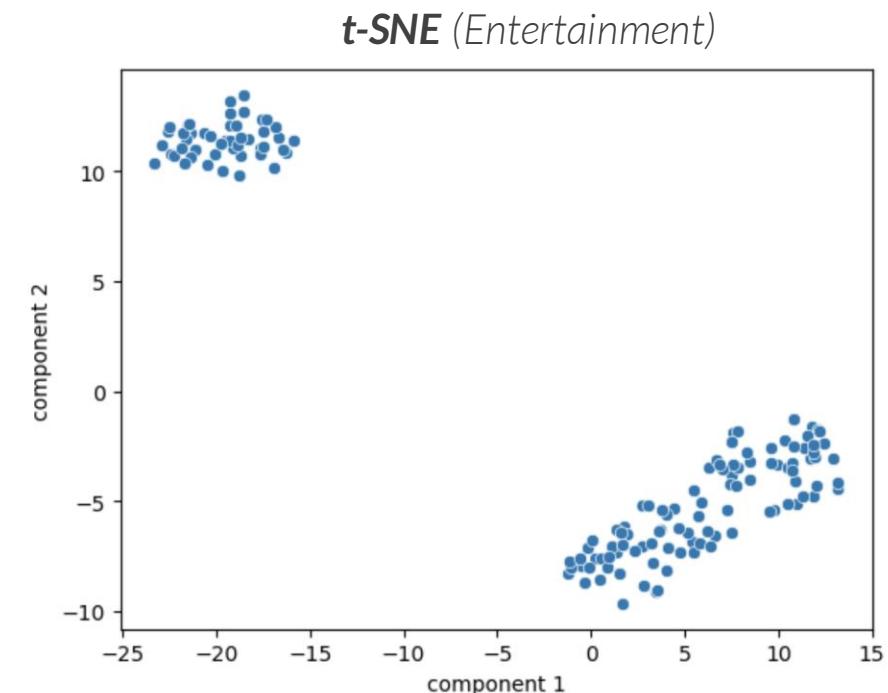
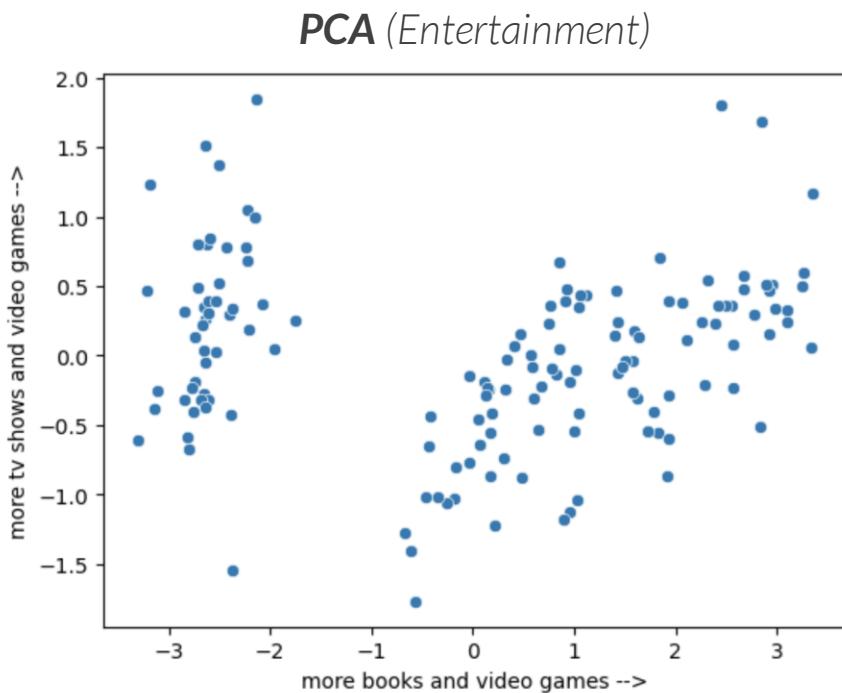


PCA VS T-SNE

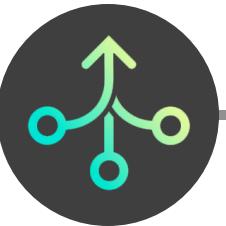
Dimensionality Reduction

PCA

t-SNE



With t-SNE, the two clusters in the data are clearly displayed



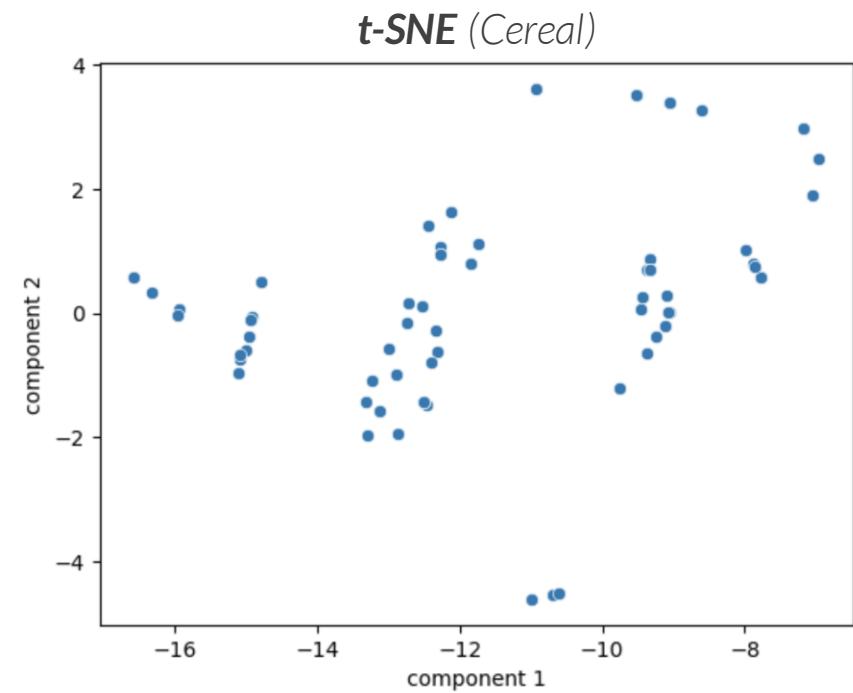
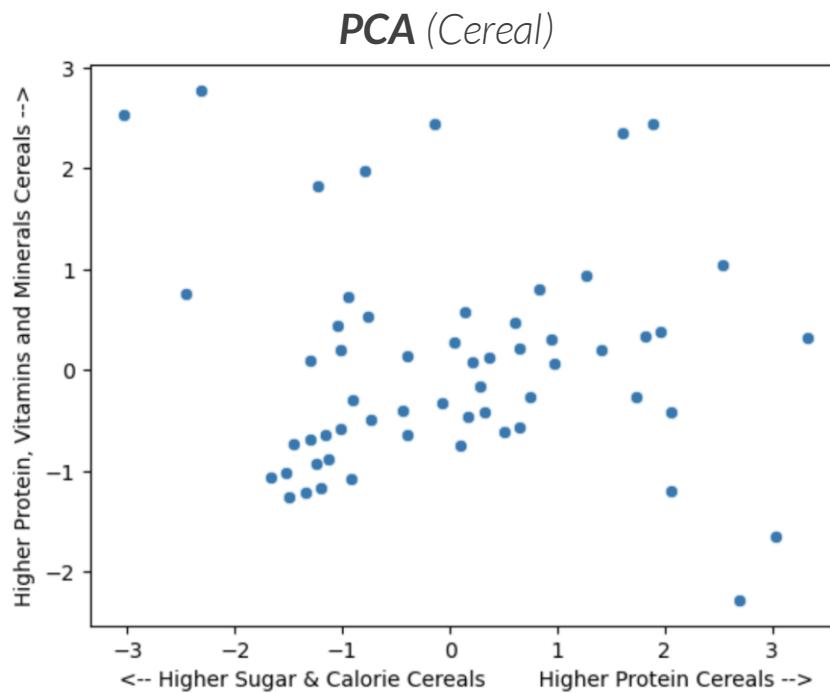
PCA VS T-SNE

Dimensionality Reduction

PCA

t-SNE

PCA visualizations are more accurate and interpretable, while **t-SNE** visualizations are better at visually separating groups of data into clusters



With t-SNE, the large blob of data is shown as more distinct groups



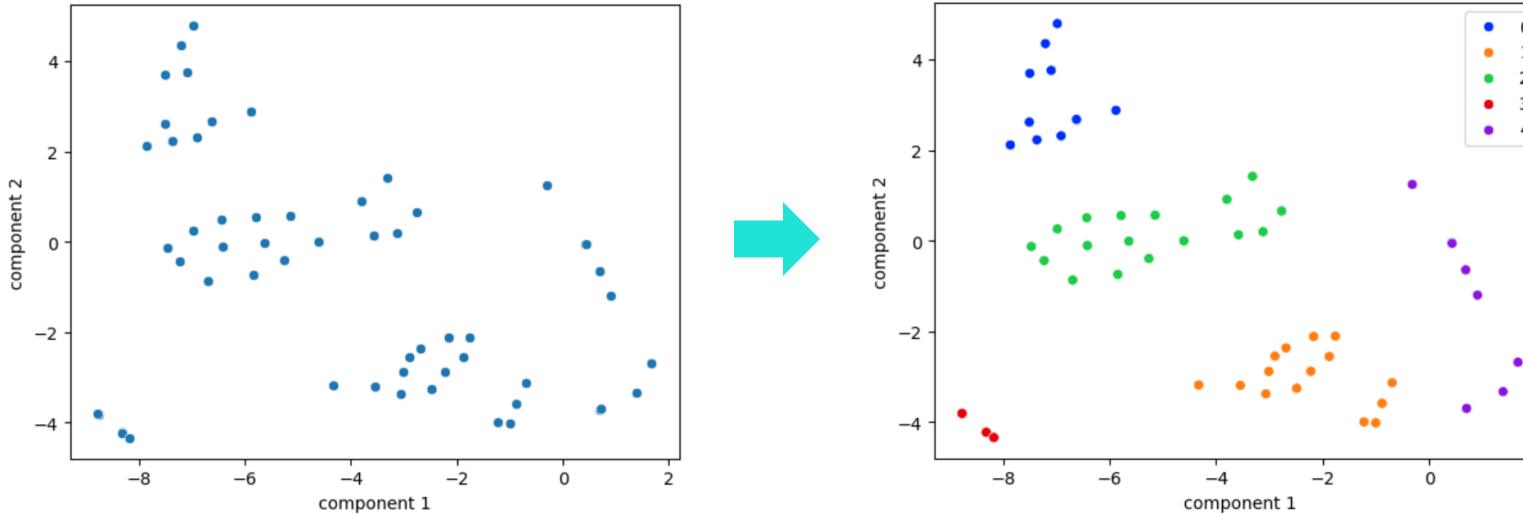
DIMENSIONALITY REDUCTION & CLUSTERING

Dimensionality Reduction

PCA

t-SNE

Visualizations created from **dimensionality reduction** techniques like PCA and t-SNE can help identify clusters while fitting **clustering** algorithms



PRO TIP: You can use this visualization approach in addition to metrics (inertia plots, silhouette scores, etc.) and intuition when deciding on the ideal number of clusters when clustering

ASSIGNMENT: T-SNE & K-MEANS CLUSTERING

 **NEW MESSAGE**
March 29, 2024

From: Tim Menschen (Assistant Principal)
Subject: RE: Student Analysis Visualization

Happy Friday!

Final request here, I promise.

Can you highlight the clusters in the t-SNE plot using colors?

I think that will really impress the staff.

Thanks,
Tim

[Reply](#) [Forward](#)

Key Objectives

1. Fit a K-Means model with 3 clusters
2. Overlay the 3 clusters onto a t-SNE plot
3. Interpret the cluster centers

KEY TAKEAWAYS



Dimensionality reduction helps reduce columns without losing information

- Common dimensionality reduction techniques include Principal Component Analysis (PCA) and t-SNE
- PCA is a good first algorithm to try, while t-SNE typically does a better job distinguishing clusters



PCA is great for both **data visualization** and **feature extraction**

- Use the explained variance ratio to understand how representative of the original data your components are
- The resulting components of PCA are somewhat interpretable, and can be input into other ML algorithms



t-SNE is ideal for **data visualization**

- Even though its components are more difficult to interpret, it creates a clear visual separation between clusters
- t-SNE can also help determine the ideal number of clusters for modeling

RECOMMENDERS

RECOMMENDERS



In this section we'll introduce the concept of **recommenders** and cover two common ways of making recommendations: content-based filtering and collaborative filtering

TOPICS WE'LL COVER:

Recommenders Basics

Content-Based Filtering

Collaborative Filtering

Next Steps

GOALS FOR THIS SECTION:

- Understand the difference between content-based and collaborative filtering recommenders
- Use cosine similarity to make recommendations by identifying similar items
- Use Singular Value Decomposition (SVD) to make recommendations by identifying similar users



RECOMMENDERS BASICS

Recommenders
Basics

Content-Based
Filtering

Collaborative
Filtering

Next Steps

Recommenders, also known as recommender systems or recommendation engines, are used to suggest personalized items to users based on their preferences

There are two main approaches to making recommendations:

Content-Based Filtering

Recommend items to users based on items with similar characteristics

Popular approach: Cosine similarity

Required input data:

- Rows: Items
- Columns: Item characteristics

Collaborative Filtering

Recommend items to users based on the behaviors or preferences of similar users

Popular approach: SVD

Required input data:

- Rows: Users
- Columns: Items



In addition to using cosine similarity and SVD, there are **many other approaches to creating recommenders**, including algorithms like regression, classification, clustering, association rule mining, and more



CONTENT-BASED FILTERING

Recommenders
Basics

Content-Based
Filtering

Collaborative
Filtering

Next Steps

Content-based filtering recommenders are used to suggest personalized items to users based on items with similar characteristics

- In other words, if a user listens to a lot of classical music, recommend more classical music

EXAMPLE

Recommending fruits based on their nutritional profiles

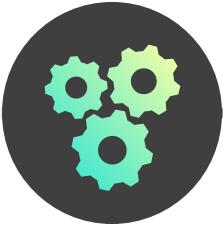
Each row
represents
an **item** to
recommend

The features are **item characteristics**

Fruit	Energy	Vitamin C	Sugar
Banana	89	8.7	12.2
Lime	30	29.1	1.69
Mango	46	4.1	8.39
Peach	60	36.4	13.7



What other fruit would you recommend to someone who likes mangos?



COSINE SIMILARITY

Recommenders
Basics

Content-Based
Filtering

Collaborative
Filtering

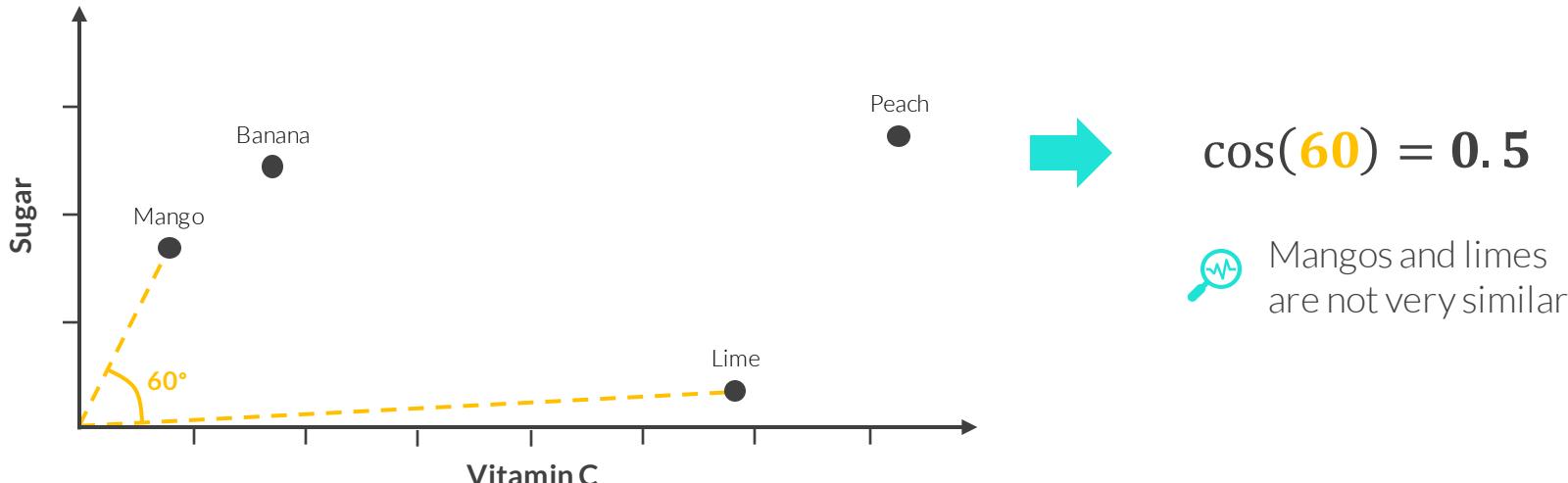
Next Steps

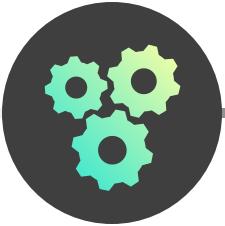
Cosine similarity is a metric used to calculate the similarity between observations

- Values range from -1 (dissimilar) to 1 (similar)
- If all data points are in the first quadrant, the values will range from 0 (dissimilar) to 1 (similar)

EXAMPLE

Recommending fruits based on their nutritional profiles





COSINE SIMILARITY

Recommenders
Basics

Content-Based
Filtering

Collaborative
Filtering

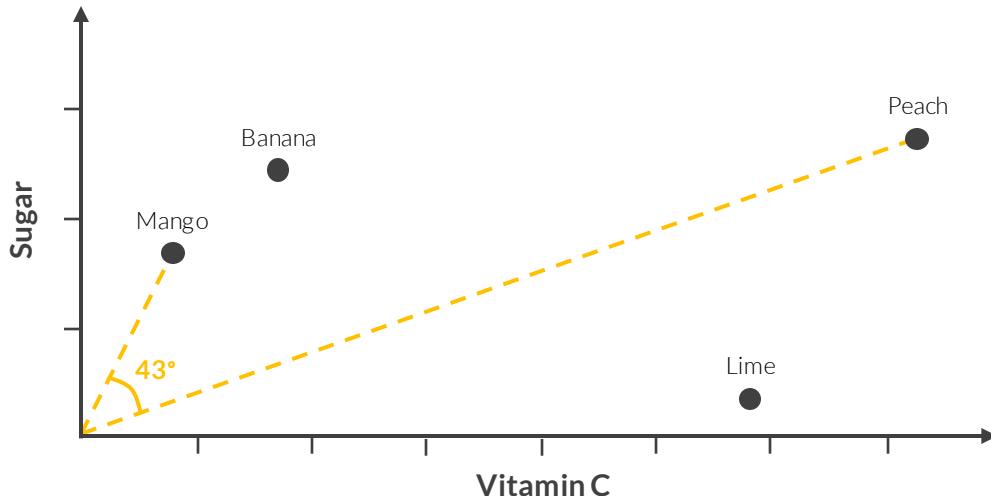
Next Steps

Cosine similarity is a metric used to calculate the similarity between observations

- Values range from -1 (dissimilar) to 1 (similar)
- If all data points are in the first quadrant, the values will range from 0 (dissimilar) to 1 (similar)

EXAMPLE

Recommending fruits based on their nutritional profiles



$$\cos(43^\circ) = 0.73$$



Peaches are more similar
to mangos than limes



COSINE SIMILARITY

Recommenders
Basics

Content-Based
Filtering

Collaborative
Filtering

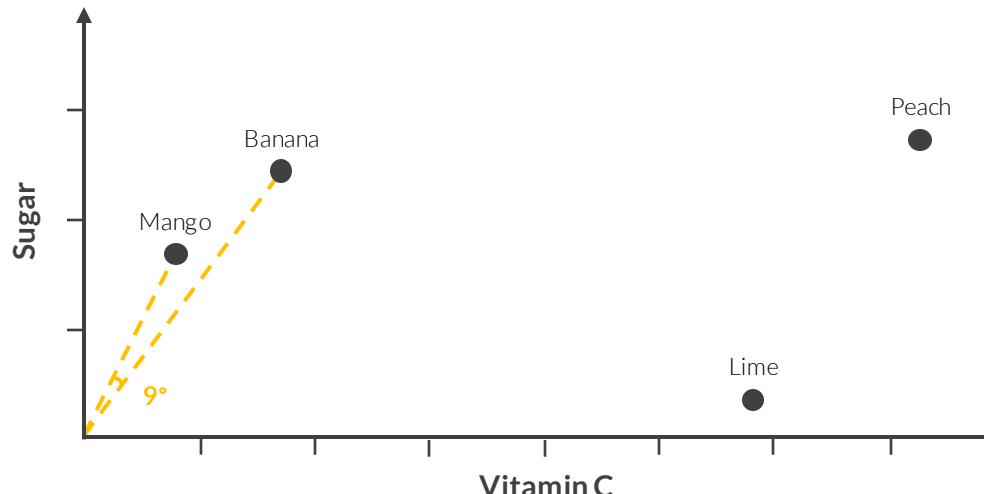
Next Steps

Cosine similarity is a metric used to calculate the similarity between observations

- Values range from -1 (dissimilar) to 1 (similar)
- If all data points are in the first quadrant, the values will range from 0 (dissimilar) to 1 (similar)

EXAMPLE

Recommending fruits based on their nutritional profiles



$$\cos(9^\circ) = 0.98$$



Bananas are the most similar fruit to mangos – we should recommend them!



COSINE SIMILARITY

Recommenders
Basics

Content-Based
Filtering

Collaborative
Filtering

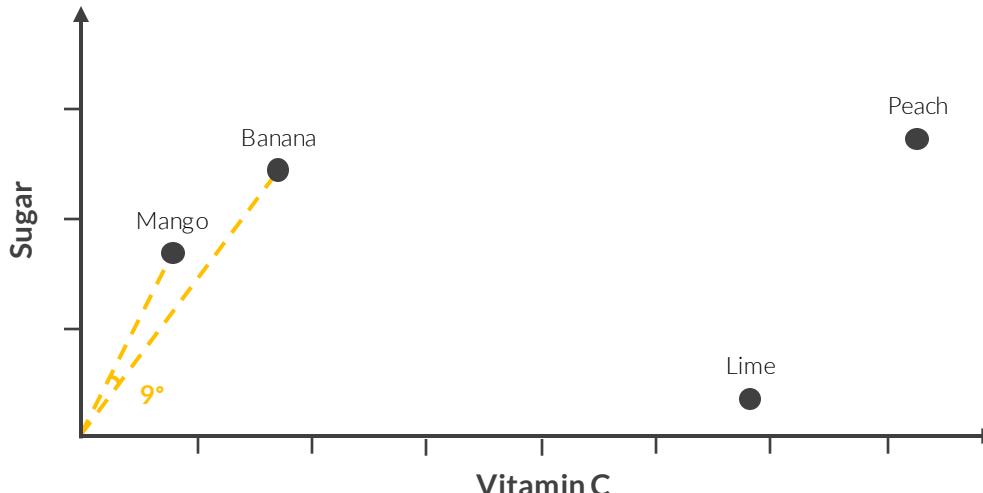
Next Steps

Cosine similarity is a metric used to calculate the similarity between observations

- Values range from -1 (dissimilar) to 1 (similar)
- If all data points are in the first quadrant, the values will range from 0 (dissimilar) to 1 (similar)

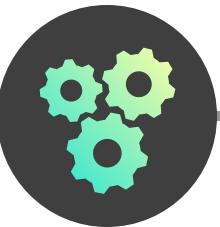
EXAMPLE

Recommending fruits based on their nutritional profiles



There are many similarity metrics to choose from, but cosine similarity is popular in machine learning because:

- It focuses on **direction** instead of magnitude
- It can handle **high dimensions**
- It works well on **sparse data** (data containing many 0 values)



COSINE SIMILARITY IN PYTHON

Recommenders
Basics

Content-Based
Filtering

Collaborative
Filtering

Next Steps

You can use sklearn's **cosine_similarity()** function to calculate the cosine similarity between two items (rows) in a DataFrame

```
df = nutrition.iloc[:, 5:]  
df
```

	sugars_g	vitaminc_mg
Banana	12.20	8.7
Lemon	2.50	53.0
Lime	1.69	29.1
Mango	8.39	4.1
Peach	13.70	36.4
Pineapple	9.85	47.8



```
from sklearn.metrics.pairwise import cosine_similarity  
  
cosine_matrix = cosine_similarity(df)  
cosine_matrix  
  
array([[1.          , 0.61832414, 0.62683477, 0.9864306 , 0.83018982, 0.73298243],  
       [0.61832414, 1.          , 0.99994086, 0.48090193, 0.95146351, 0.98784307],  
       [0.62683477, 0.99994086, 1.          , 0.49040872, 0.95475426, 0.98947527],  
       [0.9864306 , 0.48090193, 0.49040872, 1.          , 0.72739811, 0.61135407],  
       [0.83018982, 0.95146351, 0.95475426, 0.72739811, 1.          , 0.98773952],  
       [0.73298243, 0.98784307, 0.98947527, 0.61135407, 0.98773952, 1.        ]])
```

Each row / column in the matrix contains the cosine similarities between that item and the rest





COSINE SIMILARITY IN PYTHON

Recommenders
Basics

Content-Based
Filtering

Collaborative
Filtering

Next Steps

You can use sklearn's **cosine_similarity()** function to calculate the cosine similarity between two items (rows) in a DataFrame

```
df = nutrition.iloc[:, 5:]  
df
```

	sugars_g	vitaminc_mg
Banana	12.20	8.7
Lemon	2.50	53.0
Lime	1.69	29.1
Mango	8.39	4.1
Peach	13.70	36.4
Pineapple	9.85	47.8



```
from sklearn.metrics.pairwise import cosine_similarity  
  
cosine_matrix = cosine_similarity(df)  
  
cosine_df = pd.DataFrame(cosine_matrix, index=df.index, columns=df.index)  
cosine_df
```

	Banana	Lemon	Lime	Mango	Peach	Pineapple
Banana	1.000000	0.618324	0.626835	0.986431	0.830190	0.732982
Lemon	0.618324	1.000000	0.999941	0.480902	0.951464	0.987843
Lime	0.626835	0.999941	1.000000	0.490409	0.954754	0.989475
Mango	0.986431	0.480902	0.490409	1.000000	0.727398	0.611354
Peach	0.830190	0.951464	0.954754	0.727398	1.000000	0.987740
Pineapple	0.732982	0.987843	0.989475	0.611354	0.987740	1.000000



Bananas are the most similar fruit to mangos!
(if we only use sugar & vitamin C as characteristics)



CONTENT-BASED FILTERING RECOMMENDATIONS

Recommenders
Basics

Content-Based
Filtering

Collaborative
Filtering

Next Steps

To **make content-based recommendations** using cosine similarity, simply filter the cosine similarity matrix to the relevant item and sort the similarity scores

```
# if i like mangos, what other fruits might i like?  
cosine_df[['Mango']].sort_values('Mango', ascending=False)
```

Mango	
Mango	1.000000
Banana	0.986431
Peach	0.727398
Pineapple	0.611354
Lime	0.490409
Lemon	0.480902



If you like mangos, I'd recommend trying bananas

ASSIGNMENT: CONTENT-BASED FILTERING

 **NEW MESSAGE**
April 1, 2024

From: Becca Mender (Data Scientist)
Subject: Movie Recommender Project

Hi there, I'm creating a website that recommends movies. I've manually labelled 1600+ movies with 18 genres, and would like to create a proof of concept to see if this data can actually be effective at recommending movies.

Can you help me:

- Find the similarity between Toy Story and Get Shorty
- Find the top 5 movies that are most similar to Toy Story

Thanks!
Becca

 Movie_Ratings.xlsx  Reply  Forward

Key Objectives

1. Read in the data from the first tab of the spreadsheet, which contains genre labels
2. Remove the non-genre columns in the data
3. Calculate the cosine similarity between Toy Story (1995) and Get Shorty (1995)
4. Calculate the cosine similarity between Toy Story (1995) and the rest of the movies
5. Return the 5 movies with the highest similarity



COLLABORATIVE FILTERING

Recommenders
Basics

Content-Based
Filtering

Collaborative
Filtering

Next Steps

Collaborative filtering recommenders are used to suggest personalized items to users based on the behaviors or preferences of similar users

- **User-based:** find users that like the same songs as me and recommend other songs they like
- **Item-based:** find songs similar to songs that I like and recommend those songs

EXAMPLE

Recommending fruits based on personal taste preferences

The features are **items** to recommend

Each row
represents a
user

User	Banana	Lime	Mango
Adam	1.1	4.7	1
Liz	4.2	2.3	3.2
Daisy	3.7	1.4	4.9
Bob	1.2	4.5	1.3

The values represent **ratings**

This is known as a **user-item matrix**



What other fruit would you recommend to someone who likes mangos?



USER-ITEM MATRIX IN PYTHON

Recommenders
Basics

Content-Based
Filtering

Collaborative
Filtering

Next Steps

```
# fruit ratings  
fruit_ratings.head(10)
```

	User	Fruit	Rating
0	user0	Lemon	4.3
1	user1	Lemon	4.6
2	user2	Lemon	4.2
3	user3	Lemon	4.4
4	user4	Lemon	4.1
5	user5	Lemon	4.5
6	user6	Lemon	4.3
7	user7	Lemon	4.6
8	user8	Lemon	4.2
9	user9	Lemon	4.4

```
# restructure the data into a user-item matrix  
X = (fruit_ratings.pivot(index='User', columns='Fruit', values='Rating').fillna(3))  
X.head()
```

	Fruit	Banana	Lemon	Lime	Mango	Peach	Pineapple
User							
user0		1.1	4.3	4.7	1.0	2.4	4.9
user1		1.2	4.6	4.5	1.3	2.1	4.8
user10		4.8	3.7	4.1	2.2	3.3	2.4
user11		2.5	2.9	3.7	4.1	1.8	4.1
user12		4.2	4.1	2.3	3.2	5.0	1.7

Make sure to fill the
blank values with the
mean of the ratings



Notice that the data has changed from each row
representing a rating to each row representing a user

ASSIGNMENT: USER-ITEM MATRIX

 **NEW MESSAGE**
April 2, 2024

From: Becca Mender (Data Scientist)
Subject: Collaborative Filtering Data Prep

Hi again,

Thanks for your help earlier providing recommendations using the movies data set!

I'd like to try using a collaborative filtering approach as well, so I've collected movie ratings from almost a thousand people who responded to my survey.

Can you help me restructure the ratings data into a user-item matrix to get it ready for modeling?

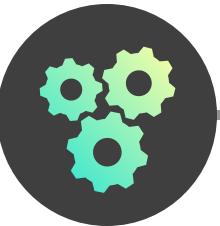
Thanks!

 Movie_Ratings.xlsx

Reply Forward

Key Objectives

1. Read in the movies, users, and ratings tabs of the spreadsheet into three DataFrames
2. Use `.pivot()` to restructure the ratings data into a user-item matrix



SINGULAR VALUE DECOMPOSITION

Recommenders
Basics

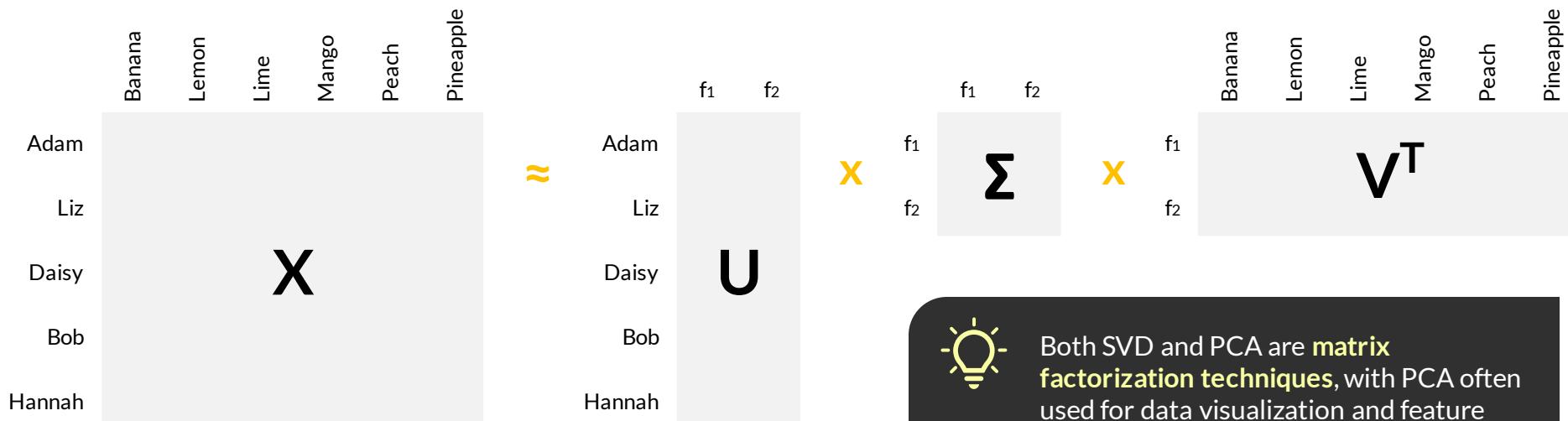
Content-Based
Filtering

Collaborative
Filtering

Next Steps

Singular Value Decomposition (SVD)* is a matrix factorization technique used to decompose the user-item matrix (X) into three other matrices:

- \mathbf{U} : Represents the users as rows and latent features as columns
- Σ : Represents the importance, or magnitude, of each latent feature across the diagonal
- \mathbf{V}^T : Represents the latent features as rows and the items as columns



Both SVD and PCA are **matrix factorization techniques**, with PCA often used for data visualization and feature extraction, and Truncated SVD for recommenders and other applications



SINGULAR VALUE DECOMPOSITION

Recommenders
Basics

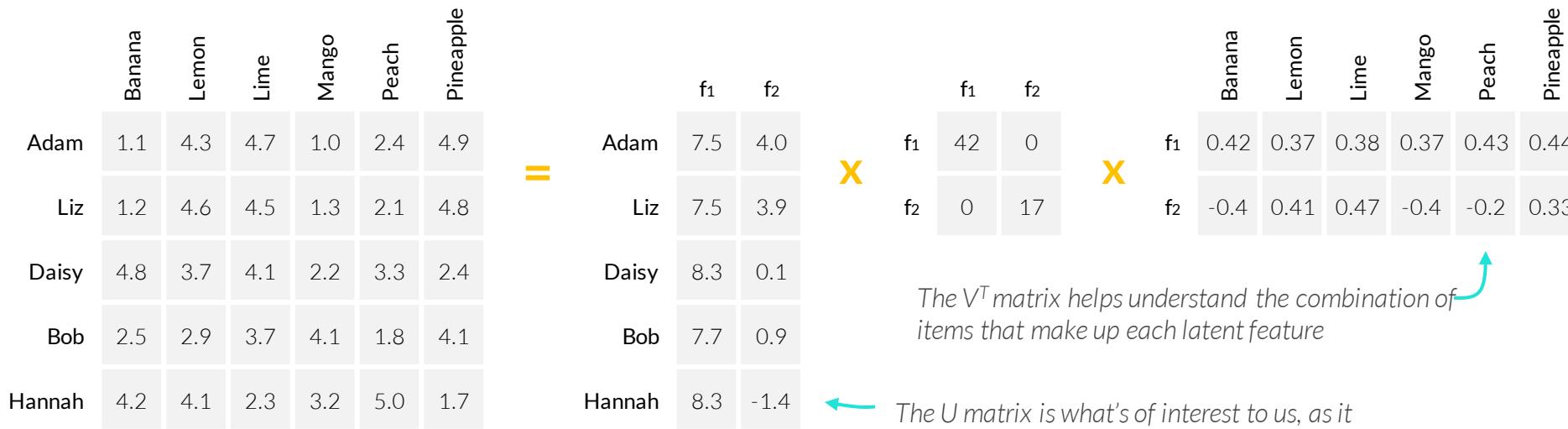
Content-Based
Filtering

Collaborative
Filtering

Next Steps

Singular Value Decomposition (SVD)* is a matrix factorization technique used to decompose the user-item matrix (X) into three other matrices:

- \mathbf{U} : Represents the users as rows and latent features as columns
- Σ : Represents the importance, or magnitude, of each latent feature across the diagonal
- \mathbf{V}^T : Represents the latent features as rows and the items as columns





SINGULAR VALUE DECOMPOSITION IN PYTHON

Recommenders
Basics

Content-Based
Filtering

Collaborative
Filtering

Next Steps

You can use sklearn's **TruncatedSVD()** function to apply the SVD algorithm

- Use **.fit_transform(X)** to return the U matrix
- Use **.singular_values_** to return the importance for each latent feature, or the sigma matrix
- Use **.components_** to return the V^T matrix

```
from sklearn.decomposition import TruncatedSVD  
  
# apply SVD to the ratings  
svd = TruncatedSVD(n_components=2)  
U = svd.fit_transform(X)  
  
pd.DataFrame(U, index=X.index).head()
```

0 1

User	0	1
user0	7.501225	4.006685
user1	7.516644	3.916673
user10	8.342821	0.115237
user11	7.735173	0.872330
user12	8.355260	-1.447248

You can choose the
number of latent
features to extract

```
sigma = svd.singular_values_  
  
sigma  
array([42.56011748, 17.72073293])
```

```
VT = svd.components_  
  
pd.DataFrame(VT, columns=X.columns)
```

Fruit	Banana	Lemon	Lime	Mango	Peach	Pineapple
0	0.420285	0.375942	0.389082	0.379614	0.437680	0.441557
1	-0.469867	0.417385	0.470985	-0.428490	-0.293848	0.336505

ASSIGNMENT: SINGULAR VALUE DECOMPOSITION

 **NEW MESSAGE**
April 3, 2024

From: Becca Mender (Data Scientist)
Subject: Let's create a recommender!

Hi again,
Thanks for pulling together the user-item matrix yesterday.
Next, let's create a recommender!
Can you apply TruncatedSVD to the user-item matrix using 2 components?
Thanks!
Becca

Reply **Forward**

Key Objectives

1. Apply TruncatedSVD to the user-item matrix from the last assignment
2. View the user-item matrix and the shape of the user-item matrix
3. View the U matrix and the shape of the U matrix
4. View the VT matrix and the shape of the VT matrix



CHOOSING THE NUMBER OF COMPONENTS

Recommenders
Basics

Content-Based
Filtering

Collaborative
Filtering

Next Steps

```
# try SVD with all 6 components
svd6 = TruncatedSVD(n_components=6)
U6 = svd6.fit_transform(X)

svd6.explained_variance_ratio_.round(2)
array([0.02, 0.83, 0.08, 0.05, 0.01, 0.01])
```

```
import numpy as np

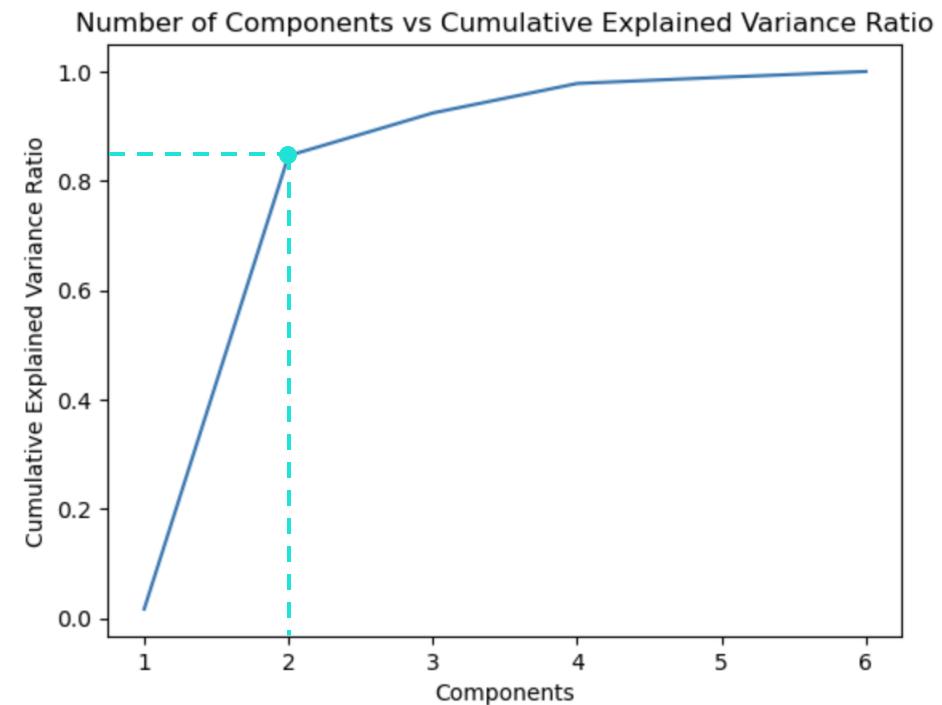
np.cumsum(svd6.explained_variance_ratio_.round(2))
array([0.02, 0.85, 0.93, 0.98, 0.99, 1.  ])
```



The first two components capture 85% of the variance in the data



PRO TIP: You typically want to capture around 80%+ of the explained variance with your model



ASSIGNMENT: CHOOSING THE NUMBER OF COMPONENTS

 **NEW MESSAGE**
April 4, 2024

From: Becca Mender (Data Scientist)
Subject: Tuning the recommender

Hi again,

Using the code from yesterday, can you try applying TruncatedSVD with 500 components?

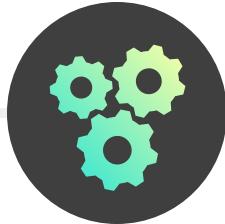
1. Fit a model with 500 components, and suggest a good number of components for this data set
2. Fit a model using your suggested number of components

Thanks!
Becca

[Reply](#) [Forward](#)

Key Objectives

1. Fit a TruncatedSVD model with 500 components
2. Plot the cumulative explained variance ratios
3. Suggest a “good” number of components that best captures the variance in the data
4. Fit another TruncatedSVD model with the “good” number of components



COLLABORATIVE FILTERING RECOMMENDATIONS

Recommenders
Basics

Content-Based
Filtering

Collaborative
Filtering

Next Steps

STEP 1: Select and fit a model using **TruncatedSVD()** and **.fit_transform(X)**

```
# apply SVD to the ratings
svd = TruncatedSVD(n_components=2)
U = svd.fit_transform(X)

# calculate the explained variance
sum(svd.explained_variance_ratio_)

0.8459496262760737
```



The model captures 85% of the variance in the data

STEP 2: Use **.transform()** to get the latent features for a new user

```
# preferences for new user
new_user_6d
```

Fruit	Banana	Lemon	Lime	Mango	Peach	Pineapple
0	NaN	2.0	NaN	5.0	NaN	NaN



All we know is the user likes mangos more than lemons

```
# transform with our SVD model
new_user_2d = svd.transform(new_user_6d.fillna(3))
```

```
array([[ 7.71576637, -1.17635595]])
```

You need to fill in the blank ratings with the mean (3)



COLLABORATIVE FILTERING RECOMMENDATIONS

Recommenders
Basics

Content-Based
Filtering

Collaborative
Filtering

Next Steps

STEP 3: Use `np.dot()` and `.components_` to get the model's recommendations

```
# make recommendations by reconstructing 1x6 matrix
new_user_pred = np.dot(new_user_2d, svd.components_)
recs = pd.DataFrame(new_user_pred, columns=X.columns)
recs
```

Fruit	Banana	Lemon	Lime	Mango	Peach	Pineapple
0	3.795554	2.409686	2.448016	3.43307	3.722708	3.011103

```
# sorted recommendations
recs.T.sort_values(0, ascending=False)
```

Fruit	0
Banana	3.795554
Peach	3.722708
Mango	3.433070
Pineapple	3.011103
Lime	2.448016
Lemon	2.409686



If they like mangos and don't like lemons, we should suggest bananas and peaches



How does this work?

- By multiplying the new user's U matrix with the model's V^T matrix (`svd.components_`), we get an approximation of their ratings

ASSIGNMENT: COLLABORATIVE FILTERING

 **NEW MESSAGE**
April 5, 2024

From: Becca Mender (Data Scientist)
Subject: Recommend movies

Happy Friday!

I have one final request for you. Let's test out our SVD model on a new user.

In the attached notebook, I've included the movie ratings that a new user provided.

Can you generate 10 movie recommendations for her using our final SVD model (with 250 components) and see if they make sense?

Thanks for all your help this week!

 section08_recommenders_assignments.ipynb  

Key Objectives

1. Open the notebook and take a look at the new user DataFrame
2. Transform the user into the latent space using `.transform()`
3. Reconstruct the user-item matrix for the user using `np.dot()`
4. Make 10 movie recommendations for the user
5. Review them to determine if they make sense
6. *Optional:* Try playing around with the number of components to see the difference in movie recommendations



RECOMMENDERS NEXT STEPS

Recommenders
Basics

Content-Based
Filtering

Collaborative
Filtering

Next Steps

Once you've fit an initial content-based filtering or collaborative filtering model, some potential **next steps** are to:

1

Consider a hybrid approach

- In practice, you often combine approaches (*content-based, collaborative, popularity, rule-based, etc.*) to create a hybrid model and make the best recommender

2

Evaluate the quality of recommendations

- A common approach is to deploy your model, see how users react to the recommendations, and then make tweaks to create a better recommender

3

Deal with the difficulties of recommenders

- Tuning recommenders is an on-going process, and includes difficulties like changes to items, data sparsity, popularity bias, and the “cold start” problem

4

Be creative in your approach

- All the concepts covered in this course can be mixed and matched when it comes to recommenders (*clustering, anomaly detection, dimensionality reduction, etc.*)

KEY TAKEAWAYS



You can use **content-based** and **collaborative filtering** for recommenders

- Content-based filtering recommenders are based on the characteristics of the items, while collaborative filtering recommenders are based on user-item interactions and preferences



Cosine similarity is often used to find the distance between items

- The range of cosine similarities falls between -1 and +1, with positive numbers being more similar



SVD is a matrix factorization technique commonly used for recommenders

- Modeling & interpreting the results of Singular Value Decomposition (SVD) in Python is very similar to PCA



Tuning recommenders is a long and difficult process

- In practice, hybrid recommenders that combine multiple recommender techniques are often used
- Models can be evaluated further down the line to note user interactions with their recommendations

RECOMMENDER PROJECT

PROJECT: RECOMMENDING RESTAURANTS



THE SITUATION

You've just been hired as a Data Scientist for **MavenEats**, a restaurant review website that suggests restaurants to users based on their past restaurant ratings



THE ASSIGNMENT

You have access to data on each **user's restaurant ratings** on the website
Your task is to **create a recommender** that (1) displays five suggested restaurants on a user's homepage and (2) five similar restaurants on each restaurant's details page



THE OBJECTIVES

1. **Prep** the data for modeling
2. **Create a recommender** using TruncatedSVD
3. **Recommend** five restaurants based on a user, and five restaurants based on a restaurant



UNSUPERVISED LEARNING REVIEW

UNSUPERVISED LEARNING REVIEW



In this section we'll review the foundations of **unsupervised learning** covered in the course, including techniques, applications, and its place in the data science workflow

TOPICS WE'LL COVER:

Unsupervised Learning

Techniques & Applications

Data Science Workflow

GOALS FOR THIS SECTION:

- Review the basics of unsupervised learning
- Define a flowchart for selecting the correct unsupervised learning technique
- Revisit the different applications for each unsupervised learning technique
- Identify where each application of unsupervised learning fits within the data science workflow



UNSUPERVISED LEARNING FLOWCHART

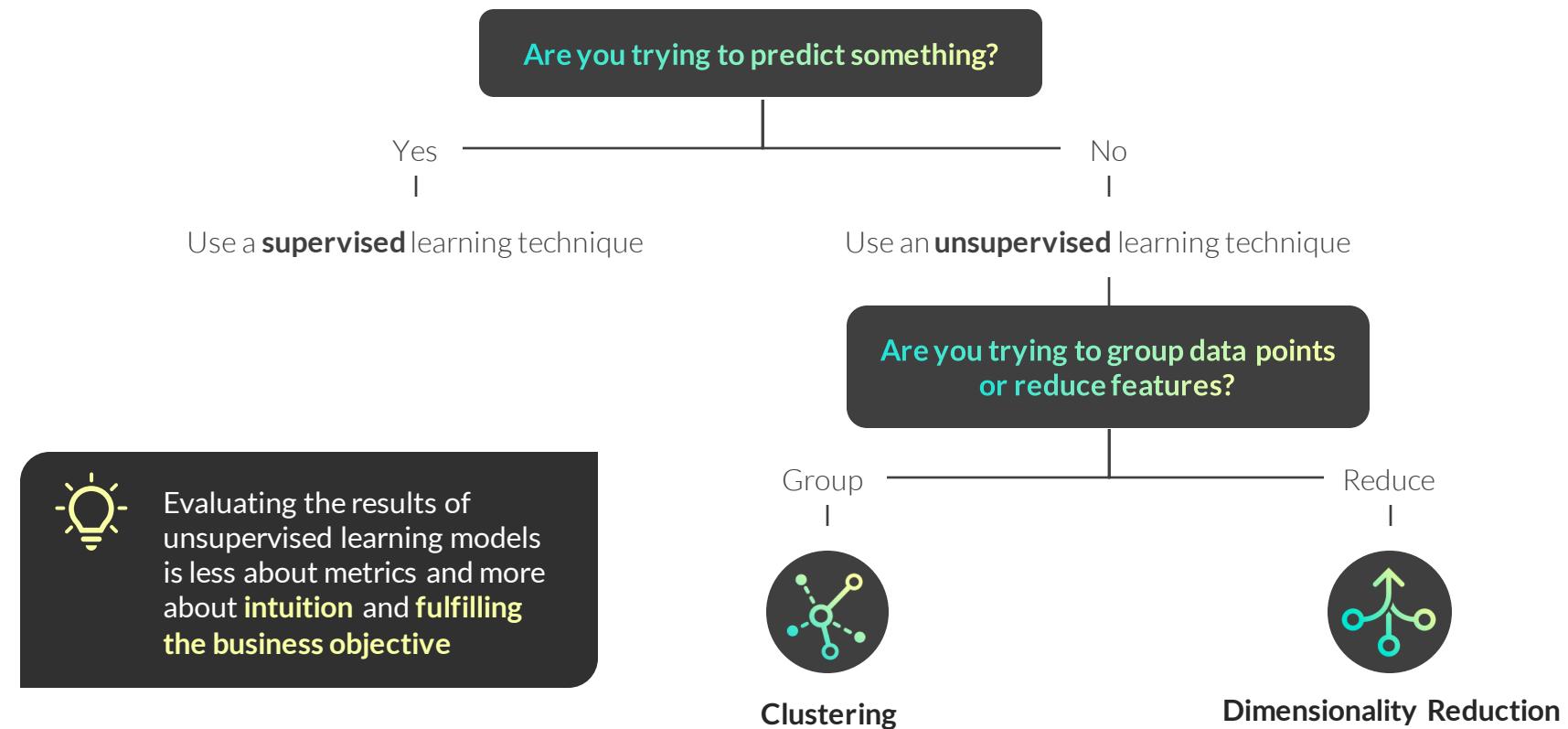
Unsupervised Learning

Techniques & Applications

Data Science Workflow

Unsupervised learning is about **finding insights & patterns** hidden in the data

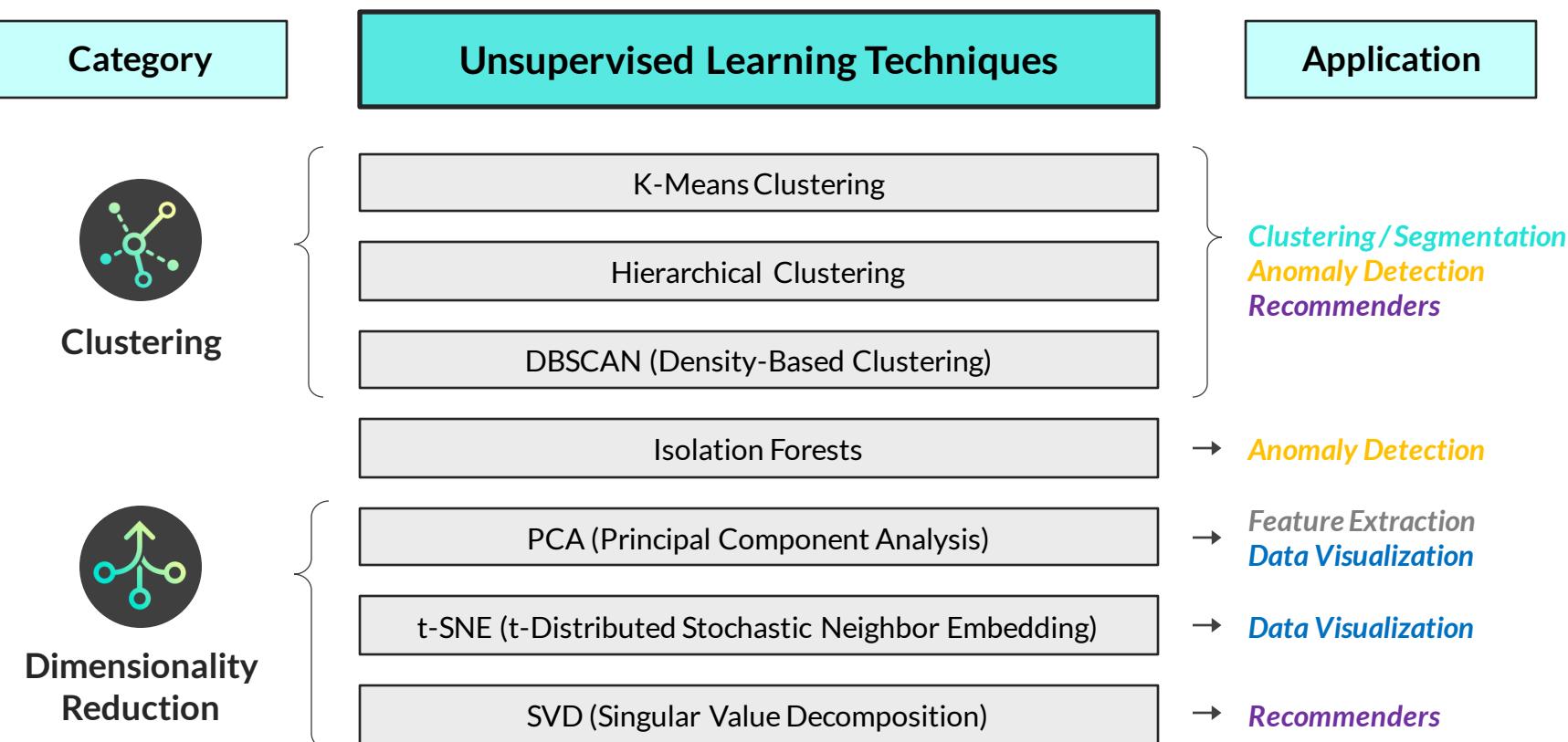
- We care about understanding the relationships in *unlabeled* data, not making predictions





UNSUPERVISED TECHNIQUES & APPLICATIONS

Unsupervised learning techniques can be used for **multiple applications**:





UNSUPERVISED LEARNING IN THE DS WORKFLOW

Unsupervised learning can be applied at **multiple phases of the workflow:**



KEY TAKEAWAYS



Unsupervised learning is used to **find patterns & relationships** in data

- *There are no predictions or labels with unsupervised learning – we are just trying to better understand the data's non-obvious structure, organization, and relationships between data points*



Unsupervised learning model evaluation focuses on **intuition**

- *While there are evaluation metrics available, the focus for unsupervised learning is on using intuition and domain expertise to make recommendations for specific business objectives*



There are **multiple applications** for unsupervised learning **techniques**

- *While the two main categories of unsupervised learning techniques fall under clustering and dimensionality reduction, these techniques can be applied to segmentation, anomaly detection, recommenders, and more*



The techniques can be used at **multiple steps** of the data science workflow

- *In addition to using unsupervised learning techniques during the modeling step of the data science workflow, select techniques can also be used during the data cleaning, exploration, and feature engineering phases*

FINAL PROJECT

RECAP: THE COURSE PROJECT



THE **SITUATION**

You've just been hired as an Associate Data Scientist for the **HR Analytics** team at a medium-sized software company that's trying to increase employee retention



THE **ASSIGNMENT**

You have access to the company's employee database, including demographic info, performance history, tenure at the company, attrition, and more

Your task is to use **unsupervised learning techniques** to define employee segments and make recommendations to increase retention within each one



THE **OBJECTIVES**

1. **Prepare** the data for unsupervised modeling
2. **Segment** the employees using clustering
3. **Visualize** the clusters using dimensionality reduction
4. **Explore** the employees within each cluster
5. **Recommend** next steps to increase retention

