

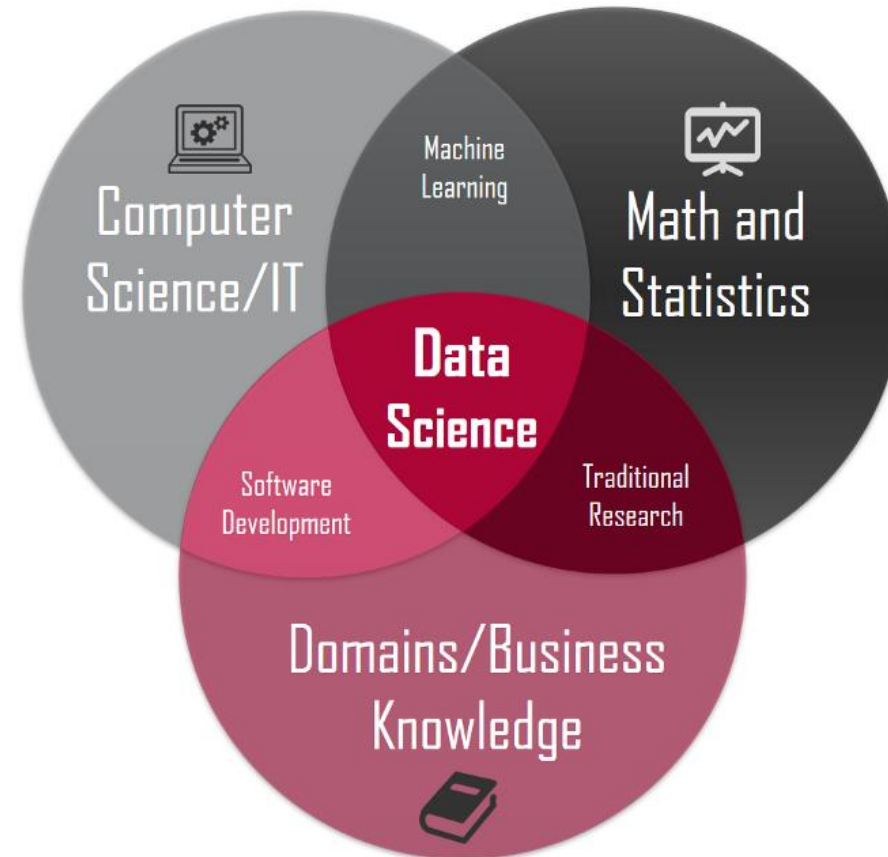
MACHINE LEARNING

A hands-on Introduction

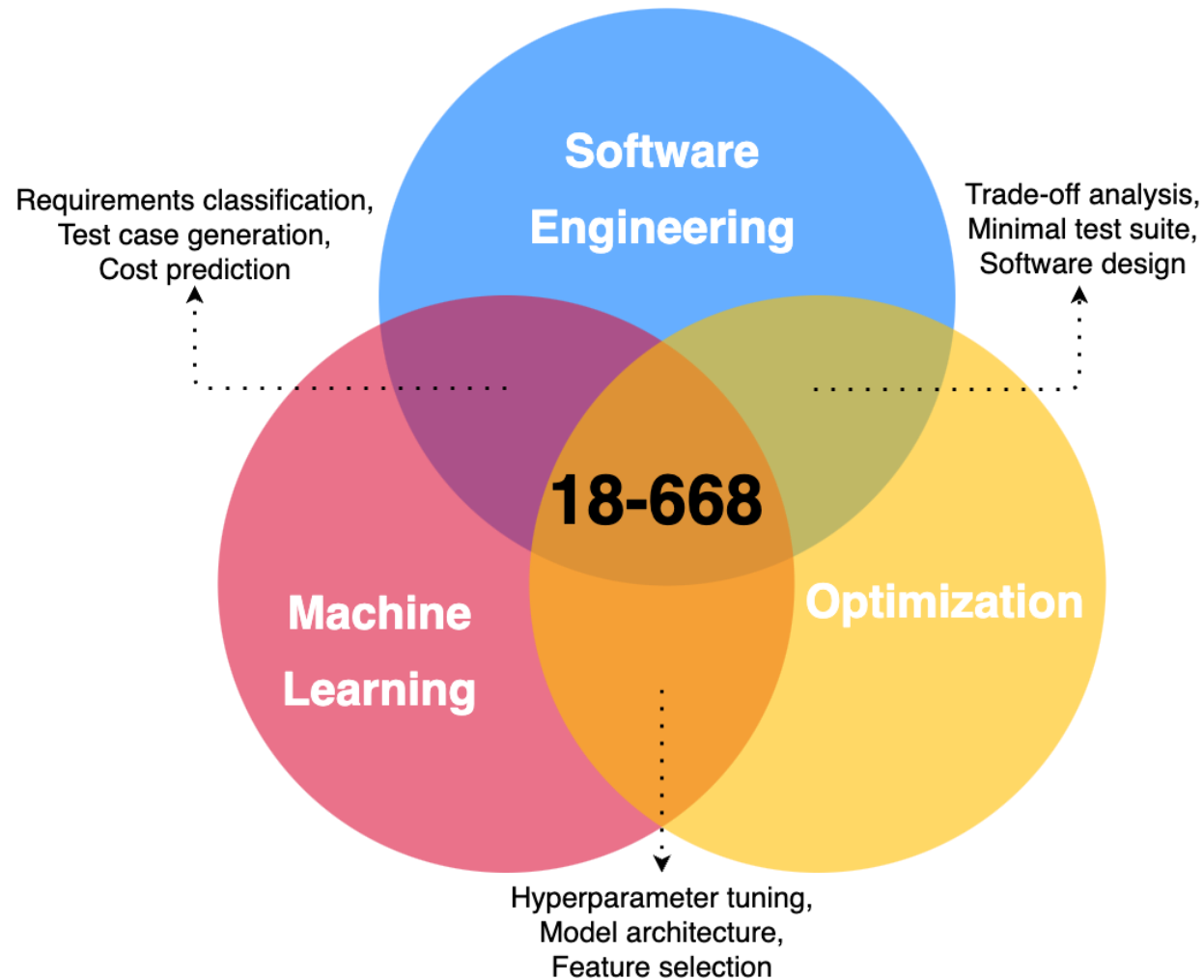


Data Science

- The science in which we gather, organize and analyze large set of structure and unstructured [data](#)

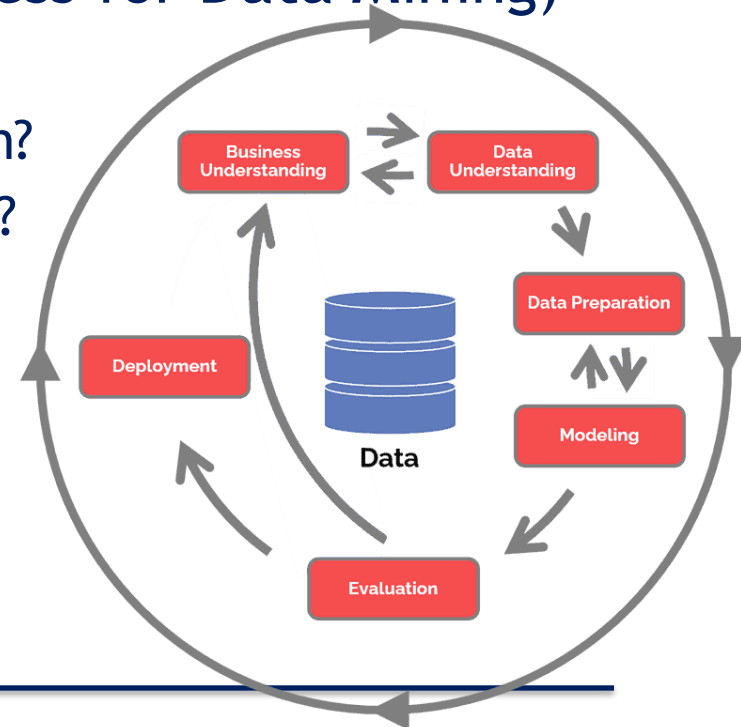


Data Science for Software Engineering

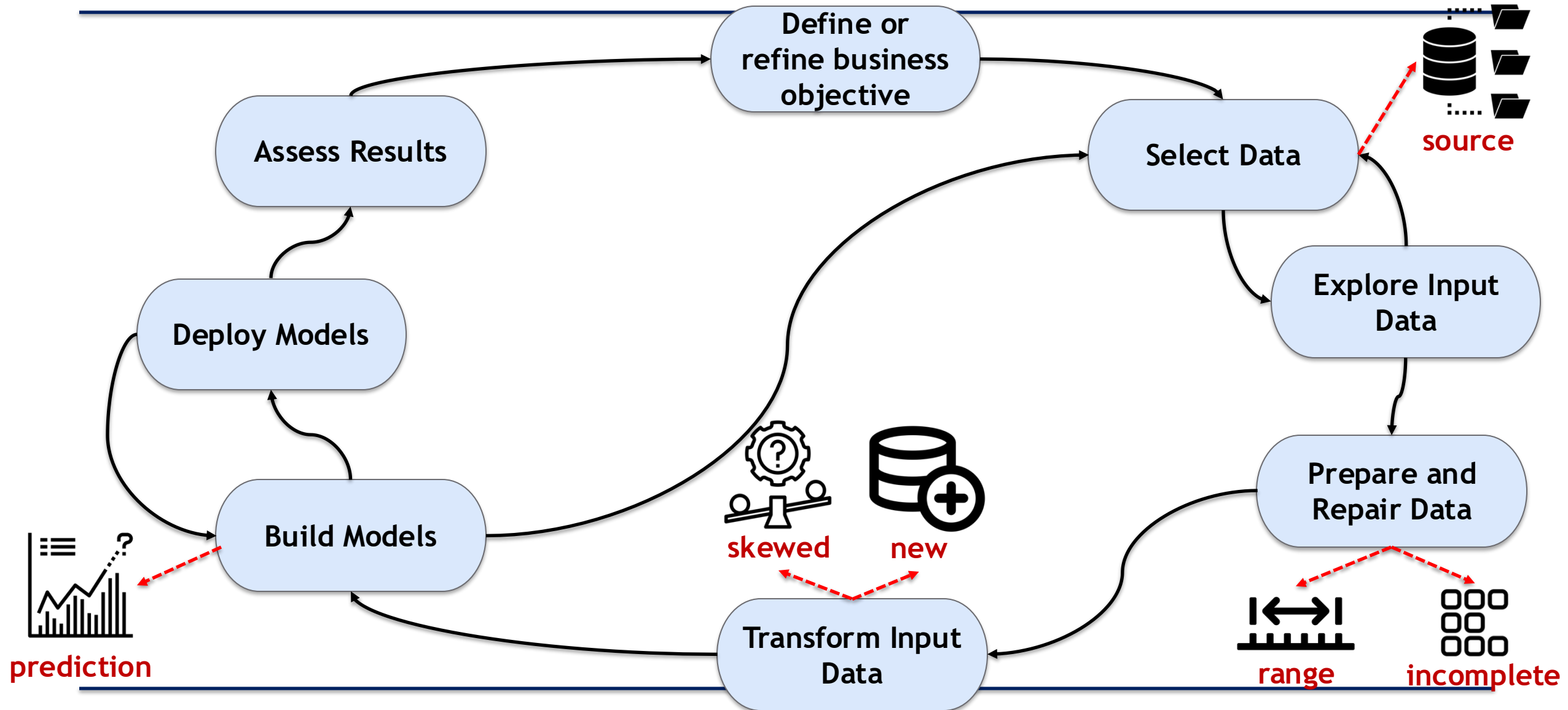


Data Science Methodologies

- They indicate the routine for finding solutions to a specific problem
 - Teams often struggle to apply an appropriate data science methodology and team-based collaboration framework
- Data Mining → CRISP-DM (**C**ross Industry Standard Process for **D**ata **M**ining)
 1. Business understanding - What does the business need?
 2. Data understanding - What data do we have / need? Is it clean?
 3. Data preparation - How do we organize the data for modeling?
 4. Modeling - What modeling techniques should we apply?
 5. Evaluation - Which model best meets the business objectives?
 6. Deployment - How do stakeholders access the results?

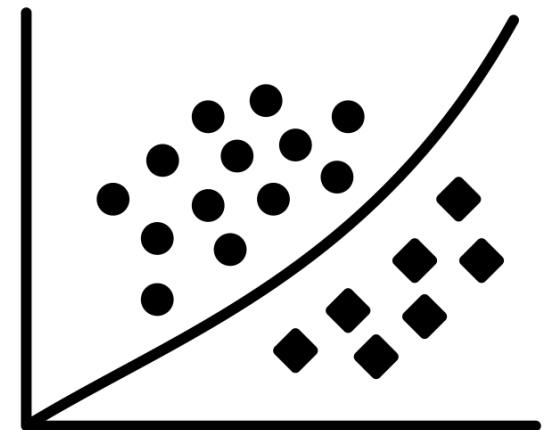


Data Science Project - Process Overview



Machine Learning

- The study of approaches to computation that improve with use
- Re-formulate software engineering problems as learning problems
 - *Classification*, *regression* and *clustering*
 - Application on all types of problems
- E.g.: Classification of software requirements





DATA MINING

Ai
ARTIFICIAL
INTELLIGENCE

PROBLEM
SOLVING

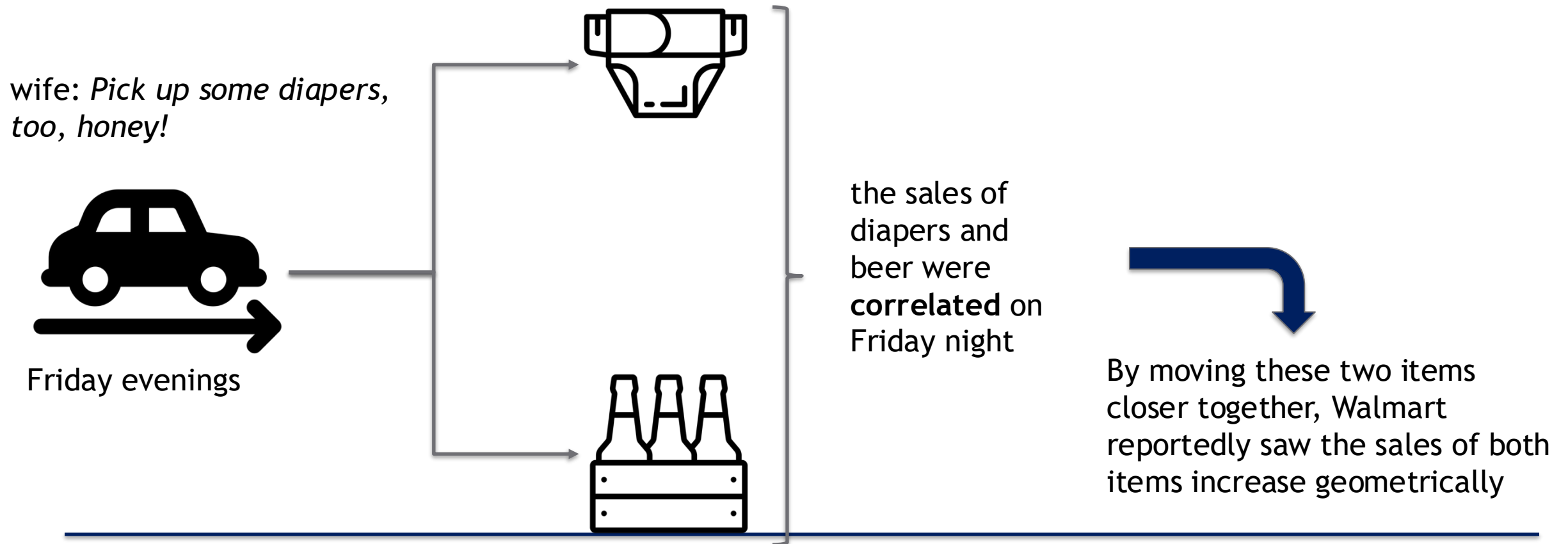
AUTOMATION

MACHINE
LEARNING

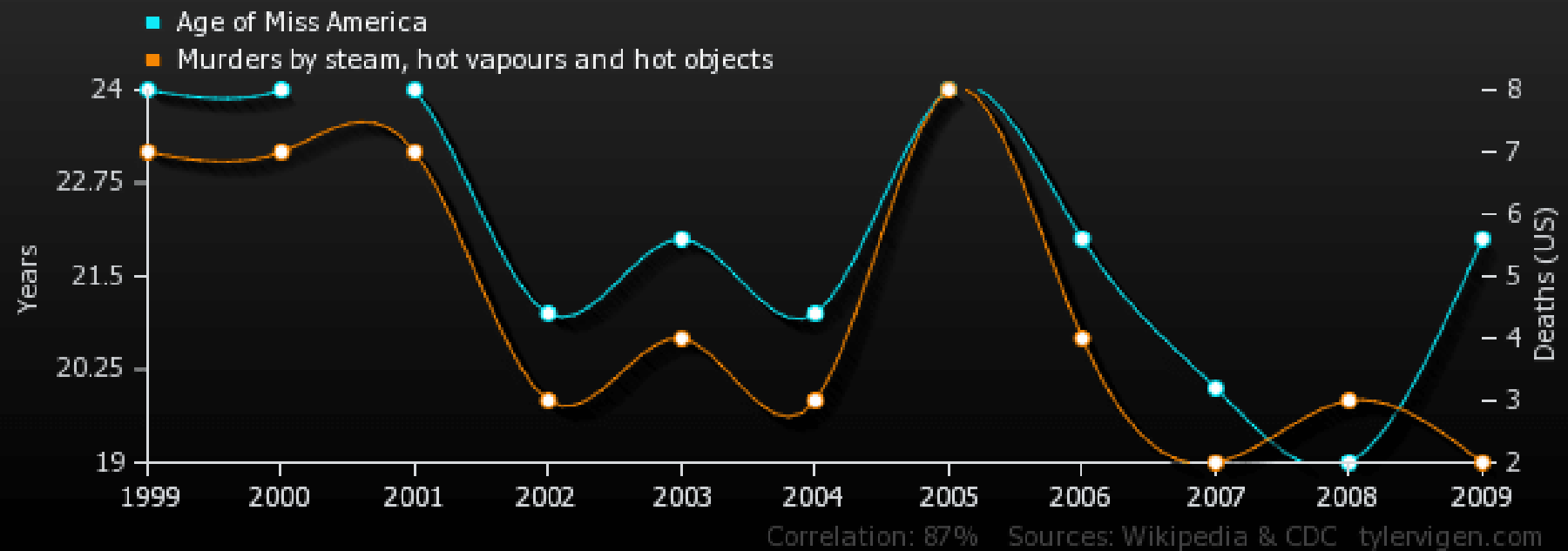
Data Mining

Beer and Diapers

- Walmart *supposedly* found out that there are certain times at which beer and diapers sell particularly well together

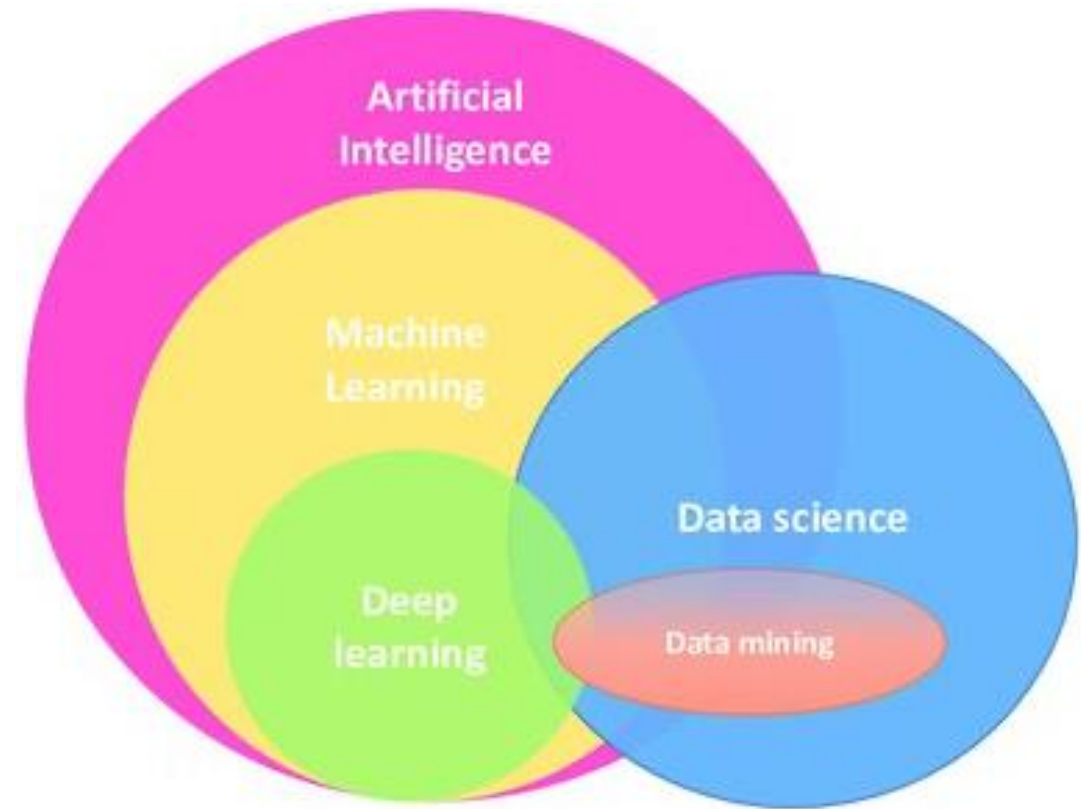


Correlation isn't Causation!

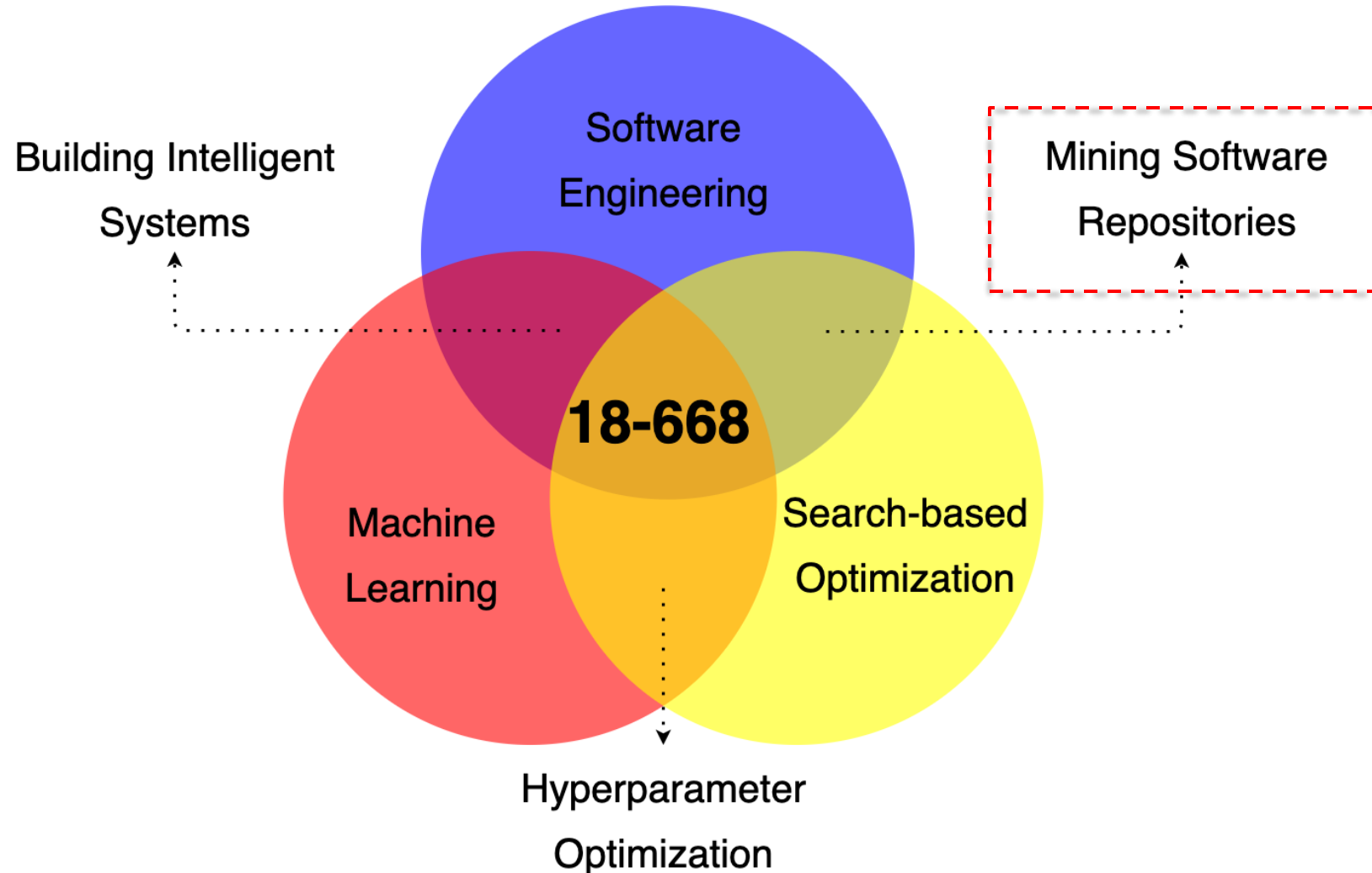


Data Science and Data Mining

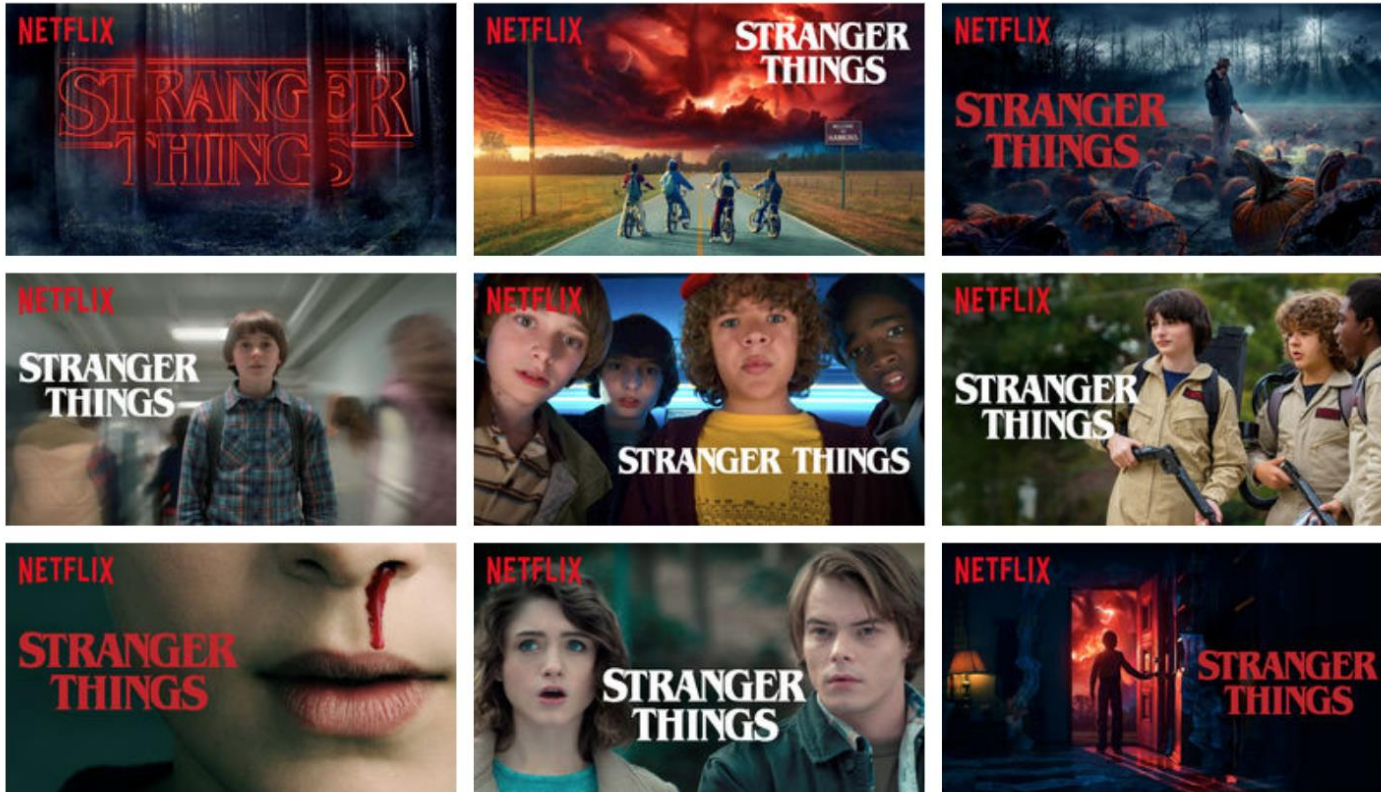
- Data science is the science in which we gather, organize and analyze large set of structure and unstructured data
- Data mining is about processing data and identifying patterns and trends in that information so that you can decide or judge



Data Science for Software Engineering



Netflix: Thumbnail Selection (Collecting Data)



Machine learning to select a thumbnail best suited for each user

Aesthetic Visual Analysis (AVA)

1. *Frame annotation*

- Brightness, contrast, skin tone, probability of nudity...

2. *Frame grading*

- rule of thirds, symmetry, depth of field, shot angle...

3. *Image Raking*

- Bright colors that grab attention
- Expressive faces of the people in the video
- Main characters from the series or the movie

WHICH DATA SHOULD WE COLLECT?



MOTIVATING QUESTION



YOU HAVE BEEN HIRED AS A DATA
SCIENTIST



TRAIN A MODEL TO PREDICT THE
RESOLUTION TIME OF BUGS

How would you tackle this task?

Motivation

- Prior experiences and dominant patterns are the driving force for many decision-processes in modern software organizations
- Example: Developers commonly use their experience when adding a new feature or *fixing a bug*

Bug Priority*	Production	Test Environment
P1	ASAP. Drop whatever you do.	24 Hours
P2	24 Hours	48 Hours
P3	n/a*	72 hours
P4	n/a**	Fix if possible

Motivation (cont'd)

- Managers allocate development and testing resources based on their experience in previous projects and their intuition about the complexity of the new project relative to prior projects
- Developers often rely on their experience, intuition and gut feeling in making important decisions
- Testers usually prioritize the testing of features that are known to be error prone based on field and bug reports

What is the issue in relying on our own experience?

Data Mining

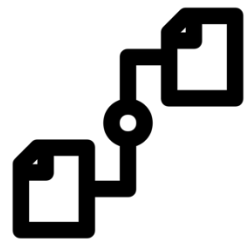
- Find hidden relationships and patterns in data that human analysts and other analysis techniques are likely to miss



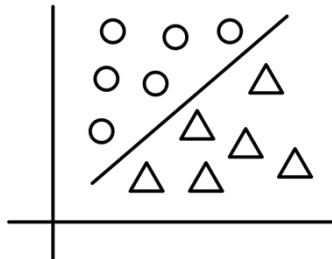
- Overview:
 - Start with historical data → collecting data
 - Analyze the historical data → looking for patterns and relationships in the data
 - Write rules → express the patterns and relationships as rules
 - Apply the rules → data mining model is applied to a new database

Data Mining Process

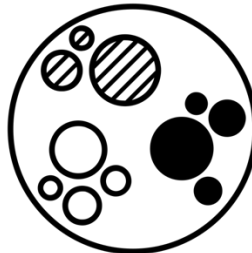
- The process of data analysis, discovery, and model-building is often iterative as you target and identify the different information that you can extract
 - Setting goal
 - **Gather the data**
 - Cleanse the data
 - **Build a model**
 - Evaluate the results



association



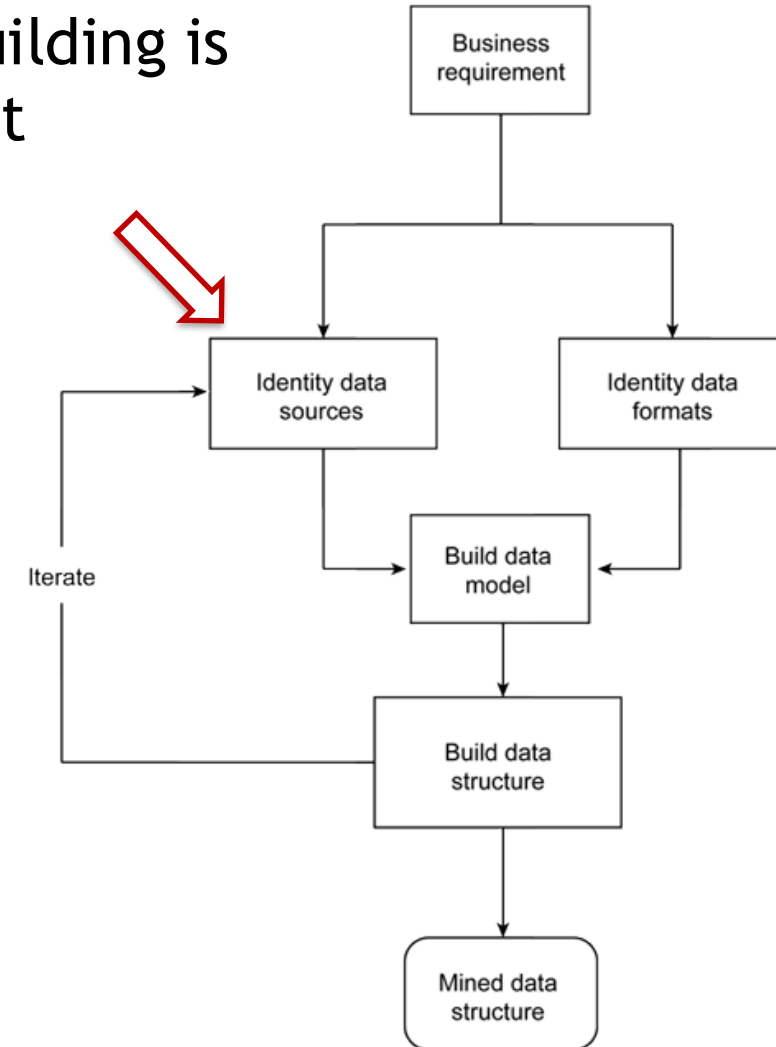
classification



clustering



prediction



Mining Software Repositories



MSR is the field that analyzes and cross-link the data available in Software Repositories



Uncover interesting and actionable information about software systems and projects



Using the information stored in these repositories, practitioners can depend less on their intuition and experience, and depend more on historical and field data

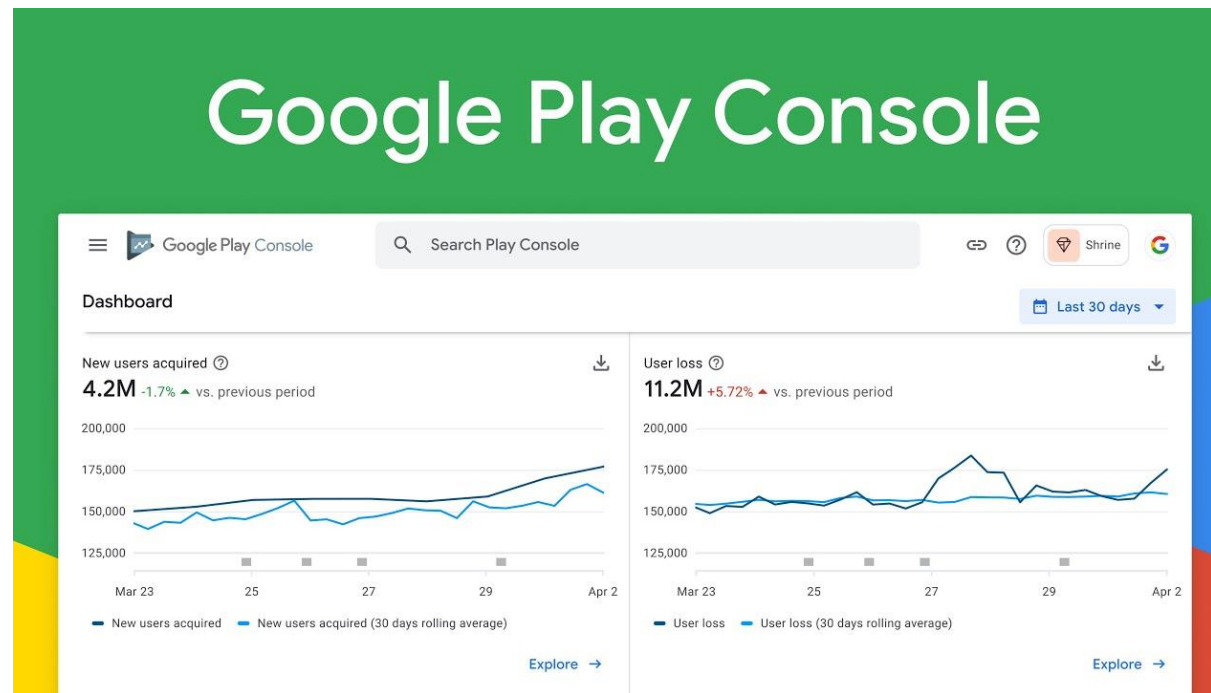
Examples of Software Repositories

- **Historical repositories** such as source control repositories, bug repositories, and archived communications record several information about the evolution and progress of a project



Examples of Software Repositories (cont'd)

- **Run-time repositories** such as deployment logs contain information about the execution and the usage of an application at a single or multiple deployment sites



Examples of Software Repositories (cont'd)

- **Code repositories** such as Bitbucket and Google Cloud code contain the source code of various applications developed by several developers



GitHub

WHAT ARE THE BENEFITS OF MINING SOFTWARE REPOSITORIES?



Software Repositories

- They are commonly used in practice as record-keeping repositories
- They are rarely used to support decision making processes
- Large amount of raw data
 - **how to explore them?**

MSR: Benefits

- **Understand Software Systems**
 - Hipikat¹: tool that indexes historical repositories, and displays on-demand relevant historical information within the development environment
- **Propagate Code Changes**
 - Propagating code changes to other entities of a software system
 - Goal: ensure the consistency of assumptions in the system after changing an entity

1. Cubranic et al. Hipikat: A Project Memory for Software Development. 2015

MSR: Benefits

- **Understand Team Dynamics**
 - Team discussions cover many important topics such as...
 - future plans, design decisions, project policies, and code or patch reviews
- **Improve the User Experience**
 - Stabilizer tool¹: mines reported bugs and execution logs to prevent an application from crashing
 - Goal: prevent users to perform an action that leads to a bug

1. A. Michail and T. Xie. Helping Users Avoid Bugs in GUI Applications. 2005

Example 1

- Historical repositories are used to track the history of a bug or a feature

Bug 39884 - cee27995-8985-4cfe-aa54-a9813cec9e0f ([edit](#)) Save Changes

Status: CONFIRMED ([edit](#))

Alias: None
Add
Bug

Product: MyOwnBadSelf ▾
Component: comp2 ▾ ([show other bugs](#))
Version: unspecified
Hardware: All ▾ All ▾

Importance: P2 ▾ normal ▾
Target Milestone: ---
Assignee: [tara@tequilarista.org](#) ([edit](#)) ([take](#))
QA Contact: ([edit](#)) ([take](#))


URL:
Whiteboard:
Keywords:
Personal Tags:

Depends on: [38933](#) ([edit](#))
Blocks:

[Show dependency tree / graph](#)

Reported: 2017-02-19 12:40 PST by [Ashish](#)
Modified: 2017-02-19 12:40 PST ([History](#))
CC List: ☐ Add me to CC list
2 users ([edit](#))
Ignore Bug Mail: ☐ (never email me about this bug)

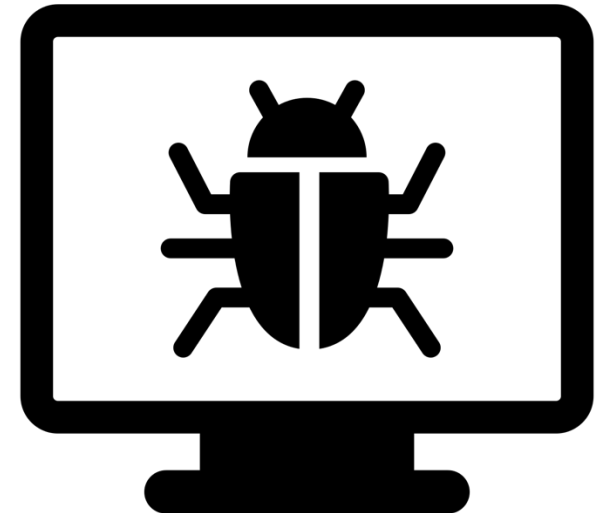
See Also: ([add](#))
Large text box: ([edit](#))
A multiple-select box: Always Appears
Also Always Appears
Third Value, Always

Drop Down List: --- ▾
Date Time: 
Bug ID Field:

Flags: None yet set ([set flags](#))

Example 1 (cont'd)

- Repository stores resolution time of previously closed bugs
- They are not used to determine the expected resolution time of an open bug
 - How can we predict the resolution time?

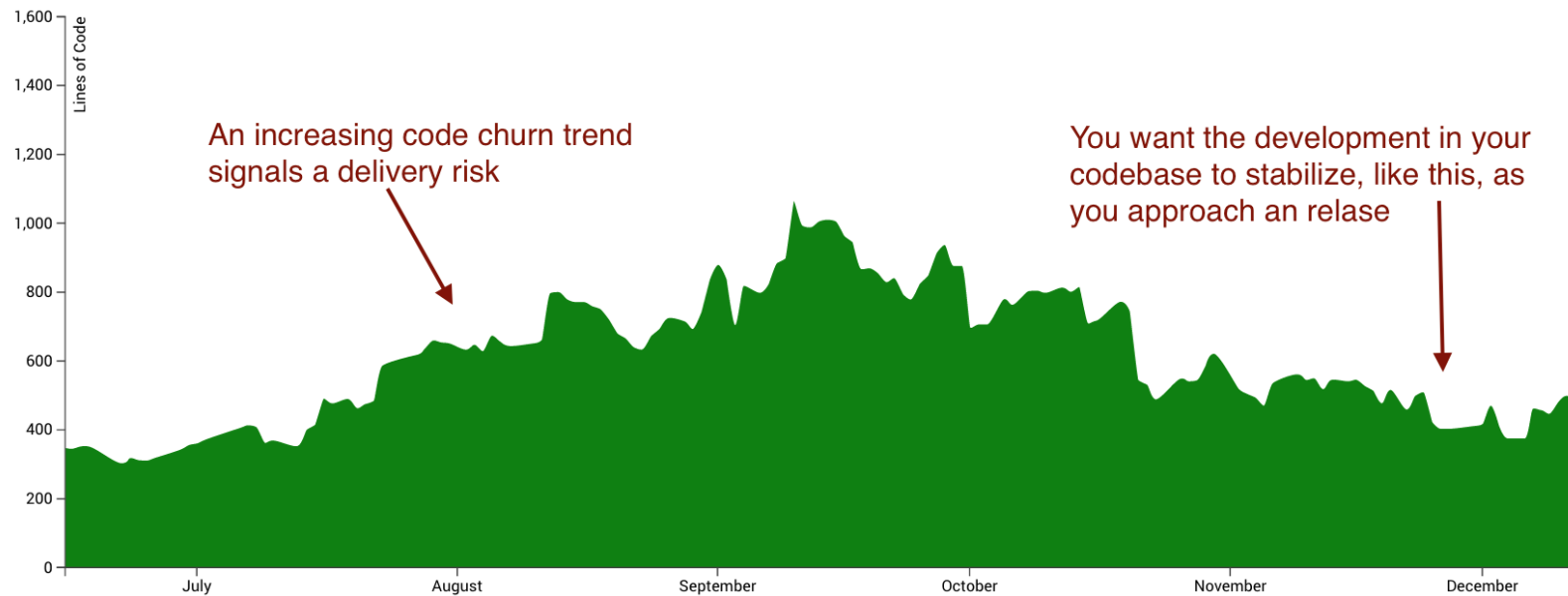


Example 2: Predicting and Identifying Bugs

- Predicting the occurrence of bugs remains one of the most active areas in SE
- Beneficiaries
 - Managers can allocate testing resource appropriately
 - Developers can review risky code more closely
 - Testers can prioritize their testing efforts
- Data: **code change metrics** mined from source control repositories

Code Change Metrics

- Metrics that capture details of changes made to the code base over time
 - N° of commits, code churn (added, deleted lines), authors, **commit entropy**...



- Goal: predict program files that are going to be bug prone due to various changes that happened in software over period of time

Predicting Buggy Files¹

- Java Development Tools repository of Eclipse
 - 12 code change metrics: commits, add, deleted, commits60 and entropy...
 - Bug report: bug fixes

Mining Github for Novel Change Metrics to Predict Buggy Files in Software Systems

- Models:
 - Gaussian NB,
 - CART Decision Tree,
 - Logistic Regression and
 - NB Tree

K Muthukumaran, Abhinav Choudhary, N L Bhanu Murthy
Department of Computer Science and Information Systems
BITS Pilani Hyderabad Campus
{p2011415, f2009155, bhanu}@hyderabad.bits-pilani.ac.in

Abstract—Code change metrics mined from source control repositories have proven to be the most reliable predictors of bugs in contemporary software engineering research. Yet a definitive modus operandi for obtaining the required data from a particular software configuration management (SCM) repository needs to be put forward. In this paper, we define a modus operandi to extract some popular change metrics from the Eclipse repository on Github, which can be generalized for any open-source Github repository. We define few code change metrics that are intuitively significant for predicting bugs. Bug prediction models built with these metrics along with the existing prominent code change metrics prove to be competent and consistent as per our experiments on five different versions of Eclipse JDT project. We explored Naive Bayes Tree algorithm to build a prediction model and have found it to perform better than other commonly used algorithms in this problem domain.

faults [3]. Nagappan et al. found that change burst metrics yield excellent predictive capability in projects with high-quality changes. They claim that precision and recall exceed 90% for Windows Vista: the highest predictive power ever observed [4]. Nagappan and Ball took code churn measures to build defect prediction model and found out that relative code churn measures are excellent predictors of defect density in large industrial software systems [5]. Moser et al. shows that change metrics clearly outperform predictors based on static code attributes for the Eclipse project [6] [7]. They confirm the observations made by other researchers, as change data, and more in general process related metrics, contain more discriminatory and meaningful information about the defect distribution in software than the source code itself. They suggest that while most of the past research effort has been

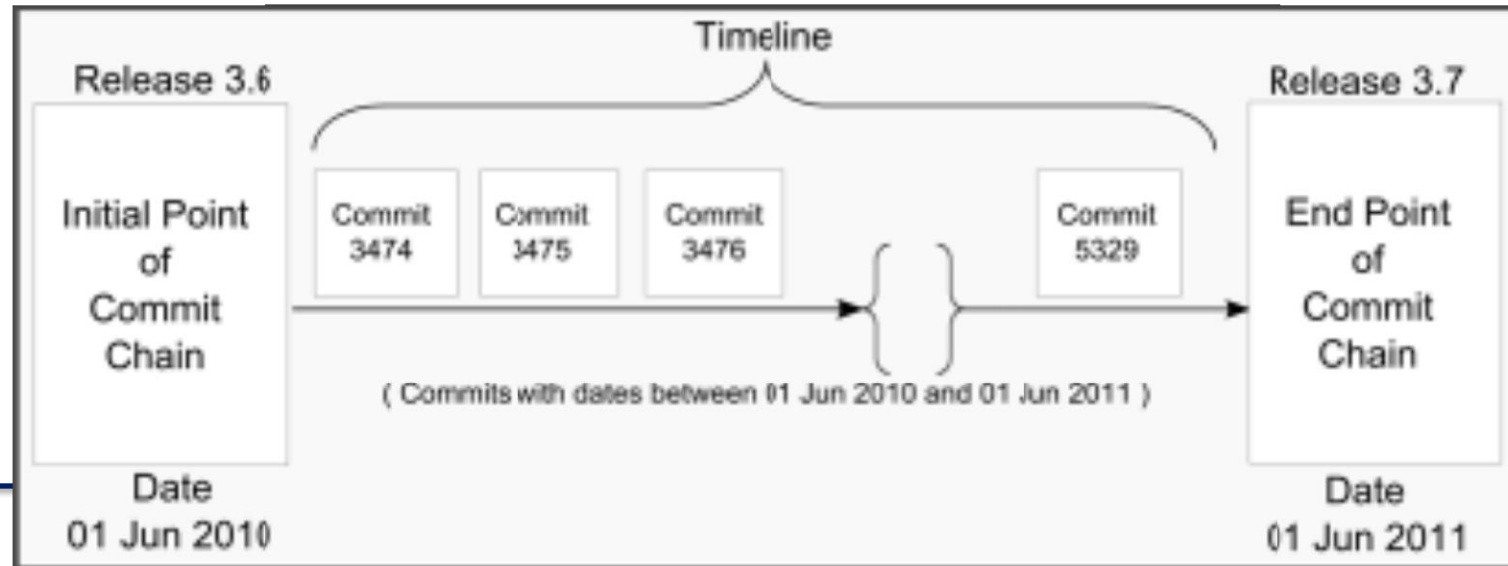
It looks easy, right?

1. Muthukumaran et al. "Mining GitHub for Novel Change Metrics to Predict Buggy Files in Software Systems" 2015

All major releases of Eclipse from 3.0 to 3.5



- For every bug fix commit within 1-year post-release:
 - i. The commit message is analyzed to strip out the bug ID number for the bug that was fixed in the commit
 - ii. The bug repository for the project is then consulted and the document for the particular bug ID is parsed to extract the report date
 - iii. The report date is checked to see if it lies between the release date of the later release and within 6 months from that
 - iv. If it does, the bug report counts for each file modified by the commit is incremented by 1.



Predicting Buggy Files: Results

- NB Tree algorithm is better than all other three algorithms individually for each release individually

	Algorithms											
	Gaussian Naïve Bayes			CART Decision Tree			Logistic Regression			Naïve Bayes Tree		
	P	R	F	P	R	F	P	R	F	P	R	F
Eclipse 3.0 - 3.1	73.31	68.448	66.074	71.019	71.034	71.023	73.3	73.3	73.2	76.4	76.4	76.3
Eclipse 3.1 - 3.2	77.445	78.168	76.788	70.621	71.016	70.8	76.3	77.2	75.7	75.4	76.4	75
Eclipse 3.2 - 3.3	72.726	76.149	73.387	70.105	70.678	70.38	73.2	77.2	71.9	74.6	77.9	73.6
Eclipse 3.3 - 3.4	73.532	74.1	71.001	70.466	70.863	70.641	72.5	73.7	71.8	75.3	76.1	75.3
Eclipse 3.4 - 3.5	72.132	74.177	72.448	70.461	69.873	70.146	71.2	73.9	70.8	73.4	75.4	72.9
Average	73.829	74.2084	71.94	70.5344	70.6928	70.598	73.3	75.06	72.68	75	76.4	74.62
S D	2.0934	3.62765	3.9098	0.33045	0.48014	0.3446	1.875	1.9655	1.8913	1.11	0.91	1.363

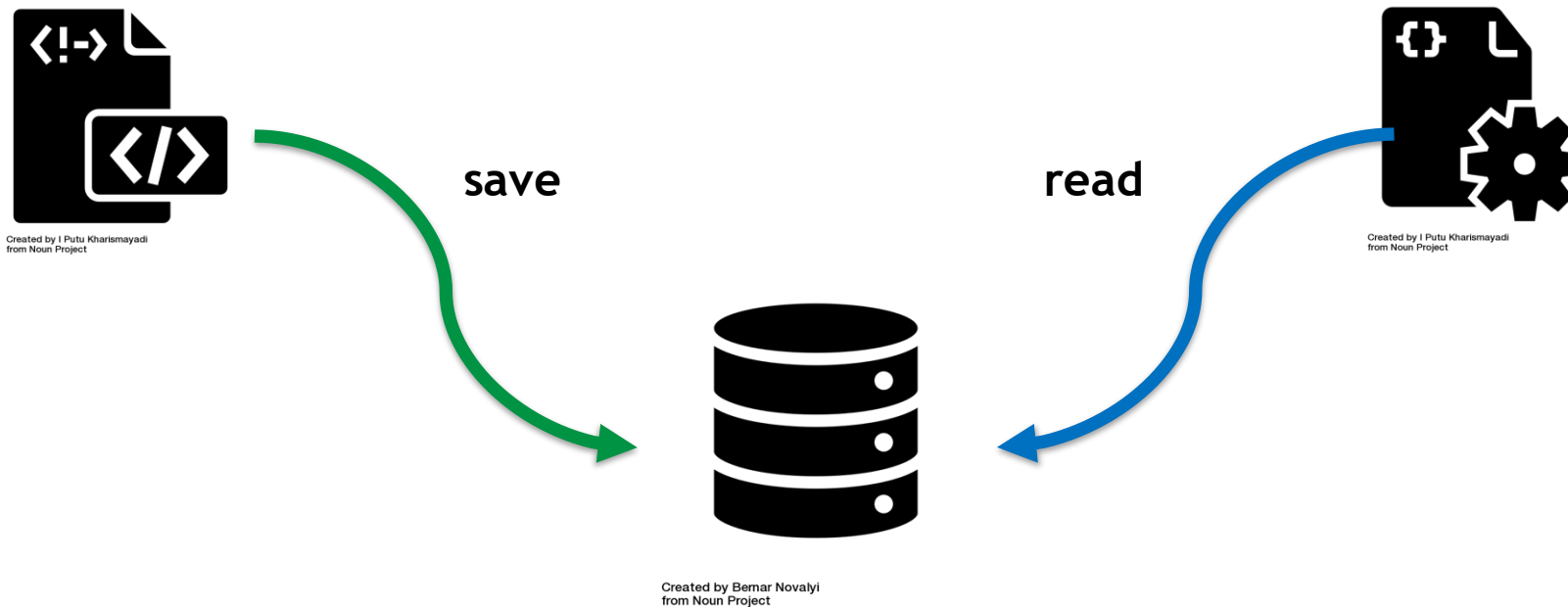
- Results are very consistent across the releases

Example 3

- Historical repositories capture important historical dependencies
 - E.g.: dependency between project artifacts, such as functions, documentation files, or configuration file
- Developers can use this information to propagate changes to related artifacts...
 - instead of only using static or dynamic code dependencies
 - which may fail to capture important dependencies

Example 3 (cont'd)

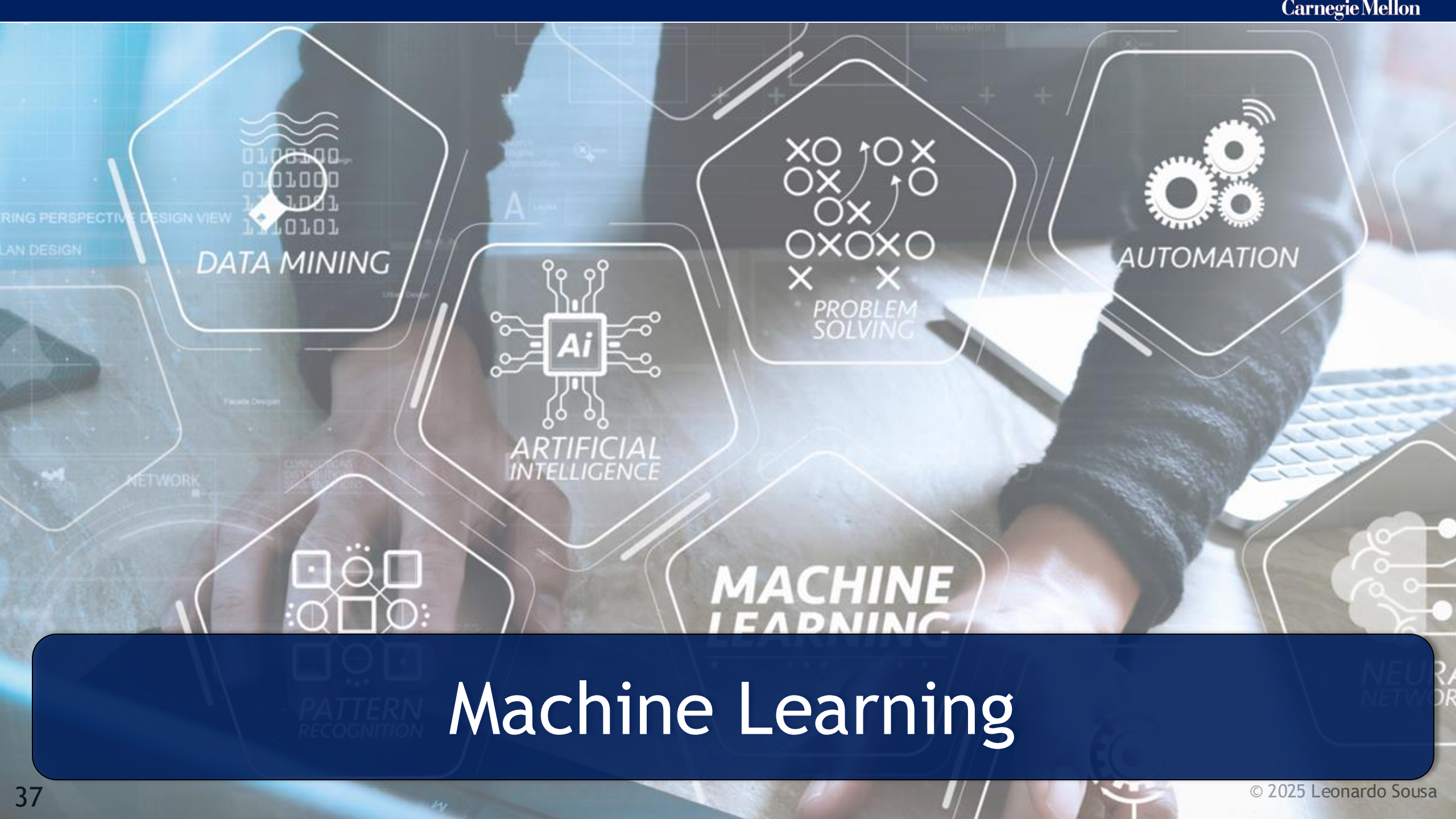
- A change to the code which writes data to a file may require changes to the code which reads data from the file



- There exist no traditional dependencies (e.g., data and control flow) between both pieces of code

Take Away Points

- The science in which we gather, organize and analyze large set of structure and unstructured **data**
- A data scientist must analyze, process and model **data**
- They must interpret the **data**
 - find patterns, trends
 - create actionable plans, etc.



Machine Learning

Machine Learning

- It is a subset of artificial intelligence (AI) that focuses on
 - developing algorithms and statistical models that
 - enable computers to learn from and make predictions or decisions based on data
- It involves
 - training models using data,
 - identifying patterns, and
 - improving decision-making or predictions
 - without being explicitly programmed to perform specific tasks

Machine Learning

- Machine learning gives computers the ability to make decisions by inferring data
 - Format the data in appropriate ways
 - Give it as input to the ML algorithm that will perform the learning process
- So, we can **solve some complex problems** without having to code them step by step if we have enough data
 - Classify emails as spam
 - Categorize requirements as *functional* or *non-functional requirements*
- Types of Learning
 - *Supervised Learning*
 - *Unsupervised Learning*
 - *Reinforcement Learning*

Supervised Learning

1 Labeled Data

Supervised learning algorithms are trained on labeled data, meaning each data point has a known outcome.

- the data has input-output pairs

2 Predictive Models

The goal is to create a model that can predict the outcome for new, unseen data

3 Regression & Classification

Examples include predicting house prices (regression) or classifying emails as spam or not (classification)

4 Examples

Image recognition, spam detection, fraud detection, and medical diagnosis



Unsupervised Learning

Unlabeled Data

Unsupervised learning algorithms are trained on unlabeled data, without any predefined outcome

Finding Patterns

The goal is to discover hidden patterns, structures, or relationships within the data

Clustering & Dimensionality Reduction

Examples include grouping customers based on their purchasing behavior (*clustering*) or simplifying complex data into a lower-dimensional representation (*dimensionality reduction*)

Reinforcement Learning

1

Agent-Environment Interaction

Reinforcement learning involves an agent that interacts with an environment to learn through trial and error

2

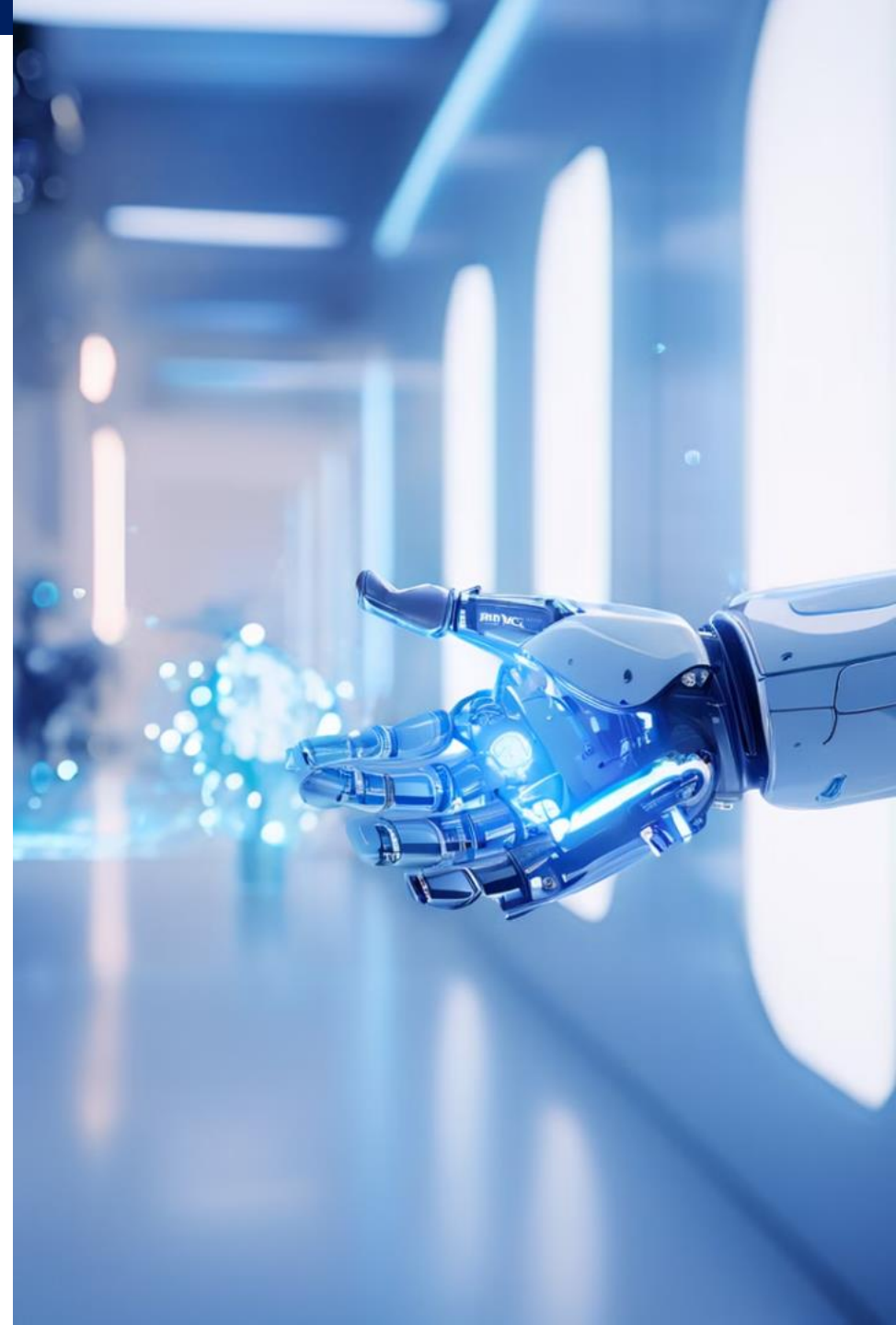
Rewards & Penalties

The agent receives rewards for desirable actions and penalties for undesirable ones

3

Optimal Policy

The goal is to find an optimal policy that maximizes the cumulative reward over time



Differences between the Learning Paradigms

Learning Paradigm	Data	Goal	Examples
Supervised	Labeled	Prediction	Image classification, spam detection
Unsupervised	Unlabeled	Pattern discovery	Customer segmentation, anomaly detection
Reinforcement	Interactive	Optimal policy	Game playing, robotics





Applications of Supervised Learning

Fraud Detection

Identifying fraudulent transactions based on historical data

Spam Filtering

Classifying emails as spam or not spam based on content and sender information

Medical Diagnosis

Predicting the likelihood of a disease based on patient symptoms and medical history

Image Recognition

Identifying objects, scenes, and faces in images

Applications of Unsupervised Learning



Customer Segmentation

Grouping customers into distinct segments based on their purchasing behavior and demographics



Anomaly Detection

Identifying unusual or unexpected patterns in data that may indicate errors, fraud, or security breaches



Dimensionality Reduction

Simplifying complex data into a lower-dimensional representation while preserving important information



Recommendation Systems

Suggesting products or services to users based on their past preferences and behaviors

Applications of Reinforcement Learning

- 1 — Game Playing**
Developing AI agents that can play games like chess, Go, and video games at a superhuman level
- 2 — Robotics**
Training robots to perform complex tasks in real-world environments, such as navigation, manipulation, and assembly
- 3 — Personalized Recommendations**
Creating dynamic recommendation systems that adapt to user preferences and provide personalized suggestions
- 4 — Resource Optimization**
Optimizing the use of resources in areas such as energy management, traffic control, and logistics



Supervised Learning

- The model is trained on a labeled dataset
- In supervised learning, each training example is a pair consisting of an input object (**features**) and a desired output value (**label** or target)
- The model learns to map inputs to the correct output based on these examples



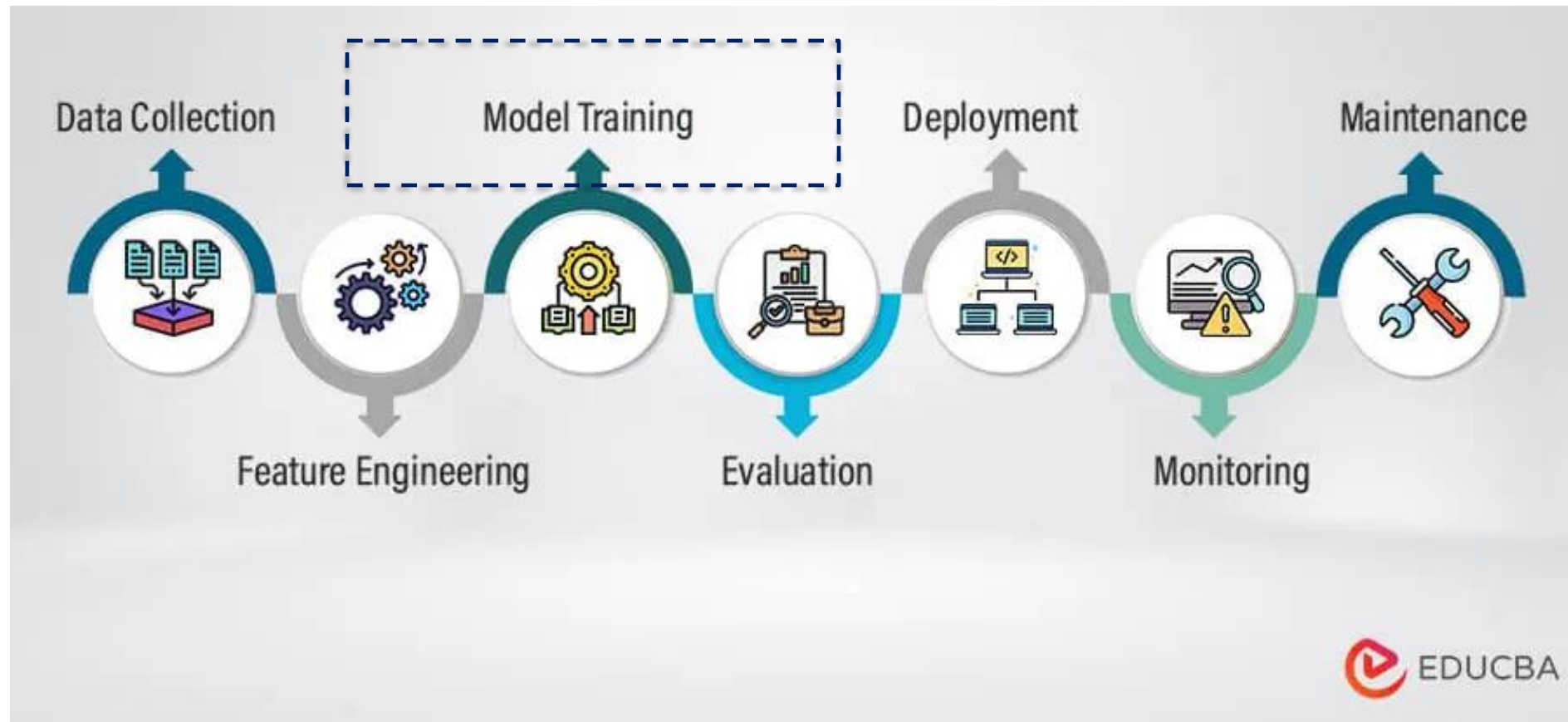
Some terms

- Features = predictor variables = independent variables
- Label = Target/response variable = dependent variable

Predictor variables					Target variable	
	sepal length (cm)	sepal width (cm)	petal length (cm)	petal width (cm)	species	
0	5.1	3.5	1.4	0.2	setosa	
1	4.9	3.0	1.4	0.2	setosa	
2	4.7	3.2	1.3	0.2	setosa	
3	4.6	3.1	1.5	0.2	setosa	
4	5.0	3.6	1.4	0.2	setosa	

This table belongs to the Iris dataset, which is commonly used when a machine learning novice.

Machine Learning Pipeline

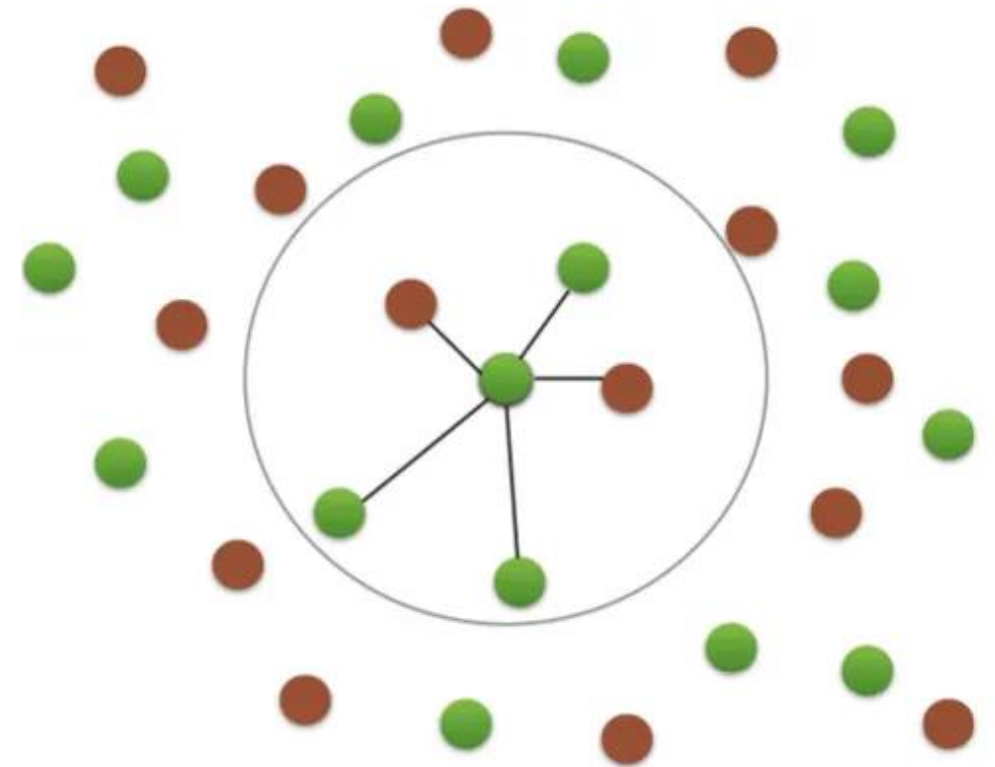




Training a ML model

K-Nearest-Neighbor (KNN)

- The k-nearest neighbors (KNN) algorithm is a supervised machine learning algorithm
 - used to solve both classification and regression problems
 - however, it is mainly used for classification problems
- Assumption: similar things exist in close proximity
 - Look at the k close neighbors to decide
- Two characteristics
 - Lazy learning algorithm
 - Non-parametric learning algorithm



Understanding KNN Algorithm

1 KNN's Core Principle

KNN classifies a data point based on its proximity to its k-nearest neighbors.

2 The Role of 'k'

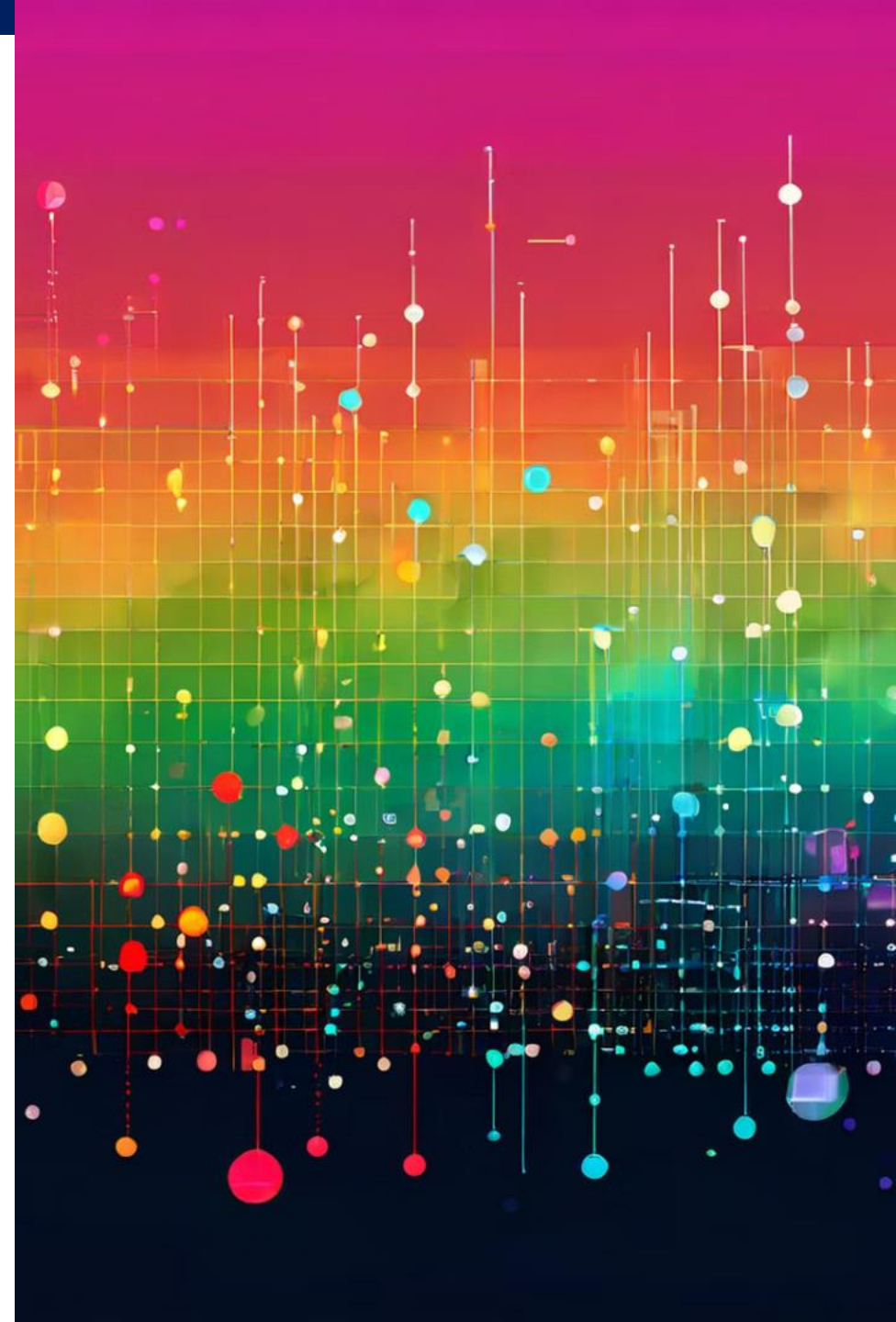
'k' represents the number of neighbors considered during classification or regression.

3 Distance Metrics

KNN employs distance metrics like Euclidean distance to measure the similarity between data points.

4 Majority Voting (Classification)

For classification, the class with the majority among the 'k' nearest neighbors is assigned to the target data point.

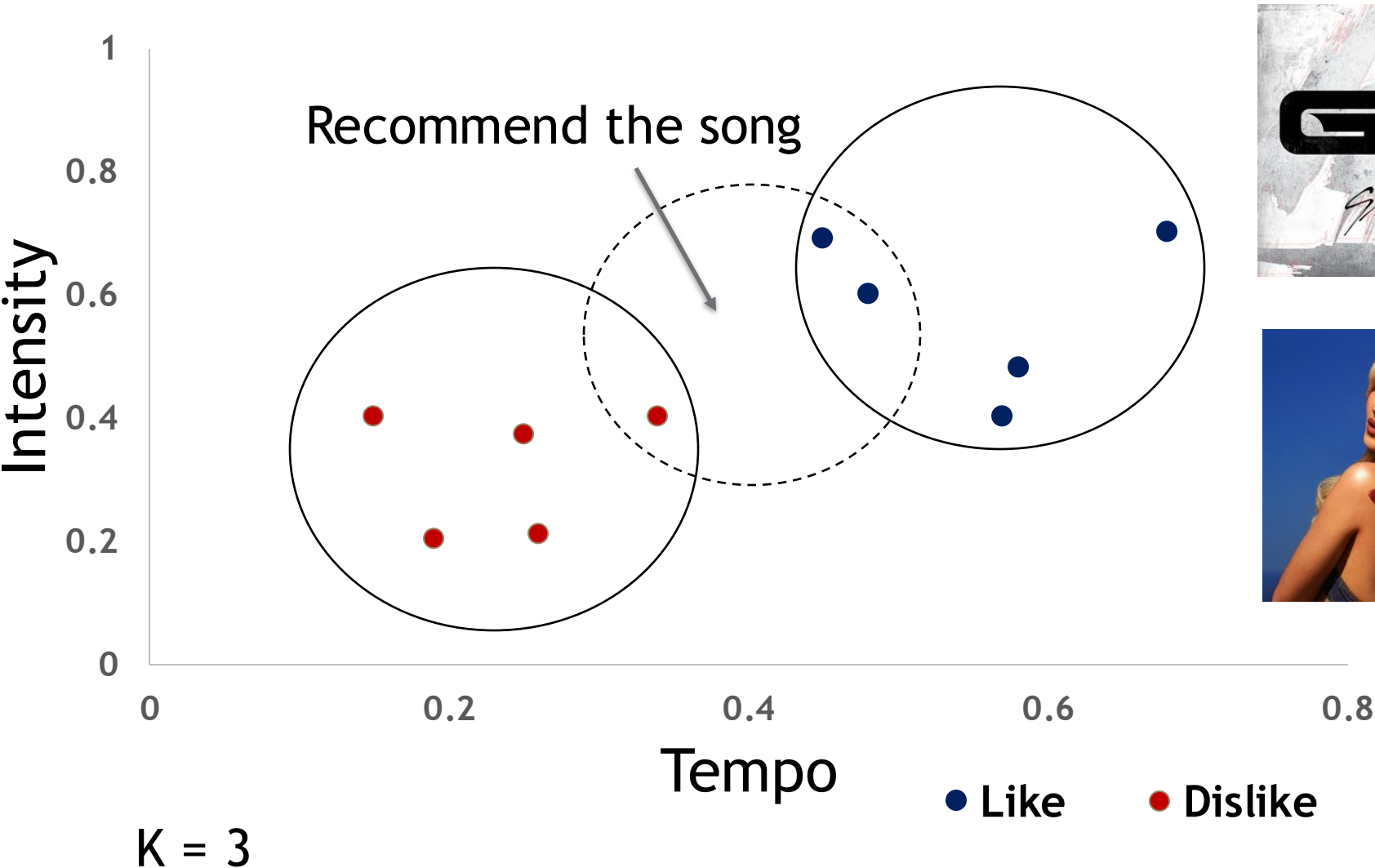


Example

- Suppose you developed your own audio streaming service
 - Spotify Competitor
- You want to leverage your service by using users' data
- New feature: recommendation of songs
 - You will let users to upvote/downvote songs
 - Train a model to recommend songs



KNN Recommender



Song A:

- **God's Menu (Stray Kids)**
- *Fast tempo and Soaring intensity*



Song B:

- **Please Please Please (Sabrina Carpenter)**
- *Midtempo and soaring intensity*

Distance

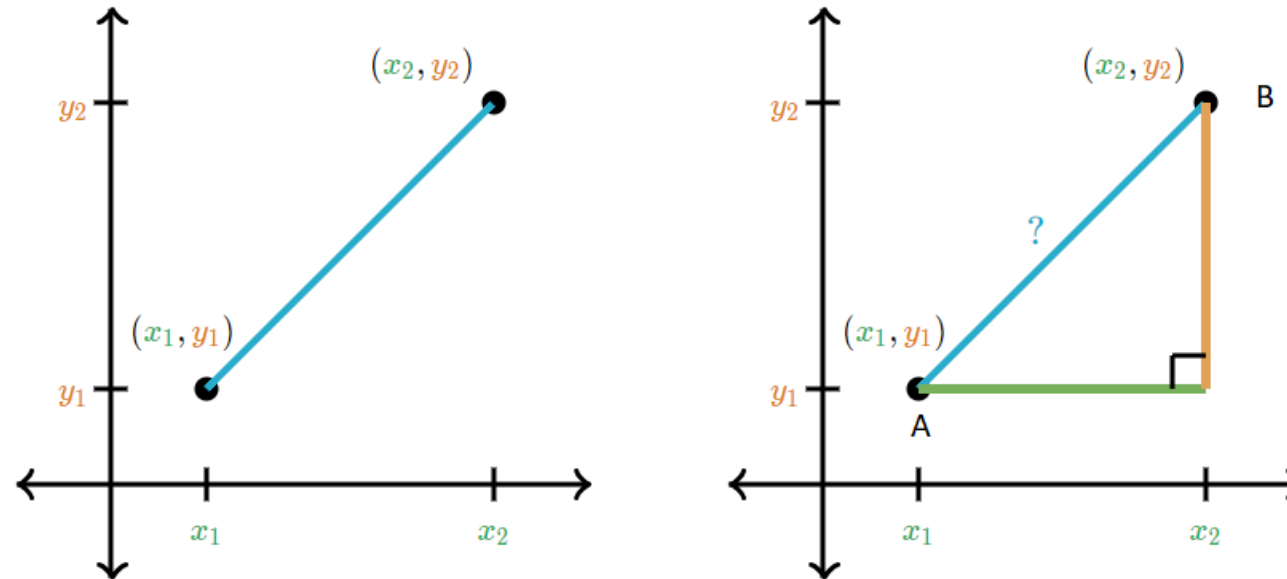
- KNN uses distance metrics to find **similarities** or **dissimilarities** among elements
 - A distance metric relies on a distance function to compute the degree of similarity (or dissimilarity)
- You must be wondering...
 - *How does a distance function indicate the similarity between elements?*

Distance


- A distance function provides distance between the elements of a set
- The distance is a degree of similarity
 - If the distance is zero, then elements are equivalent
 - Otherwise, they are different from each other
- There are several distance metrics
 - For example, *Euclidian distance*, *Manhattan distance* and *Cosine distance*

Distance Function

- Pythagorean theorem: calculating the distance between to data points



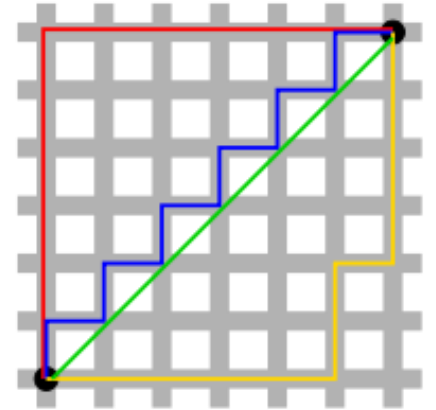
- $a^2 + b^2 = c^2$

- $d(A, B) = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2}$  Euclidean Distance

Other Distance Metrics

- Manhattan Distance is used when we need to calculate the distance between two data points in a grid like path

$$d = \sum_{i=1}^n |x_i - y_i|$$



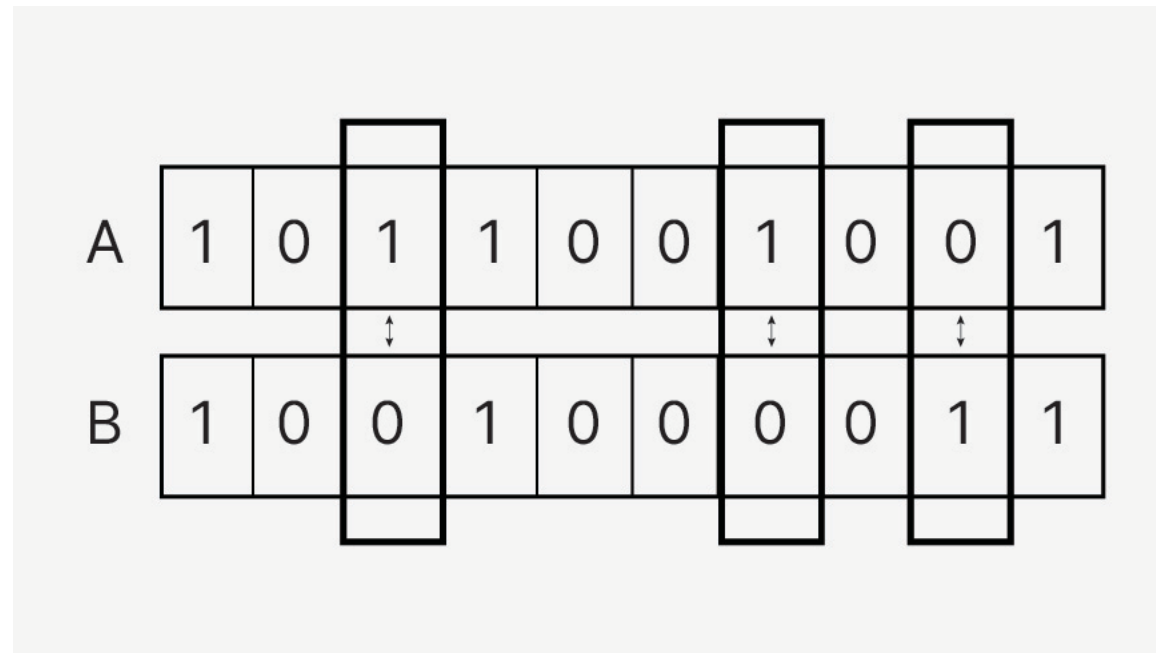
- Minkowski Distance** combines the Manhattan and Euclidian distances

$$d = \left(\sum_{i=1}^n |x_i - y_i|^p \right)^{1/p}$$

- $p = 1$, Manhattan Distance
- $p = 2$, Euclidean Distance

Hamming Distance

- It is calculated by counting the number of positions at which the corresponding symbols are different



hamming distance = 3



Hyperparameter Tuning for KNN

Optimal 'k' Value

The choice of 'k' directly influences model performance. Experimenting with different 'k' values is crucial.

Distance Metric Selection

Choosing the appropriate distance metric is vital, considering the characteristics of the data.

Weighting Techniques

Weighting neighbors based on their distance can improve accuracy by giving more influence to closer neighbors.

Choosing the Optimal Number of Neighbors (k)

1

Splitting dataset

Divide data into training and testing sets. Apply KNN with varying 'k' values on the training set and evaluate performance on the test set.

2

Hyperparameter tuning

Train models with different 'k' values.
(We can also train models with different distance metrics)

3

Performance Evaluation

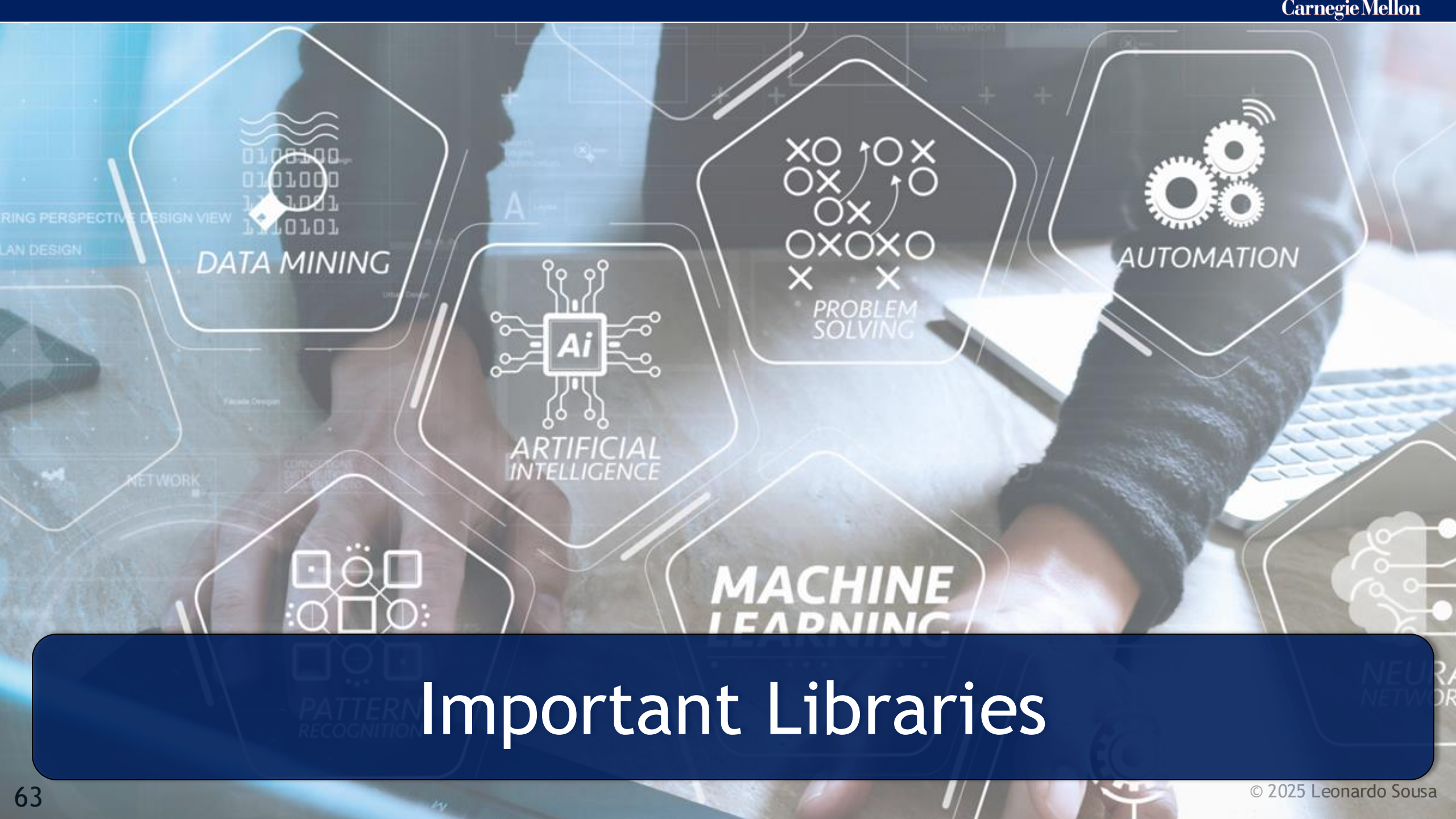
Use metrics like accuracy, precision, recall, and F1-score to assess model performance across different 'k' values.



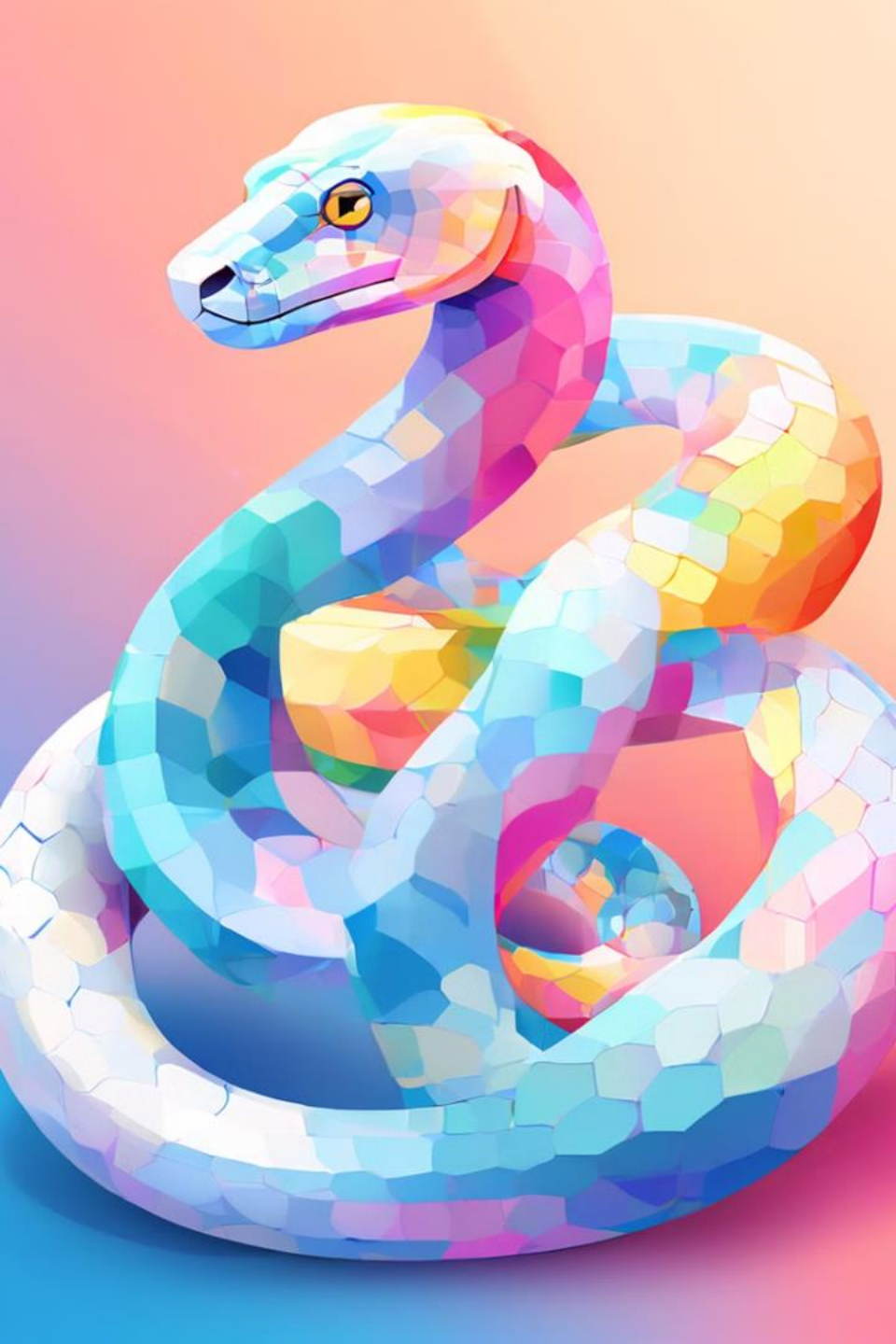
Evaluating KNN Performance

Metric	Description
Accuracy	Overall correct predictions.
Precision	Correct positive predictions among all predicted positives.
Recall	Correct positive predictions among all actual positives.
F1-Score	Harmonic mean of precision and recall.





Important Libraries



Numpy: The Foundation of Numerical Computing

1 Powerful Arrays

Numpy's core data structure is the array, optimized for efficient numerical operations on large datasets

2 Mathematical Functions

Numpy provides a wide range of mathematical functions, from basic arithmetic to linear algebra, for manipulating numerical data

3 Broadcasting

Numpy's broadcasting mechanism automatically expands the dimensions of arrays during operations, simplifying calculations

4 Random Number Generation

Numpy enables the generation of random numbers and arrays, crucial for tasks like data sampling and model initialization

Pandas: Your Data Wrangling Toolkit

Data Structures

Pandas introduces DataFrames, tabular data structures, and Series, one-dimensional labeled arrays, to organize and analyze data.

Data Manipulation

Pandas offers powerful functions for filtering, sorting, grouping, and aggregating data, enabling data cleaning and preprocessing.

Missing Data Handling

Pandas provides methods for identifying and managing missing data, crucial for maintaining data integrity and accuracy.



Visualizing Insights with Matplotlib and Seaborn

Matplotlib: Core Visualization

Matplotlib is a foundational plotting library in Python, offering a wide range of customization options for creating static, interactive, and animated visualizations

Seaborn: Statistical Visualization

Seaborn builds on Matplotlib, providing high-level functions for creating informative statistical visualizations, such as heatmaps, scatter plots, and violin plots

Visual Exploration

Data visualization plays a crucial role in understanding patterns, identifying outliers, and gaining insights from your data



Training the Model

- Import a dataset set about the [Congressional Voting Records](#);
 - (available on Canvas)
- Conduct an Exploratory Data Analysis (EDA) and
- Train a KNN classifier
 1. Import the most important libraries
 2. Exploratory data analysis
 3. Data preprocessing
 - i. Deal with missing values
 4. Model training
 - i. Split the dataset
 - ii. Hyperparameter tuning
 5. Show accuracy

