GBDi
Government Big Data Institute
สถาบันส่งเสริมการวิเคราะห์และบริหารข้อมูลขนาดใหญ่ภาครัฐ (สวข.)

depa

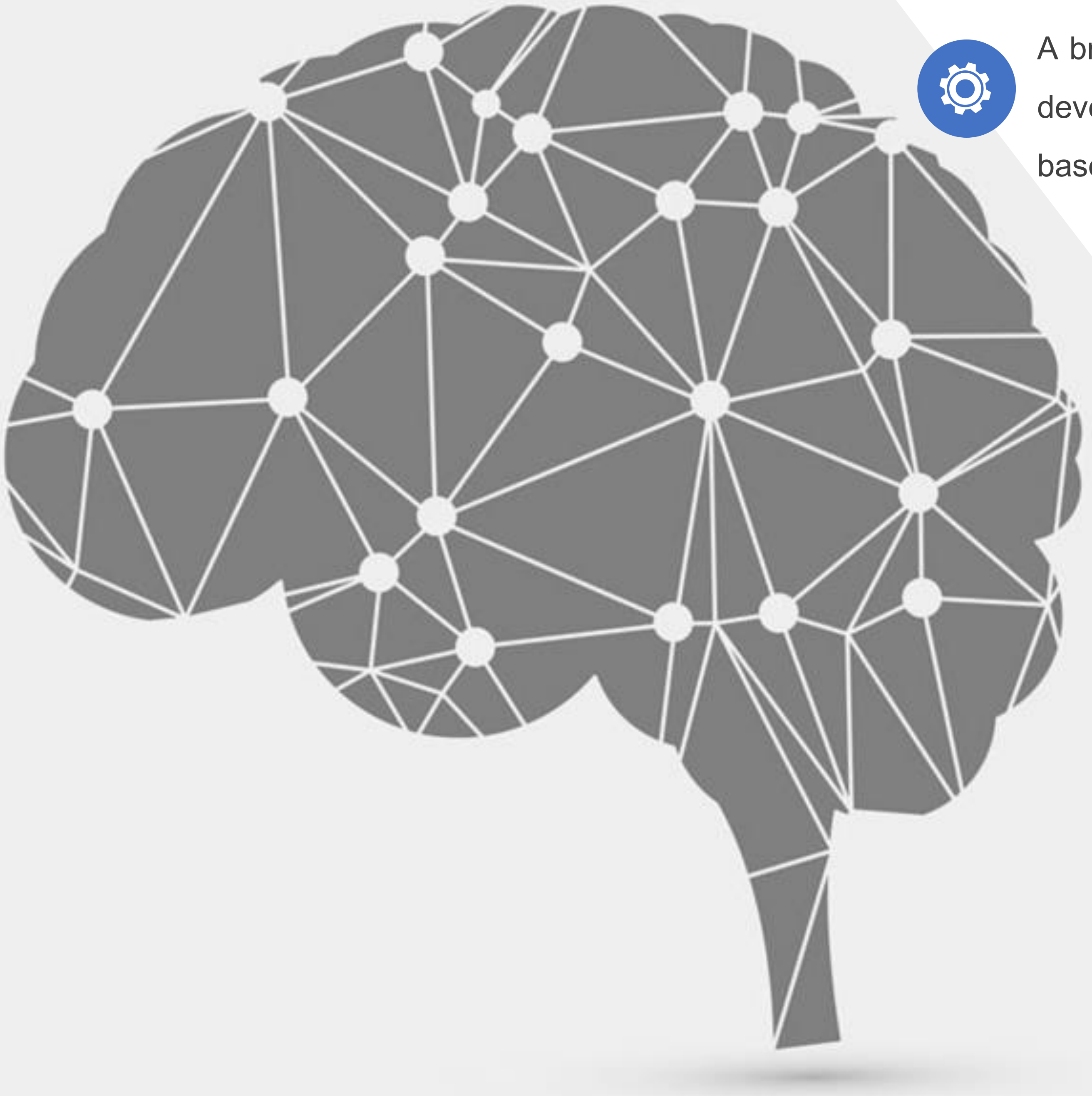
Introduction to Machine Learning:

Machine Learning Process and Model Evaluation

Thanakorn Thaminkaew

Data Scientist

Original materials by Dr. Duangjai Jitkongchuen, Papoj Thamjaroenporn, and Patipan Prasertsom



A branch of artificial intelligence, concerned with the design and development of algorithms that allow computers to evolve behaviors based on empirical data.

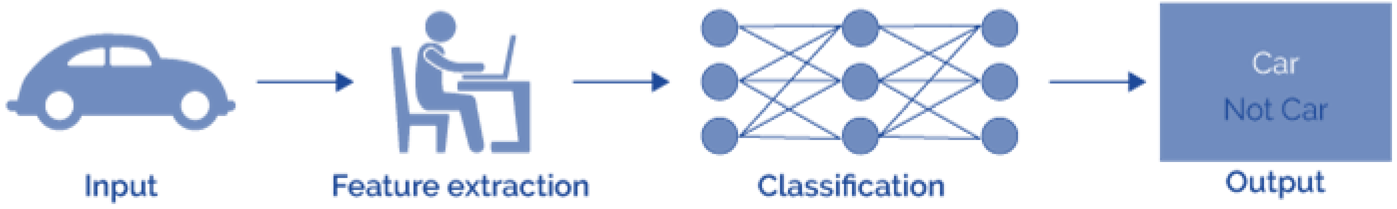
The goal of machine learning is to develop methods that can automatically detect patterns in data and then to use the uncovered patterns to predict future data or other outcomes of interest. -- Kevin P. Murphy

A computer program is said to learn from experience E with respect to some class of tasks T and performance measure P, if its performance at the tasks improves with the experiences. -- Tom Mitchell

Machine Learning

What is Machine Learning?

- Set of all tasks in which a computer can make decisions based on data
- Optimize a performance criterion using example data or experience
- It is common sense, except done by a computer
- **Role of Statistics**: Inference from a sample
- **Role of Computer science**: Efficient algorithms to
  - Solve the optimization problem
  - Representing and evaluating the model for inference



What is Machine Learning?

In general terms, we make decisions in the following two ways:

1. By using logic and reasoning
2. By using our experience

Imagine that we are trying to decide what car to buy.

- **By using logic and reasoning**: features of the car, such as price, fuel consumption, and navigation, and try to figure out the best combination of them that adjusts to our budget
- **By using our experience**: ask our friends what cars they own, and what they like and dislike about them, we form a list of information and use that list to decide, then we are using experience (in this case, our friends' experiences).

Machine learning represents the second method: making decisions using our experience.

- The term for experience is data. Therefore, in machine learning, computers make decisions based on data.

Types of Learning

Types of Learning

1. Supervised (inductive) learning

- Learn through examples of which we know the desired output (what we want to predict).
- Is this a cat or a dog?
- Are these emails spam or not?
- Predict the market value of houses, given the square meters, number of rooms, neighborhood, etc.

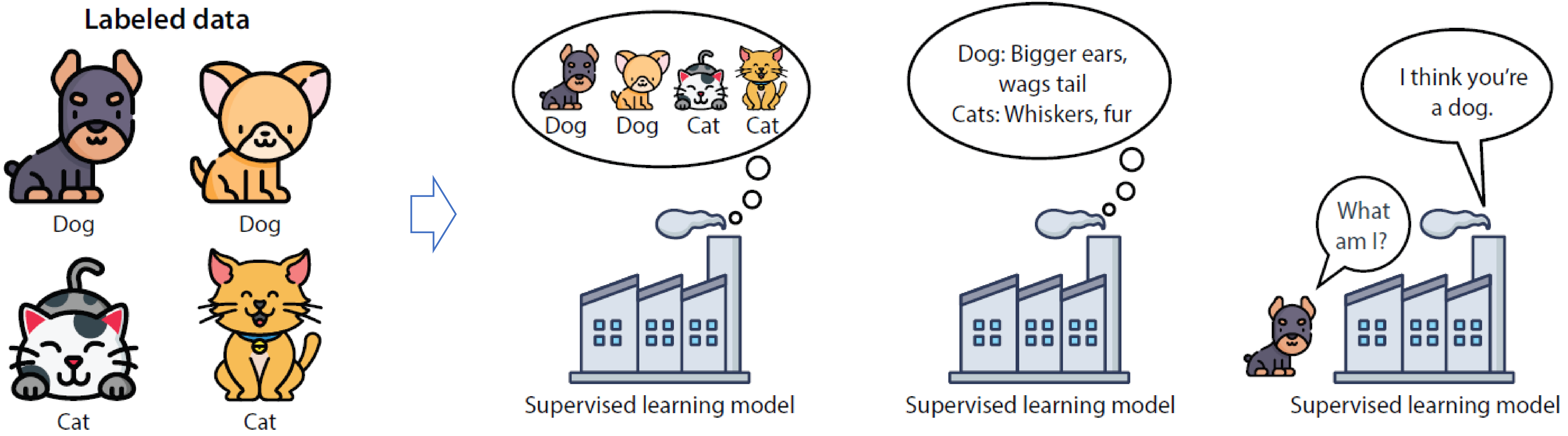
Classification

Output is a discrete variable (e.g., cat/dog)

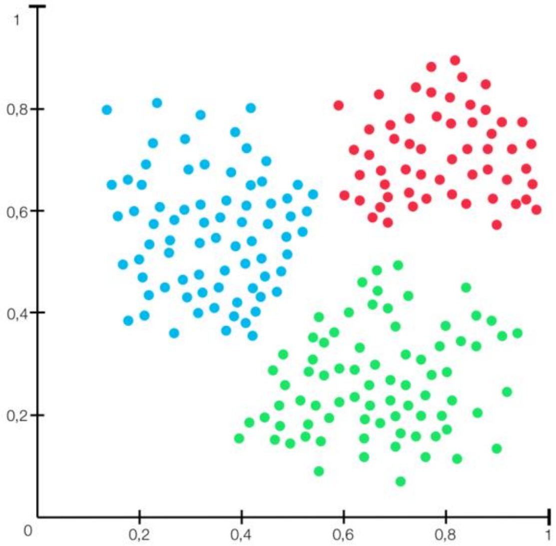
Regression

Output is continuous (e.g., price, temperature)

Supervised (inductive) learning



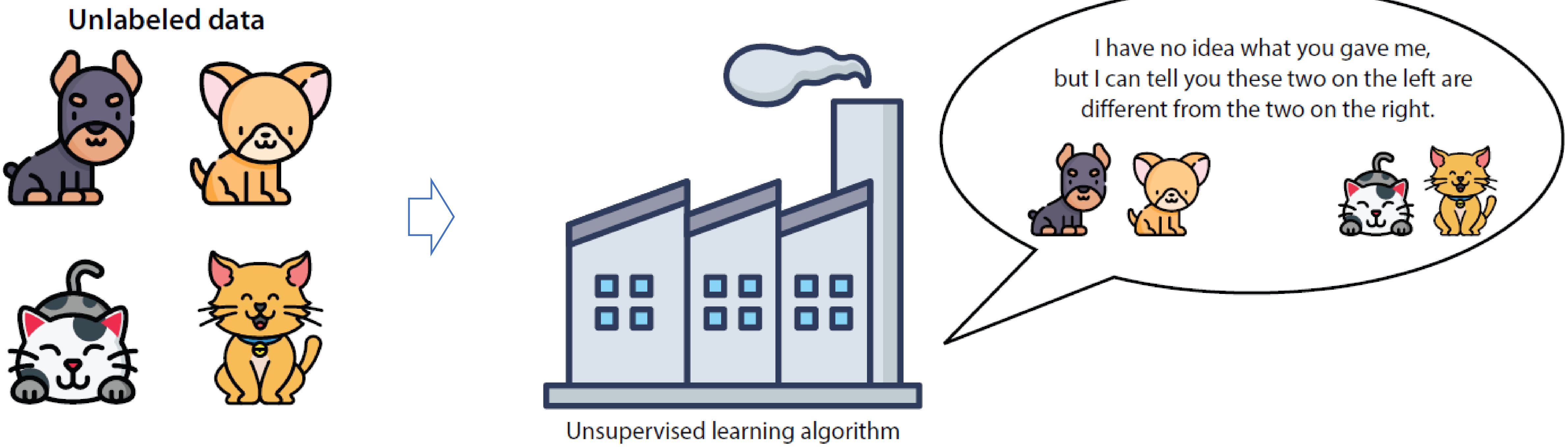
Types of Learning



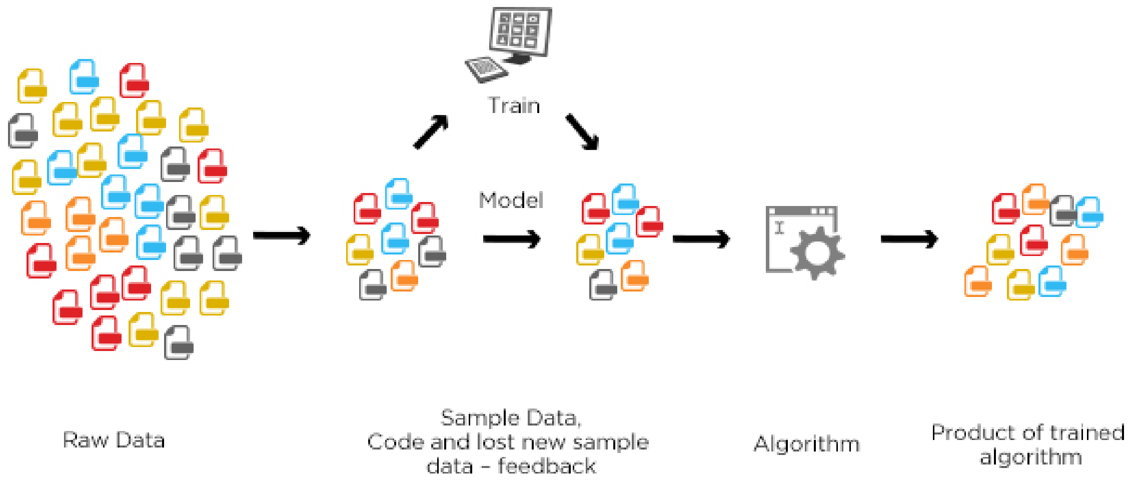
2. Unsupervised learning

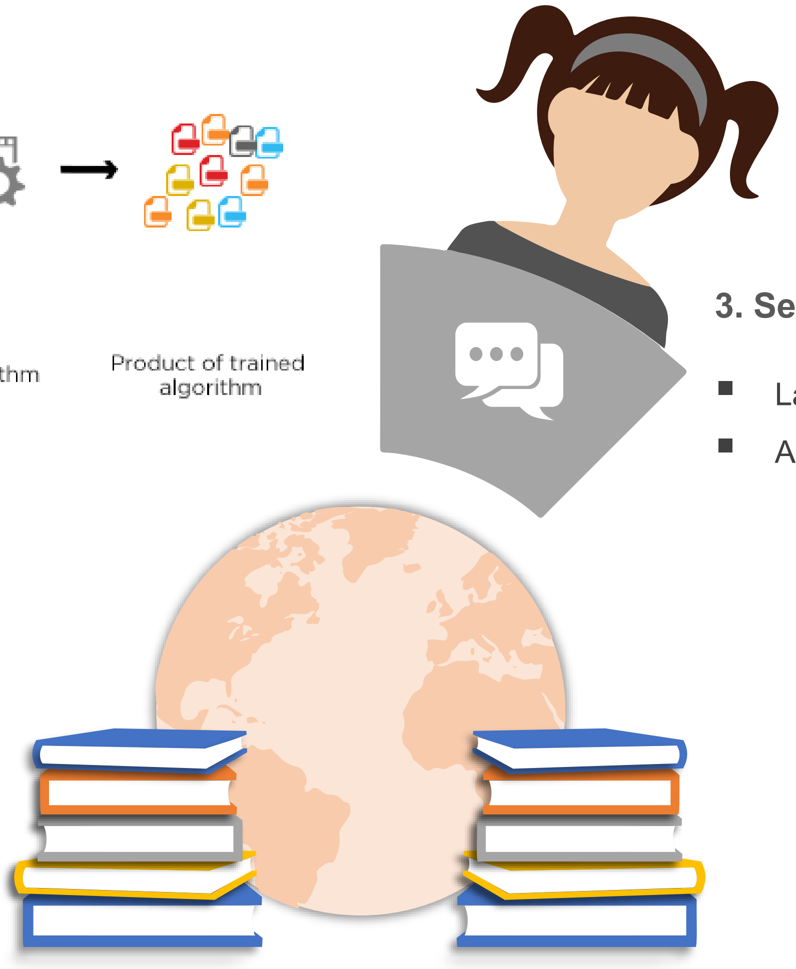
- There is no desired output. Learn something about the data. Latent relationships.
- I have photos and want to put them in 20 groups.
- I want to find anomalies in the credit card usage patterns of my customers.
- Useful for learning structure in the data (clustering), hidden correlations, reduce dimensionality, etc.

Supervised (inductive) learning



Types of Learning





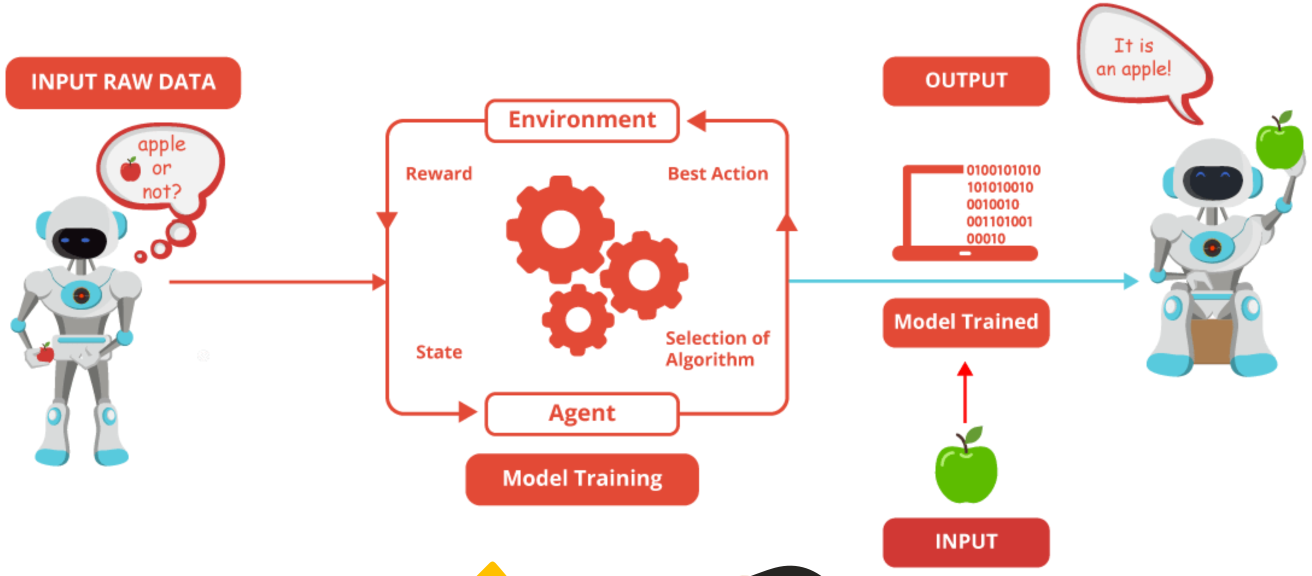
3. Semi-supervised learning

- Labels or output known for a subset of data
- A blend of supervised and unsupervised learning

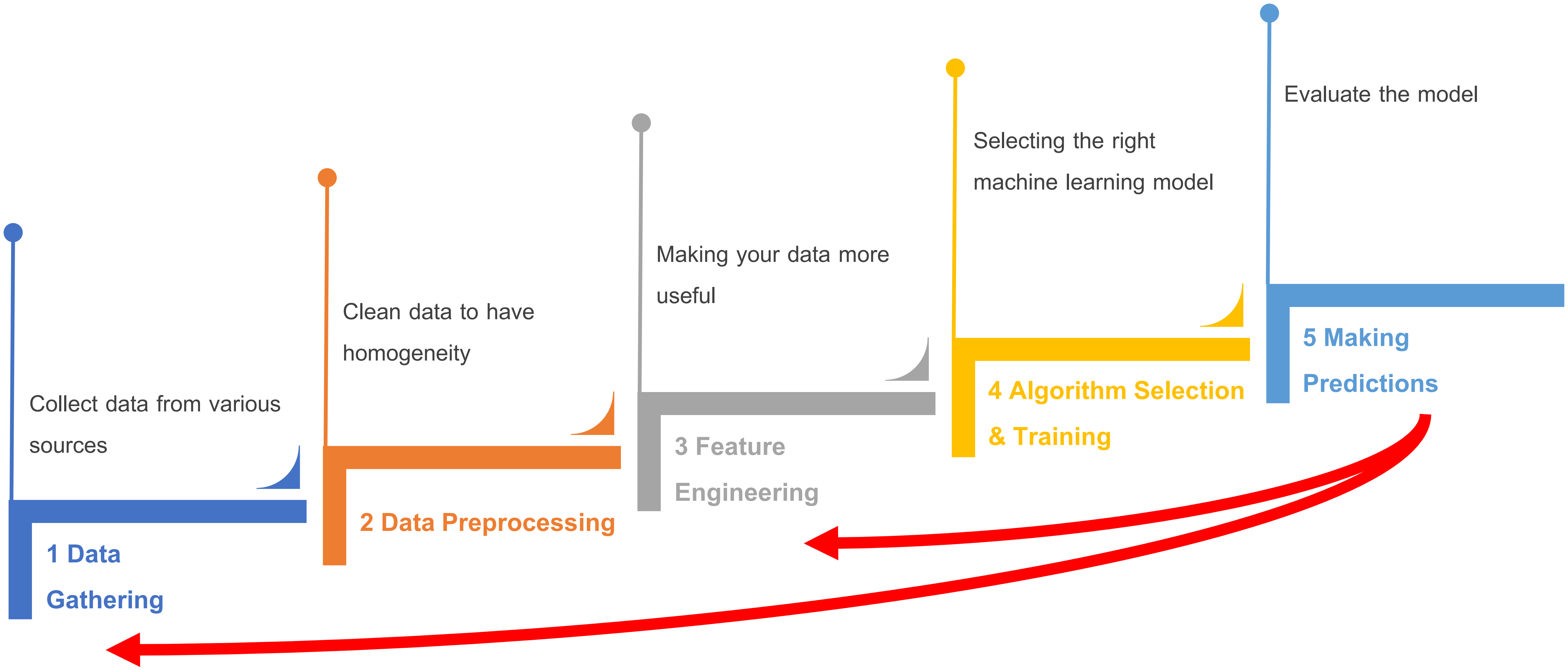
Types of Learning

4. Reinforcement learning

- Using this algorithm, the machine is trained to make specific decisions.
- It works this way: the machine is exposed to an environment where it trains itself continually using trial and error.
- This machine learns from past experiences and tries to capture the best possible knowledge to make accurate business decisions.



Steps to Solve a Machine Learning Problem

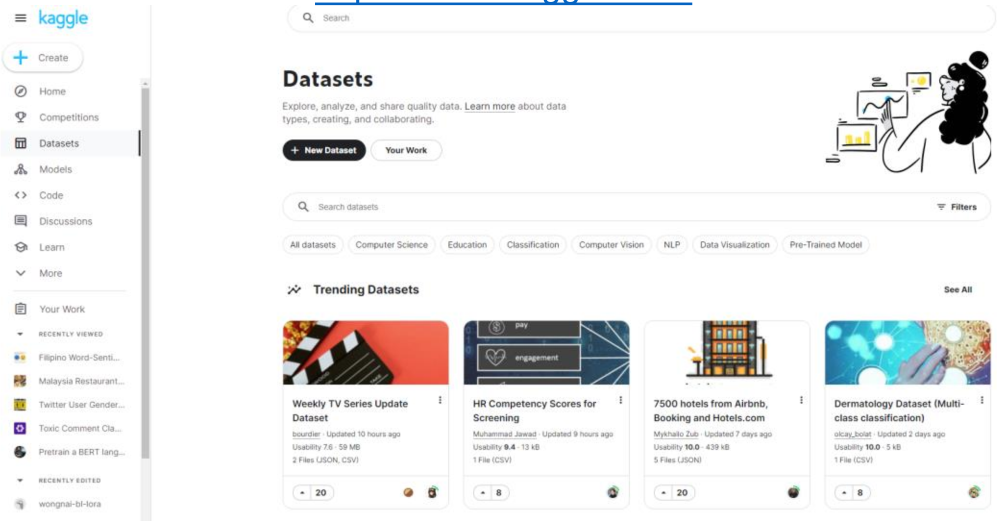


1 Data Gathering

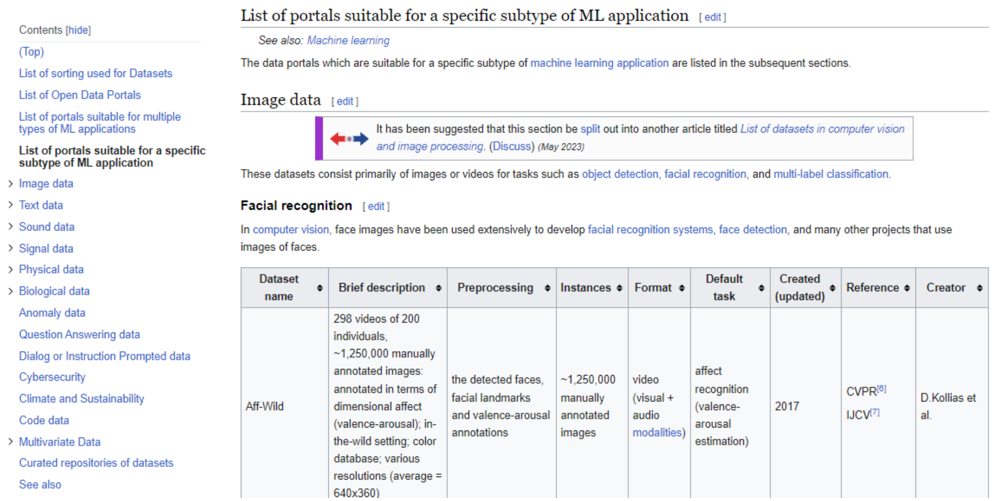
- Might depend on human work
  - Manual labeling for supervised learning.
  - Domain knowledge. Maybe even experts.
- May come for free, or "sort of“
  - E.g., Machine Translation.
- The more the better : Some algorithms need large amounts of data to be useful (e.g., neural networks).
- The quantity and quality of data dictate the model accuracy

Open Datasets

https://www.kaggle.com/

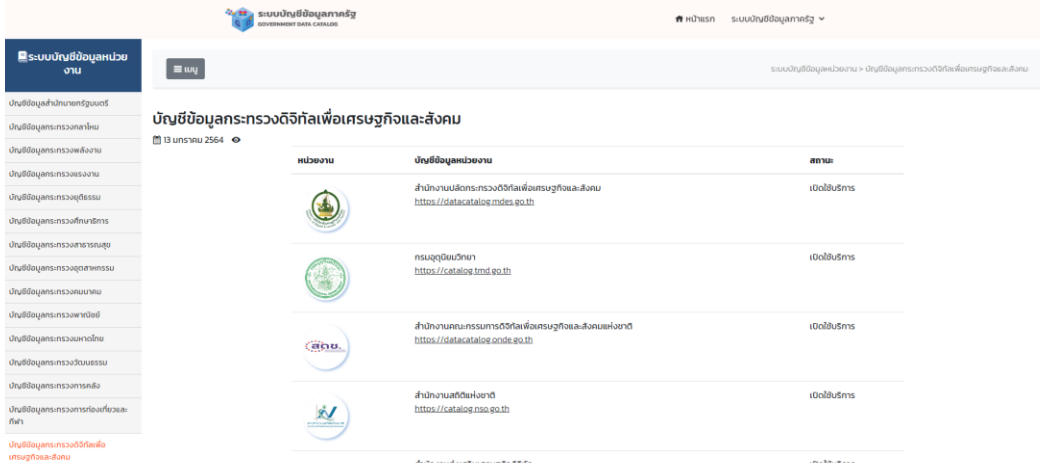


https://en.wikipedia.org/wiki/List_of_datasets_for_machine-learning_research



https://data.go.th/

https://gdhelppage.nso.go.th/p01_00.html



2 Data Preprocessing

- Perform **Exploratory Data Analysis (EDA)**
  - Essentially, **study the data**
  - This is arguably the most important step
- Is there anything wrong with the data?
  - Missing values
  - Outliers
  - Bad encoding (for text)
  - Wrongly labeled examples
  - Biased data
- Need to fix/remove data?

Python - Pandas

GBDi

https://pypi.org/project/pandas-profiling/

```python
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 545 entries, 0 to 544
Data columns (total 13 columns):
 #   Column            Non-Null Count  Dtype
---  ------            --------------  -----
 0   price(100k)       545 non-null    float64
 1   area              545 non-null    int64
 2   bedrooms          545 non-null    int64
 3   bathrooms         545 non-null    int64
 4   stories           545 non-null    int64
 5   mainroad          545 non-null    object
 6   guestroom         545 non-null    object
 7   basement          545 non-null    object
 8   hotwaterheating   545 non-null    object
 9   airconditioning   545 non-null    object
 10  parking           545 non-null    int64
 11  prefarea          545 non-null    object
 12  furnishingstatus  545 non-null    object
dtypes: float64(1), int64(5), object(7)
memory usage: 55.5+ KB
```

```python
df.describe()
```

	price(100k)	area	bedrooms	bathrooms	stories	parking
count	545.000000	545.000000	545.000000	545.000000	545.000000	545.000000
mean	47.667292	5150.541284	2.965138	1.286239	1.805505	0.693578
std	18.704396	2170.141023	0.738064	0.502470	0.867492	0.861586
min	17.500000	1650.000000	1.000000	1.000000	1.000000	0.000000
25%	34.300000	3600.000000	2.000000	1.000000	1.000000	0.000000
50%	43.400000	4600.000000	3.000000	1.000000	2.000000	0.000000
75%	57.400000	6360.000000	3.000000	2.000000	2.000000	1.000000
max	133.000000	16200.000000	6.000000	4.000000	4.000000	3.000000

```python
df.describe(include='all')
```

	price(100k)	area	bedrooms	bathrooms	stories	mainroad	guestroom	basement	hotwaterheating	airconditioning	parking	prefarea	furnishingstatus
count	545.000000	545.000000	545.000000	545.000000	545.000000	545	545	545	545	545	545.000000	545	545
unique	NaN	NaN	NaN	NaN	NaN	2	2	2	2	2	NaN	2	3
top	NaN	NaN	NaN	NaN	NaN	yes	no	no	no	no	NaN	no	semi-furnished
freq	NaN	NaN	NaN	NaN	NaN	468	448	354	520	373	NaN	417	227
mean	47.667292	5150.541284	2.965138	1.286239	1.805505	NaN	NaN	NaN	NaN	NaN	0.693578	NaN	NaN
std	18.704396	2170.141023	0.738064	0.502470	0.867492	NaN	NaN	NaN	NaN	NaN	0.861586	NaN	NaN
min	17.500000	1650.000000	1.000000	1.000000	1.000000	NaN	NaN	NaN	NaN	NaN	0.000000	NaN	NaN
25%	34.300000	3600.000000	2.000000	1.000000	1.000000	NaN	NaN	NaN	NaN	NaN	0.000000	NaN	NaN
50%	43.400000	4600.000000	3.000000	1.000000	2.000000	NaN	NaN	NaN	NaN	NaN	0.000000	NaN	NaN
75%	57.400000	6360.000000	3.000000	2.000000	2.000000	NaN	NaN	NaN	NaN	NaN	1.000000	NaN	NaN
max	133.000000	16200.000000	6.000000	4.000000	4.000000	NaN	NaN	NaN	NaN	NaN	3.000000	NaN	NaN

```python
[15]: from ydata_profiling import ProfileReport
      profile = ProfileReport(df)
```

```python
[16]: profile
```

Summarize dataset: 100% 31/31 [00:01<00:00, 11.07it/s, Completed]

Generate report structure: 100% 1/1 [00:01<00:00, 1.75s/it]

Render HTML: 100% 1/1 [00:00<00:00, 1.77it/s]

Pandas Profiling Report — Overview | Variables | Interactions | Correlations | Missing values | Sample

Overview

Overview | Alerts 4 | Reproduction

Dataset statistics

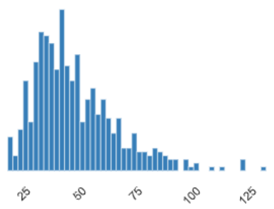
Number of variables	13
Number of observations	545
Missing cells	0
Missing cells (%)	0.0%
Duplicate rows	0
Duplicate rows (%)	0.0%
Total size in memory	55.5 KiB
Average record size in memory	104.2 B

Variable types

Numeric	3
Categorical	4
Boolean	6

price(100k)
Real number (ℝ)

Distinct	219	Minimum	17.5
Distinct (%)	40.2%	Maximum	133
Missing	0	Zeros	0
Missing (%)	0.0%	Zeros (%)	0.0%
Infinite	0	Negative	0
Infinite (%)	0.0%	Negative (%)	0.0%
Mean	47.667292	Memory size	4.4 KiB



More details

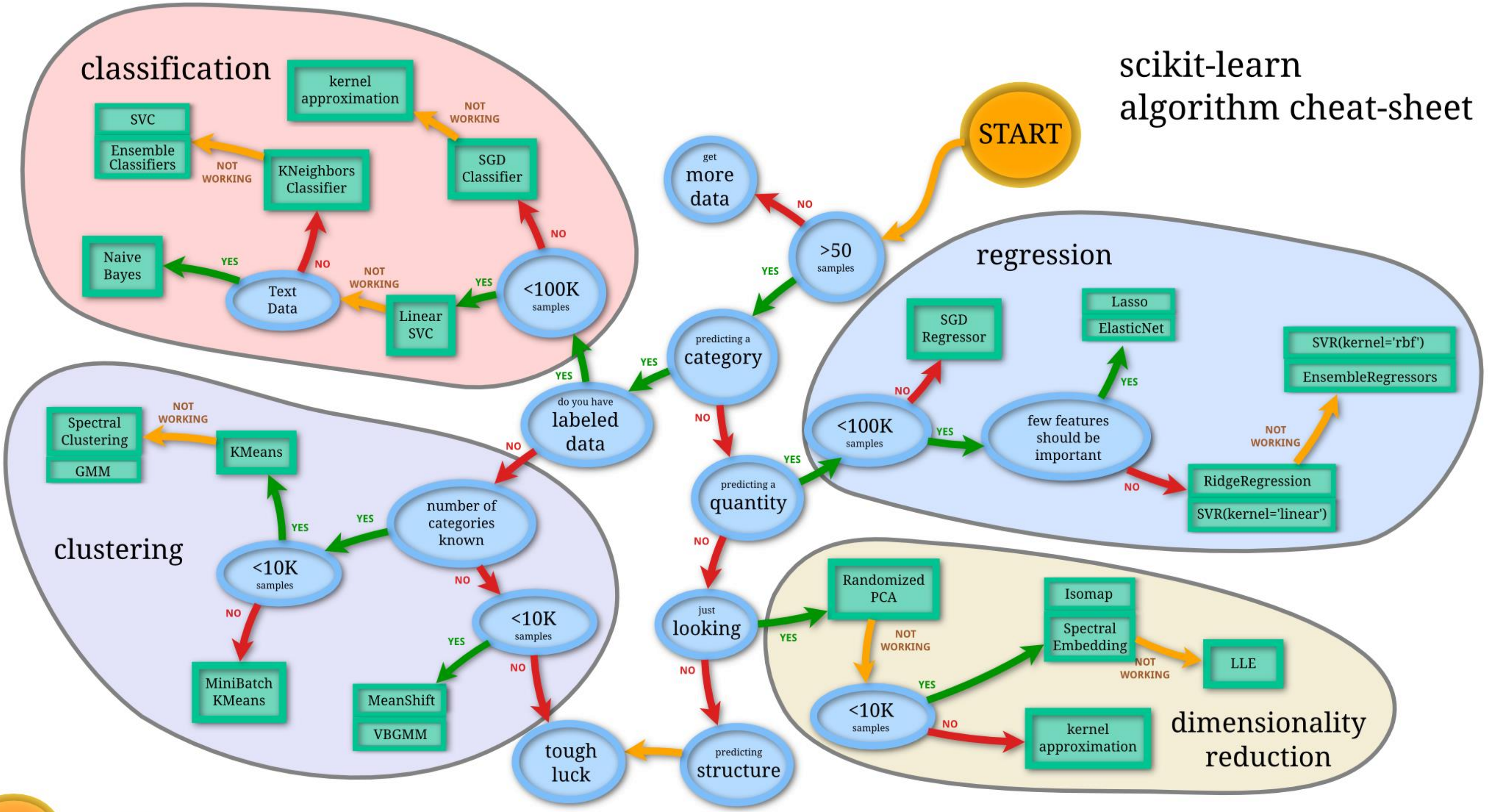
3 Feature Engineering

- What is a **feature**?
  - A feature is an individual measurable property of a phenomenon being observed
- Our inputs are represented by a set of features.
- Combining multiple columns (today – date of purchase)
- Extracting datetime features (days, month, season, night)
- Binning (gen x, y)
- One-hot encoding
- To classify spam email, features could be:
  - Language of the email (0=English,1=Spanish)
  - Number of emojis
  - Text Length

3 Feature Engineering

- Extract more information from existing data, not adding "new" data
  - Making it more useful
  - With good features, most algorithms can learn faster
- It can be an art
  - Requires thought and knowledge of the data
- Two steps:
  - Variable transformation (e.g., dates into weekdays, normalizing)
  - Feature creation (e.g., n grams for texts, if word is capitalized to detect names, etc.)

4 Algorithm Selection & Training



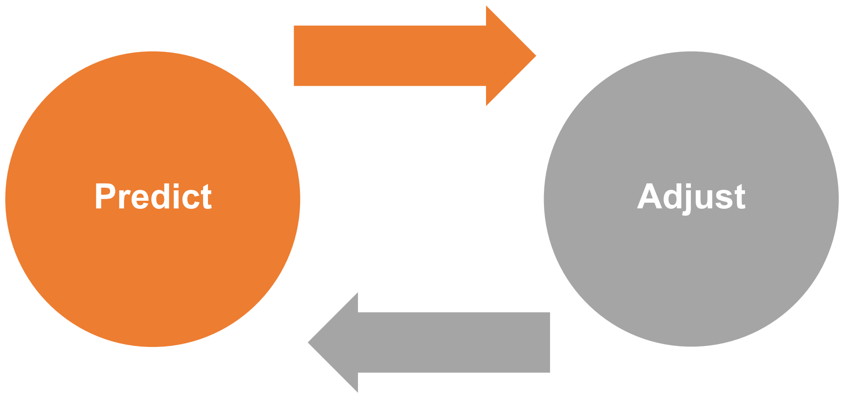
Back

scikit learn

https://scikit-learn.org/stable/tutorial/machine_learning_map/index.html

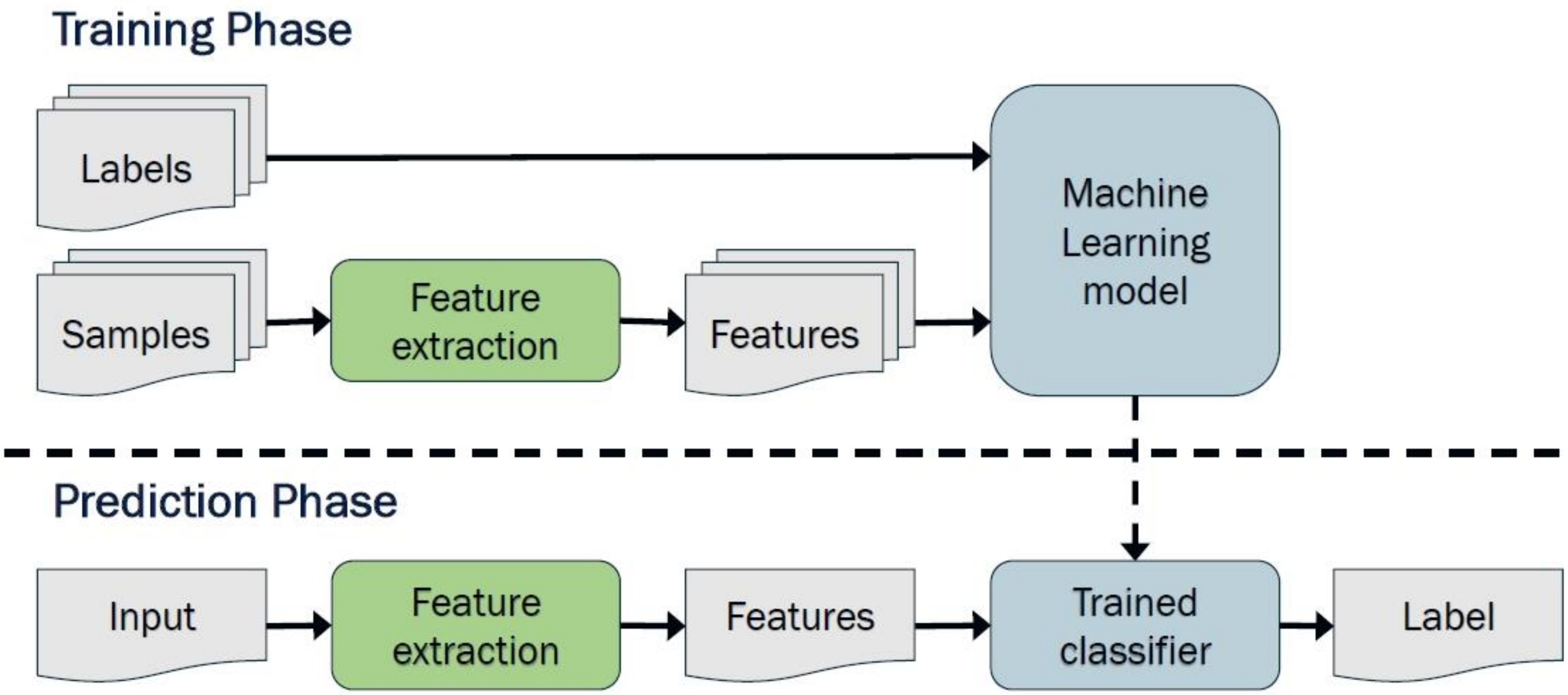
4 Algorithm Selection & Training

- **Goal of training** : making the correct prediction as often as possible
  - Incremental improvement:

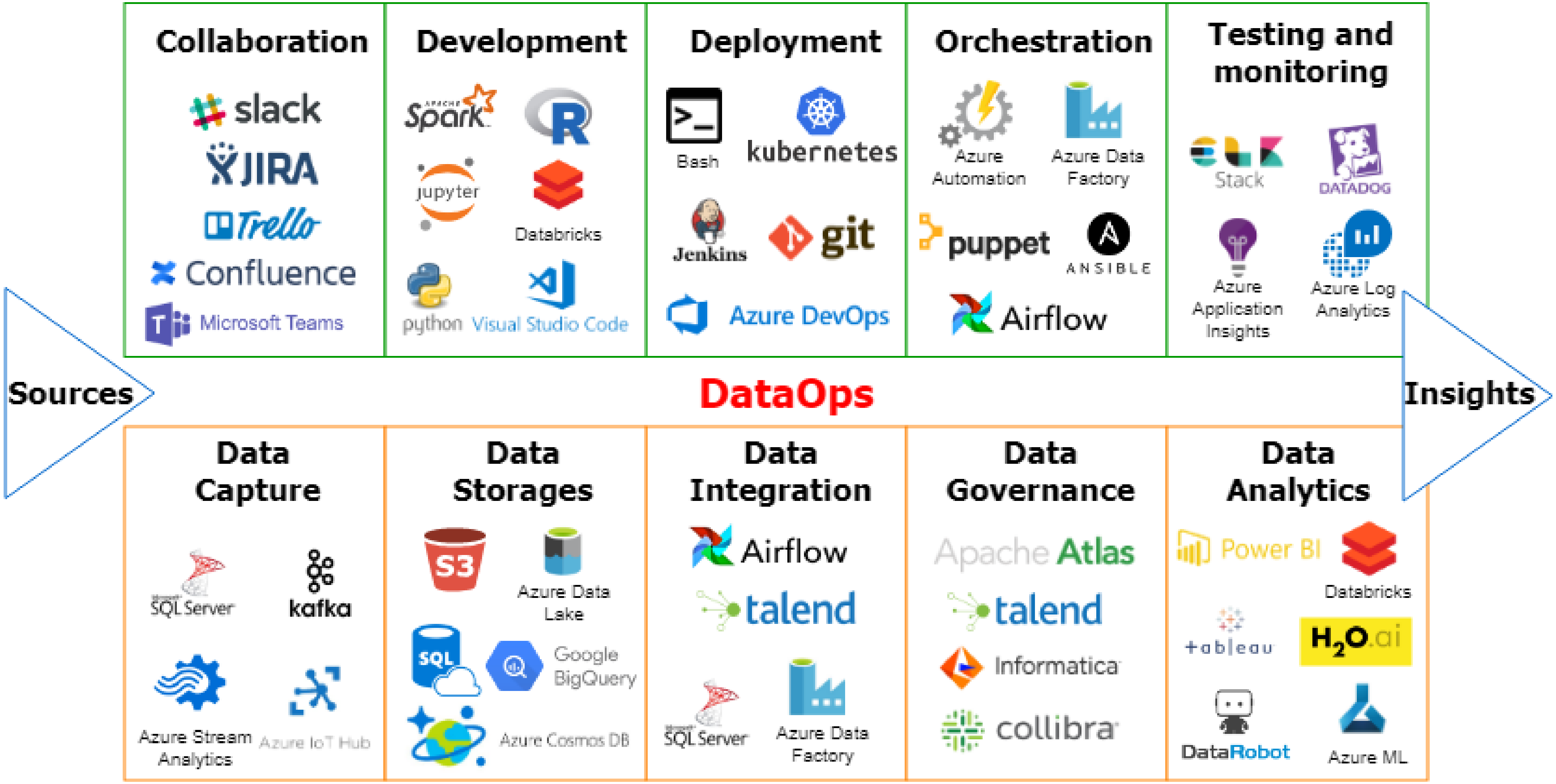


  - Use of metrics for evaluating performance and comparing solutions
  - Hyperparameter tuning more an art than a science

05 Making Predictions



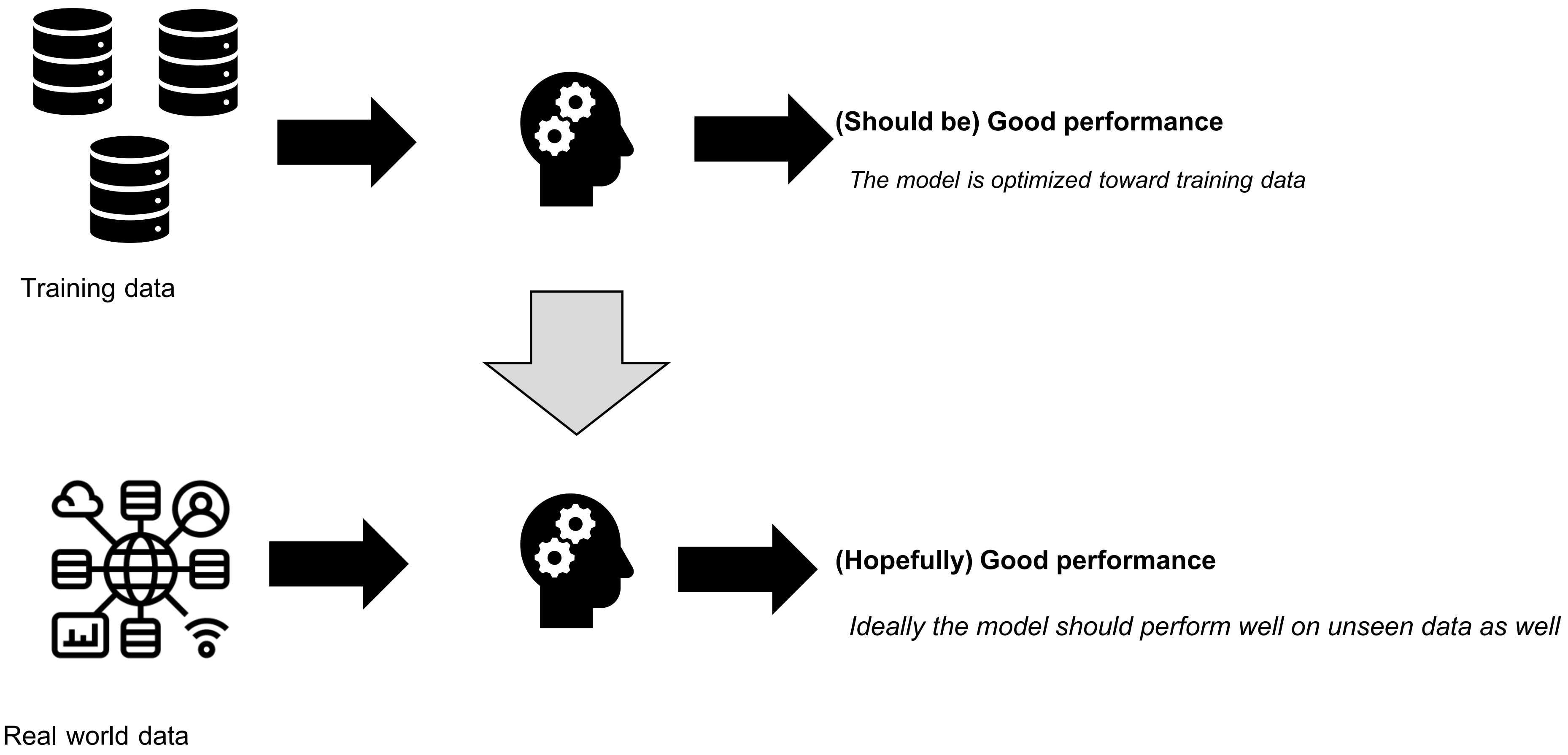
Implementation and Deployment



https://www.devopsschool.com/blog/top-20-dataops-tools-and-its-ranking/

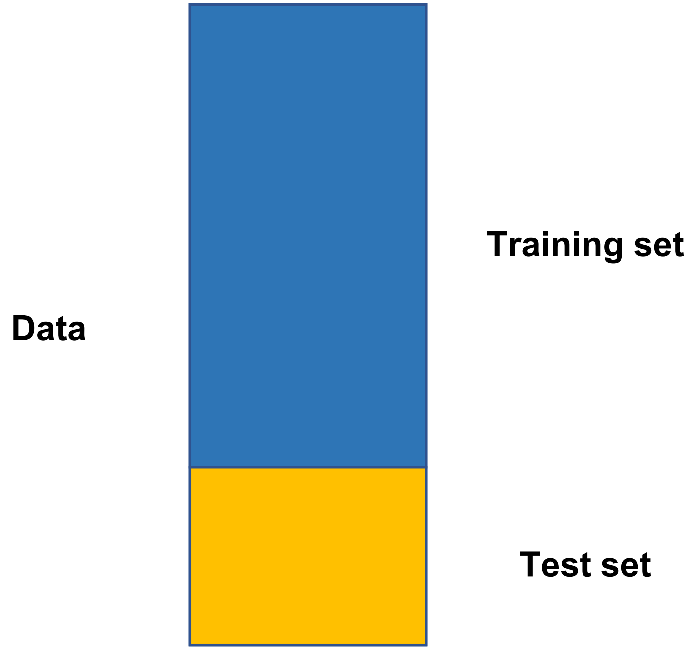
Model Evaluation

Machine Learning Model



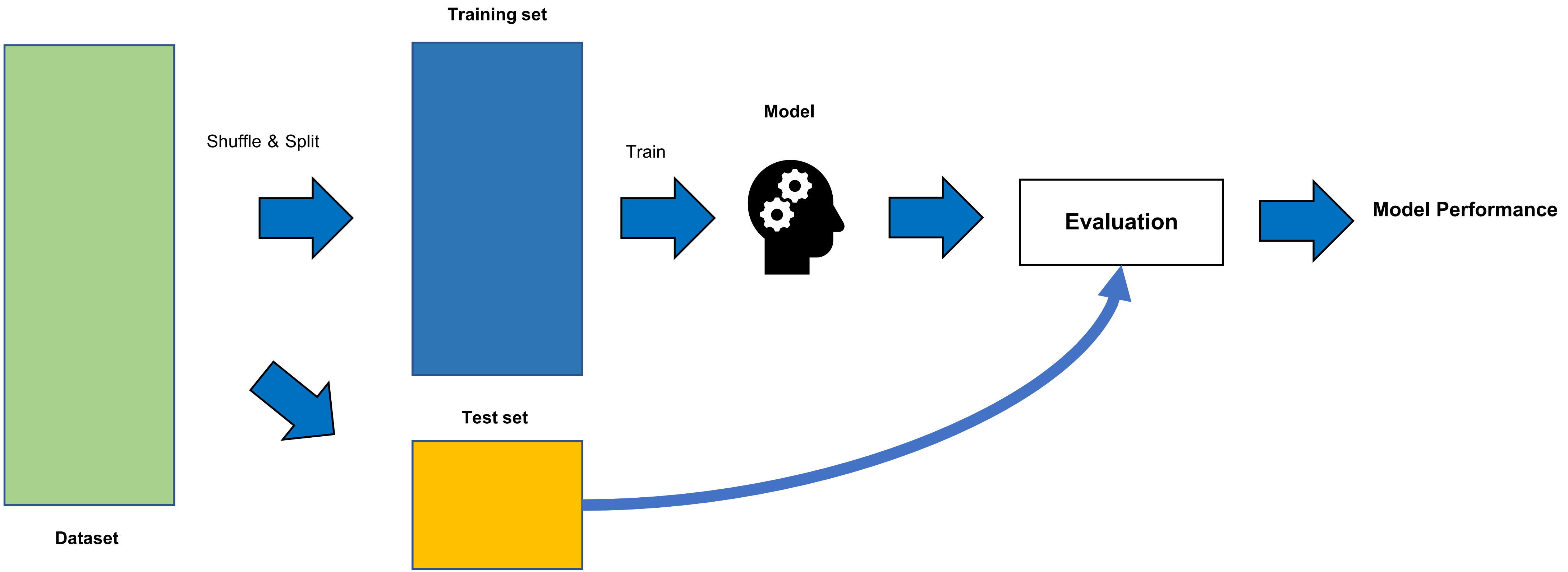
Evaluating a Model: Training Data and Test Data

- In the most basic machine learning project, we shuffle the data and split them into a training set and a test set
- A training set is used to **train** the model – creating a mathematical equations and/or adjusting the model's parameters so that it can best predict
- A test set is used to represent **unknown** data



Basic Model Evaluation

- The model is trained with training data set and evaluated with test dataset



k-Nearest Neighbors classifier (kNN)

k-Nearest Neighbors classifier (kNN)

- **The idea**: similar data points are likely to be of the same type.
- We infer a class of a data point from its ***k* most similar** data
- How do we find the "most similar" (nearest) data points?
  - There are many way of measuring the distance between data point. One frequently discussed method is the Euclidean distance.
  - Euclidean distance – a distance between 2 points in cartesian coordinates

$$Euclidean\ Distance = \sqrt{(x_i - x_j)^2 + (y_i - y_j)^2}$$

- Basically asking, "**By looking at k data points that are most similar to me, which class am I most likely to be in?**"

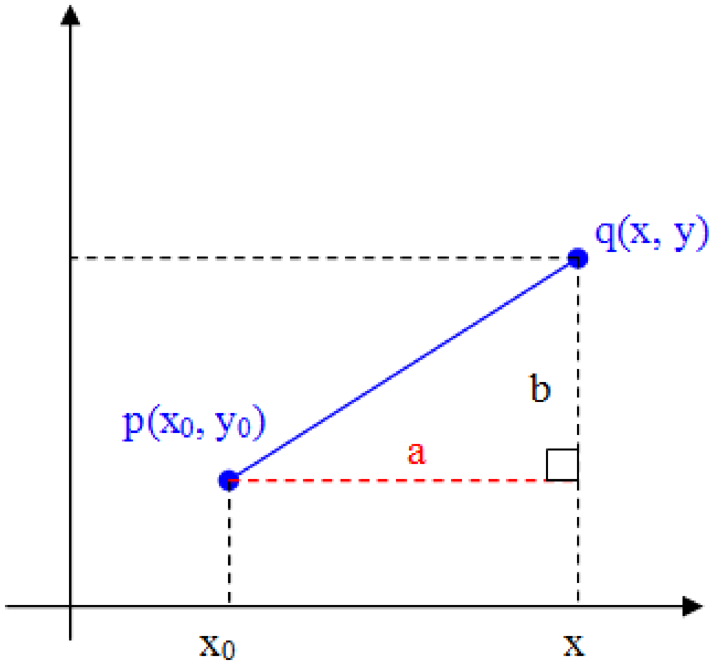


Figure from Illuyanka (https://commons.wikimedia.org/wiki/File:Dot_Product.svg)

k-Nearest Neighbors classifier (kNN)

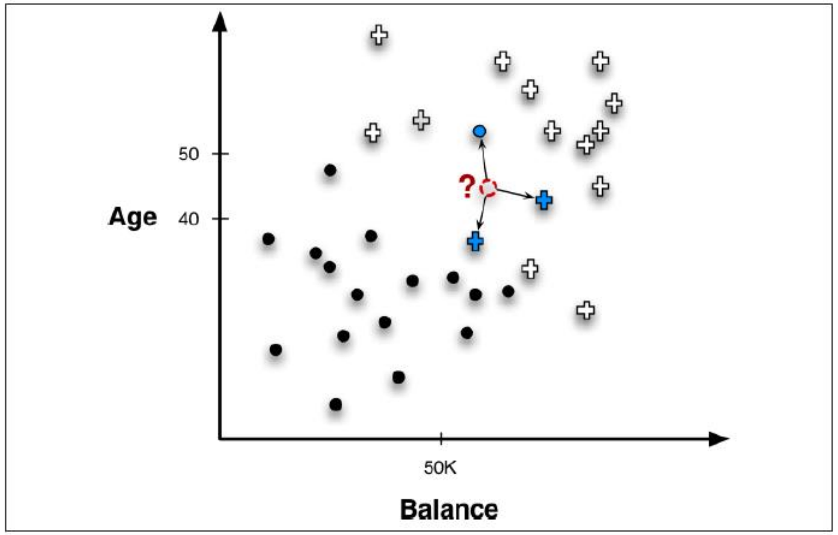
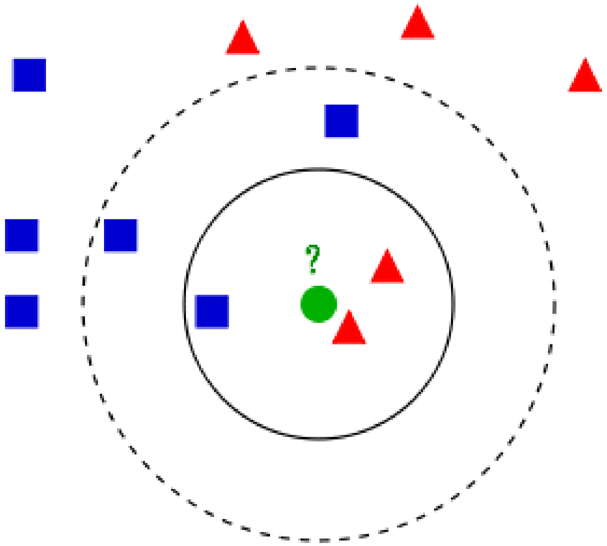


Figure 6-2. Nearest neighbor classification. The point to be classified, labeled with a question mark, would be classified + because the majority of its nearest (three) neighbors are +.



- Observes k closest data point and decide the class of data points by voting.
- Usually, k is chosen as an odd number (why?)

- The choice of ***k*** matters!
- Different number of k can result in different predictions
- Feature scale matters!

Figures from Data Science for Business by Foster Provost & Tom Fawcett and Antti Ajanki AnAj (https://commons.wikimedia.org/wiki/File:Dot_Product.svg)

k-Nearest Neighbors classifier (kNN)

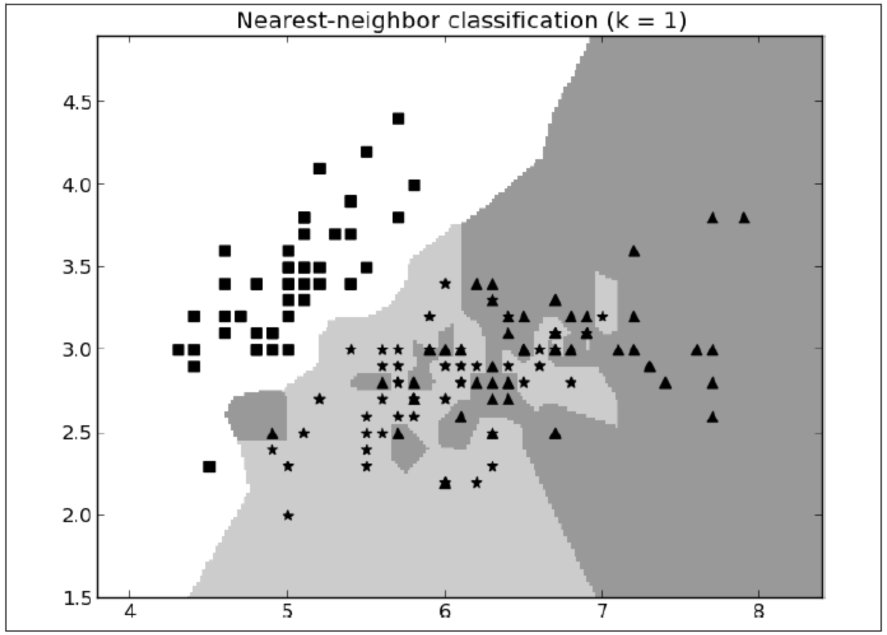


Figure 6-4. Classification boundaries created on a three-class problem created by 1-NN (single nearest neighbor).

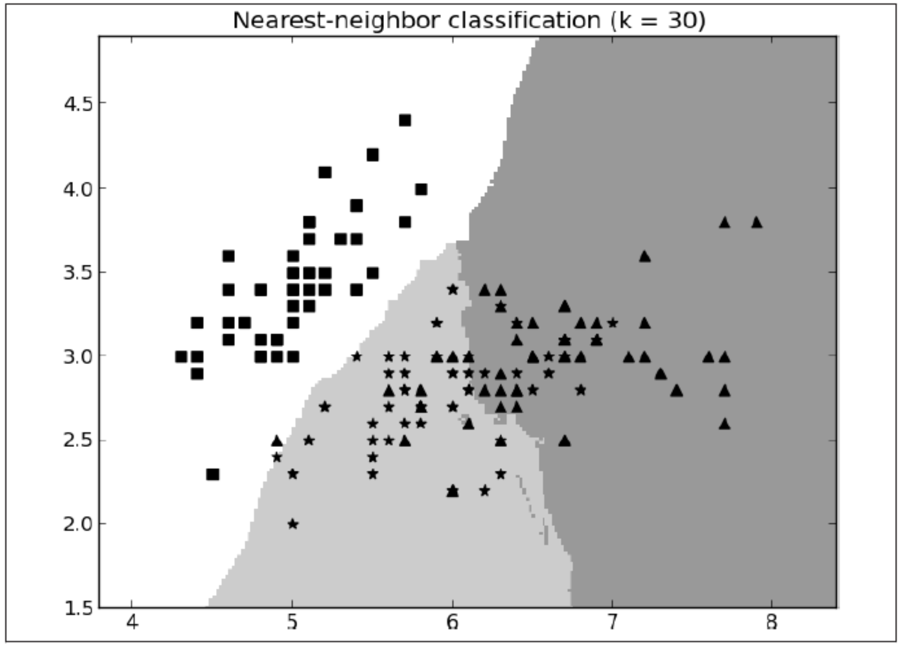


Figure 6-5. Classification boundaries created on a three-class problem created by 30-NN (averaging 30 nearest neighbors).

kNN models with a small *k*

- A finer granularity separation boundaries
- More susceptible to the presents of outlier.

kNN models with a large *k*

- More tolerant to noise
- Smooth but coarser separation boundaries

Python Tutorial

- Colab!
- 1.1 kNN (exam_data)
- 1.2 kNN (Social_Network_Ads)

Model Evaluation (Cont.)

GBDi

Making Adjustment: Hyperparameters Tuning

- Aside from selecting appropriate models for a problem, a data scientist will also need to properly configure and adjust their model to be most suitable for tasks assigned.
- Each machine learning model has their own parameters that need to be set before prior to the start of the model training (can think of it as we configure the "settings" of the model)
  - These parameters values are called **hyperparameters**
- Think of a model as a food. Even if using the same ingredient (data) but the difference in seasoning (parameters adjustment) can affect how good it taste.
  - Hyperparameter tuning can sometimes provide a noticeable improvement in a model performance.
- The process of tuning hyper parameters allows us to obtain the most optimal setting for a model that can maximize our model's target.

A Situation: Tuning the Hyperparameters

- For this example, let's say we are developing a classification model (or any model) and we are trying to adjust the parameters to maximize the model's performance in real scenario.
- How do we select appropriate values for hyperparameters?

Method 1:

- Try multiple different values for hyperparameters and pick the one that performs the best for the training data
- How is this method?
  - Overly simple and probably not generalizable

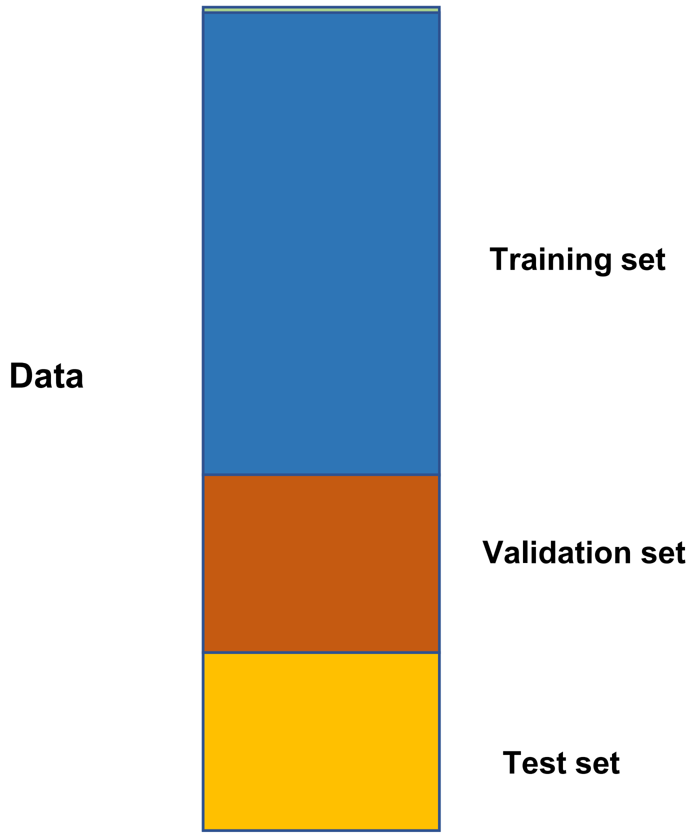
Let’s change the method

Method 2:

- Alternatively, we can try to adjust the model parameters in a way that they maximize the model’s performance on the <u>test data</u>
- How is this method?
  - It’s wrong! … but why?
- By doing so, data in the test dataset is no longer representing “**unknown**” data
  - In fact, the test set now also becomes part of the training data
- This may result in an unfair evaluation of the model performance and can cause the model to *overfit* to the test dataset.

The Correct Way: Holdout Method

- To properly perform hyperparameter tuning and model evaluation, we need to hold an additional portion of data out for final evaluation
- Hence, the data is split into 3 sets instead of 2
  - The training set is used to train the model.
  - The validation set is then used to tune the parameters
  - Finally, the test set (holdout set) is used to perform final evaluation of the model performance



The Holdout Method

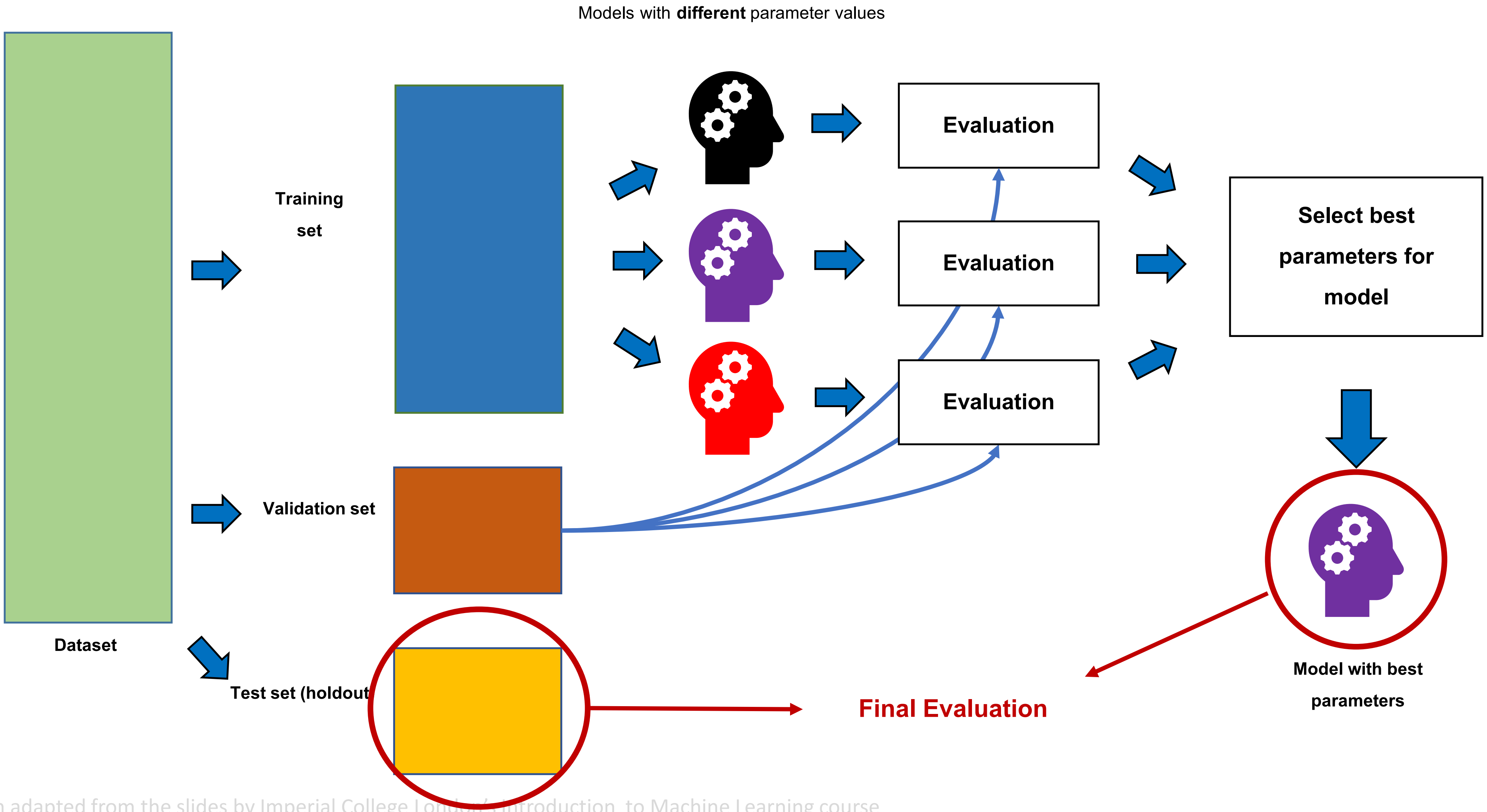


Diagram adapted from the slides by Imperial College London's Introduction to Machine Learning course

The Holdout Method (Cont.)

- Once the model with the best parameters is obtained, we can either
  - Use test set to evaluate it right away
  - Or, combine training and validation set and use them to retrain the model before evaluating with the test set

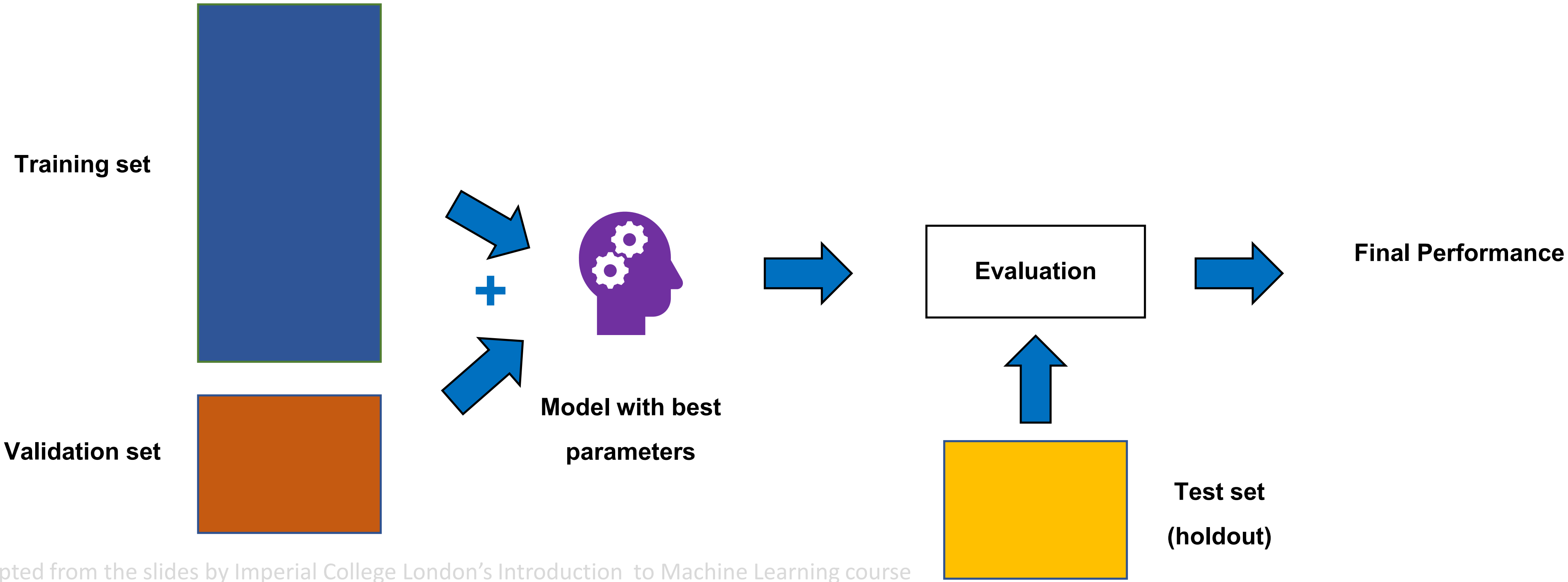
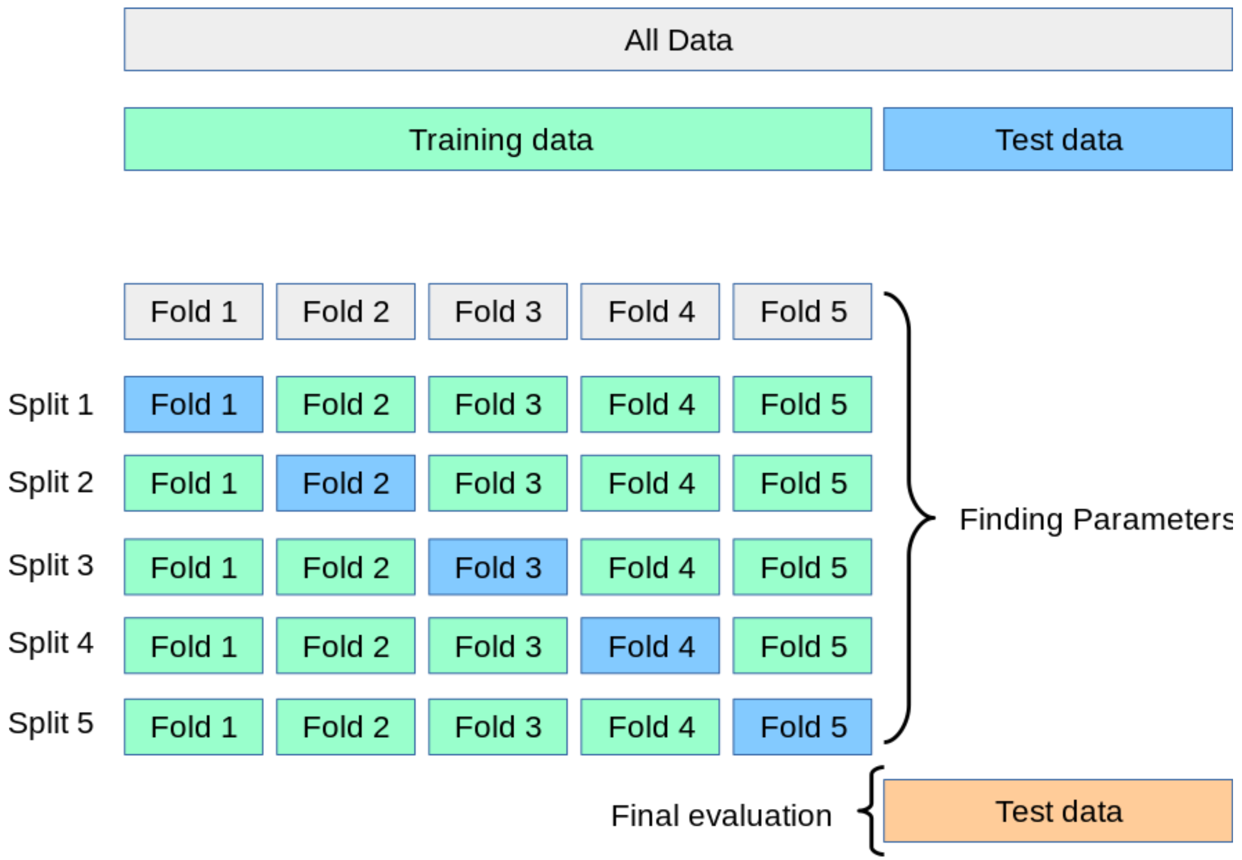


Diagram adapted from the slides by Imperial College London's Introduction to Machine Learning course

Alternative: Cross-validation

- K-fold cross validation splits data into **K folds** (without overlap)
- In each iteration, one of the folds is used as test data and the rest as training + validation data
- Provide us with average accuracy of the model and parameters that perform the best on average



When should you use each method?

If we have a lot of data available

Hold out method might be appropriate

Not a lot of data

Utilizing cross validation might be a good choice

How much should be in a training net and test set?

Normally, Train-Validation-Test ratio is 70%-15%-15% or 60%-20%-20% or 80%-10%-10%

What if we have millions of data records?

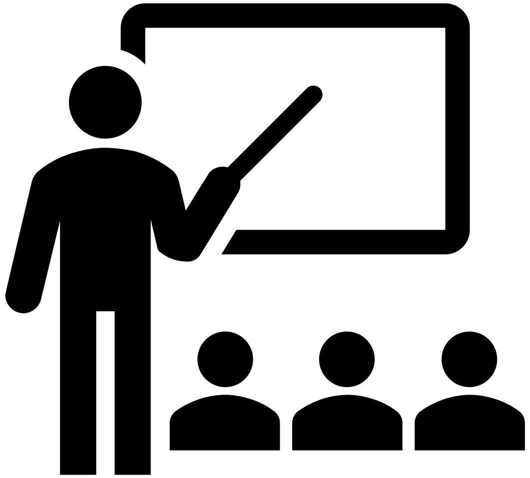
98%-1%-1% might even be ok for image recognition if 1% test data can be representative

That being said, the ratio also depends on data size and types of applications

Model Evaluation Metrics

The Situation

- You're hiring a 3rd party contractor to develop a machine learning model for your organizations (for this purpose, let's say it's a classification problem).
- The contractor go and develop model for a while, then come back and claim their model can achieve a **95%** accuracy on the problem assigned.



Is this model good?

Their Confusion Matrix: What’s Wrong?

True Label

Predicted Label

	Positive	**Negative**
Positive	0%	2%
Negative	3%	95%

What if

- The data is skewed (majority of people is in one class)
  - Ex. Fraud detection – very small fraction of people commits fraud
- We only care about one type of prediction
  - Ex. Advertisement target – the main concern is whether we can advertise successfully or not
- Is accuracy an appropriate metric?
- How do we evaluate the model?

Getting a Clearer Picture: Confusion Matrix

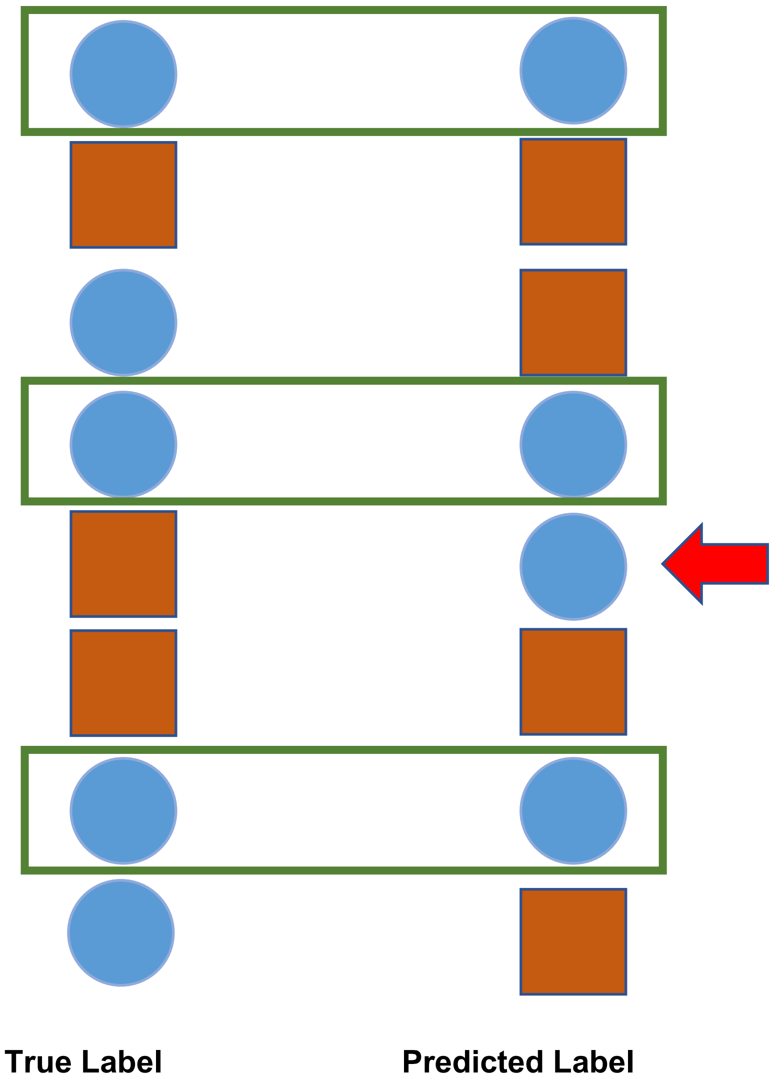
- For a binary classification problem, there are 4 possible outcomes for a prediction.

True Label

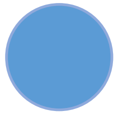
Predicted Label

	Positive	Negative
Positive	True Positive	False Positive
Negative	False Negative	True Negative

Getting a Clearer Picture: Confusion Matrix



- For a binary classification problem, there are 4 possible outcomes for a prediction.


Target of Interest (True)

True Label

Predicted Label	Positive	Negative
Positive	3	1
Negative	2	2

Precision

- Precision is a fraction of target predictions predicted correctly.
- In other words, when we consider our target class to be Positive, "how many of our **<u>positive predictions</u>** are correct?"

- $\boldsymbol{Precision = \frac{TP}{TP+FP}}$

$$\textbf{Accuracy} = ?$$
$$\textbf{Precision} = ?$$

True Label

Predicted Label

	Positive	Negative
Positive	3	1
Negative	2	2

Precision

- Precision is a fraction of target predictions predicted correctly.
- In other words, when we consider our target class to be Positive, "how many of our **positive predictions** are correct?"

$$\mathbf{Accuracy} = \frac{\mathbf{3+2}}{\mathbf{8}} = \frac{\mathbf{5}}{\mathbf{8}} = \mathbf{62.5\%}$$

$$\mathbf{Precision} = \frac{\mathbf{3}}{\mathbf{3+1}} = \frac{\mathbf{3}}{\mathbf{4}} = \mathbf{75\%}$$

- $\boldsymbol{Precision = \frac{TP}{TP+FP}}$

True Label

Predicted Label

	Positive	**Negative**
Positive	3	1
Negative	2	2

Recall

- Recall is a fraction of target class predicted correctly.
- In other words, when we consider our target class to be Positive, “how many of **positive labels** did we manage to predict correctly?”

$$\mathbf{Recall} = ?$$

- $Recall = \frac{TP}{TP+FN}$

True Label

Predicted Label	Positive	Negative
Positive	3	1
Negative	2	2

Recall

- Recall is a fraction of target class predicted correctly.
- In other words, when we consider our target class to be Positive, "how many of **positive labels** did we manage to predict correctly?"

- $Recall = \frac{TP}{TP+FN}$

$$\mathbf{Recall} = \frac{\mathbf{3}}{\mathbf{3+2}} = \frac{\mathbf{3}}{\mathbf{5}} = \mathbf{60\%}$$

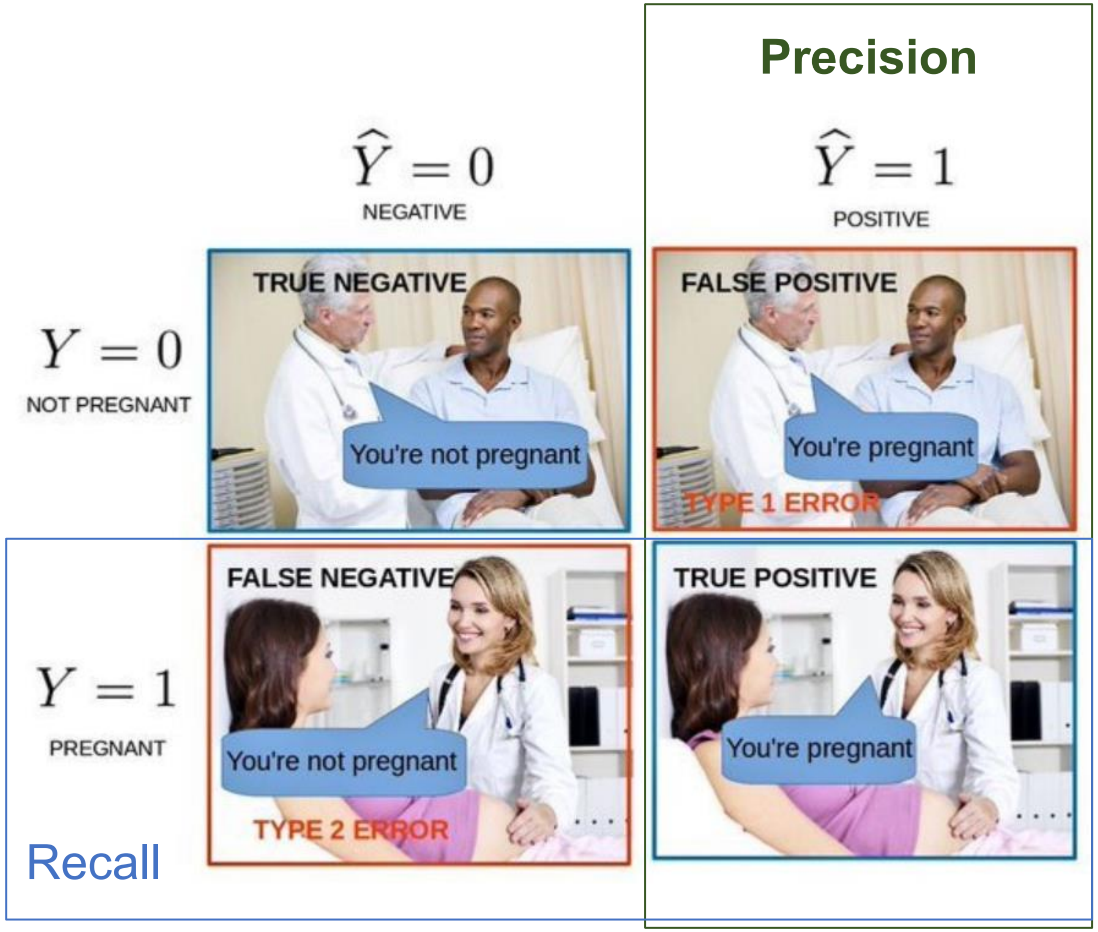
True Label

Predicted Label

	Positive	Negative
Positive	3	1
Negative	2	2

Precision vs. Recall

- Should we focus on precision or recall?
- Case 1: We're building a model to detect cancer?
  - If we focus on precision, are we taking a chance of letting cancer patients go untreated?
  - If we focus on recall, are we wasting people money and scaring them unnecessarily?
- Case 2: We're building a model to predict the likelihood that a loan applicant will pay back the loan on time?
  - If we focus on precision, are we missing out on income opportunities?
  - If we focus on recall, are we making poor decisions?
- A lot of times, this kind of decision is hard to judge



F1 (and F-Beta)

- F1 score is computed as a harmonic mean of precision and recall

$$F1 = 2 \times \frac{Precision \times Recall}{Precision + Recall}$$

  - It gives precision and recall equal importance.

- What if we don't weight each precision and recall equally?

$$F_\beta = (1 + \beta^2)\frac{Precision \times Recall}{\beta^2 Precision + Recall}$$

  - β is a measure specifying that Recall is β times as important as Precision

- Getting the right weight for β is hard.

Evaluating cost-benefit: Expected Value

- The **expected value (EV)** is the anticipated outcome value of a situations.
- EV is computed by calculated a weighted average of all possible outcome value, i.e. it is a sum of outcome values weighted by their respective probability

$$EV = prob(o_1) \times value(o_1) + prob(o_2) \times value(o_2) + \cdots$$

  - where o_1, o_2, … are possible outcomes of a situation
- The probability of each outcome can usually be approximated from data
- The values of outcomes are often harder to estimate and may require specific business domain knowledge
- Expected value can be a useful tool for choose appropriate model for the job.

Example: Target Marketing

- A company is trying to perform targeted marketing and offer a product to consumers. If a customer buys a product, the company will gain $100 of profits. However, each product offer made to a customer will cost the company $1.
- The company then builds a model to predict if a customer will buy the product.
- In this situation, the 4 possible outcomes of the predictions are as follow.
  - True Positive – a product is offered to a customer who will buy it. Profit = $100-$1 = $99
  - False Positive – a product is offered to a customer who will not buy it. Profit (loss) = -$1
  - True Negative – a product is not offered to a customer who will not buy it. Profit = $0
  - False Negative – a product is not offered to a customer who will buy it. Profit = $0
    - This is a case of a missed opportunity, but for simplicity, we will not consider that here.

Example: Target Marketing (Cont.)

- Assuming the prediction result is as shown.

True Label

Predicted Label	Positive	Negative
Positive	56	7
Negative	5	42

- Using $EV = prob(o_1) \times value(o_1) + prob(o_2) \times value(o_2) + \cdots$

Note that the result differs from the book as the number here is not rounded until the end

Example: Target Marketing (Cont.)

- Assuming the prediction result is as shown.

True Label

Predicted Label	Positive	Negative
Positive	56	7
Negative	5	42

Total # of test data =

56+7+5+42= 110

- Using $EV = prob(o_1) \times value(o_1) + prob(o_2) \times value(o_2) + \cdots$
- The expected value of the model (on this test data) is then

$$EV = \frac{56}{110}(99) + \frac{7}{110}(-1) + \frac{42}{110}(0) + \frac{5}{110}(0) = \$50.34$$

Note that the result differs from the book as the number here is not rounded until the end

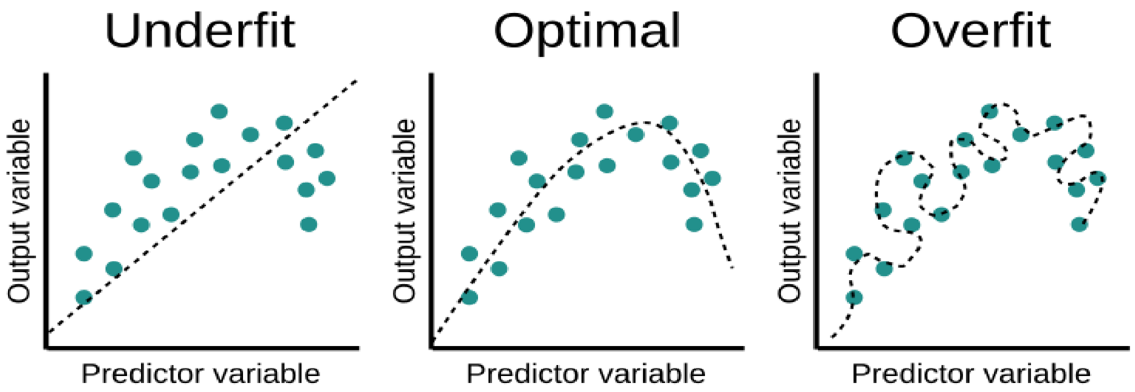
Overfitting vs. Underfitting

Overfitting

Overfitting occurs when the trained model adjust its parameters to fit the training data too well and became unable to generalize well. This behavior is observed when the model performs well with during the development with training data but performs poorly on the unseen data.

Underfitting

Underfitting happens when the trained model is unable to adjust itself to sufficiently fit the training data, resulting in poor performances overall. This behavior is observed when the model is unable to perform well even during the development with training data.



Overfitting vs. Underfitting

Possible Solution for Overfitting	Possible Solution for Underfitting
Collects more data	Increase the model complexity
Reduce number of features	Add more features
Adjust parameters to increase regularization effect	Adjust parameters to reduce regularization effect
etc.	etc.

Thank You

GBDi
Government Big Data Institute
สถาบันส่งเสริมการวิเคราะห์และบริหารข้อมูลขนาดใหญ่ภาครัฐ (สวข.)

Follow us on

GBDi

gbdi.depa.or.th

Facebook

Twitter

Blockdit

govbigdata

YouTube

Government Big Data Institute (GBDi)

Line Official

@gbdi

BIG DATA THAILAND

Follow us on

BIG DATA THAILAND

Website

Facebook

Blockdit

Twitter