

ARTIFICIAL INTELLIGENCE



# INTRODUCTION TO MACHINE LEARNING

Rathachai Chawuthai



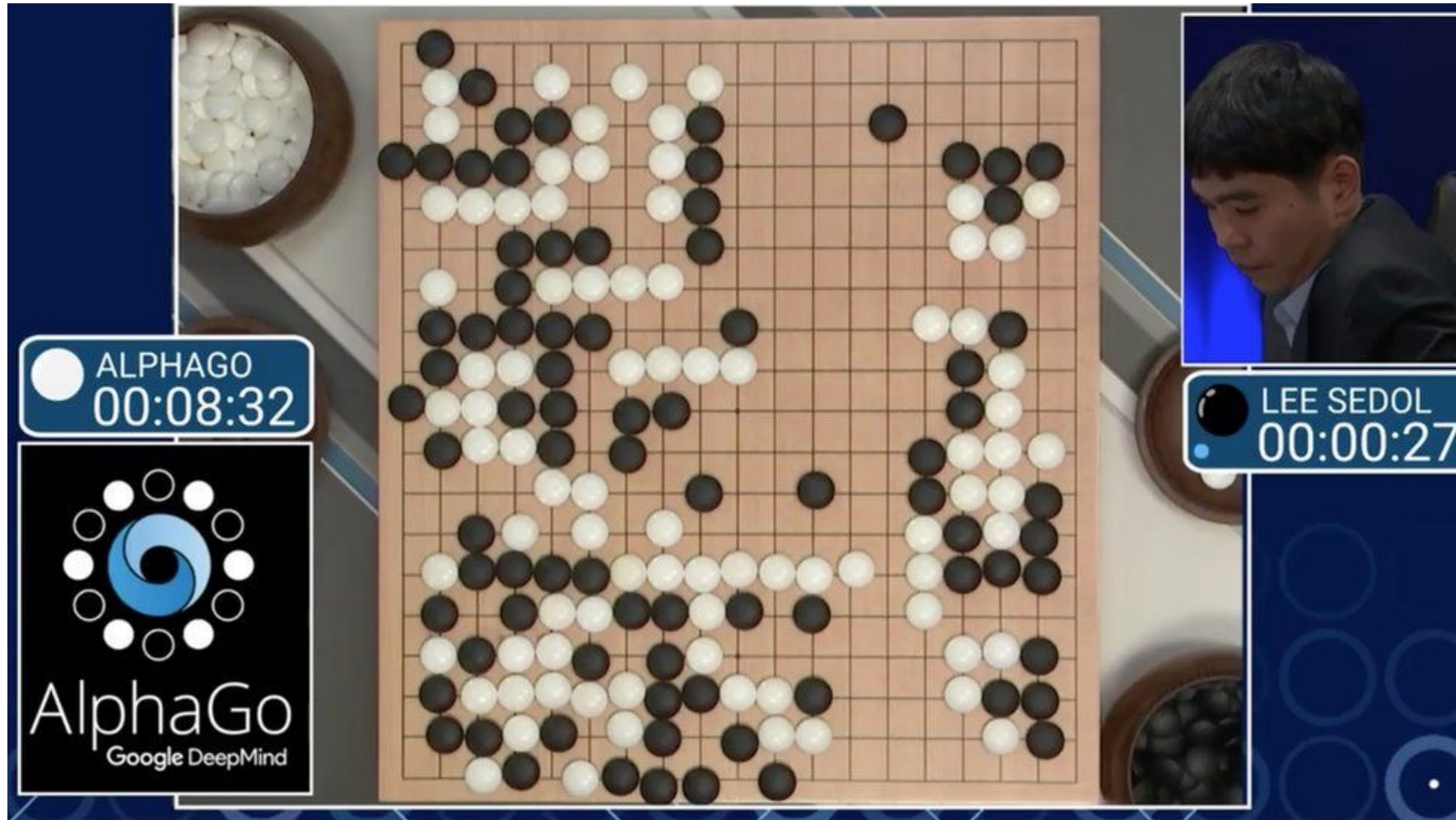
rathachai.creatier.pro

# Agenda

- Introduction to Machine Learning
- Model
- Evaluation
- Learning Steps
- Regression
- Classification

# INTRODUCTION TO MACHINE LEARNING

# Google's AlphaGo beats Go master (2016)



Ref: <https://www.bbc.com/news/technology-35785875>

# Self-Driving Car



Ref: <https://gcn.com/articles/2020/10/23/autonomous-vehicles-public-health.aspx>

# Voice Recognition



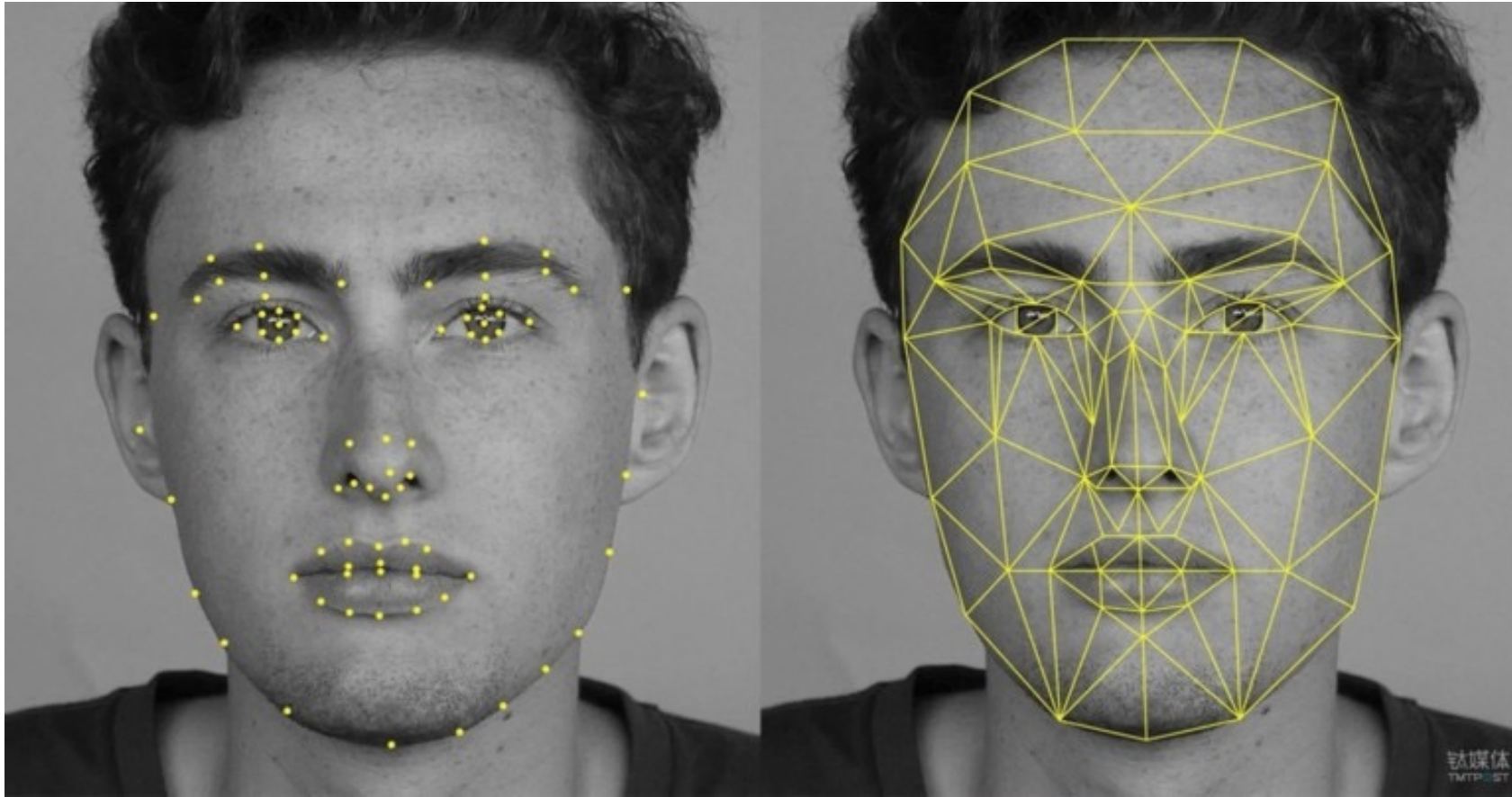
Ref: <https://www.cio.com/article/3239924/cios-listen-up-voice-recognition-meets-the-printer.html>

# Object Detection



Ref: <https://hackernoon.com/how-visual-object-detection-can-transform-manufacturing-industries-8b6698cc0a47>

# Face Detection



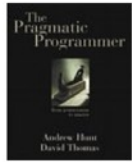
Ref: <https://rapidapi.com/blog/top-facial-recognition-apis/>

# Recommender System

## Frequently Bought Together



+



Total price: **\$83.09**

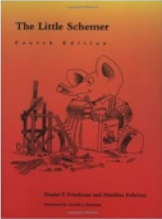
Add both to Cart

Add both to List

- ☒ **This item:** Structure and Interpretation of Computer Programs - 2nd Edition (MIT Electrical Engineering and... by Harold Abelson Paperback **\$50.50**
- ☒ The Pragmatic Programmer: From Journeyman to Master by Andrew Hunt Paperback **\$32.59**

## Customers Who Bought This Item Also Bought

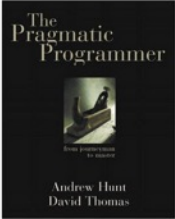
Page 1 of 13



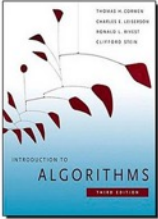
**The Little Schemer - 4th Edition**  
› Daniel P. Friedman  
★★★★☆ 64  
Paperback  
**\$36.00** ✓Prime



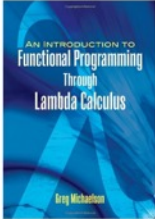
**Instructor's Manual t/a Structure and Interpretation of Computer Programs...**  
› Gerald Jay Sussman  
★★★★☆ 5  
Paperback  
**\$28.70** ✓Prime



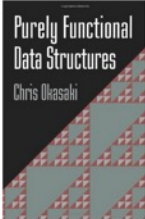
**The Pragmatic Programmer: From Journeyman to Master**  
› Andrew Hunt  
★★★★☆ 328  
Paperback  
**\$32.59** ✓Prime



**Introduction to Algorithms, 3rd Edition (MIT Press)**  
› Thomas H. Cormen  
★★★★☆ 313  
**#1 Best Seller** in Computer Algorithms  
Hardcover  
**\$66.32** ✓Prime



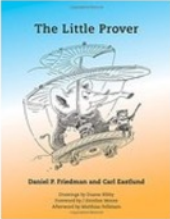
**An Introduction to Functional Programming Through Lambda...**  
› Greg Michaelson  
★★★★☆ 23  
Paperback  
**\$20.70** ✓Prime



**Purely Functional Data Structures**  
› Chris Okasaki  
★★★★☆ 19  
Paperback  
**\$40.74** ✓Prime



**Code: The Hidden Language of Computer Hardware and Software**  
› Charles Petzold  
★★★★☆ 334  
**#1 Best Seller** in Machine Theory  
Paperback  
**\$17.99** ✓Prime



**The Little Prover (MIT Press)**  
› Daniel P. Friedman  
★★★★☆ 4  
Paperback  
**\$31.78** ✓Prime



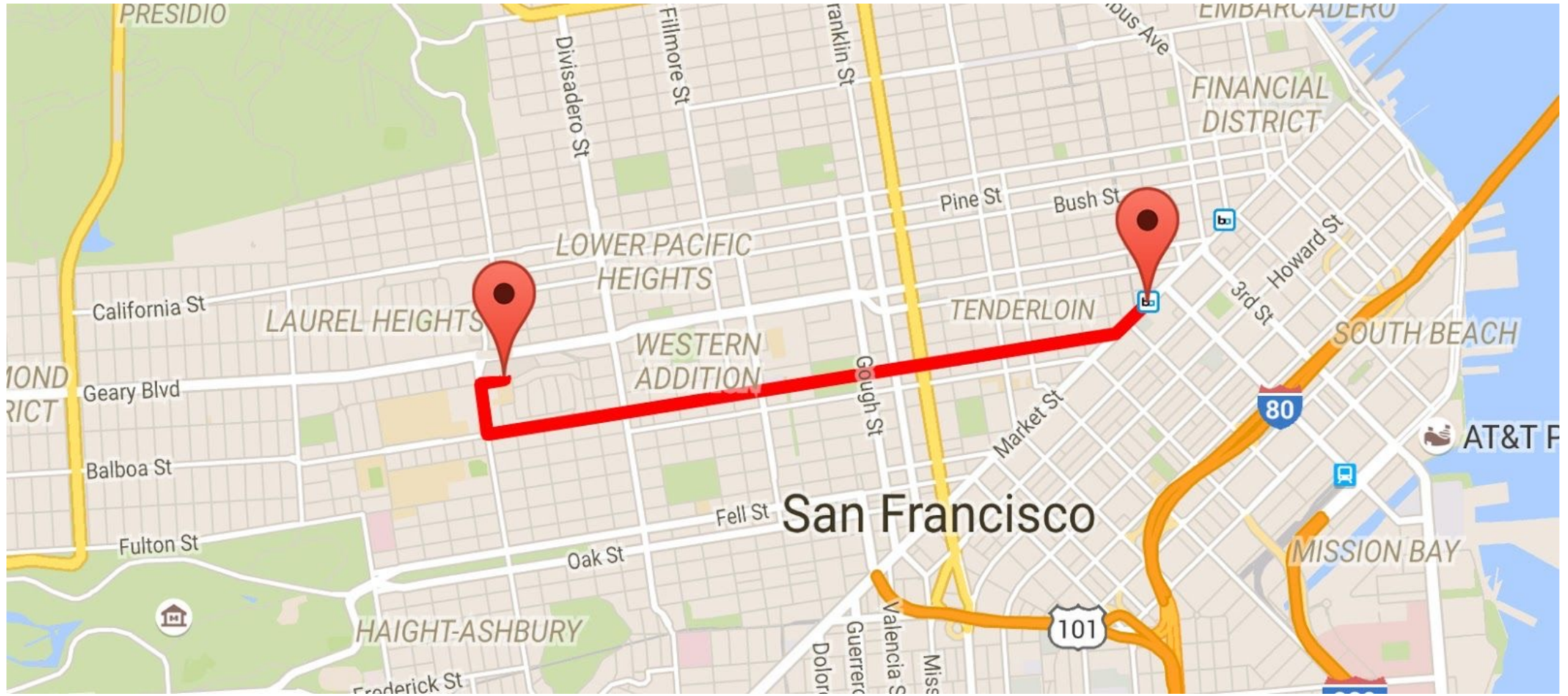
Ref: <https://madasamy.medium.com/introduction-to-recommendation-systems-and-how-to-design-recommendation-system-that-resembling-the-9ac167e30e95>

# Chatbot



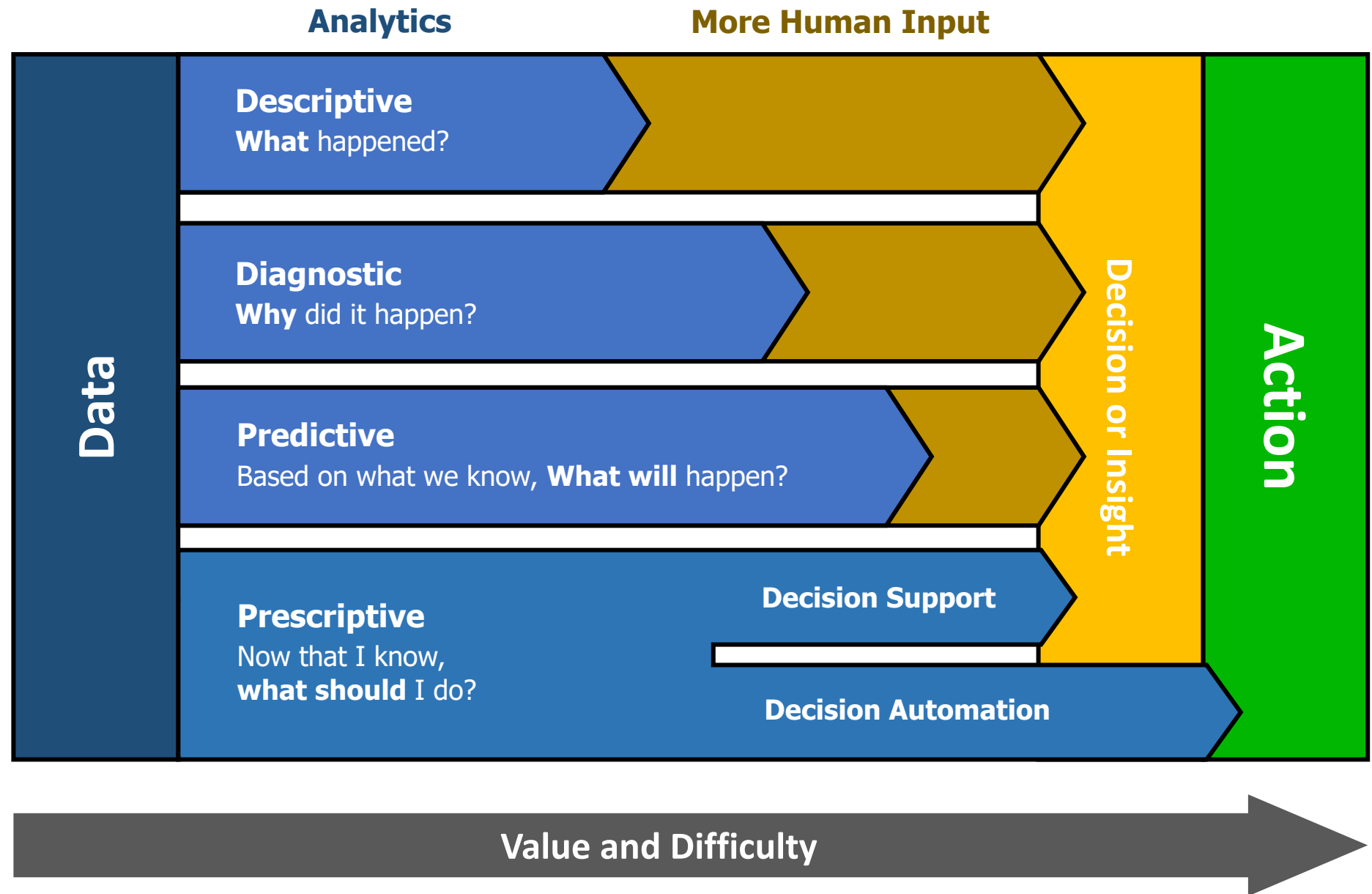
Ref: <https://700-apps.com/chatbot/>

# Map Direction

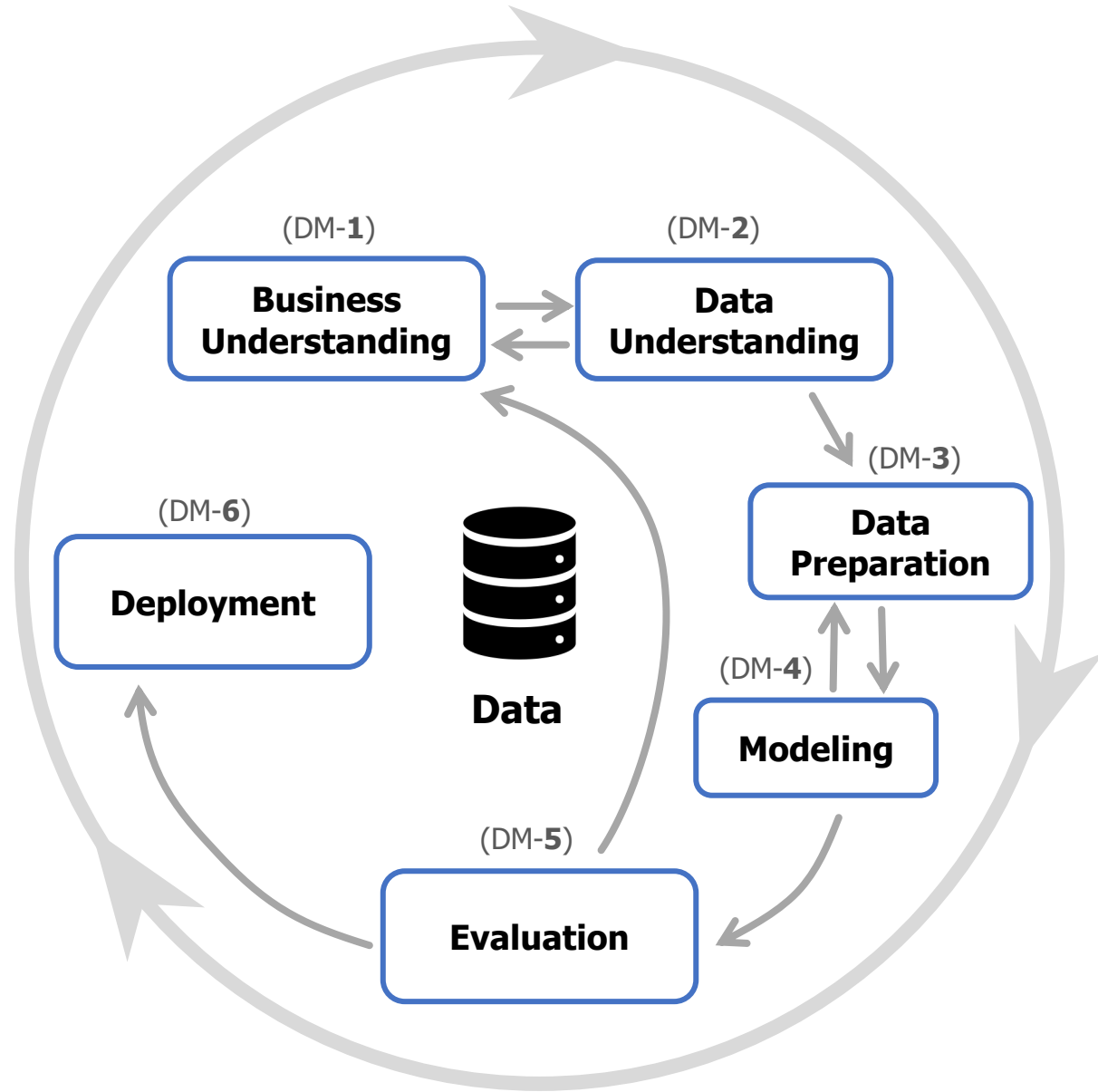


Ref: <https://akexorcist.dev/google-direction-library-for-android-th/>

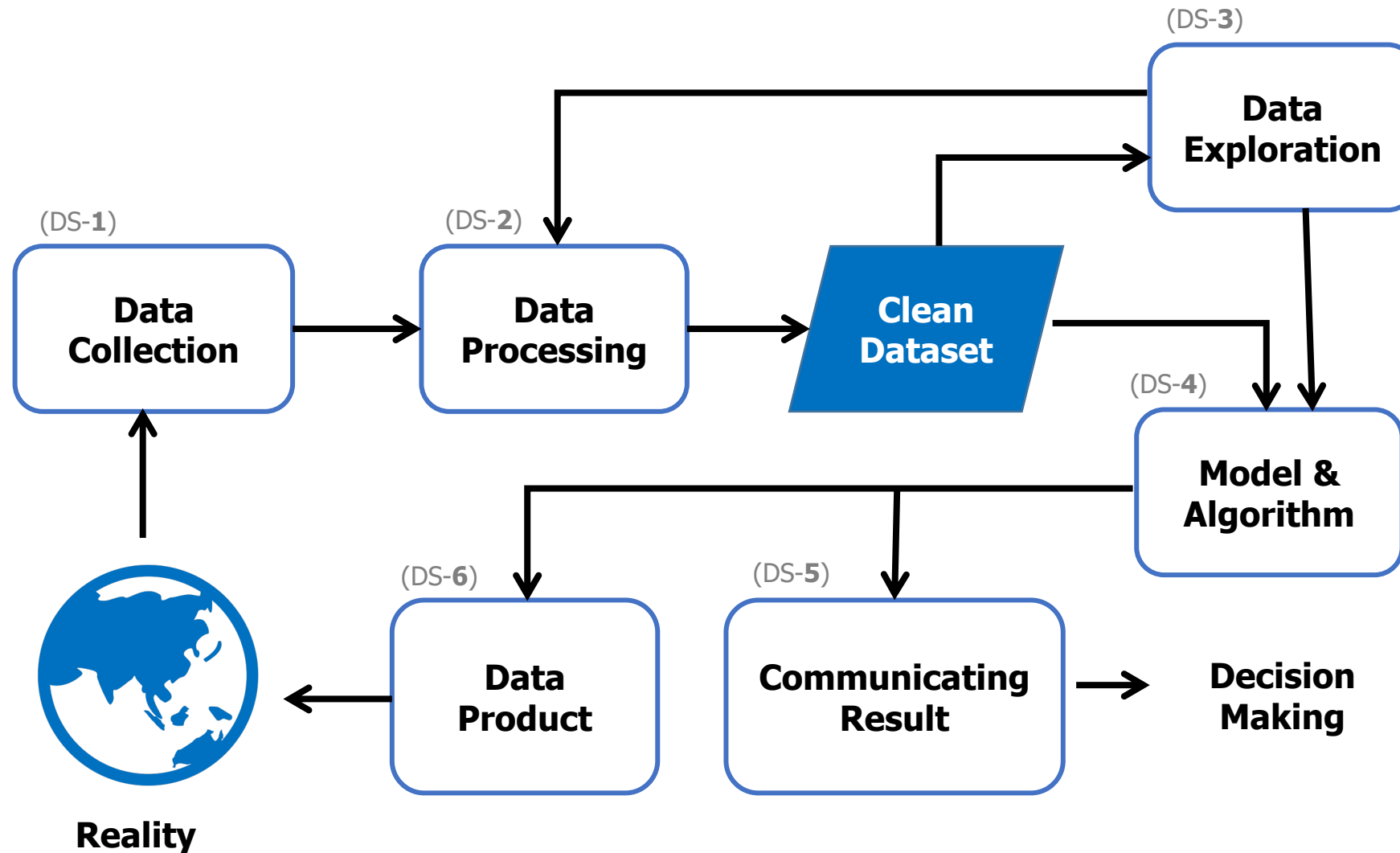
# Data Analytics



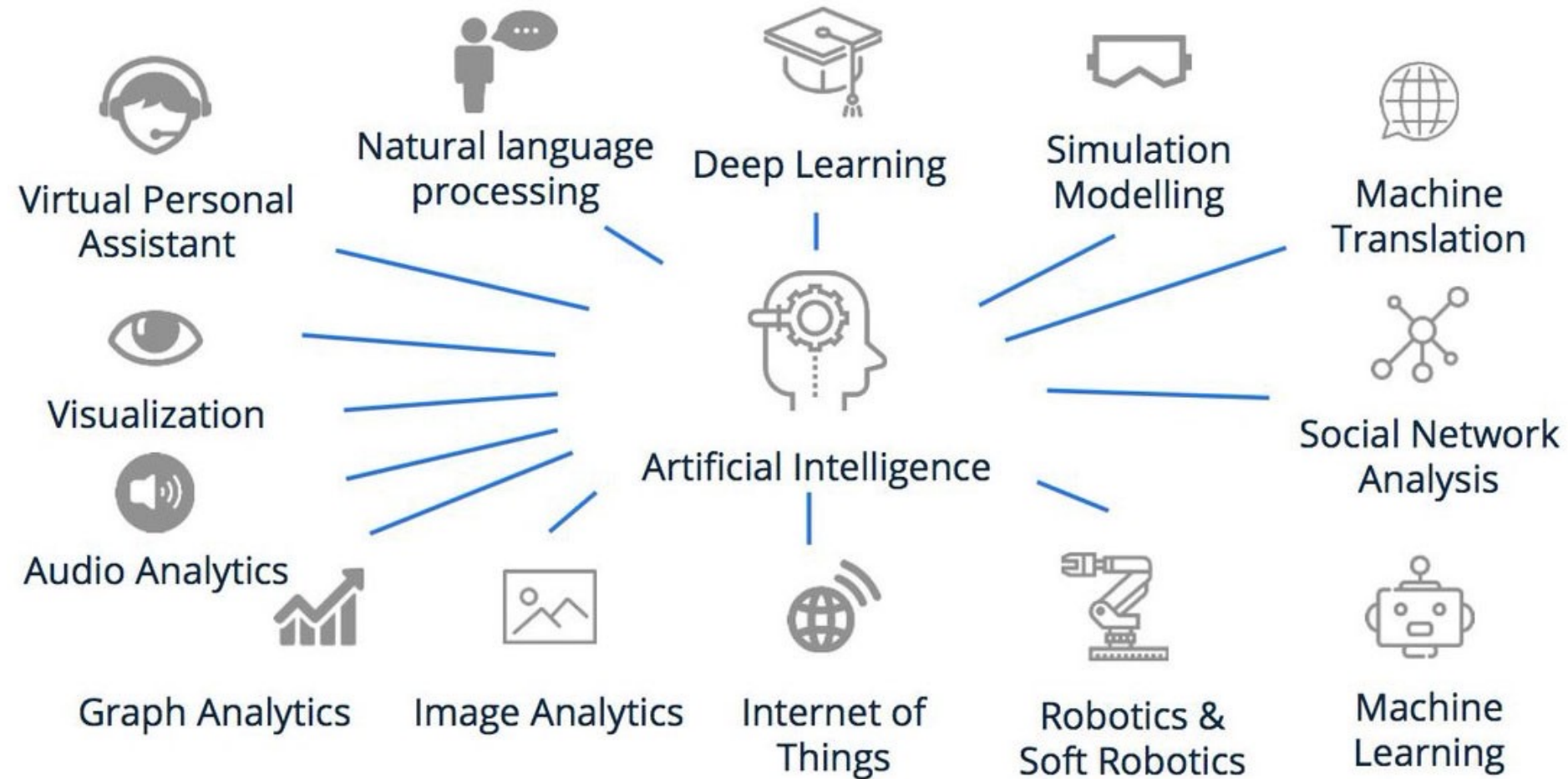
# Cross-Industry Standard Process for Data Mining: CRISP-DM



# Data Science Process

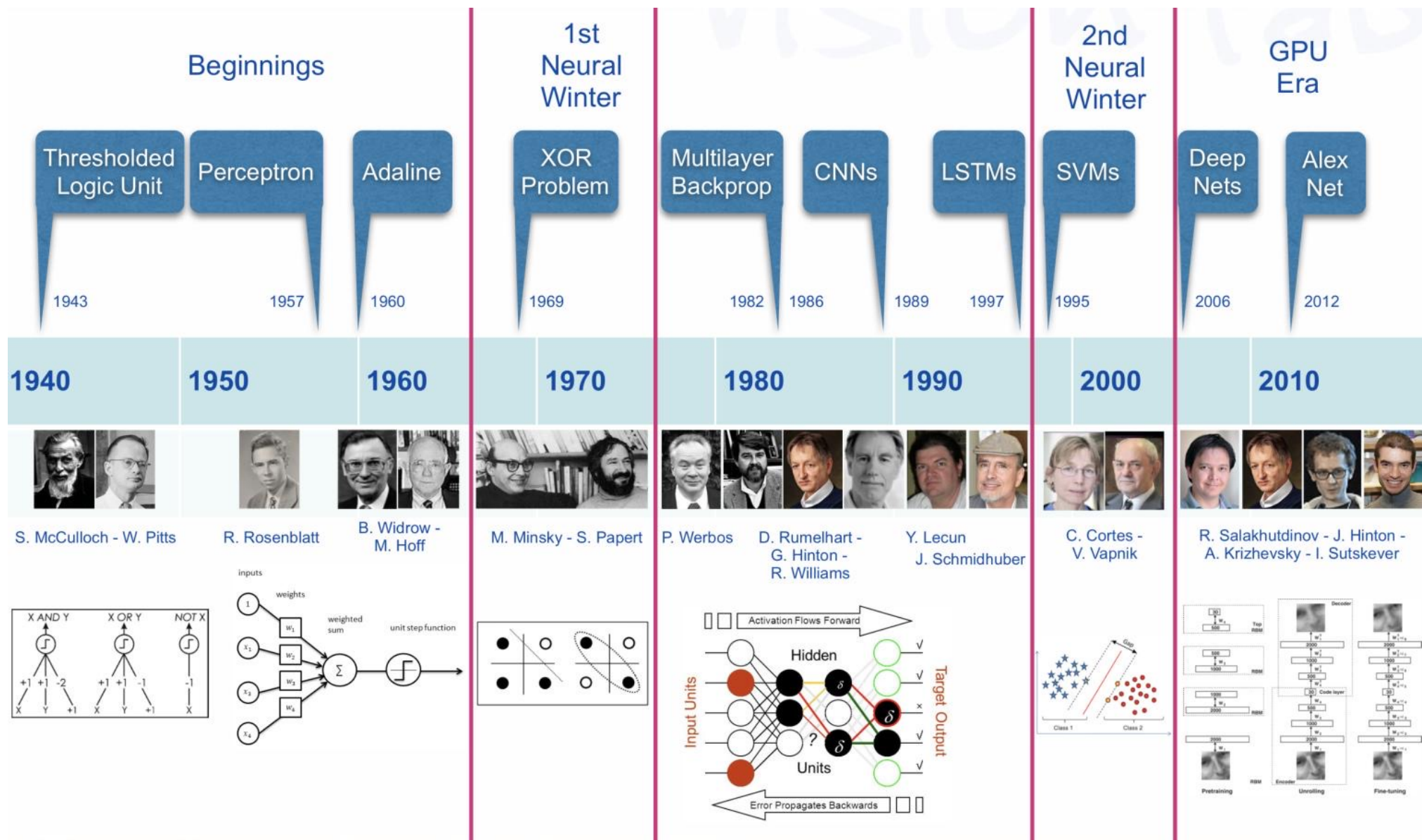


# AI



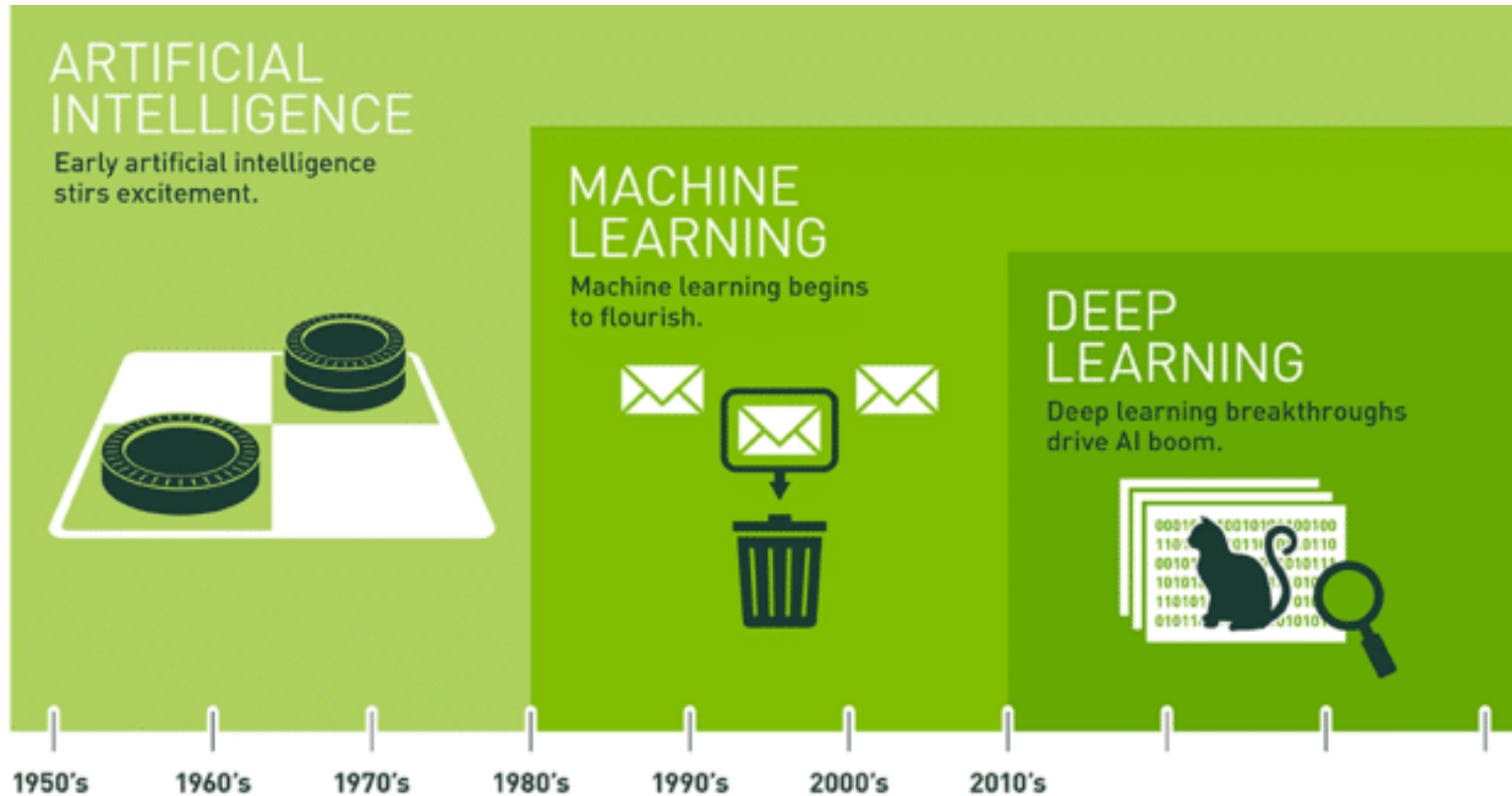
Ref: <https://medium.com/finwintech/artificial-intelligence-ai-trends-2020-a39efc8a424>

# History of AI



Ref: [https://www.reddit.com/r/artificial/comments/d4pm74/a\\_brief\\_history\\_of\\_ai\\_from\\_1940s\\_till\\_today\\_image/](https://www.reddit.com/r/artificial/comments/d4pm74/a_brief_history_of_ai_from_1940s_till_today_image/)

# AI & Machine Learning & Deep Learning



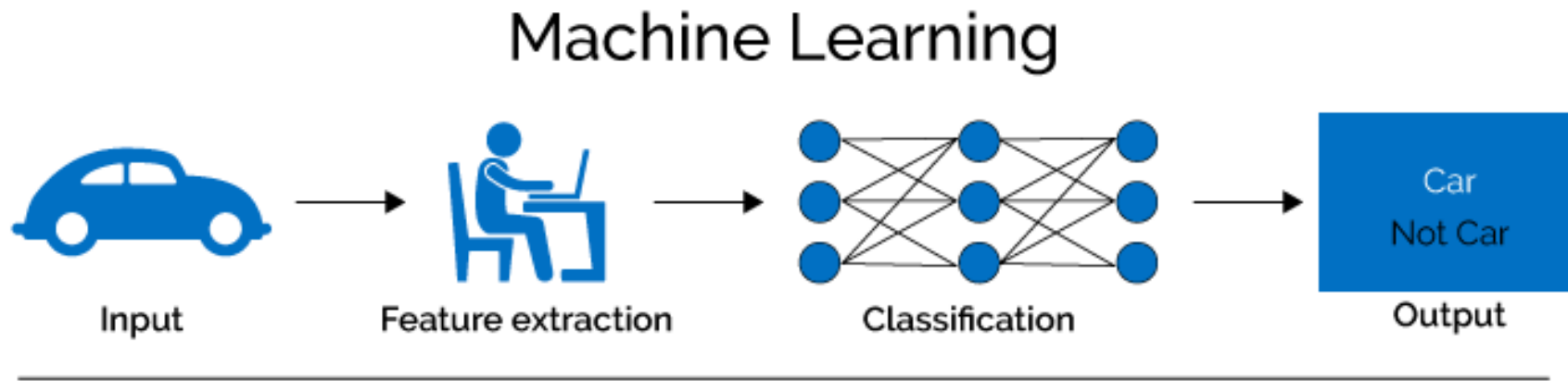
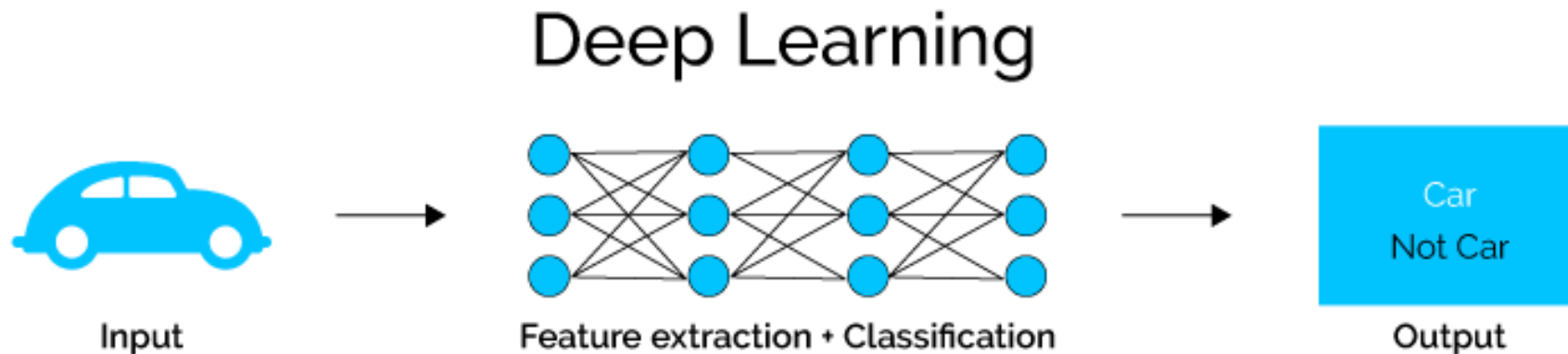
Ref: <https://blogs.nvidia.com/blog/2016/07/29/whats-difference-artificial-intelligence-machine-learning-deep-learning-ai/>

# AI & Machine Learning & Deep Learning

- Artificial Intelligence — Human Intelligence Exhibited by Machines
- Machine Learning — An Approach to Achieve Artificial Intelligence
  - Machine learning at its most basic is the practice of using algorithms to parse data, learn from it, and then make a determination or prediction about something in the world
- Deep Learning — A Technique for Implementing Machine Learning
  - Using the architecture of Artificial Neural Network

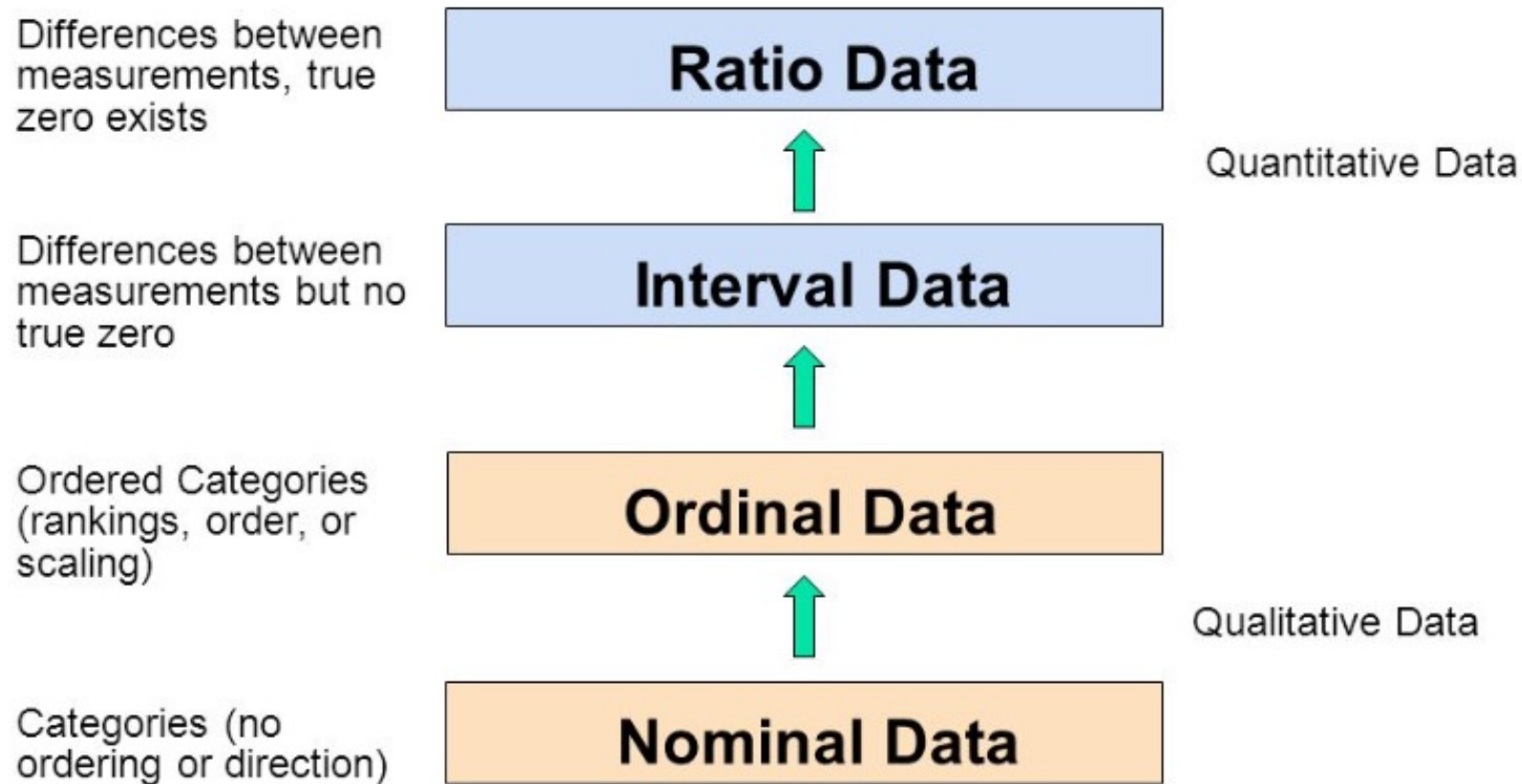
Ref: <https://blogs.nvidia.com/blog/2016/07/29/whats-difference-artificial-intelligence-machine-learning-deep-learning-ai/>

# Machine Learning & Deep Learning



Ref: <https://semiengineering.com/deep-learning-spreads/>

# Scale of Data

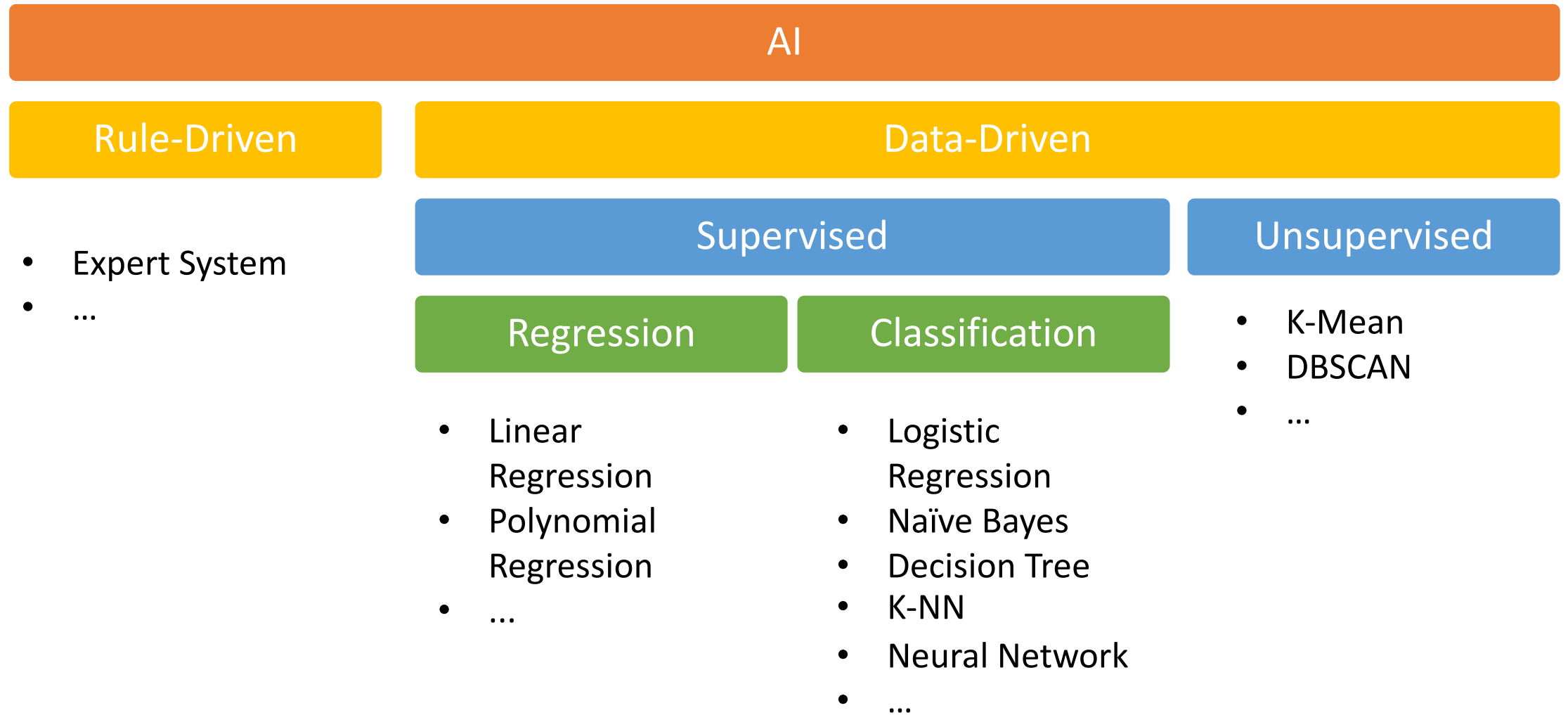


Ref: <https://www.graphpad.com/support/faq/what-is-the-difference-between-ordinal-interval-and-ratio-variables-why-should-i-care/>

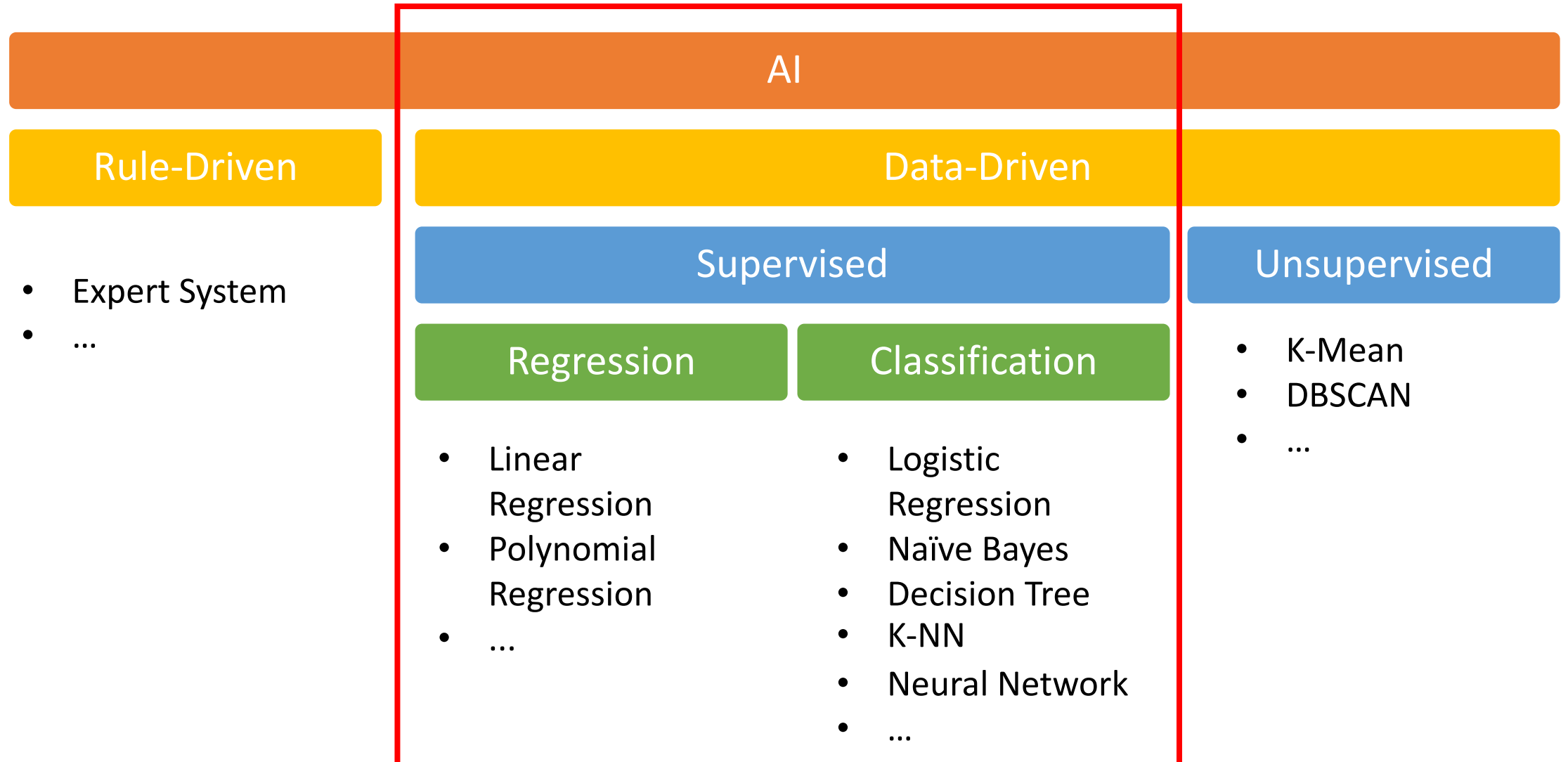
# Types of Data

- Quantitative Data
  - Information that can be measured and written down with numbers. (27, 33, 41)
  - e.g. number of things, temperature, height, age, time, ...
- Ordinal Data
  - Data type consisting of numerical scores that exist on an ordinal scale. (small, medium, large, enormous)
  - e.g. ranking, rating, ordering, ...
- Categorical Data
  - Types of data which may be divided into groups. (A, B, C)
  - e.g. gender, classification, educational level, ...

# AI Techniques



# AI Techniques



# Model

# What are Model?

A

Linear Regression

B

$$y = mx + c$$

C

$$y = 5x - 7$$

D

Gradient Decents

# What are Model?

A

Linear Regression

B

$$y = \beta_0 + \beta_1 x$$

C

$$f(x) = 5x - 7$$

D

$$y = 3 * x$$

# What are Model?

A

Decision Tree

B

$$f(w, h) = \begin{cases} car, & (w = 4) \wedge (h < 2) \\ truck, & (w = 4) \wedge (h \geq 4) \end{cases}$$

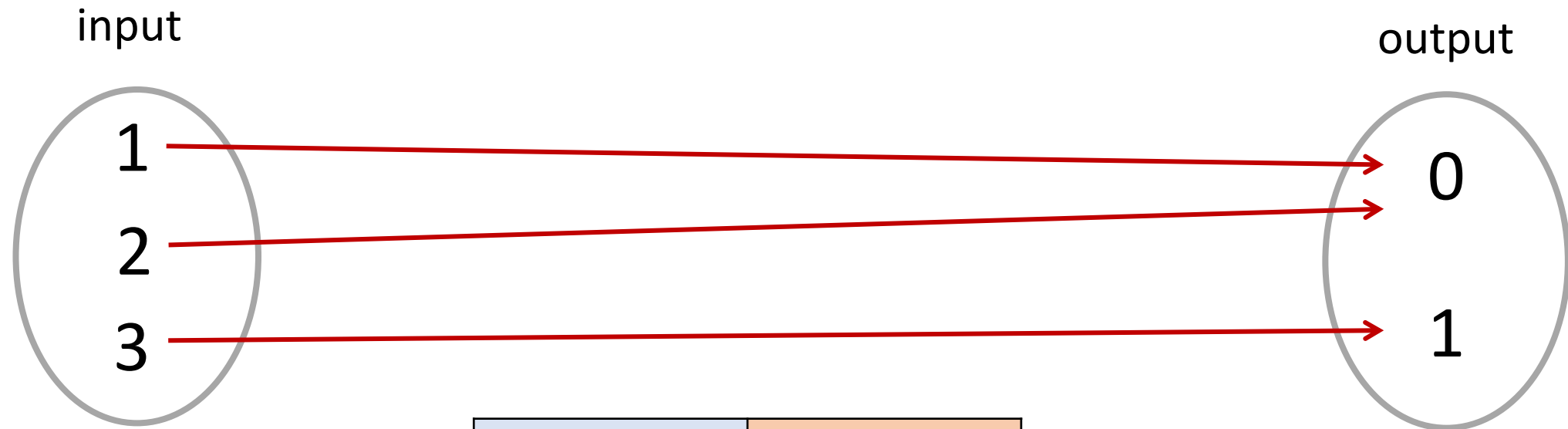
C

C3

D

```
def get_veh_type(w,h)
  if (w==4)&(h<2):
    return "car"
  else:
    return "truck"
```

# Function Mapping



| x | y |
|---|---|
| 1 | 0 |
| 2 | 0 |
| 3 | 1 |

# Model?

| x | y |
|---|---|
| 1 | 0 |
| 2 | 0 |
| 3 | 1 |

$$y = \max(x - 2, 0)$$

# Model?

| x | y |
|---|---|
| 1 | 0 |
| 2 | 0 |
| 3 | 1 |

$$y = \text{round}\left(\frac{x}{5}\right)$$

# Model?

| x | y |
|---|---|
| 1 | 0 |
| 2 | 0 |
| 3 | 1 |

$$y = \begin{cases} 0, & (x < 3) \\ 1, & (x = 3) \end{cases}$$

# Model?

| x | y |
|---|---|
| 1 | 0 |
| 2 | 0 |
| 3 | 1 |

```
if (x==1)or(x==2):  
    y=0  
elif x==3:  
    y=1
```

# Model?

| x | y |
|---|---|
| 1 | 0 |
| 2 | 0 |
| 3 | 1 |

```
if x<=2:  
    y=0  
else:  
    y=1
```

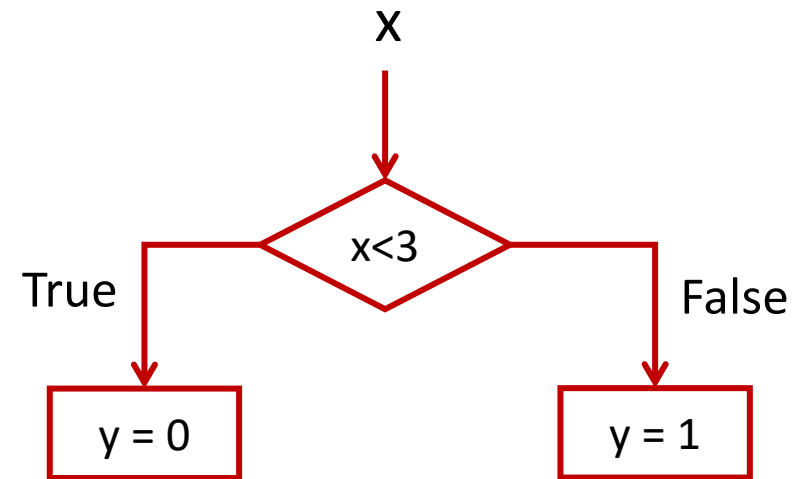
# Model?

| x | y |
|---|---|
| 1 | 0 |
| 2 | 0 |
| 3 | 1 |

$$y = \frac{1}{1 + e^{-(10x-25)}}$$

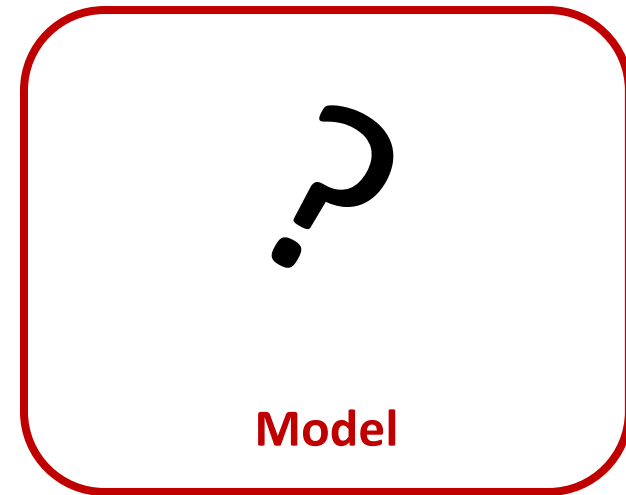
# Model?

| x | y |
|---|---|
| 1 | 0 |
| 2 | 0 |
| 3 | 1 |



# Model?

| $x_1$ | $x_2$ | $x_3$ | ... | $x_n$ | $y$ |
|-------|-------|-------|-----|-------|-----|
| 1     | 5.2   | FB    | ... | XL    | 0   |
| 2     | 1.1   | TW    | ... | M     | 0   |
| 3     | -9.6  | IG    | ... | L     | 1   |



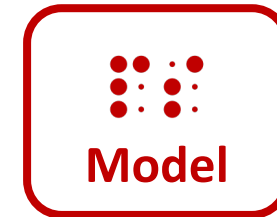
# Supervised Learning

input      output

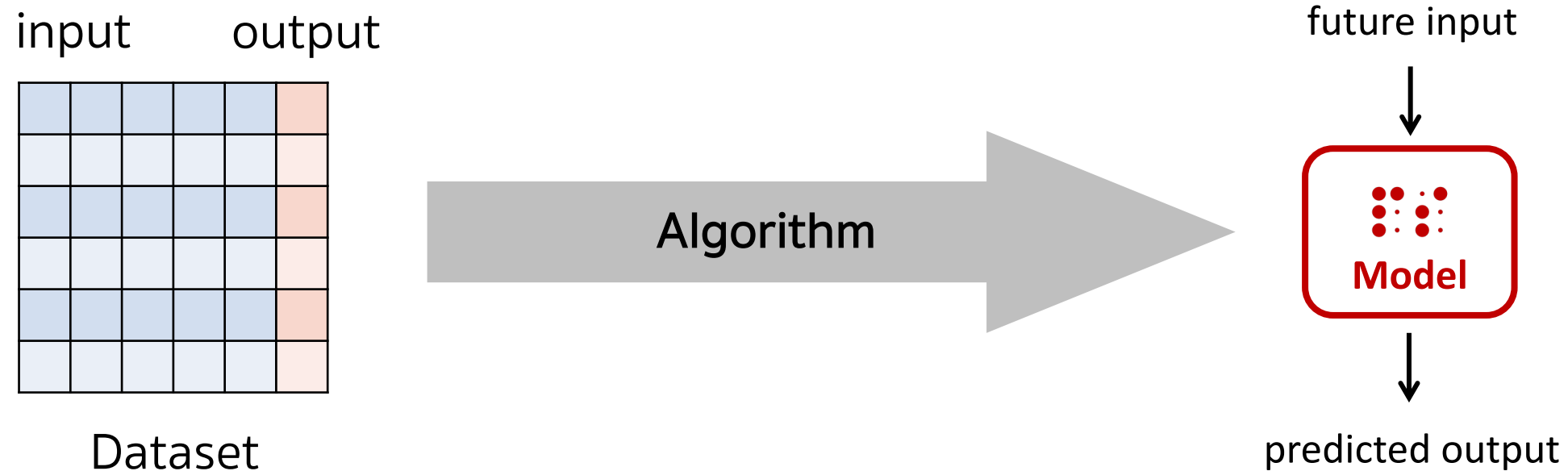
|  |  |  |  |  |  |
|--|--|--|--|--|--|
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |

Dataset

Algorithm

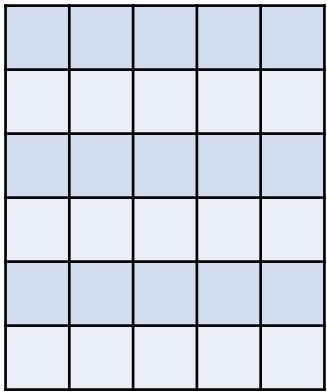


# Supervised Learning



# Unsupervised Learning

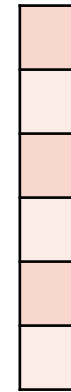
input



Dataset



output



# Demo

- Dataset
  - Travel or Not? (Classification)
  - Condominium Prices (Regression)
  - [https://docs.google.com/spreadsheets/d/1\\_xCC-V8FDW7bm1aRH7rem7FKq4-9Ly3-Daupv910B1w](https://docs.google.com/spreadsheets/d/1_xCC-V8FDW7bm1aRH7rem7FKq4-9Ly3-Daupv910B1w)
- Colab
  - <https://colab.research.google.com/drive/1H9f0Jdr81leov10np5AeGSPUd-RqRzX6>



# EVALUATION

# Evaluation

- Regression : Evaluation
- Classification : Evaluation

# Model

- Technique

- $y = mx + c$

- Model

- $y = 5x + 10$

- $y = -3x + 12$

- $y = 100x - 400$

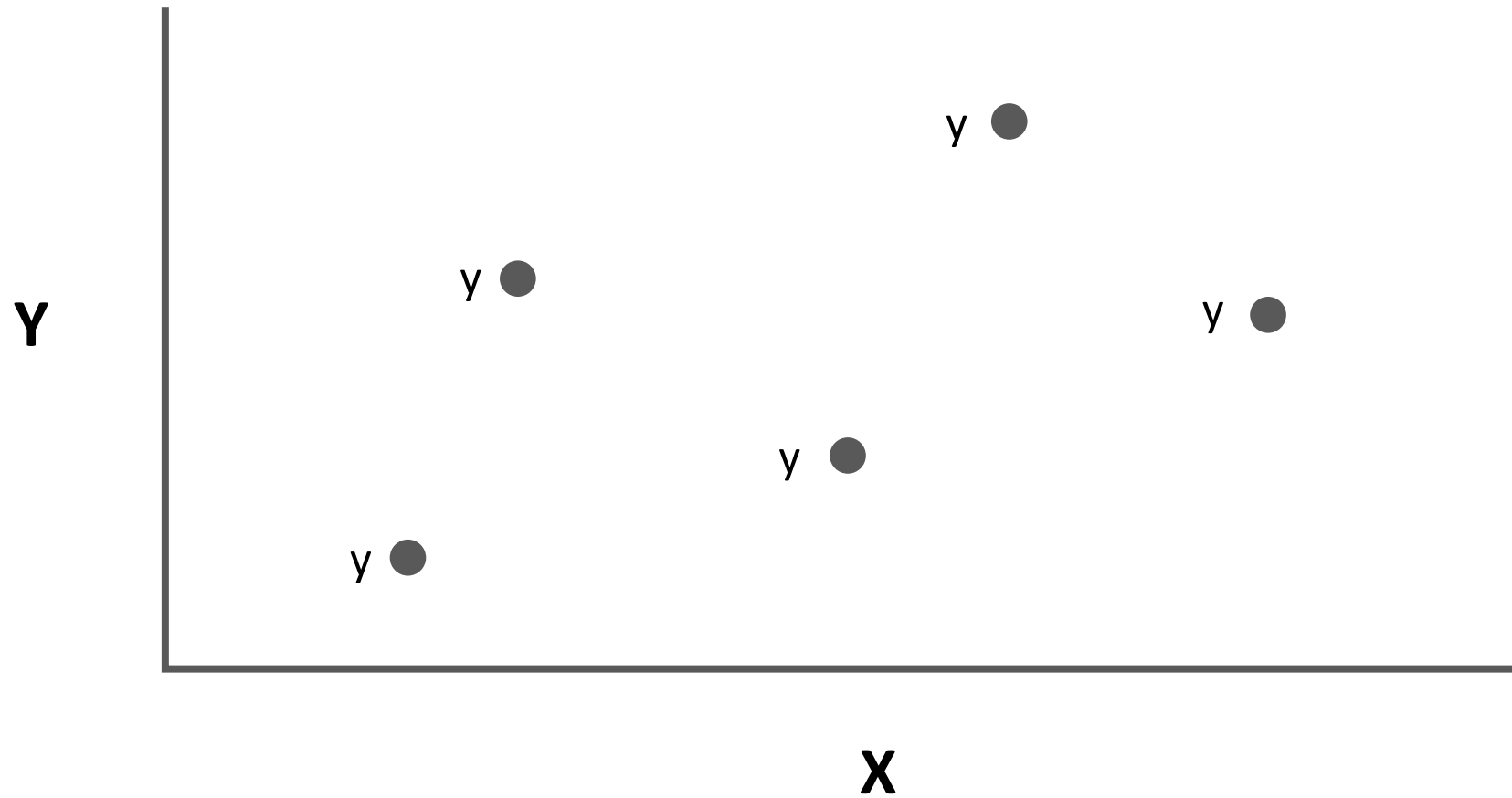
# Model Performance

- Regression
  - **RMSE**
  - MAPE
  - MAE
- Classification
  - **Accuracy**
  - Precision
  - Recall
  - F1

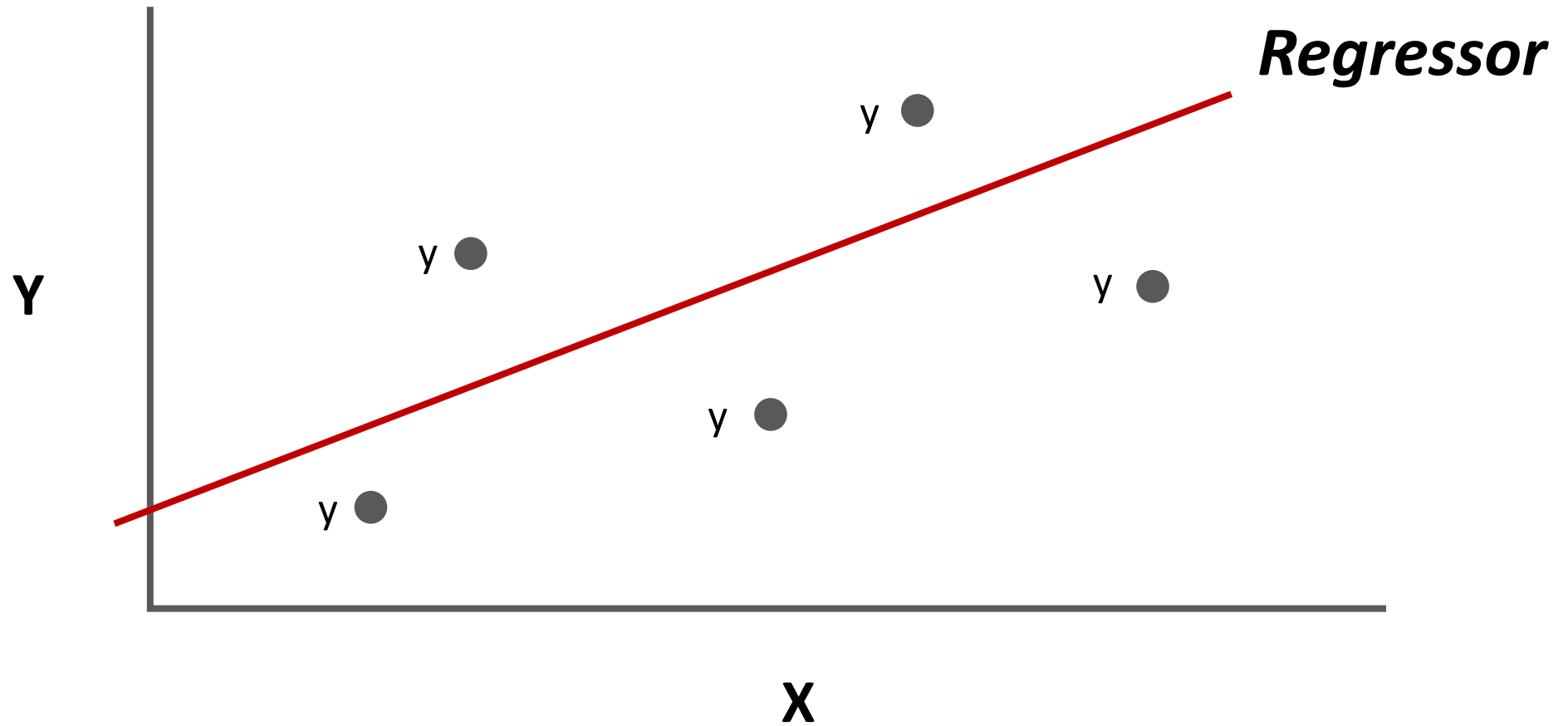


# Regression : Evaluation

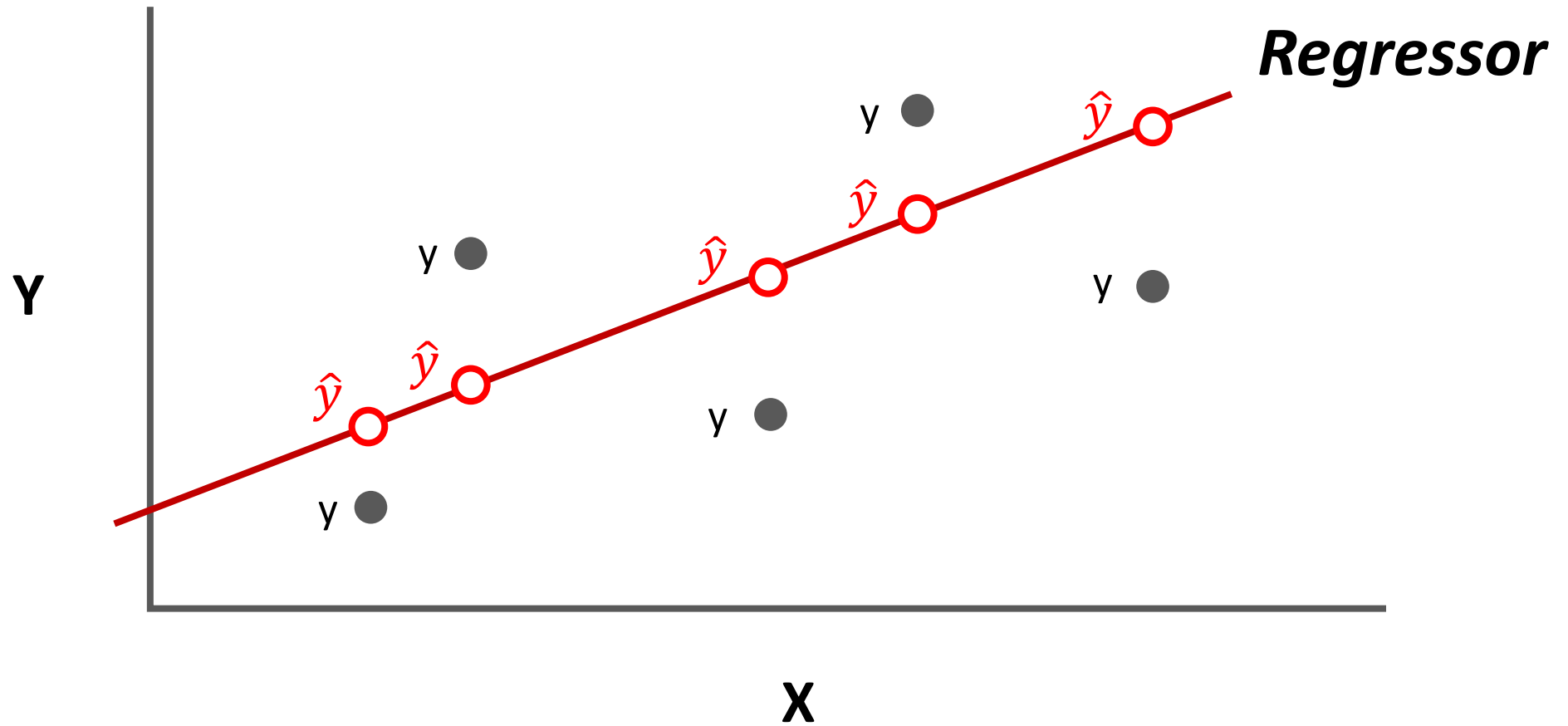
# Regression



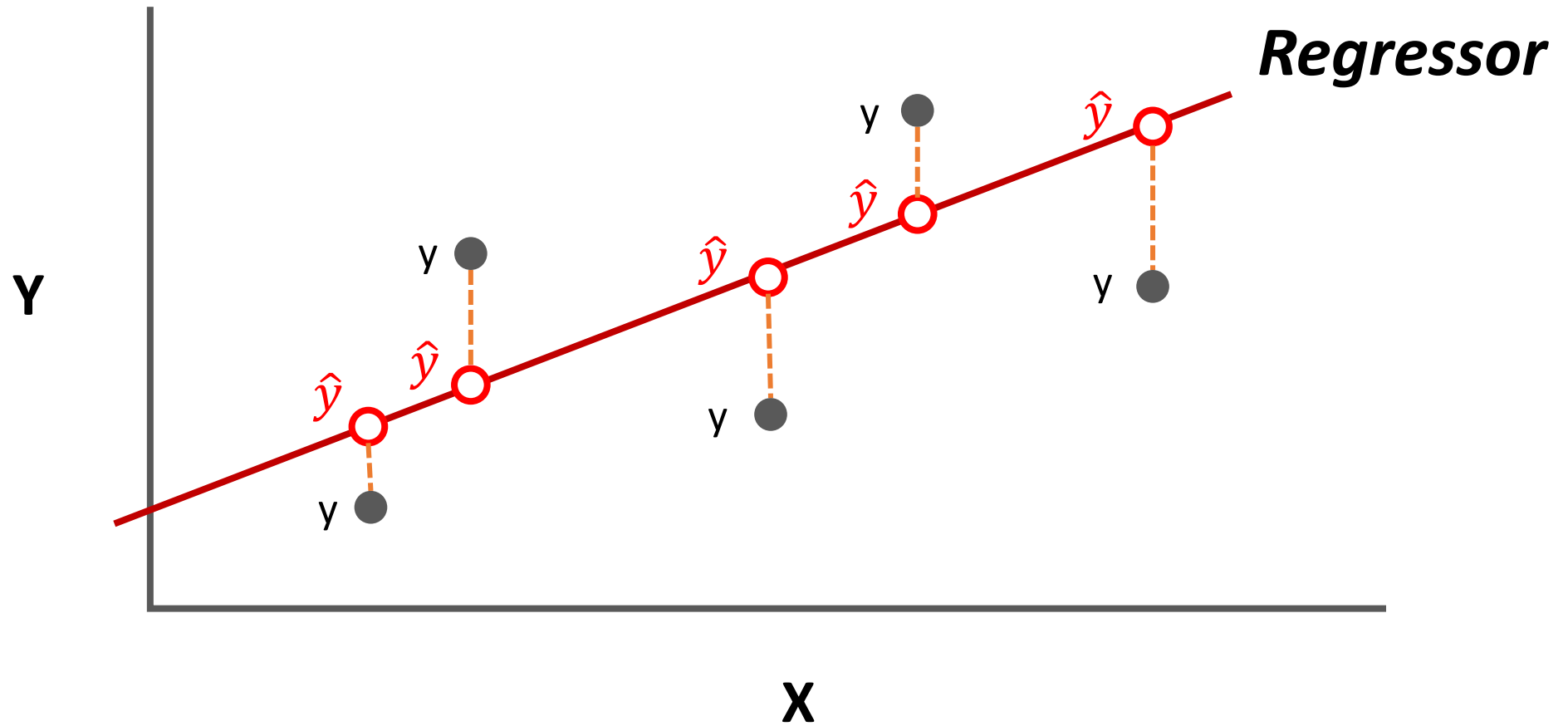
# Regression



# Regression



# Regression



# RMSE (Root Mean Square Error)

$$RMSE = \sqrt{\frac{1}{n} \sum (y - \hat{y})^2}$$

# RMSE (Root Mean Square Error)

The diagram illustrates the components of the RMSE formula:  $RMSE = \sqrt{\frac{1}{n} \sum (y - \hat{y})^2}$ . Red arrows point from numbered labels to specific parts of the formula: 1. 'Error' points to the term  $(y - \hat{y})$ . 2. 'Square' points to the exponent  $^2$ . 3. 'Mean' points to the summation symbol  $\sum$ . 4. 'Root' points to the square root symbol  $\sqrt{\phantom{x}}$ .

$$RMSE = \sqrt{\frac{1}{n} \sum (y - \hat{y})^2}$$

1 Error

2 Square

3 Mean

4 Root

# RMSE (Root Mean Square Error)

$$RMSE = \sqrt{\frac{1}{n} \sum (y - \hat{y})^2}$$

| Case # | y  | $\hat{y}$ |
|--------|----|-----------|
| 1      | 10 | 8         |
| 2      | 20 | 21        |
| 3      | 30 | 32        |

$$RMSE = \sqrt{\frac{1}{3} ((10 - 8)^2 + (20 - 21)^2 + (30 - 32)^2)}$$

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

$$= \sqrt{\frac{1}{3} (4 + 1 + 4)} = \sqrt{\frac{1}{3} (9)}$$

$$= \sqrt{3}$$

$$RMSE = \mathbf{1.732}$$

# Other Regression Measurement Methods

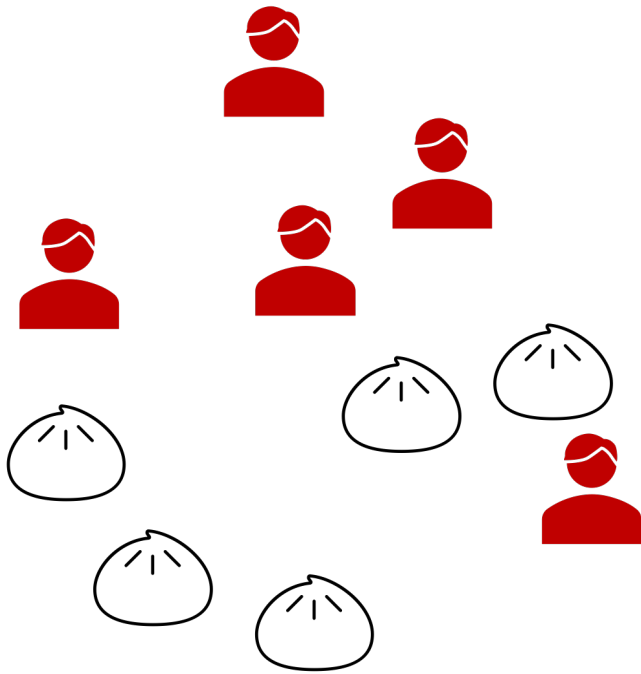
$$RMSE = \sqrt{\frac{1}{n} \sum (y - \hat{y})^2}$$

$$MAE = \frac{1}{n} \sum |y - \hat{y}|$$

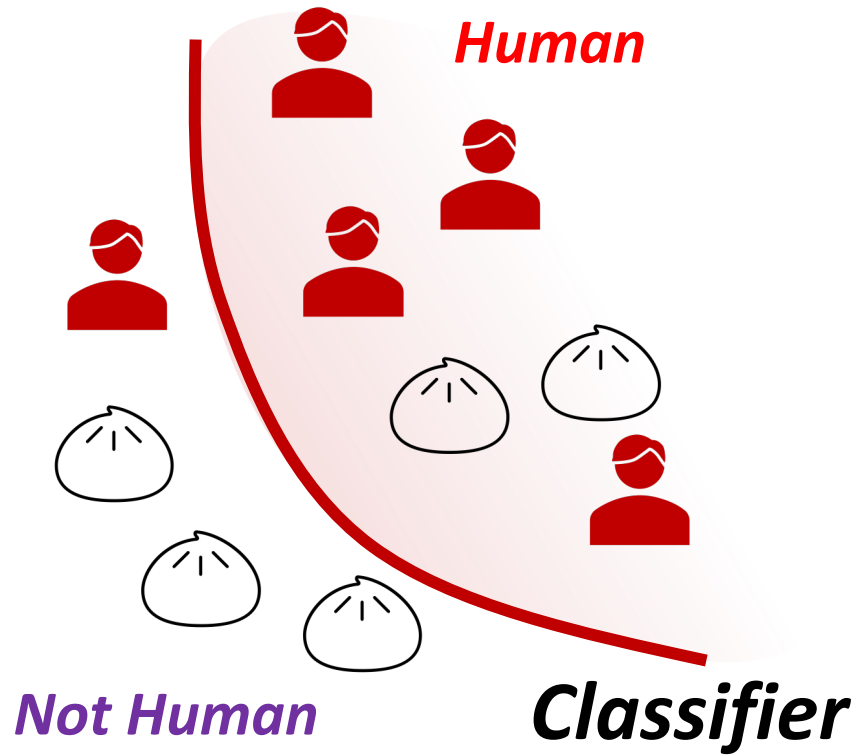
$$MAPE = \frac{100\%}{n} \sum \left| \frac{y - \hat{y}}{y} \right|$$

# Classification : Evaluation

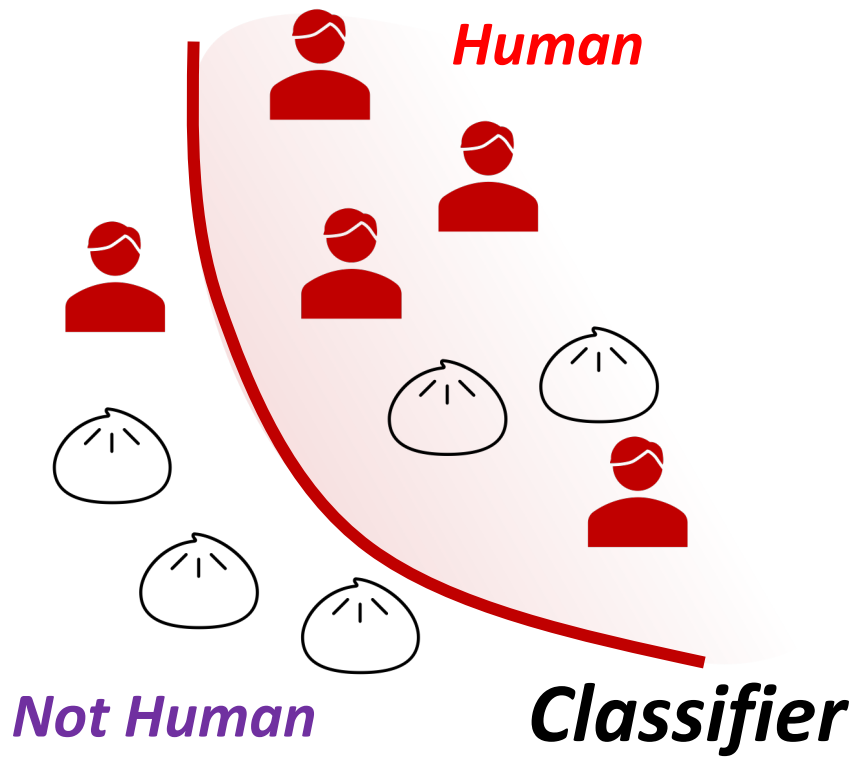
# Classification



# Classification



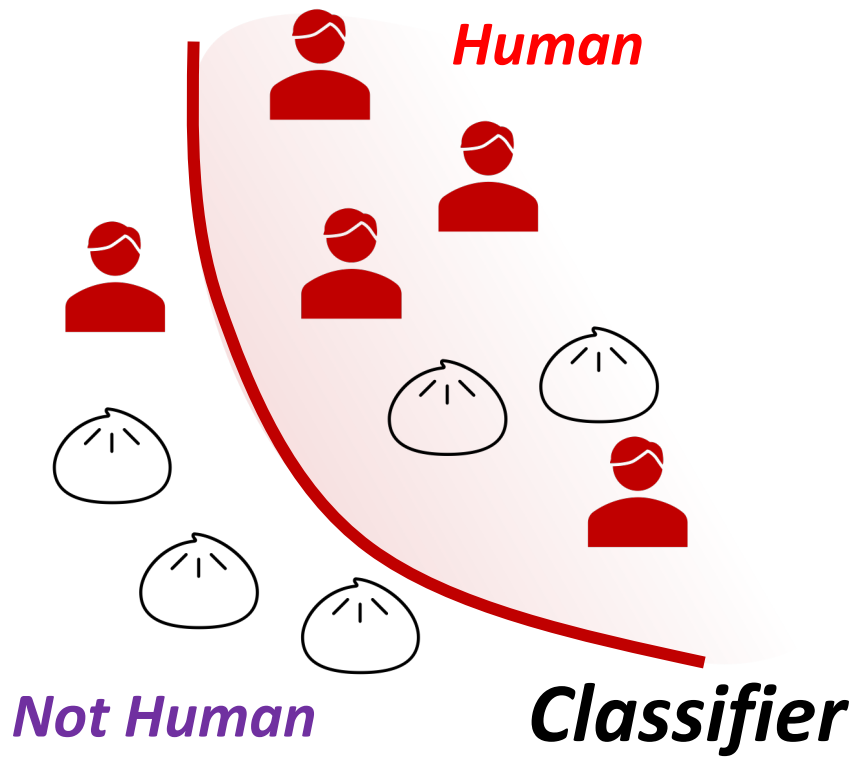
# Confusion Matrix



|        |     | Predicted |    |
|--------|-----|-----------|----|
|        |     | Yes       | No |
| Actual | Yes | TP        | FN |
|        | No  | FP        | TN |

Confusion  
Matrix

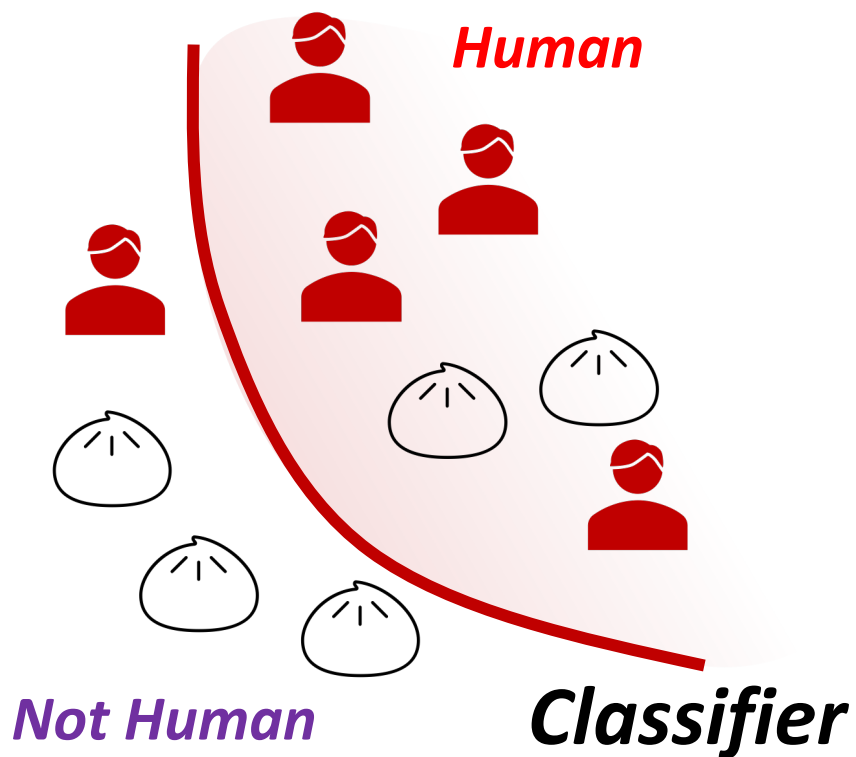
# Confusion Matrix



|        |     | Predicted |         |
|--------|-----|-----------|---------|
|        |     | Yes       | No      |
| Actual | Yes | TP<br>4   | FN<br>1 |
|        | No  | FP<br>2   | TN<br>3 |

Confusion  
Matrix

# Accuracy



|        |     | Predicted |         |
|--------|-----|-----------|---------|
|        |     | Yes       | No      |
| Actual | Yes | TP<br>4   | FN<br>1 |
|        | No  | FP<br>2   | TN<br>3 |

Confusion  
Matrix

*Accuracy*

$$= \frac{TP + TN}{TP + TN + FP + FN}$$

$$= \frac{4+3}{4+3+2+1} = 0.7$$

# Other Classification Measurement Methods

|        |     | Predicted |         |
|--------|-----|-----------|---------|
|        |     | Yes       | No      |
| Actual | Yes | TP<br>4   | FN<br>1 |
|        | No  | FP<br>2   | TN<br>3 |

Confusion  
Matrix

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN}$$

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

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

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

# LEARNING STEPS

# Learning Steps

- Machine Learning Steps
- Train-Test Split
- K-Fold Cross-Validation

# Dataset

inputs,  
independent variables  
attributes,  
features,  
 $X$ , ...

output,  
dependent variables  
label,  
actual  $y$ ,  
 $y$  true, ...



| $x_1$ | $x_2$ | $x_3$ | ... | $x_n$ | $y$ |
|-------|-------|-------|-----|-------|-----|
| 1     | 5.2   | FB    | ... | XL    | 0   |
| 2     | 1.1   | TW    | ... | M     | 0   |
| 3     | -9.6  | IG    | ... | L     | 1   |

# Dataset

Dataset

|  | X |  |  |  |  | y |
|--|---|--|--|--|--|---|
|  |   |  |  |  |  |   |
|  |   |  |  |  |  |   |
|  |   |  |  |  |  |   |
|  |   |  |  |  |  |   |
|  |   |  |  |  |  |   |
|  |   |  |  |  |  |   |
|  |   |  |  |  |  |   |
|  |   |  |  |  |  |   |
|  |   |  |  |  |  |   |



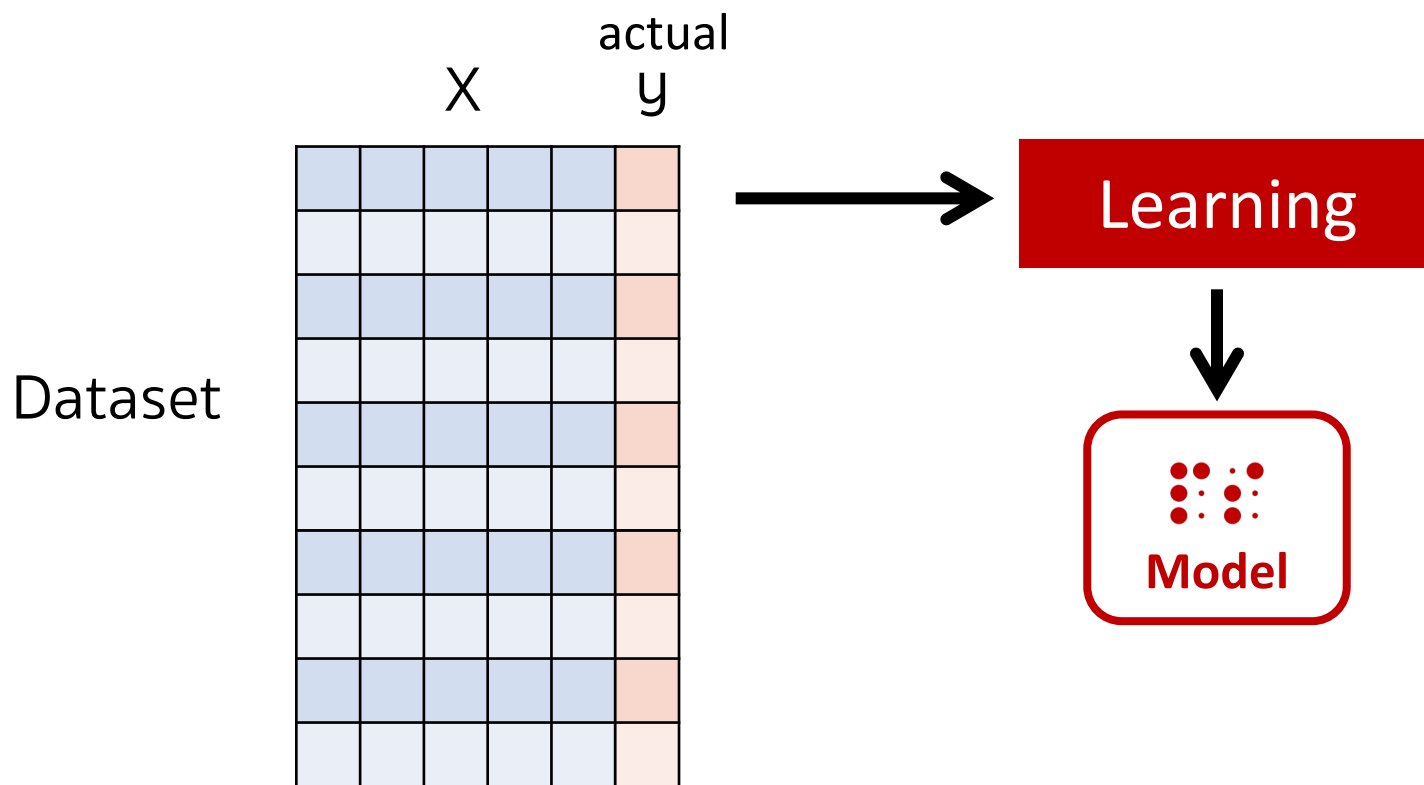
# Machine Learning Steps

# Dataset

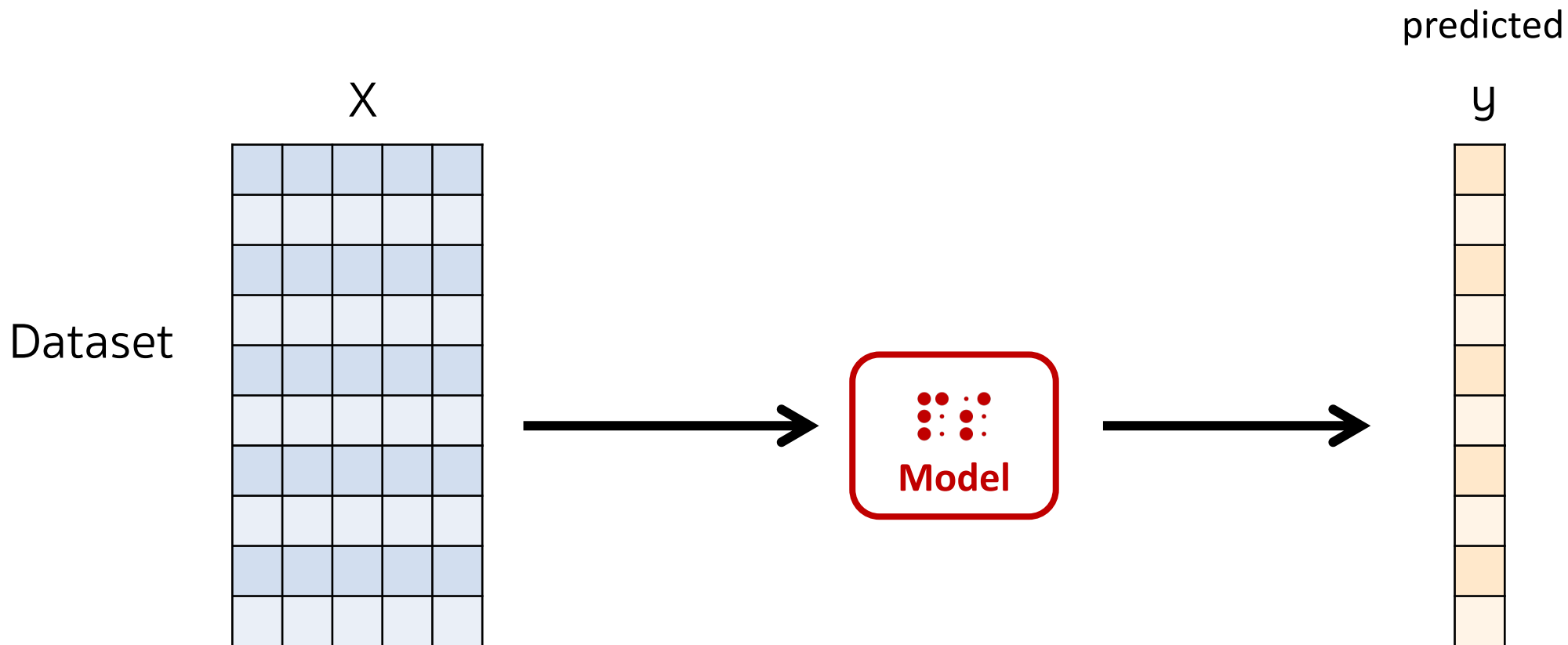
Dataset

| X |  |  |  |  | y |
|---|--|--|--|--|---|
|   |  |  |  |  |   |
|   |  |  |  |  |   |
|   |  |  |  |  |   |
|   |  |  |  |  |   |
|   |  |  |  |  |   |
|   |  |  |  |  |   |
|   |  |  |  |  |   |
|   |  |  |  |  |   |
|   |  |  |  |  |   |

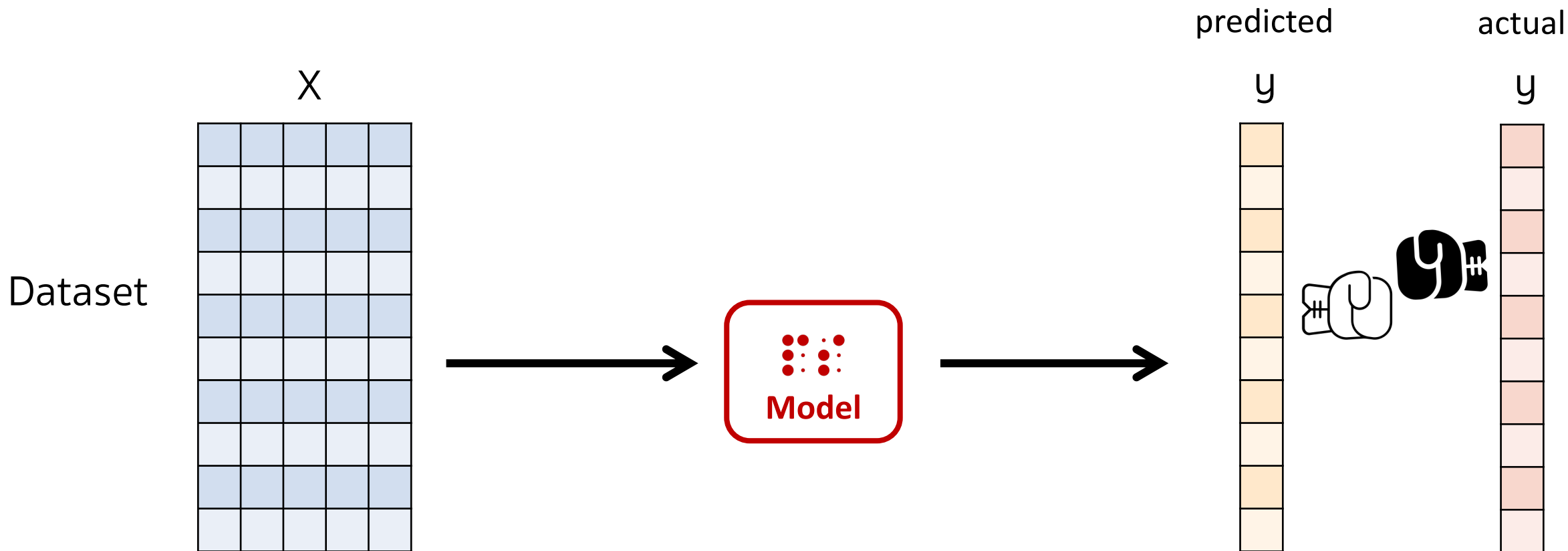
# To Learning



# To Predict



# To Evaluate





# Train-Test Split

# Machine Learning Steps

Dataset

| X |  |  |  |  | y |
|---|--|--|--|--|---|
|   |  |  |  |  |   |
|   |  |  |  |  |   |
|   |  |  |  |  |   |
|   |  |  |  |  |   |
|   |  |  |  |  |   |
|   |  |  |  |  |   |
|   |  |  |  |  |   |
|   |  |  |  |  |   |
|   |  |  |  |  |   |

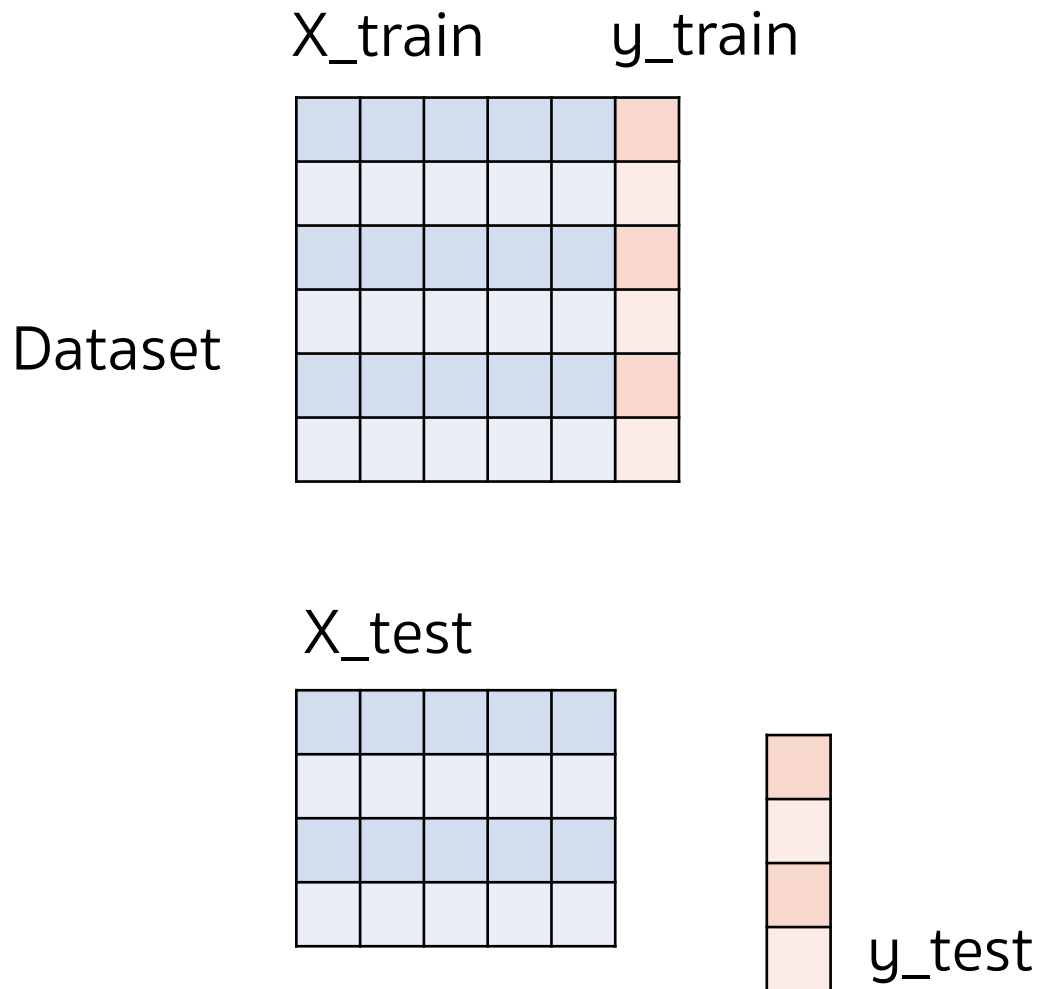
# Machine Learning Steps

Dataset

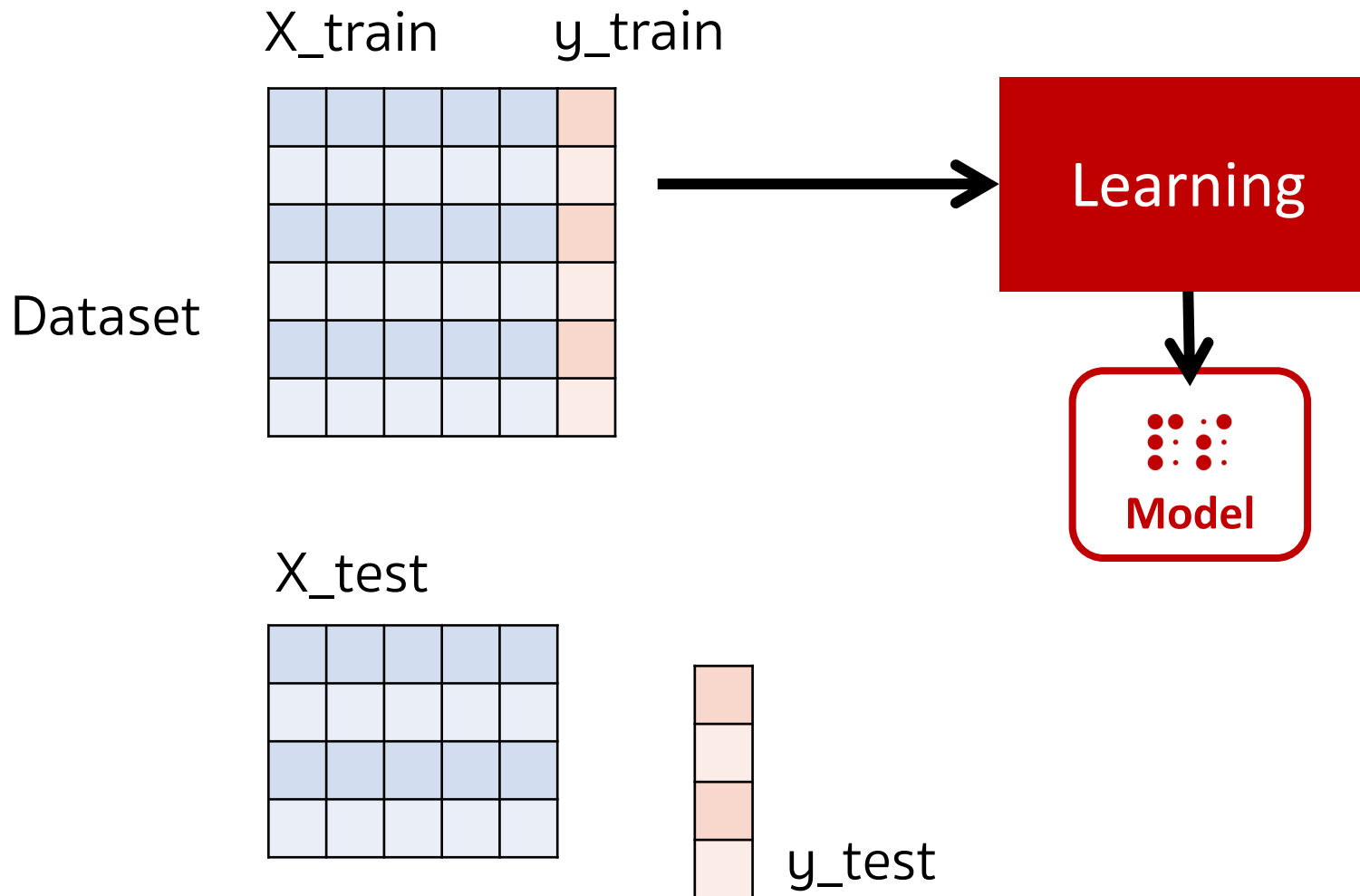
| X_train | y_train |
|---------|---------|
|         |         |
|         |         |
|         |         |
|         |         |
|         |         |
|         |         |

| X_test | y_test |
|--------|--------|
|        |        |
|        |        |
|        |        |
|        |        |

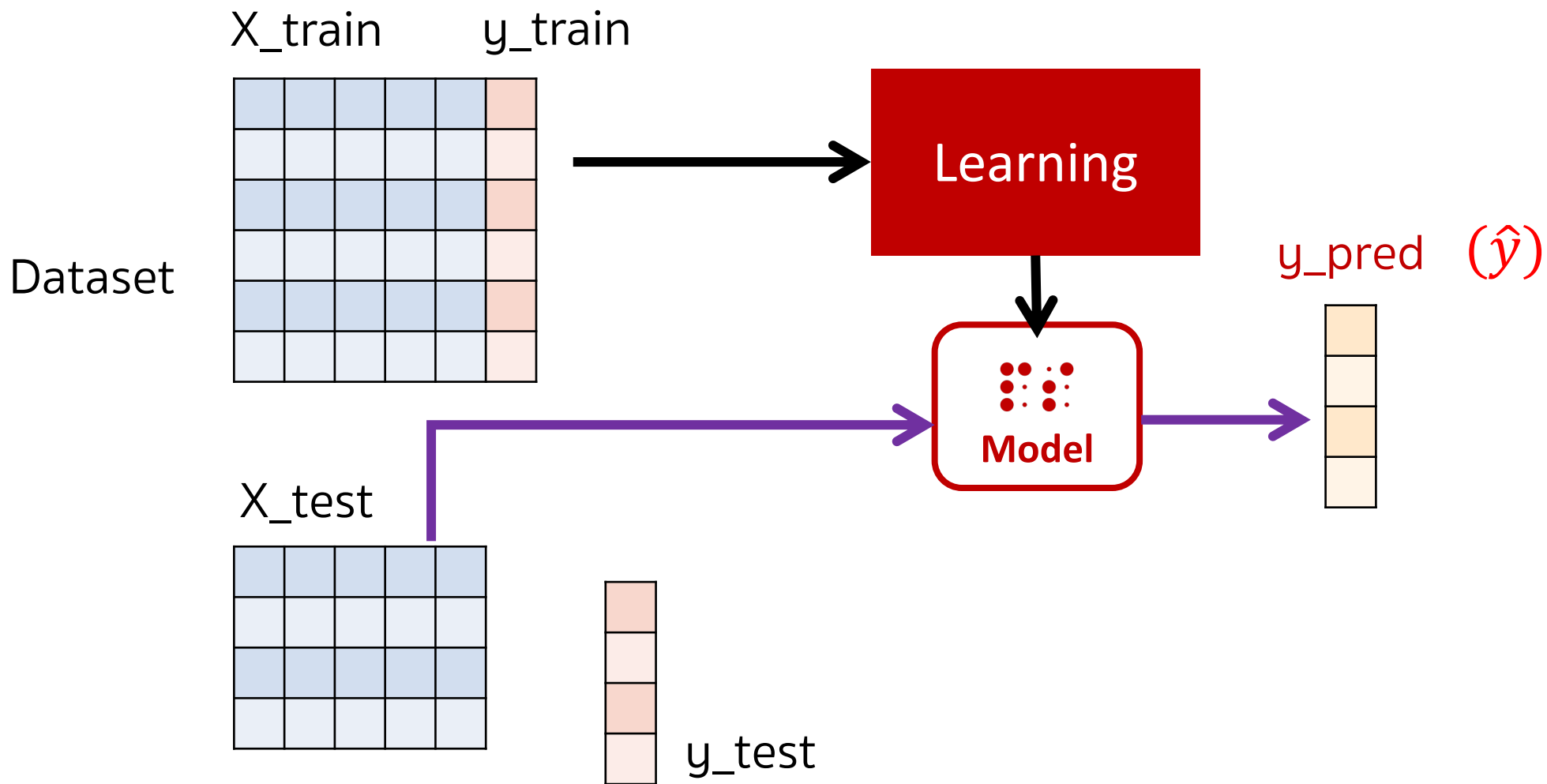
# Machine Learning Steps



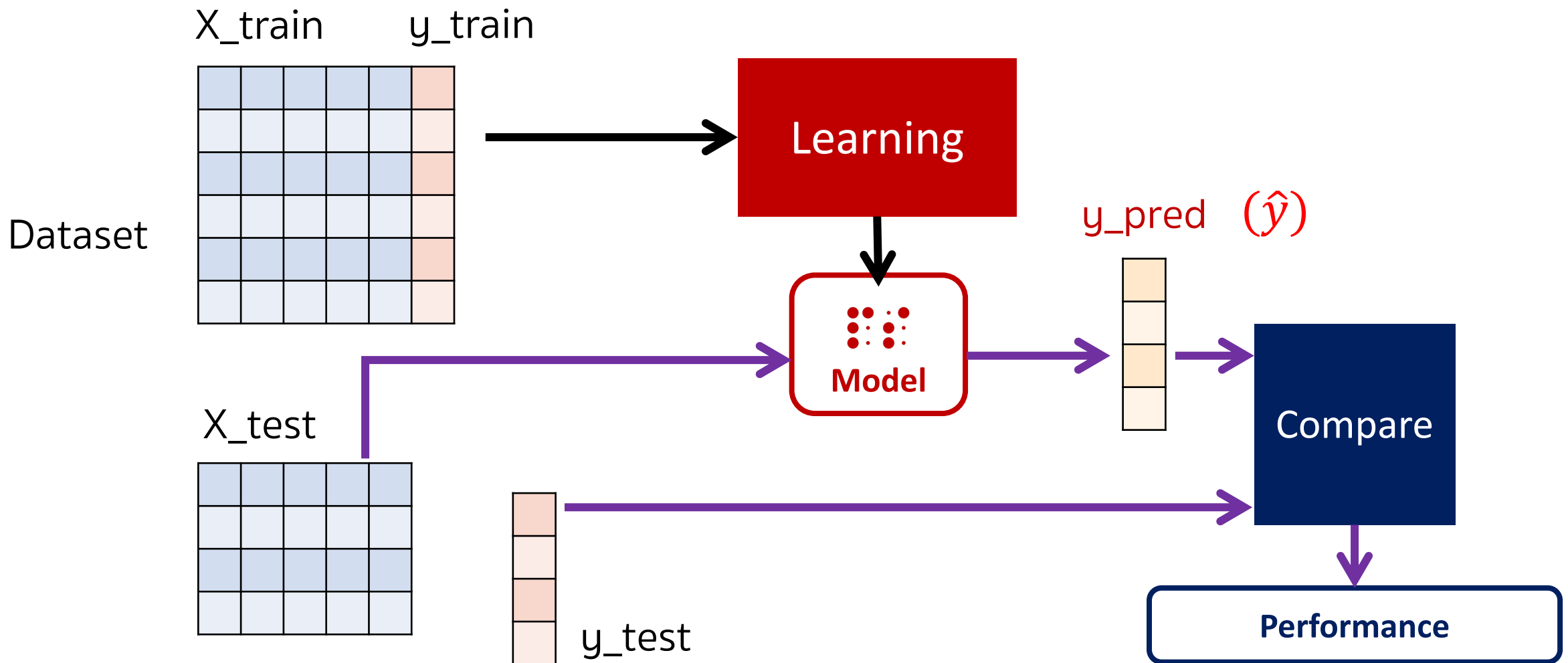
# Machine Learning Steps



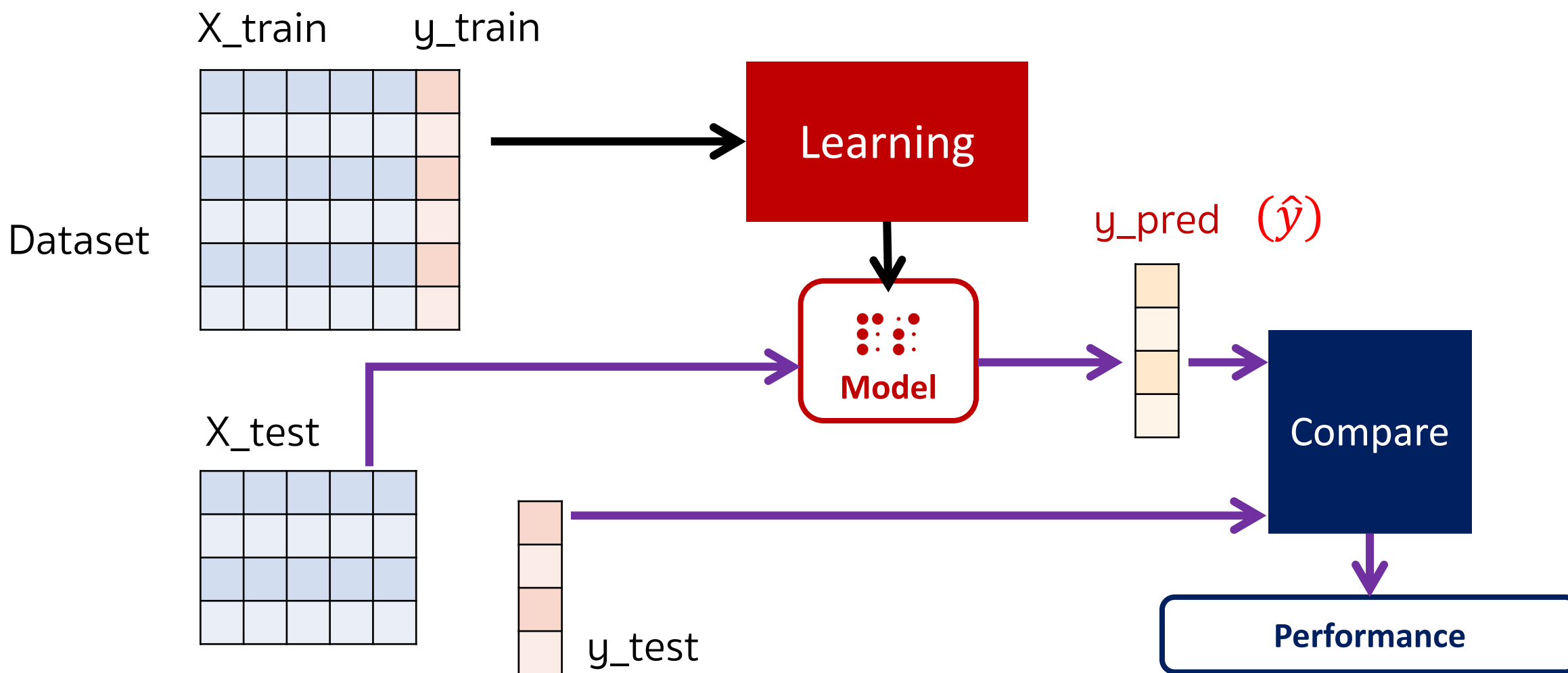
# Machine Learning Steps



# Machine Learning Steps



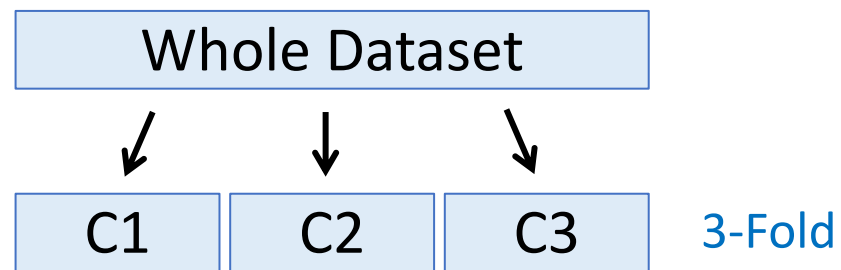
# Train-Test Split



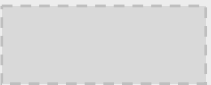


# K-Fold Cross-Validation

# K-Fold Cross-Validation

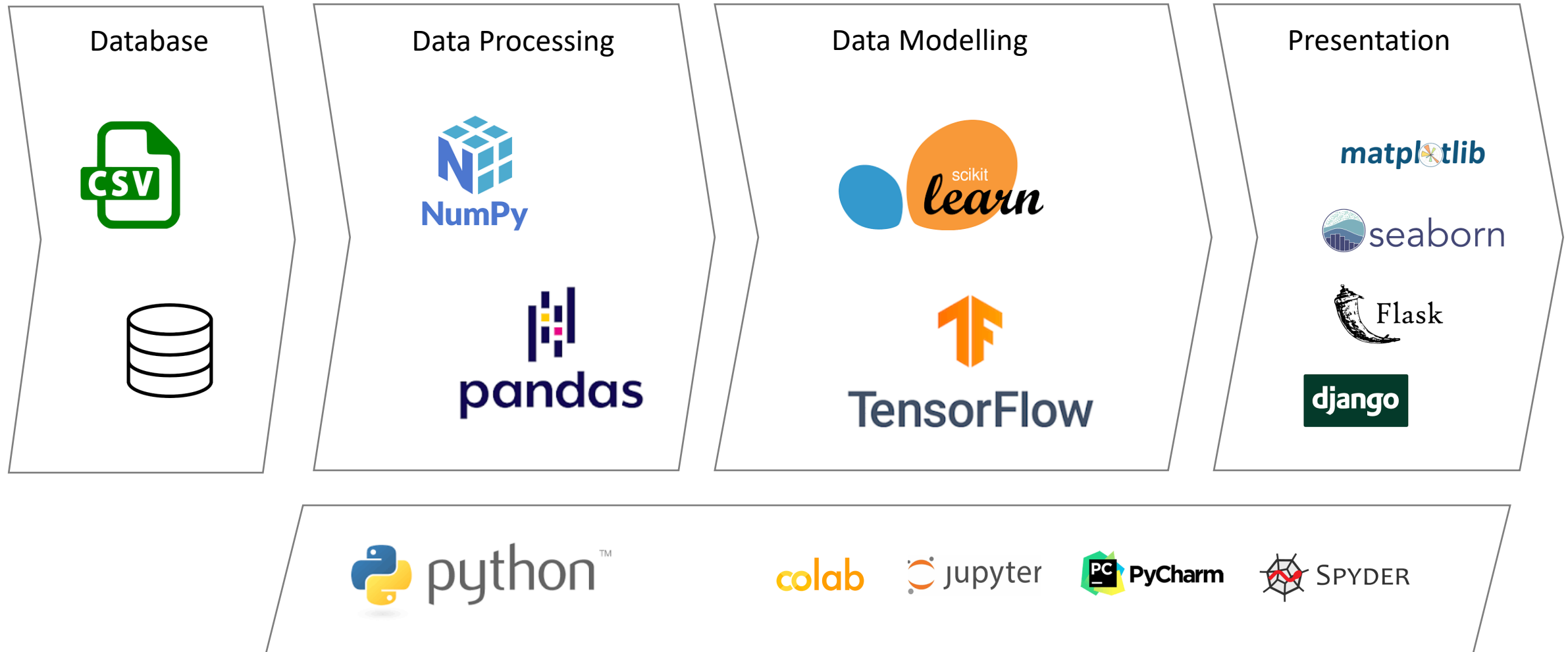


3-Fold Cross-Validation

| Round          | Training Set                                                                                | Test Set | Evaluation               |
|----------------|---------------------------------------------------------------------------------------------|----------|--------------------------|
| 1              |  C2 C3   | C1       | e1                       |
| 2              | C1  C3   | C2       | e2                       |
| 3              | C1 C2  | C3       | e3                       |
| Average Result |                                                                                             |          | $\frac{e1 + e2 + e3}{3}$ |

# Scikit Learn

# Python Ecosystem for AI



# Scikit Learn

- Simple and efficient tools for predictive data analysis
- Accessible to everybody, and reusable in various contexts
- Built on NumPy, SciPy, and matplotlib
- Open source, commercially usable - BSD license

ref: <https://scikit-learn.org/>

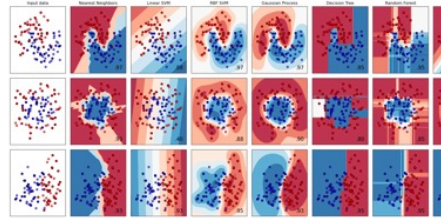
# Scikit Learn

## Classification

Identifying which category an object belongs to.

**Applications:** Spam detection, image recognition.

**Algorithms:** SVM, nearest neighbors, random forest, and more...



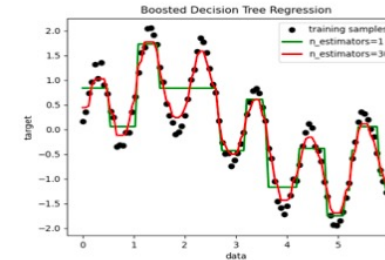
Examples

## Regression

Predicting a continuous-valued attribute associated with an object.

**Applications:** Drug response, Stock prices.

**Algorithms:** SVR, nearest neighbors, random forest, and more...



Examples

## Clustering

Automatic grouping of similar objects into sets.

**Applications:** Customer segmentation, Grouping experiment outcomes

**Algorithms:** k-Means, spectral clustering, mean-shift, and more...



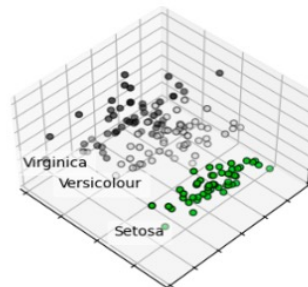
Examples

## Dimensionality reduction

Reducing the number of random variables to consider.

**Applications:** Visualization, Increased efficiency

**Algorithms:** k-Means, feature selection, non-negative matrix factorization, and more...



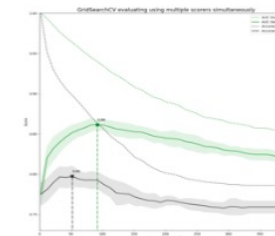
Examples

## Model selection

Comparing, validating and choosing parameters and models.

**Applications:** Improved accuracy via parameter tuning

**Algorithms:** grid search, cross validation, metrics, and more...



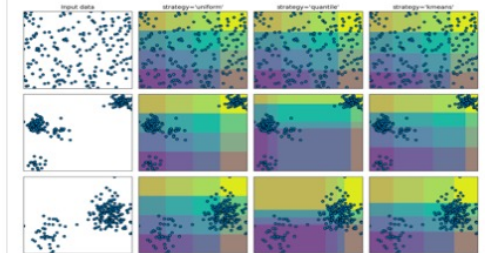
Examples

## Preprocessing

Feature extraction and normalization.

**Applications:** Transforming input data such as text for use with machine learning algorithms.

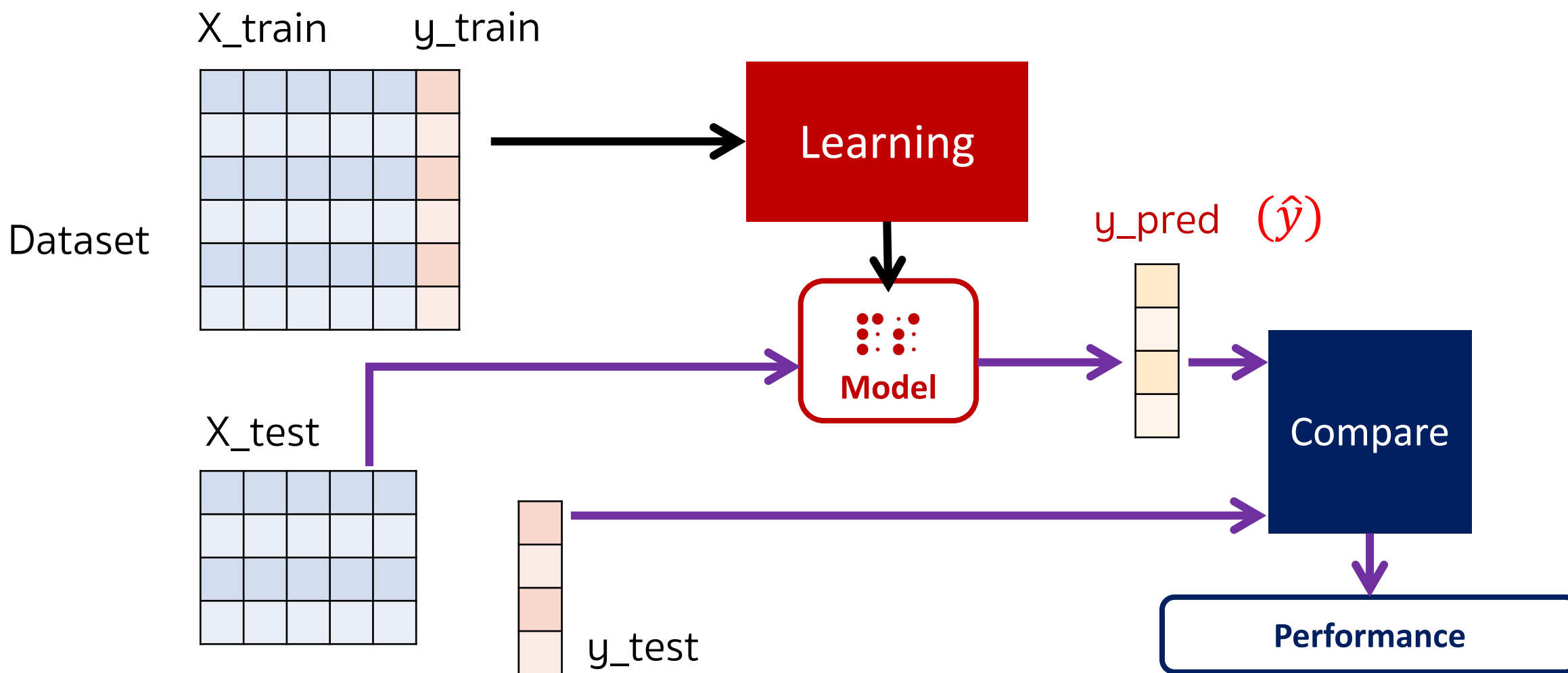
**Algorithms:** preprocessing, feature extraction, and more...



Examples

ref: <https://scikit-learn.org/>

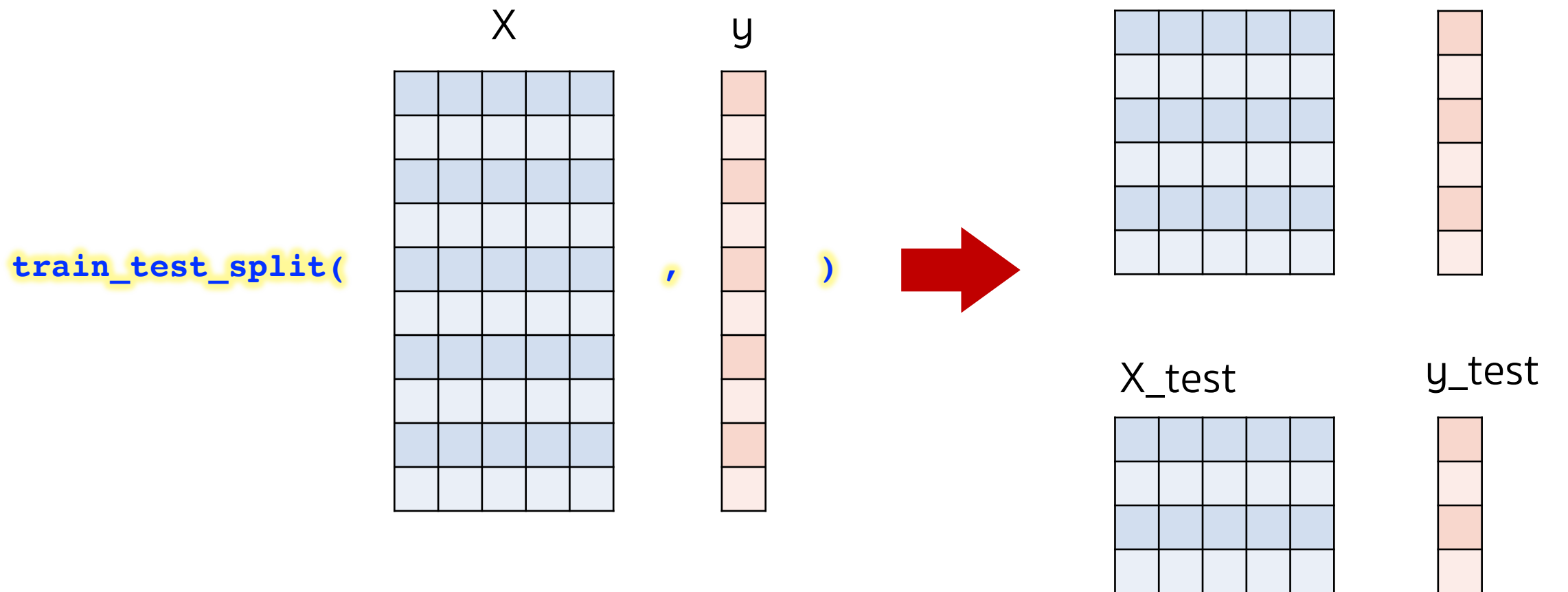
# Train-Test Split



# Major Steps for Train-Test Split

- To Split Train-Test
- To Create a Model
- To Predict
- To Evaluate

# To Split Train-Test



# To Create a Model

`{technique}.fit(`

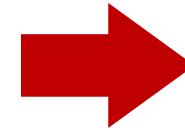
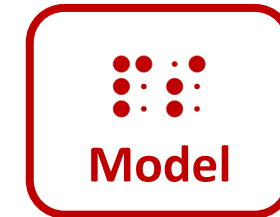
X\_train

|  |  |  |  |  |
|--|--|--|--|--|
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |

y\_train

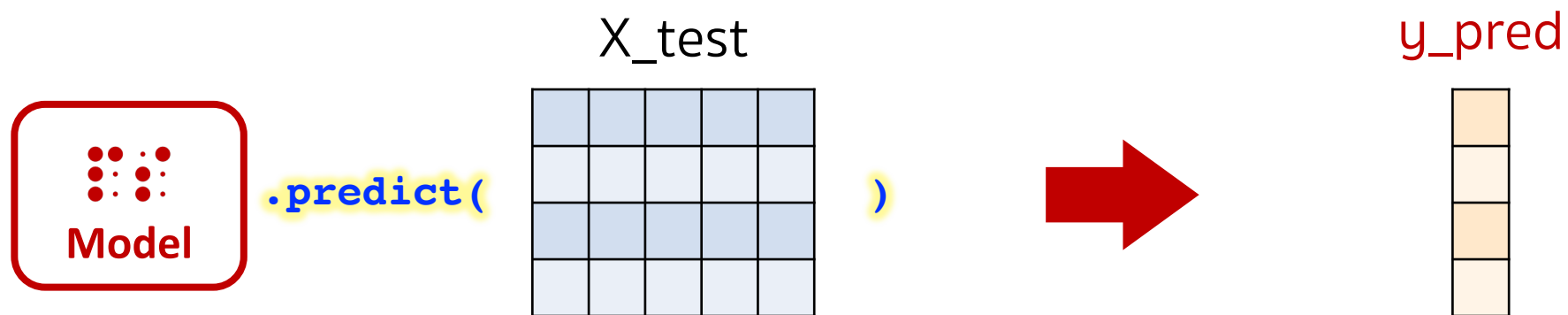
|  |
|--|
|  |
|  |
|  |
|  |
|  |
|  |
|  |

, )

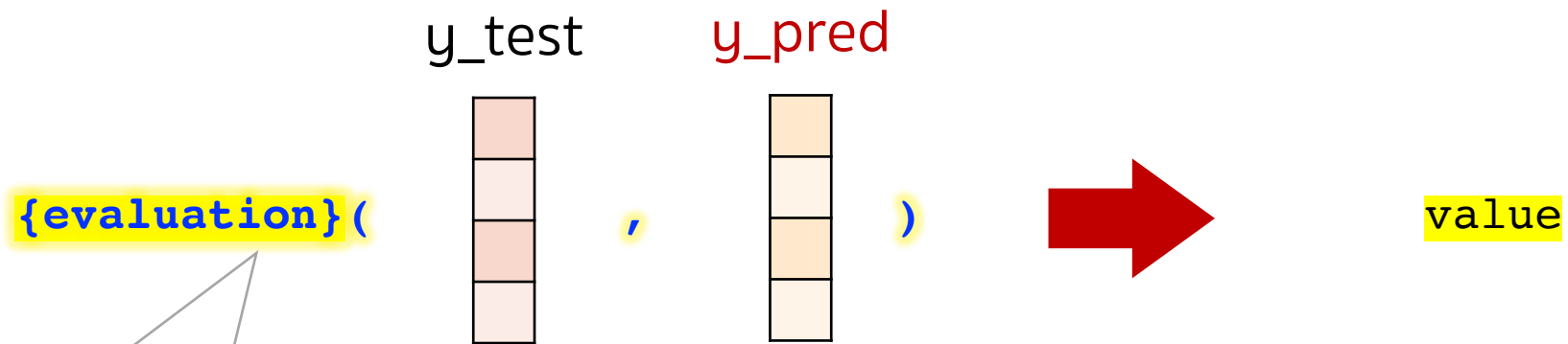


e.g.  
LinearRegression()  
LogisticRegression()  
DecisionTreeClassifier()  
...

# To Predict



# To Evaluate



e.g.  
mean\_squared\_error  
mean\_absolute\_percentage\_error  
accuracy\_score  
f1\_score  
...

# Coding : Linear Regression

```
1) import pandas as pd
2) from sklearn.model_selection import train_test_split
3) from sklearn.linear_model import LinearRegression
4) from sklearn.metrics import mean_squared_error

5) df = pd.read_csv('example.csv')
6) X = df[['x1', 'x2', 'x3']]
7) y = df['y']
8) X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3)
9) model = LinearRegression()
10) model.fit(X_train, y_train)
11) y_pred = model.predict(X_test)
12) mse = mean_squared_error(y_test, y_pred)
```

# Coding : Decision Tree

```
1) import pandas as pd
2) from sklearn.model_selection import train_test_split
3) from sklearn.tree import DecisionTreeClassifier
4) from sklearn.metrics import accuracy_score

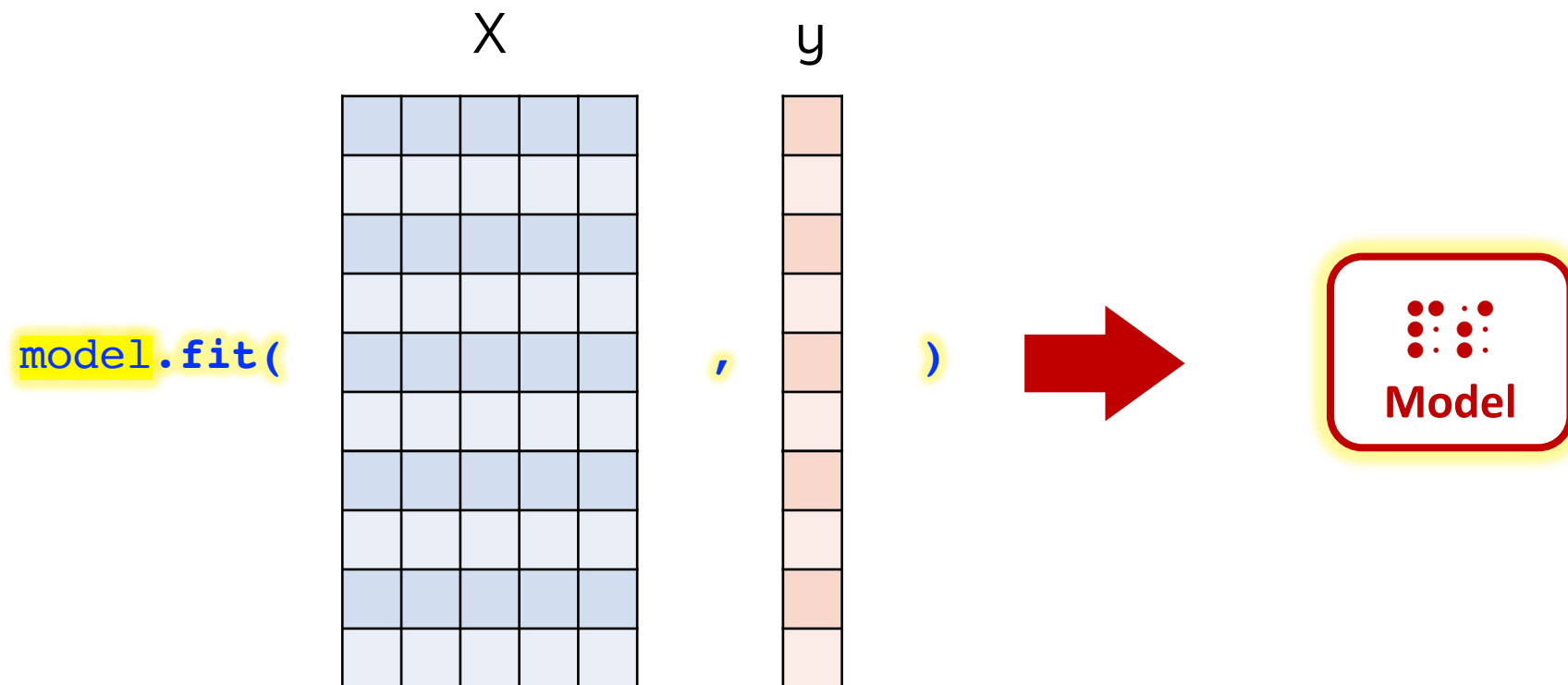
5) df = pd.read_csv('example.csv')
6) X = df[['x1', 'x2', 'x3']]
7) y = df['y']
8) X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3)
9) model = DecisionTreeClassifier()
10) model.fit(X_train, y_train)
11) y_pred = model.predict(X_test)
12) accuracy = accuracy_score(y_test, y_pred)
```

# Coding : Linear K-Fold

```
1) from sklearn.model_selection import KFold
2) from sklearn.linear_model import LinearRegression
3) from sklearn.metrics import mean_squared_error

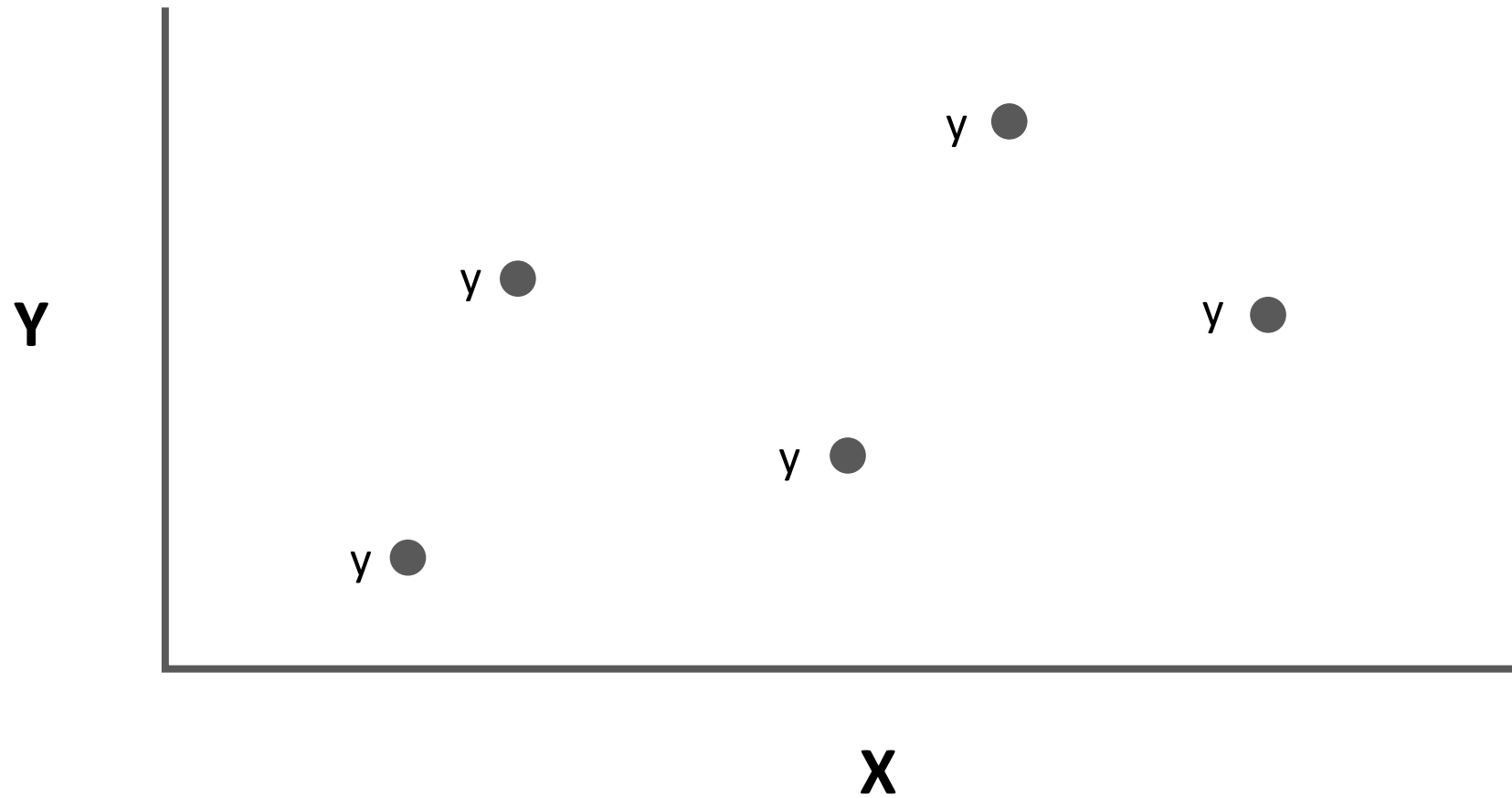
4) kf = KFold(n_splits=4)
5) for train_index, test_index in kf.split(X):
6)     X_train, X_test = X.loc[train_index], X.loc[test_index]
7)     y_train, y_test = y.loc[train_index], y.loc[test_index]
8)     model = LinearRegression()
9)     model.fit(X_train, y_train)
10)    y_pred = model.predict(X_test)
11)    mse_i = mean_squared_error(y_test, y_pred)
```

# To Deploy

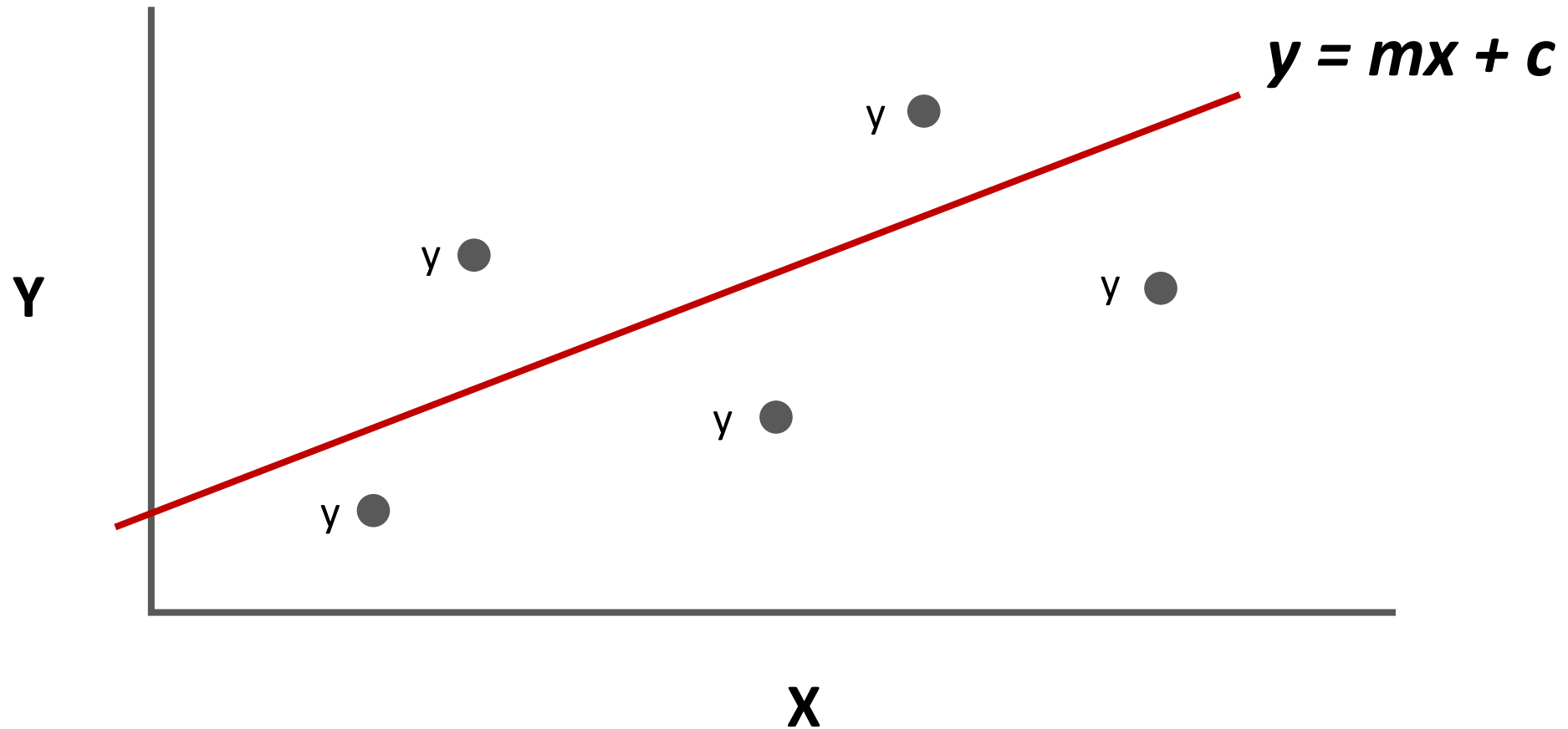


# REGRESSION

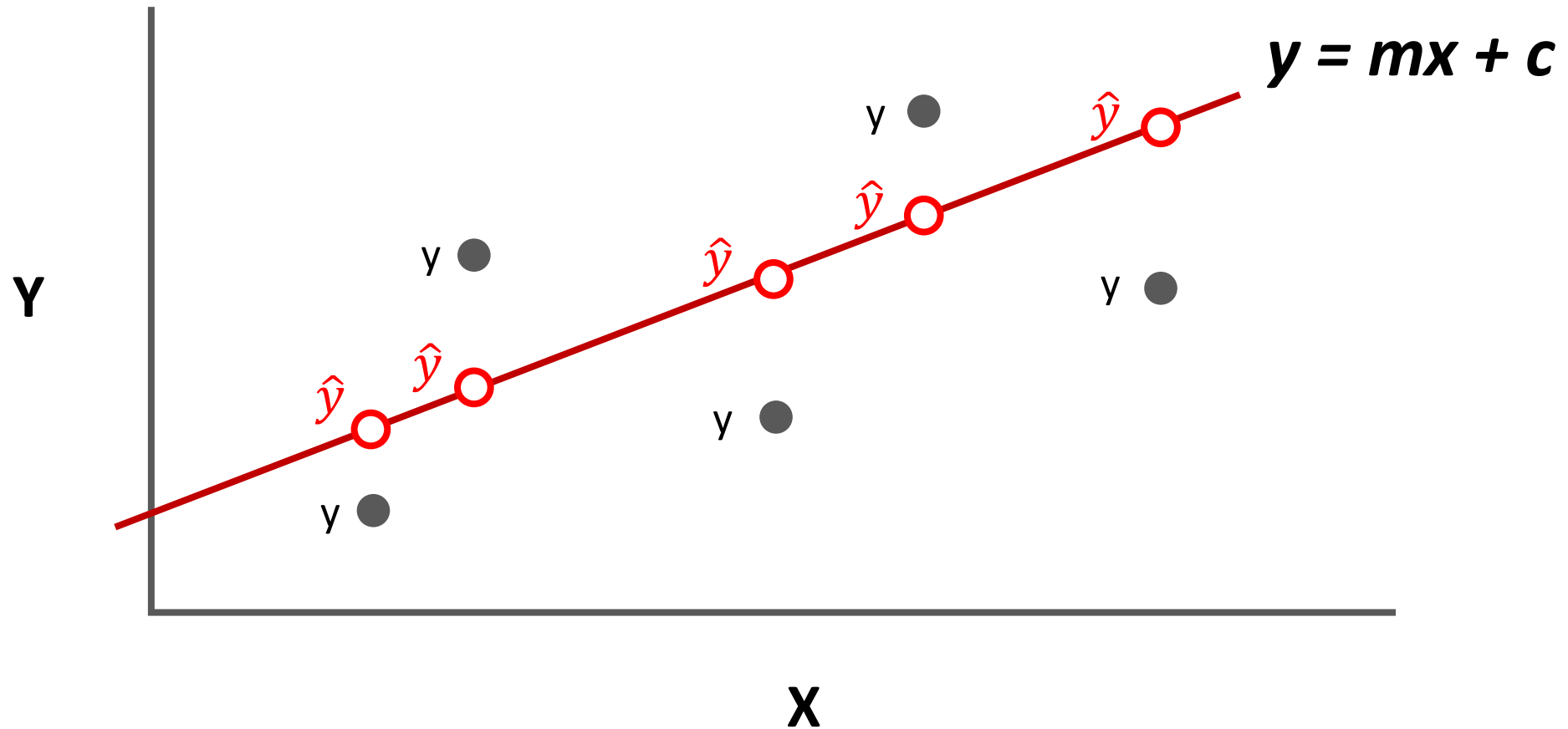
# Linear Regression



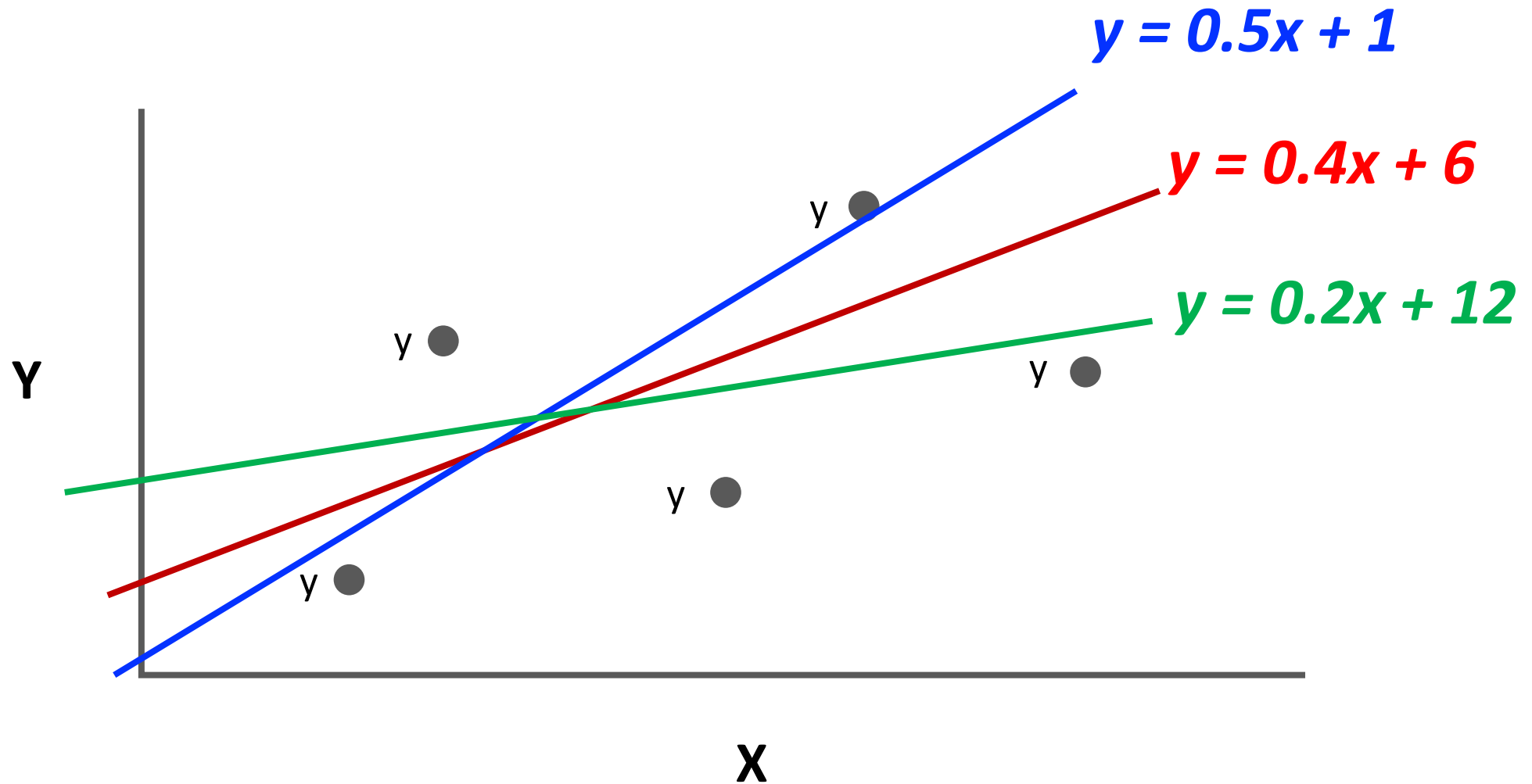
# Linear Regression



# Linear Regression



# Linear Regression



# Linear Regression

The diagram illustrates the components of the linear regression equation  $y = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \dots + \beta_n x_n$ . Annotations include:

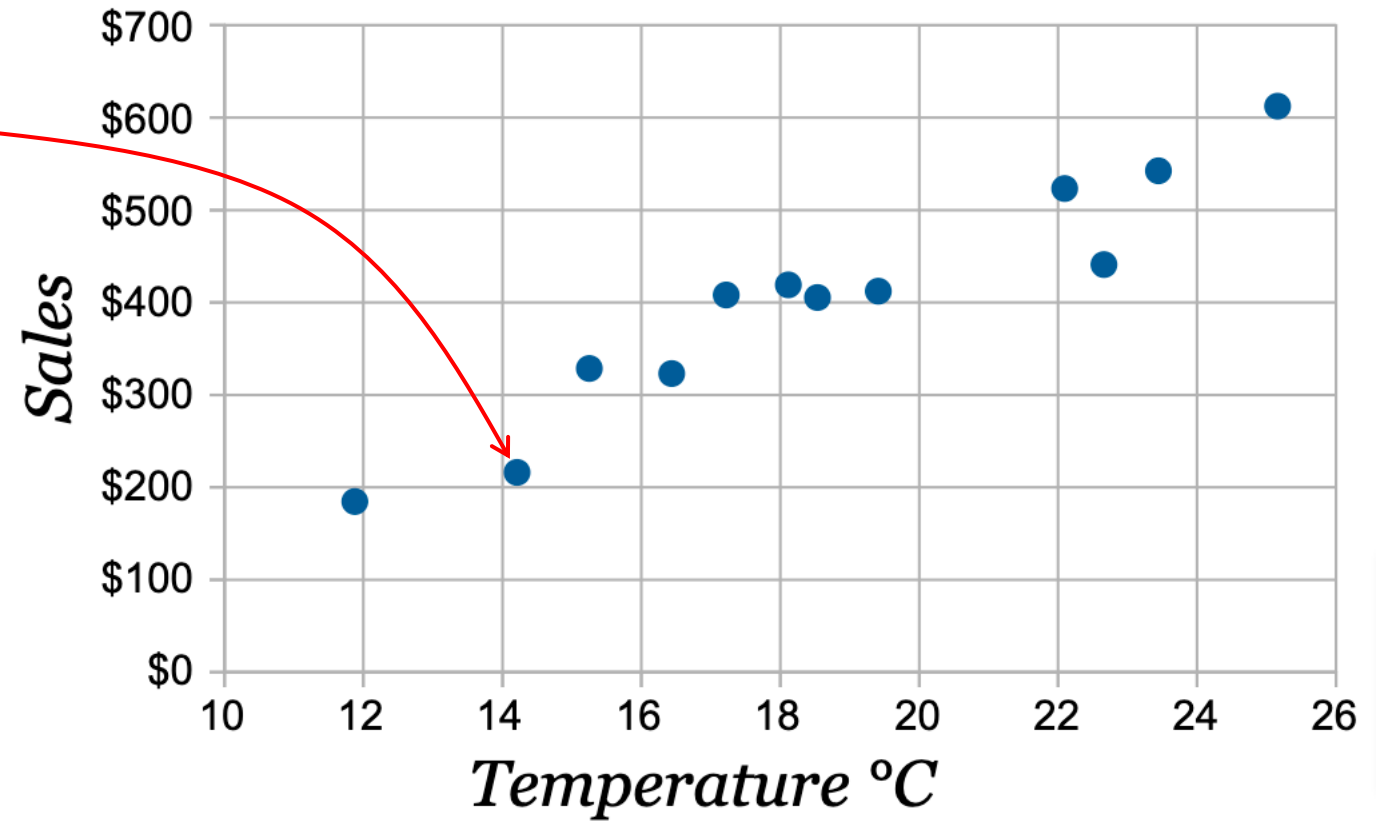
- A green arrow labeled "label" points to  $y$ .
- A red arrow labeled "features / attributes / variables" points to the  $x$  terms ( $x_1, x_2, \dots, x_n$ ).
- A blue arrow labeled "intercept" points to  $\beta_0$ .
- A blue arrow labeled "coefficients" points to the  $\beta$  terms ( $\beta_1, \beta_2, \dots, \beta_n$ ).

$$y = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \dots + \beta_n x_n$$

# Feature Selection

## Correlation

| <i>Ice Cream Sales vs Temperature</i> |                 |
|---------------------------------------|-----------------|
| Temperature °C                        | Ice Cream Sales |
| 14.2°                                 | \$215           |
| 16.4°                                 | \$325           |
| 11.9°                                 | \$185           |
| 15.2°                                 | \$332           |
| 18.5°                                 | \$406           |
| 22.1°                                 | \$522           |
| 19.4°                                 | \$412           |
| 25.1°                                 | \$614           |
| 23.4°                                 | \$544           |
| 18.1°                                 | \$421           |
| 22.6°                                 | \$445           |
| 17.2°                                 | \$408           |

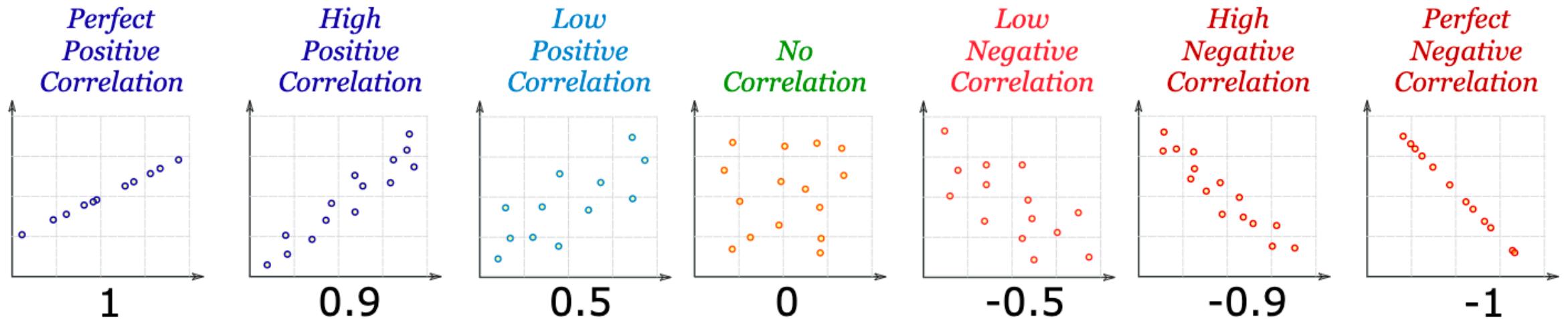


Ref: <https://www.mathsisfun.com/data/correlation.html>

# Feature Selection

## Pearson Correlation

$$r_{xy} = \frac{\sum_{i=1}^n (x_i - \bar{x})(y_i - \bar{y})}{\sqrt{\sum_{i=1}^n (x_i - \bar{x})^2 \sum_{i=1}^n (y_i - \bar{y})^2}}$$



Ref: <https://www.mathsisfun.com/data/correlation.html>

# COLAB: Linear Regression

- Go to the URL:
  - <https://colab.research.google.com/drive/1DdhhalFbgNnJv98WgdBrF8J7nNKwpLHX#scrollTo=HoVU3NP1wCa7>



# CLASSIFICATION

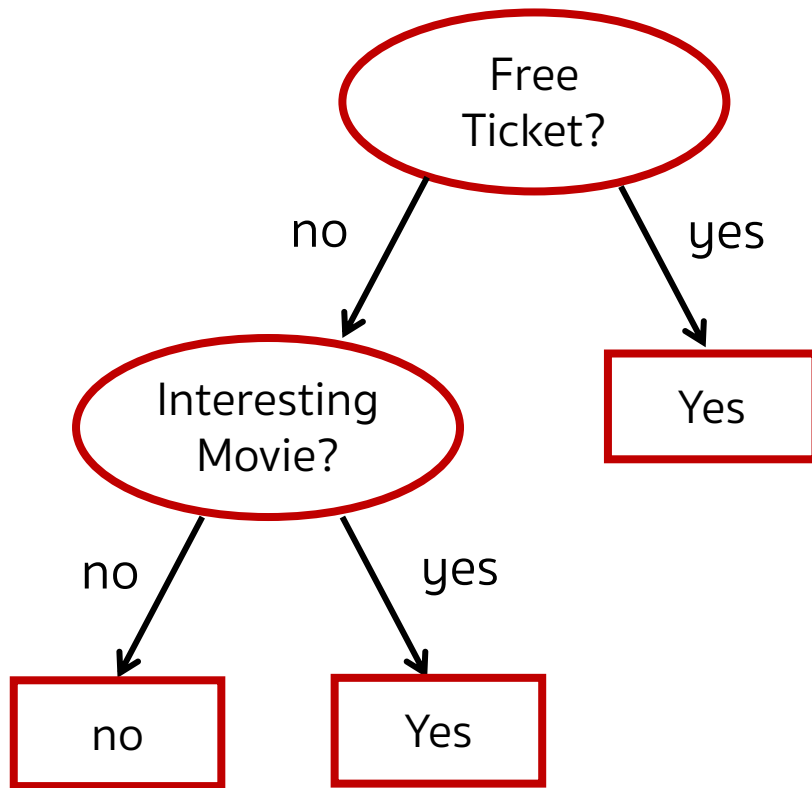
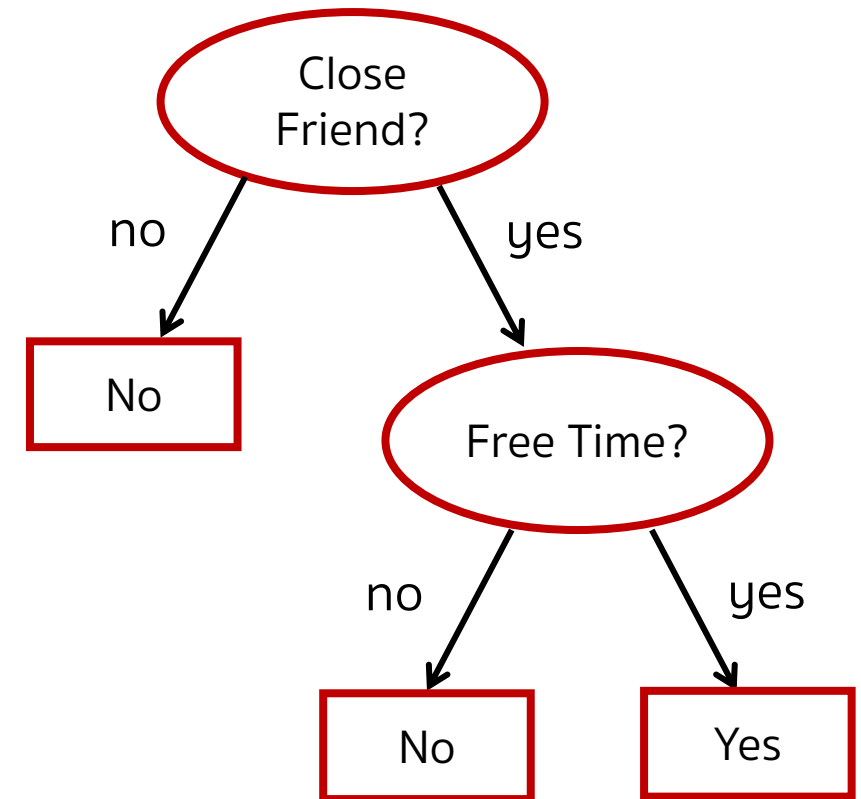
# Classification

- Decision Tree
- Logistic Regression
- Naïve Bayes
- K-Nearest Neighbors
- Artificial Neural Network

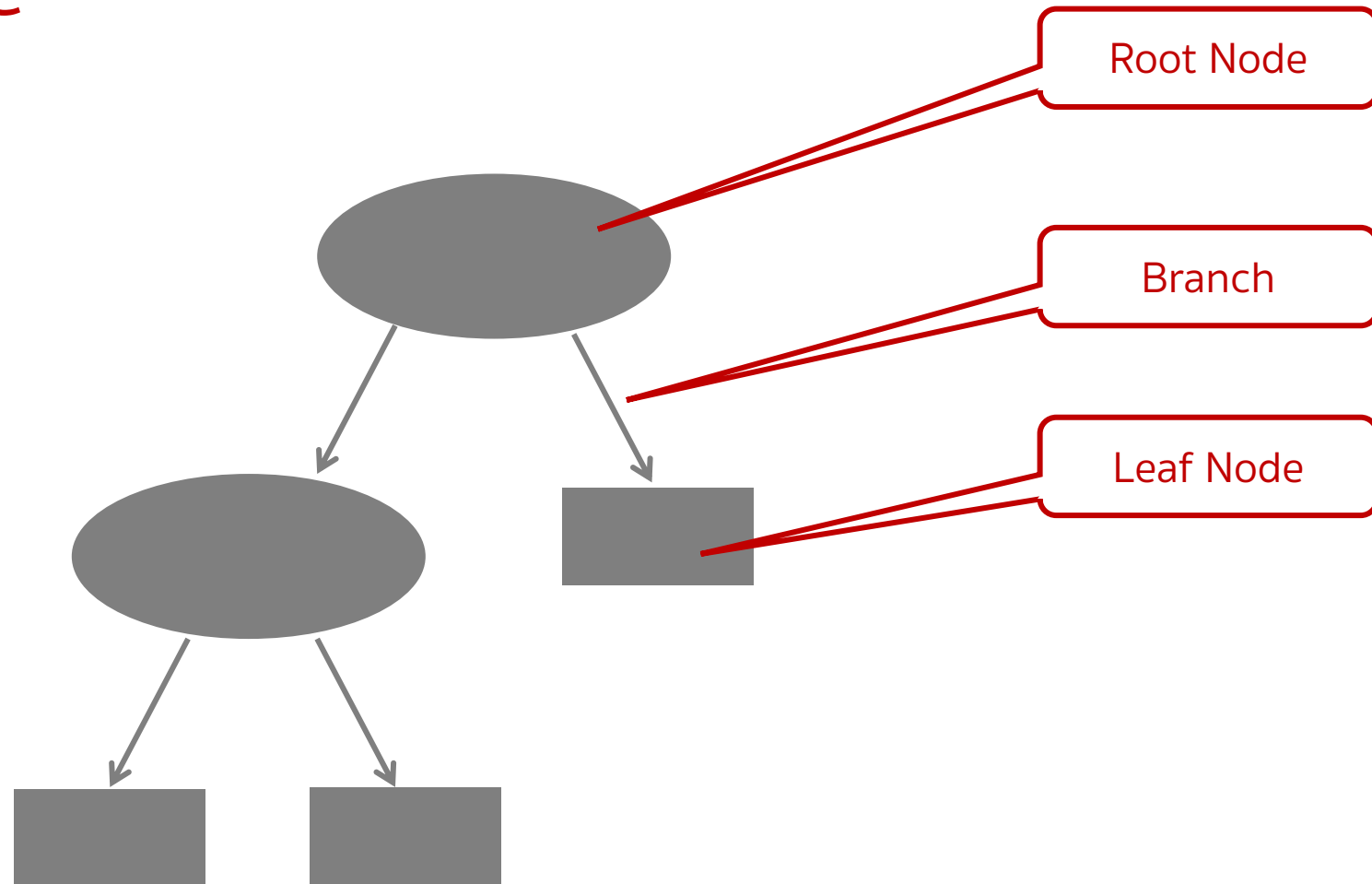


# Decision Tree

# Decision Tree

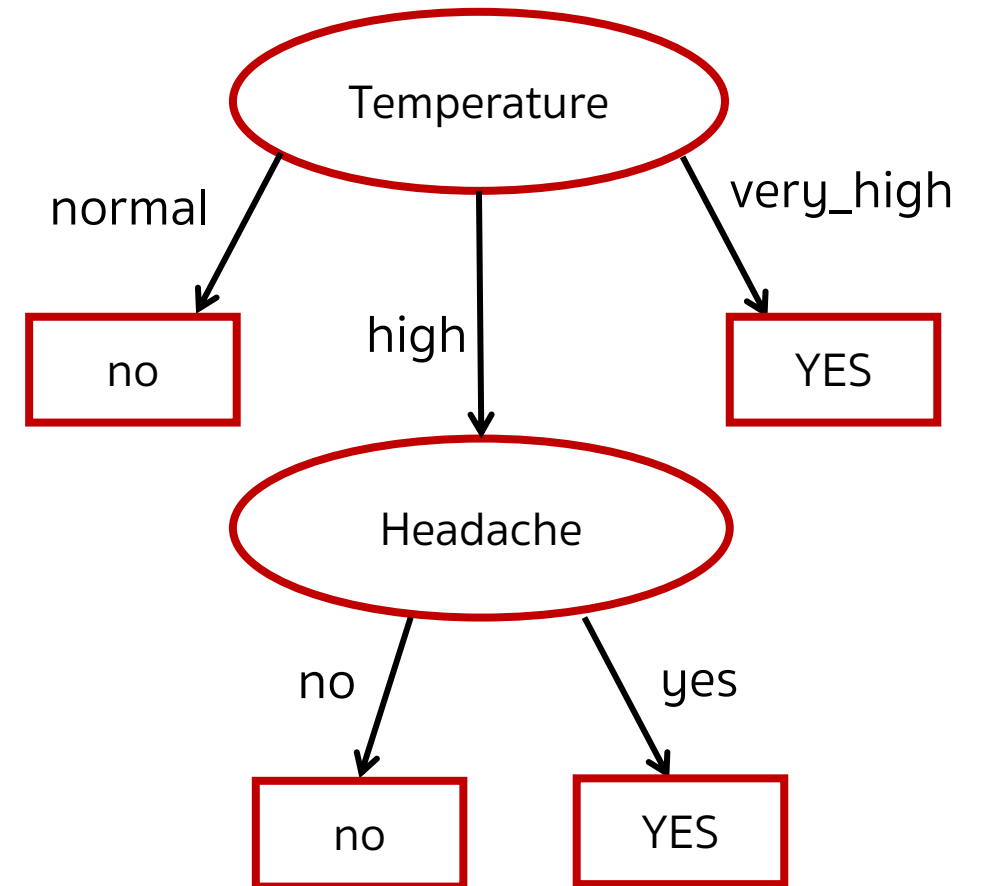


# Decision Tree



# Decision Tree

| CASE | TEMPERATURE | HEADACHE | NAUSEA | Flu |
|------|-------------|----------|--------|-----|
| 1    | high        | YES      | no     | YES |
| 2    | very_high   | YES      | YES    | YES |
| 3    | normal      | no       | no     | no  |
| 4    | high        | YES      | YES    | YES |
| 5    | high        | no       | YES    | no  |
| 6    | normal      | YES      | no     | no  |
| 7    | normal      | no       | YES    | no  |



# Iris Dataset

**iris setosa**



petal    sepal

**iris versicolor**



petal    sepal

**iris virginica**



petal    sepal

Iris Dataset

| sepal_length | sepal_width | petal_length | petal_width | species   |
|--------------|-------------|--------------|-------------|-----------|
| 5.1          | 3.5         | 1.4          | 0.2         | setosa    |
| 4.9          | 3.0         | 1.4          | 0.2         | setosa    |
| 4.7          | 3.2         | 1.3          | 0.2         | setosa    |
| 4.6          | 3.1         | 1.5          | 0.2         | setosa    |
| 5.0          | 3.6         | 1.4          | 0.2         | setosa    |
| ...          | ...         | ...          | ...         | ...       |
| 6.7          | 3.0         | 5.2          | 2.3         | virginica |
| 6.3          | 2.5         | 5.0          | 1.9         | virginica |
| 6.5          | 3.0         | 5.2          | 2.0         | virginica |
| 6.2          | 3.4         | 5.4          | 2.3         | virginica |
| 5.9          | 3.0         | 5.1          | 1.8         | virginica |

150 rows x 5 columns

ref: <https://machinelearninghd.com/iris-dataset-uci-machine-learning-repository-project/>

# COLAB: Decision Tree

- Go to the URL:
  - <https://colab.research.google.com/drive/1DdhhaIFbgNnJv98WgdBrF8J7nNKwpLHX#scrollTo=APQ3LUiwwvjg>

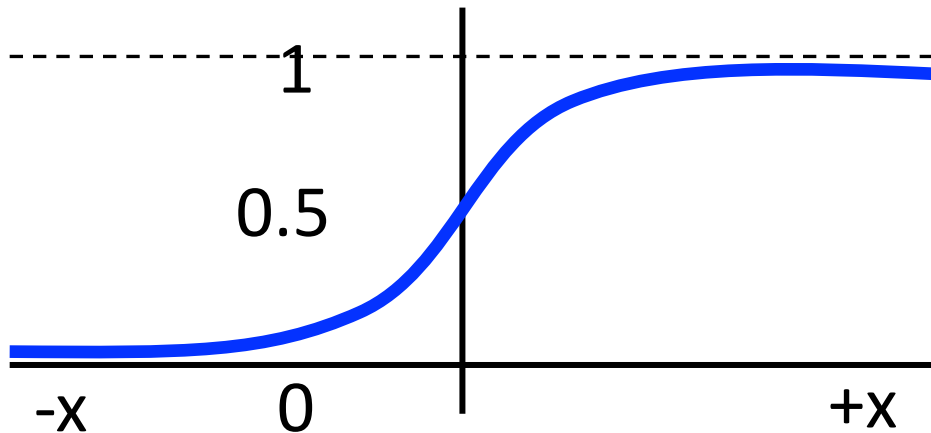




# Logistic Regression

# Logistic Regression

$$y = \frac{1}{1 + e^{-(\beta_0 + \beta_1 x_1 + \beta_2 x_2 + \dots + \beta_n x_n)}}$$



# COLAB: Logistic Regression

- Go to the URL:
  - <https://colab.research.google.com/drive/1DdhhalFbgNnJv98WgdBrF8J7nNKwpLHX#scrollTo=8aZWNrjMxJoy>





# Naiïve Bayes

# Naïve Bayes

The diagram shows the Naïve Bayes formula with arrows pointing to its components:  $P(c|x)$  is labeled as Posterior Probability,  $P(x|c)$  is labeled as Likelihood,  $P(c)$  is labeled as Class Prior Probability, and  $P(x)$  is labeled as Predictor Prior Probability.

$$P(c|x) = \frac{P(x|c)P(c)}{P(x)}$$

$$P(c|X) = P(x_1|c) \times P(x_2|c) \times \cdots \times P(x_n|c) \times P(c)$$

ref: <https://ichi.pro/pt/os-5-principais-algoritmos-de-classificacao-que-voce-realmente-usara-na-vida-204676738787569>

# COLAB: Naïve Bayes

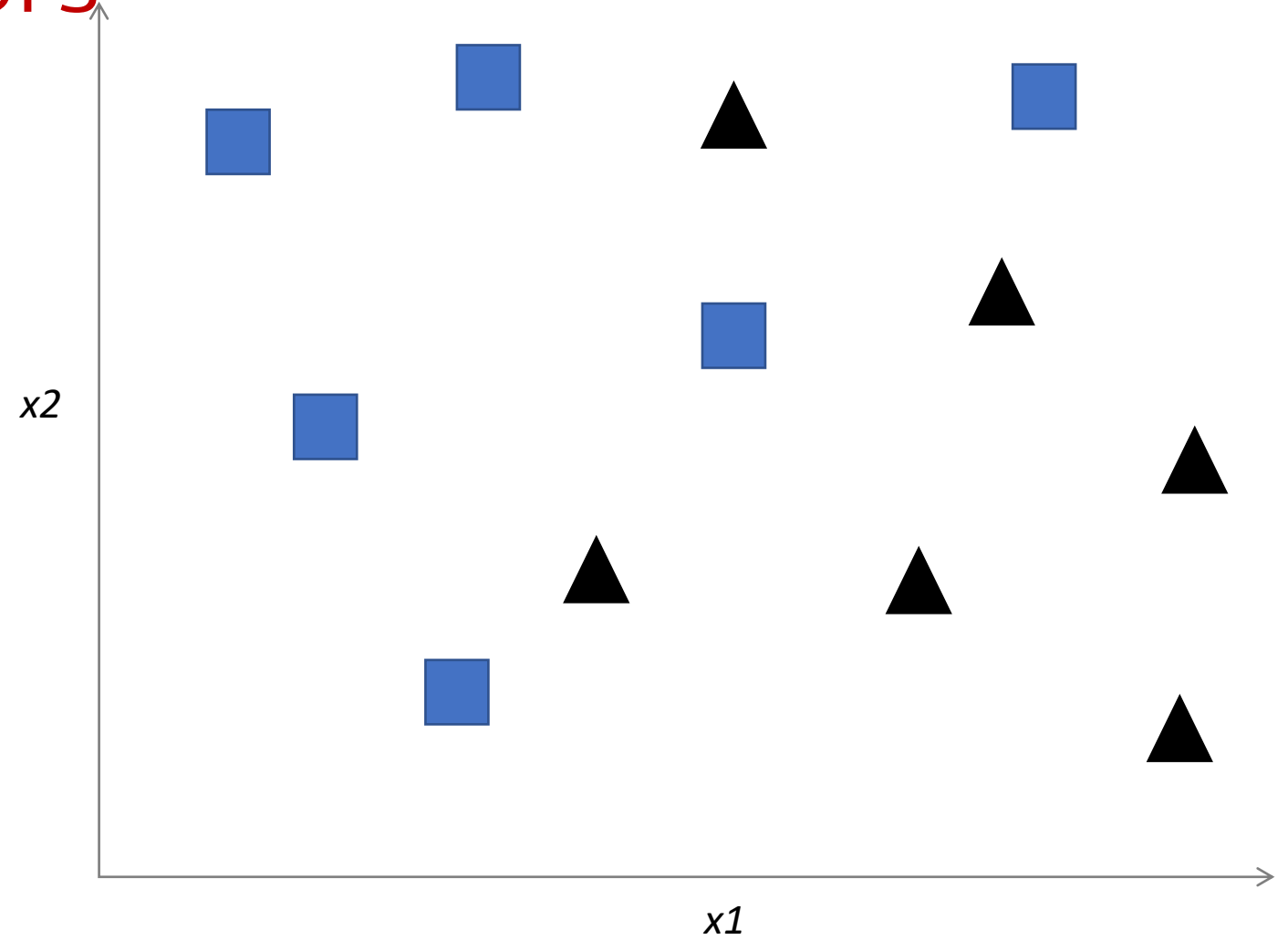
- Go to the URL:
  - <https://colab.research.google.com/drive/1DdhhalFbgNnJv98WgdBrF8J7nNKwpLHX#scrollTo=fyDdDgKXxaOv>





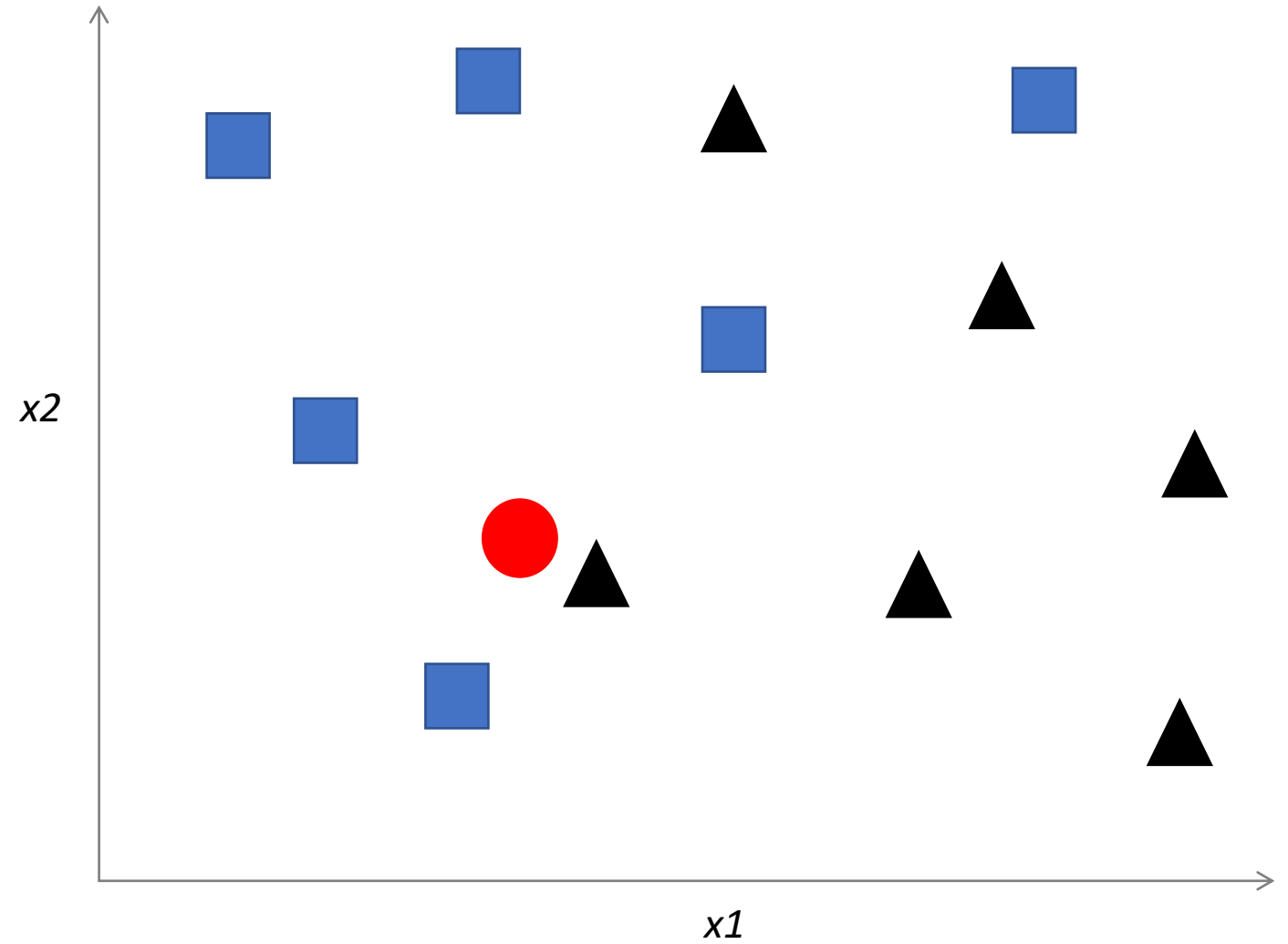
# K-Nearest Neighbors

# K-Nearest Neighbors



# K-NN

Which Classe?

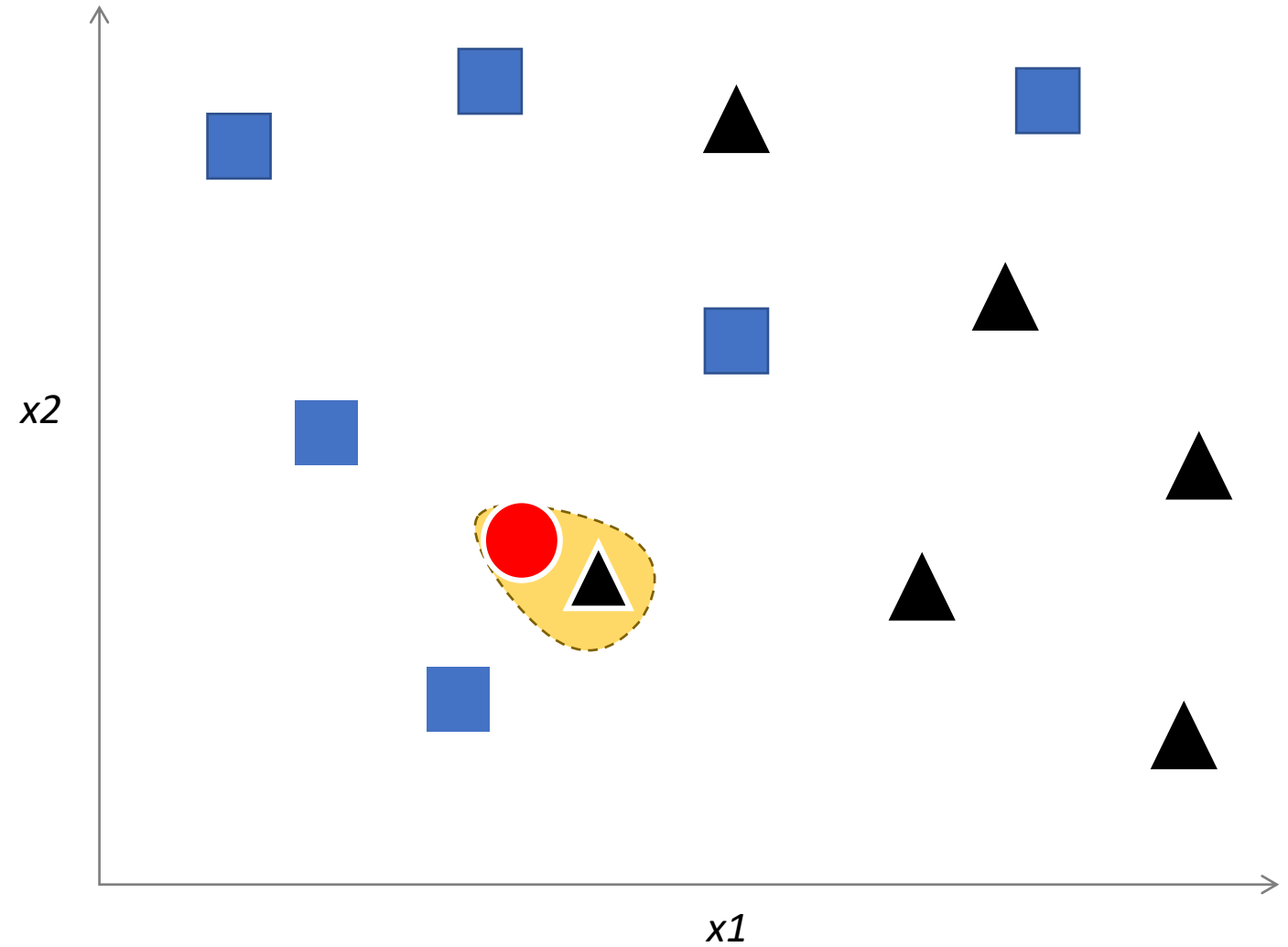


# K-NN

Which Classe?

-  $K = 1$

→ Triangle

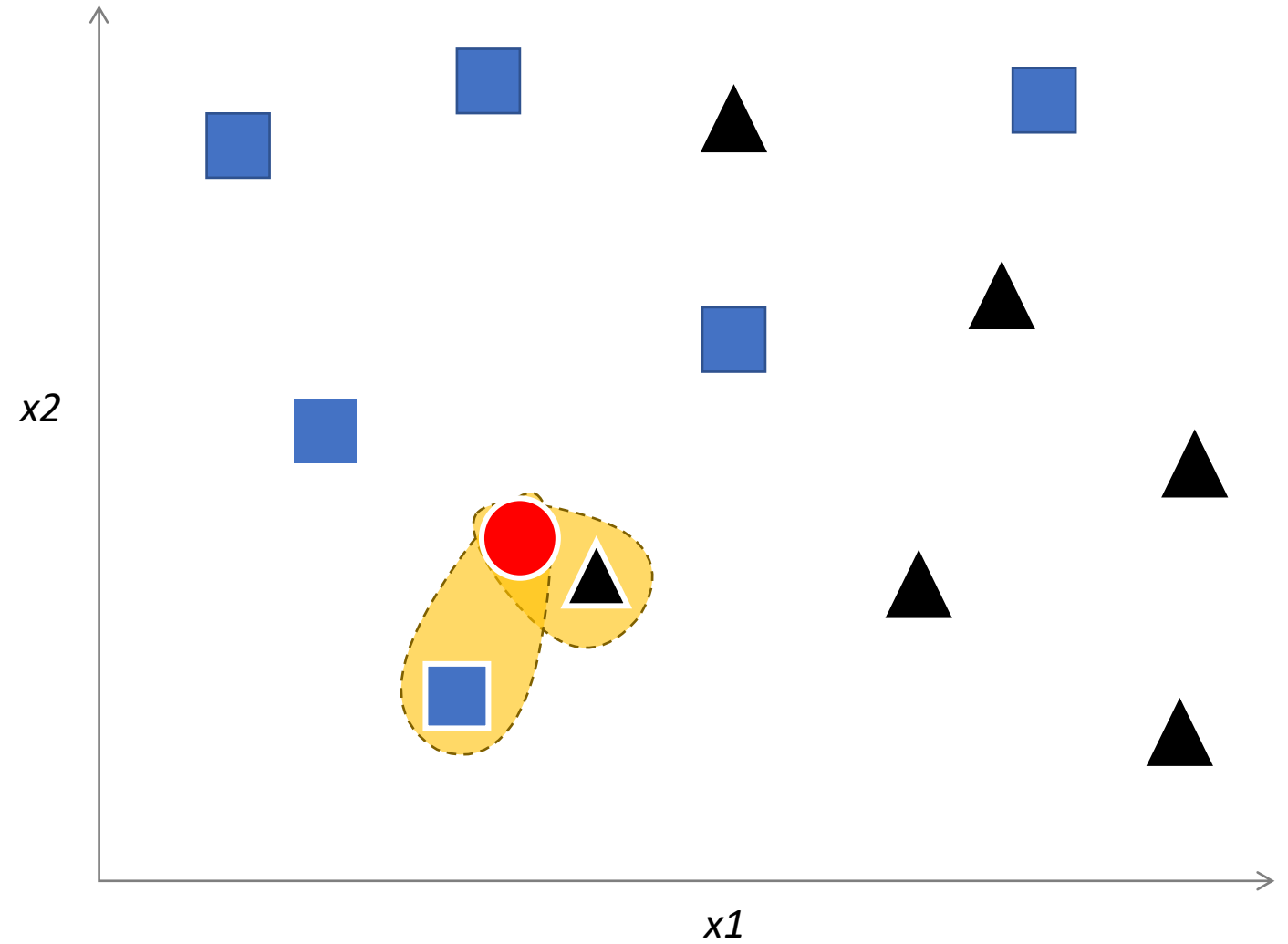


# K-NN

Which Classe?

-  $K = 2$

→ ???

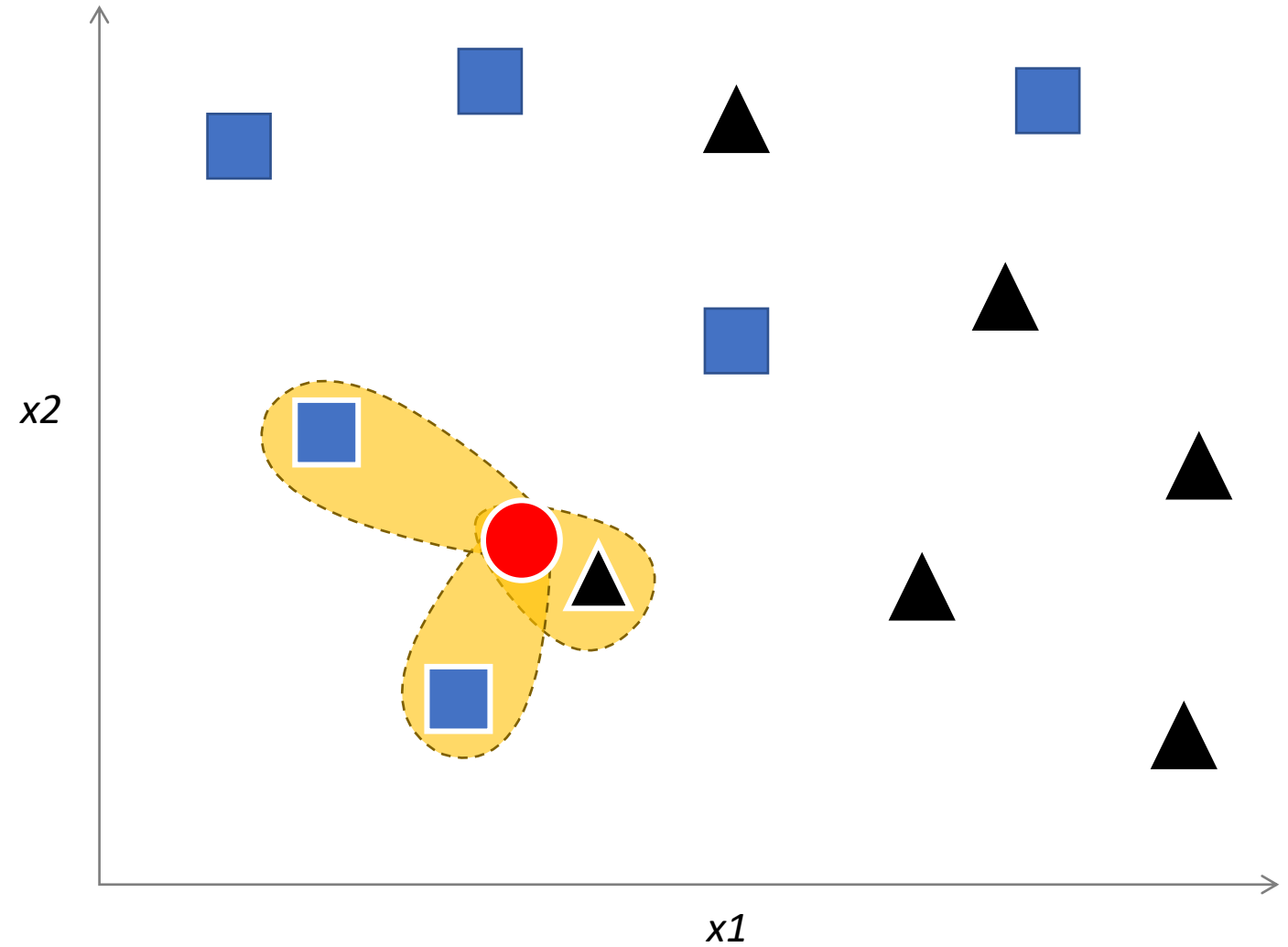


# K-NN

Which Classe?

-  $K = 3$

→ Square



# COLAB: K-NN

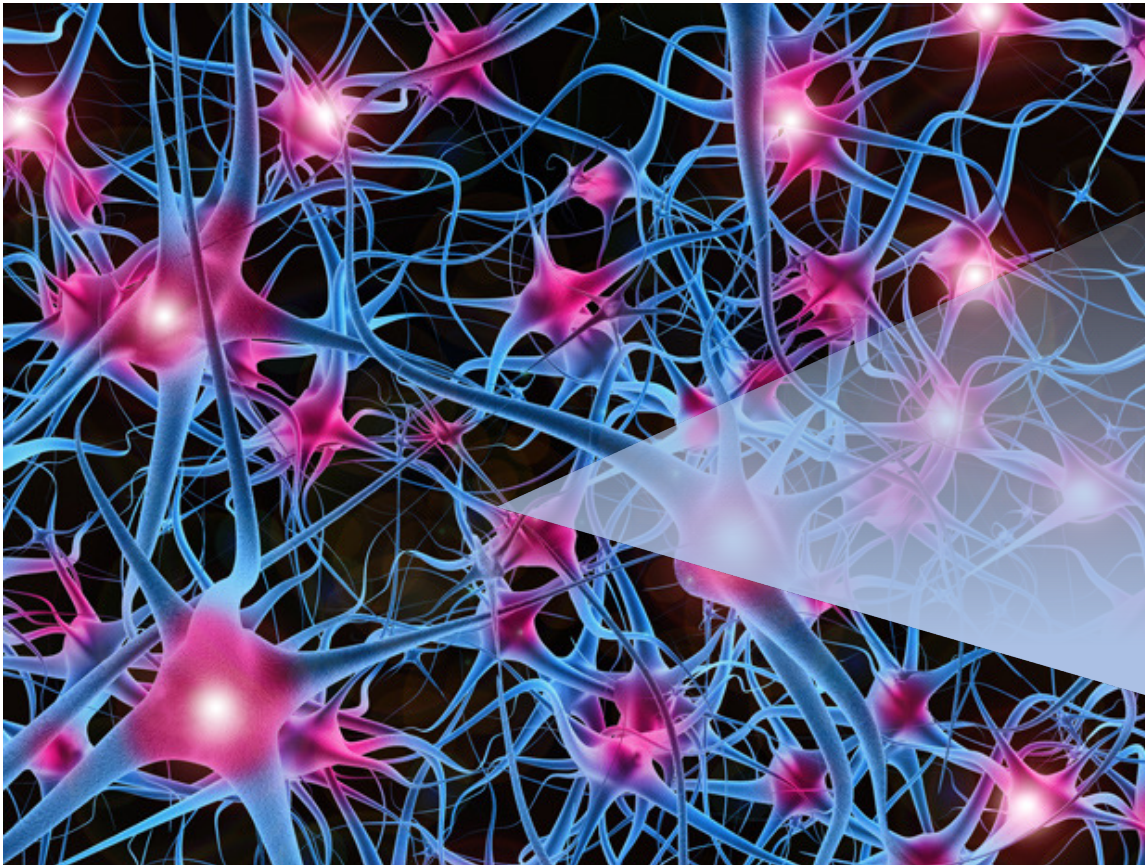
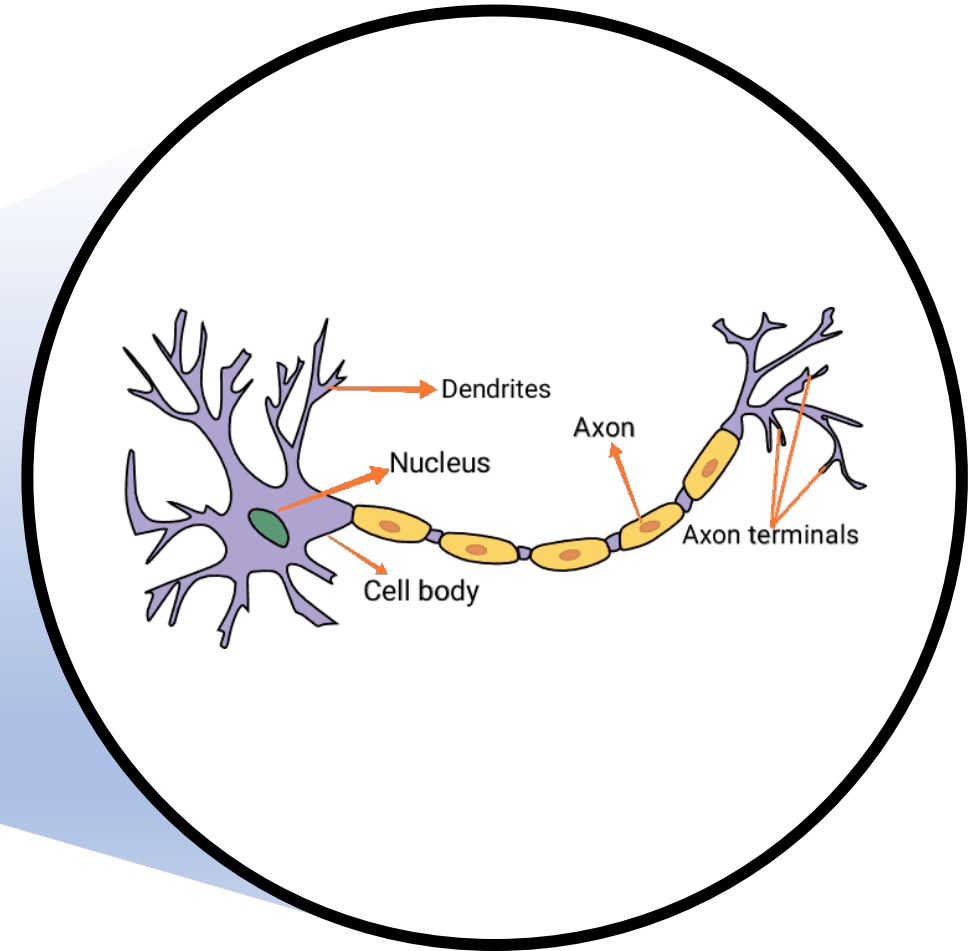
- Go to the URL:
  - <https://colab.research.google.com/drive/1DdhhalFbgNnJv98WgdBrF8J7nNKwpLHX?authuser=0#scrollTo=xH7YNQBQoOvD>





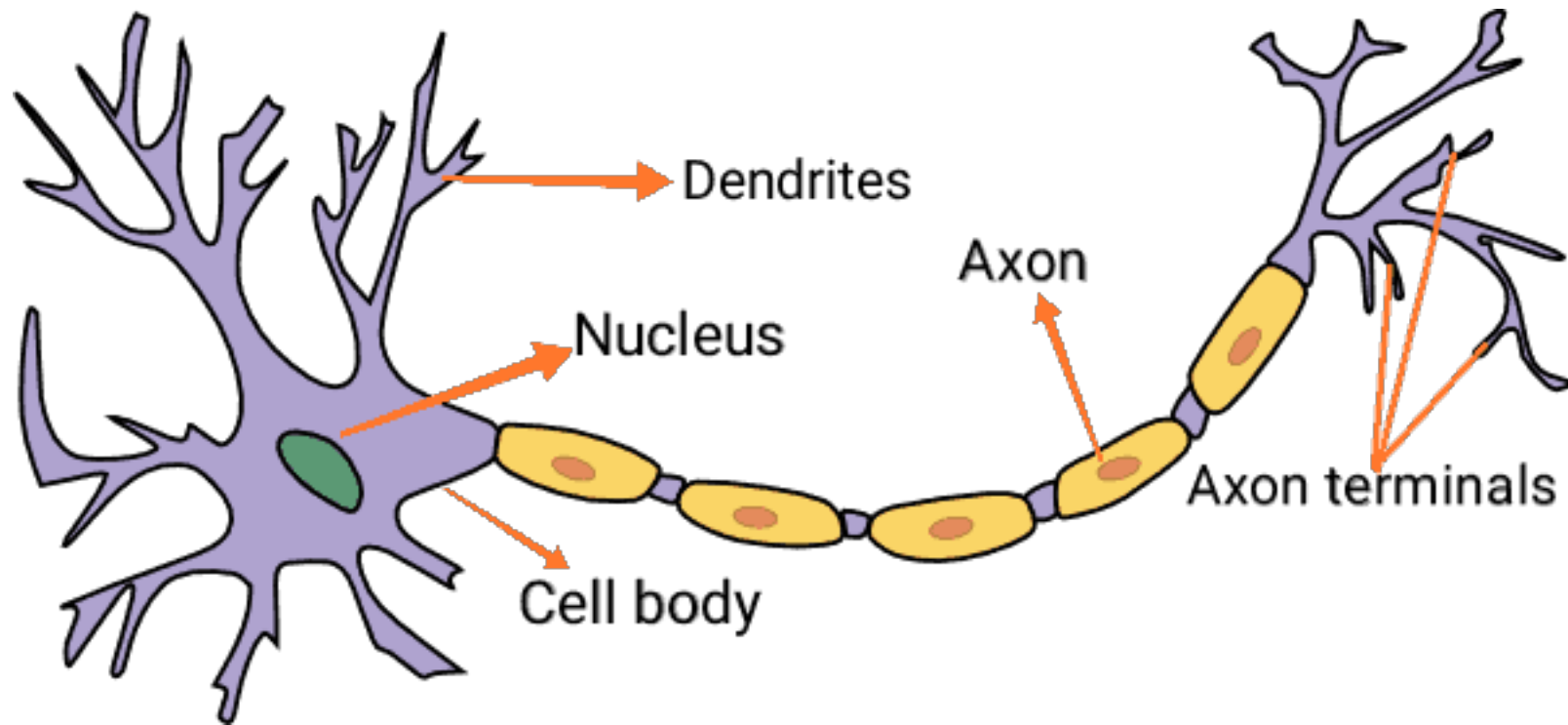
# Artificial Neural Network

# Neuron System



Ref: <https://spectrum.ieee.org/tech-talk/computing/software/biggest-neural-network-ever-pushes-ai-deep-learning>  
[https://aibusiness.com/document.asp?doc\\_id=761027](https://aibusiness.com/document.asp?doc_id=761027)

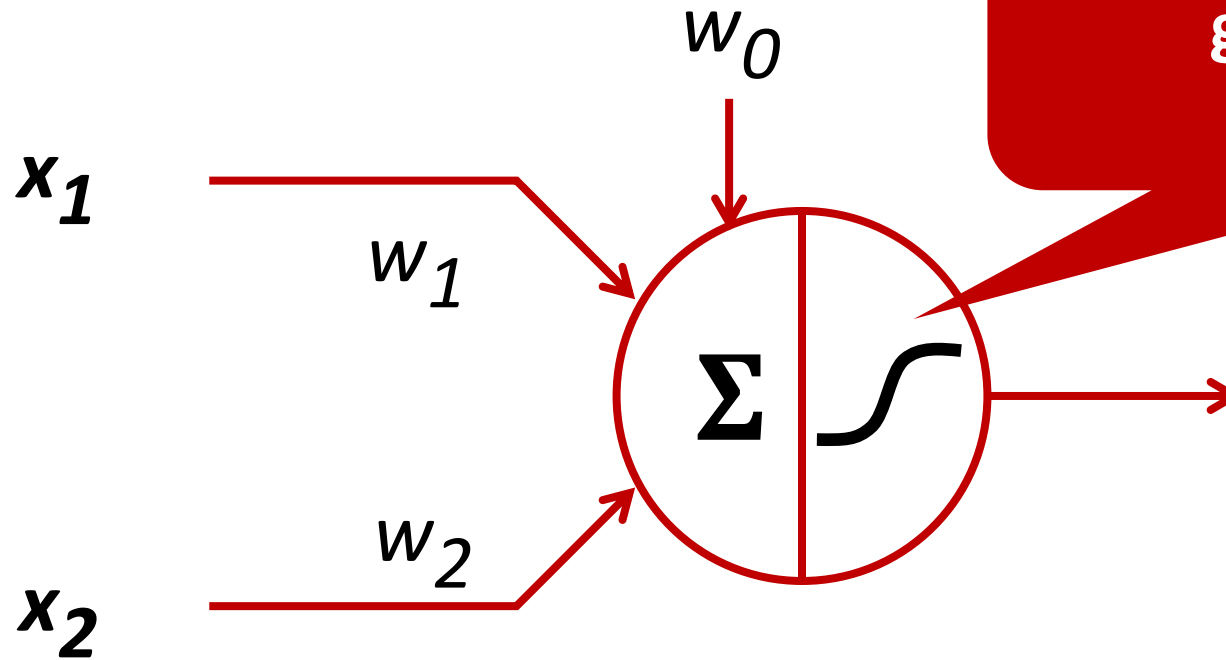
# Neuron



Ref: [https://aibusiness.com/document.asp?doc\\_id=761027](https://aibusiness.com/document.asp?doc_id=761027)

# Perceptron : Activation Function

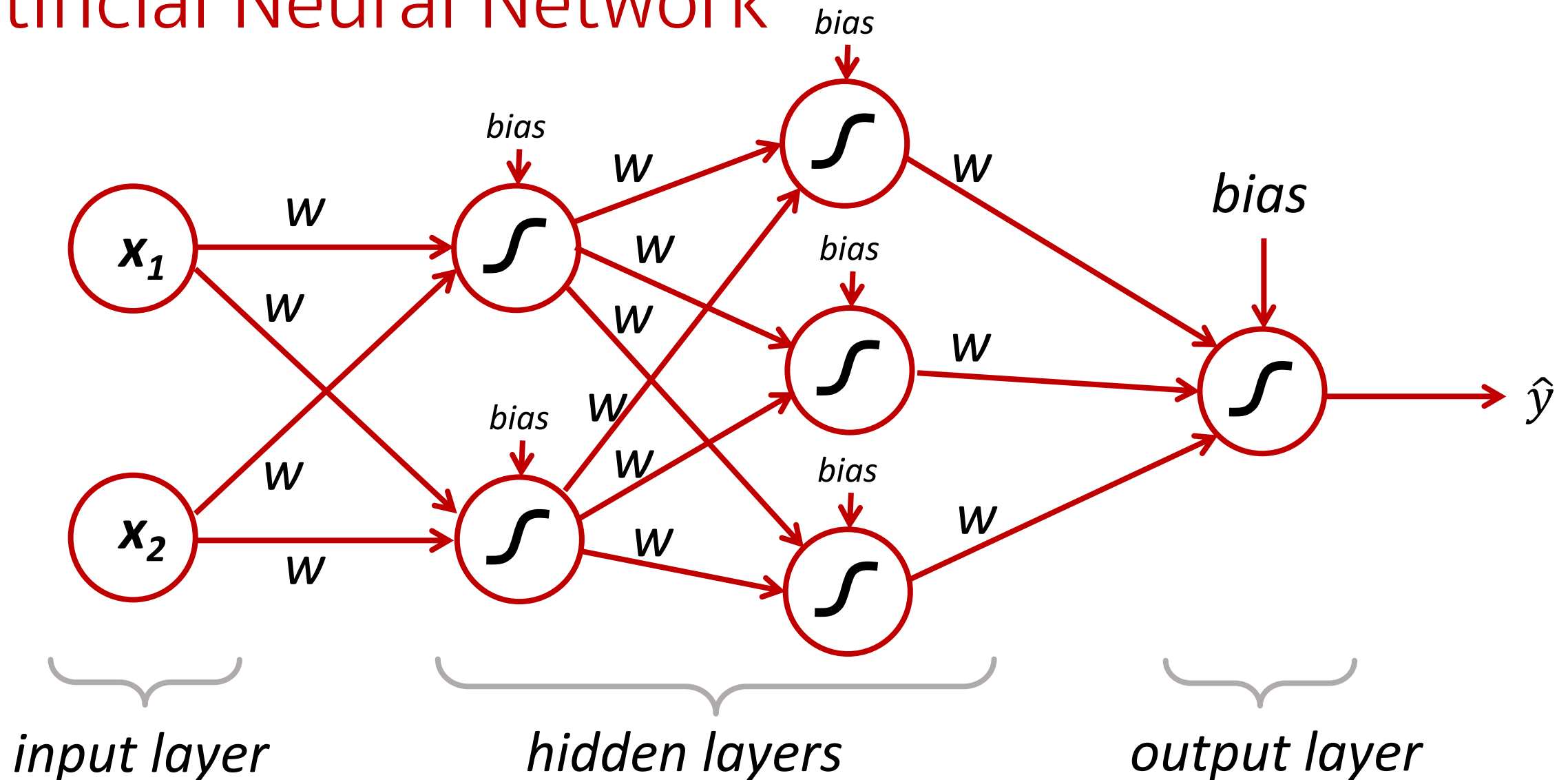
*Sigmoid Function (Logistic)*



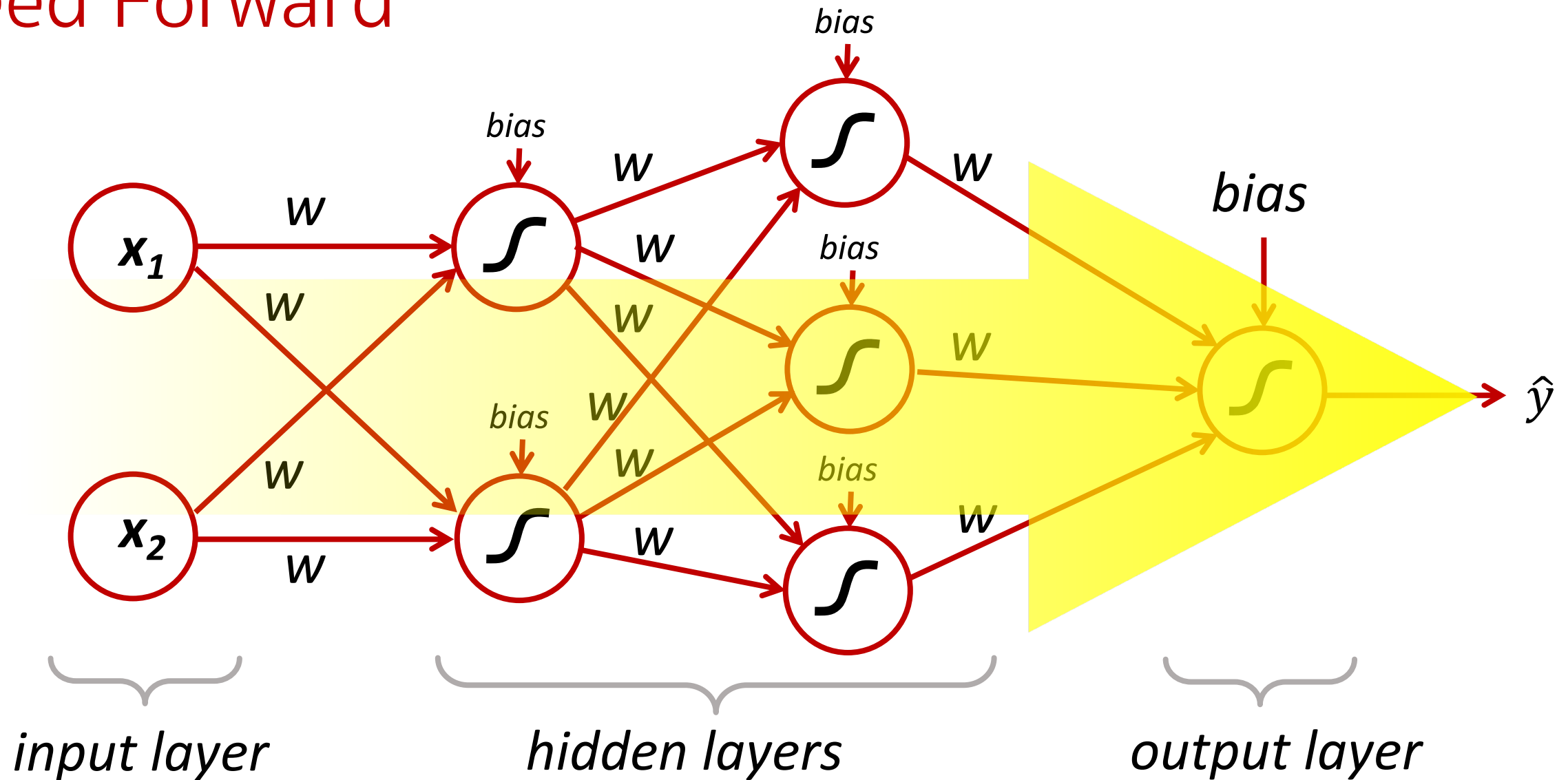
$$g(x) = \frac{1}{1 + e^{-h(X)}}$$

$$h(X) = w_0 + x_1 w_1 + x_2 w_2$$

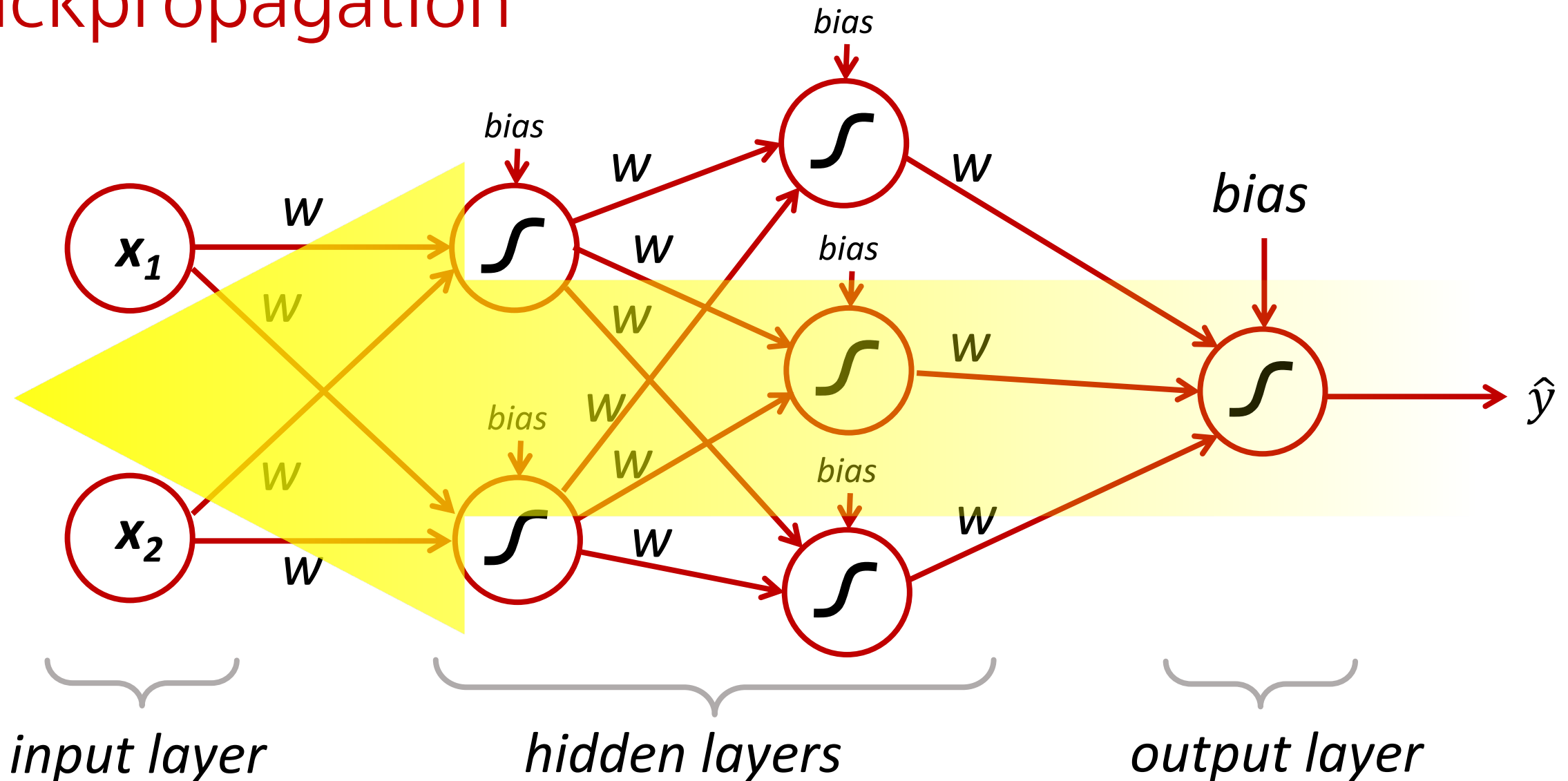
# Artificial Neural Network



# Feed Forward



# Backpropagation



# Weight Updated

$$w_j = w_j - \alpha \frac{\partial}{\partial w_j} J(W)$$



$$J(W) = \frac{1}{2m} \sum_{i=1}^m (\widehat{y^{(i)}} - y^{(i)})^2$$



$$\widehat{y^{(i)}} = w_0 + w_1 x_1^{(i)} + w_2 x_2^{(i)}$$

# Chain Rule With Partial Derivatives

$$\frac{\partial}{\partial w_j} J(\mathbf{w}) = \frac{1}{m} \sum_{i=1}^m \left( (\widehat{\mathbf{y}}^{(i)} - y^{(i)}) \cdot \frac{\partial}{\partial w_j} \widehat{\mathbf{y}}^{(i)} \right)$$

$$\frac{1}{2} \frac{\partial}{\partial \widehat{\mathbf{y}}^{(i)}} (\widehat{\mathbf{y}}^{(i)} - y^{(i)})^2 = (\widehat{\mathbf{y}}^{(i)} - y^{(i)})$$

$$\frac{\partial}{\partial w_0} \widehat{\mathbf{y}}^{(i)} = 1$$

$$\frac{\partial}{\partial w_1} \widehat{\mathbf{y}}^{(i)} = x_1^{(i)}$$

$$\frac{\partial}{\partial w_2} \widehat{\mathbf{y}}^{(i)} = x_2^{(i)}$$

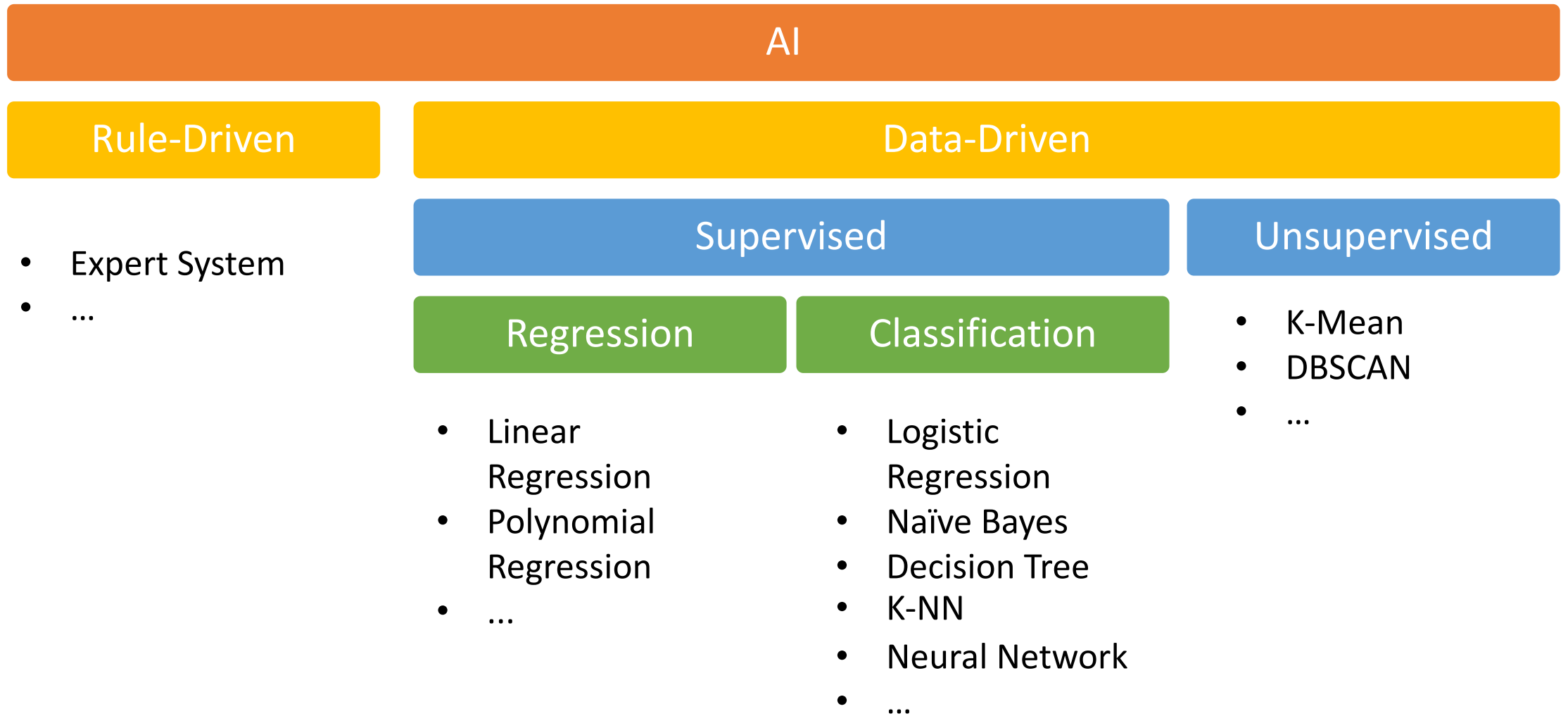
# COLAB: Artificial Neural Network

- Go to the URL:
  - <https://colab.research.google.com/drive/1DdhhaIFbgNnJv98WgdBrF8J7nNKwpLHX?authuser=0#scrollTo=r0znhtVBpyDY>

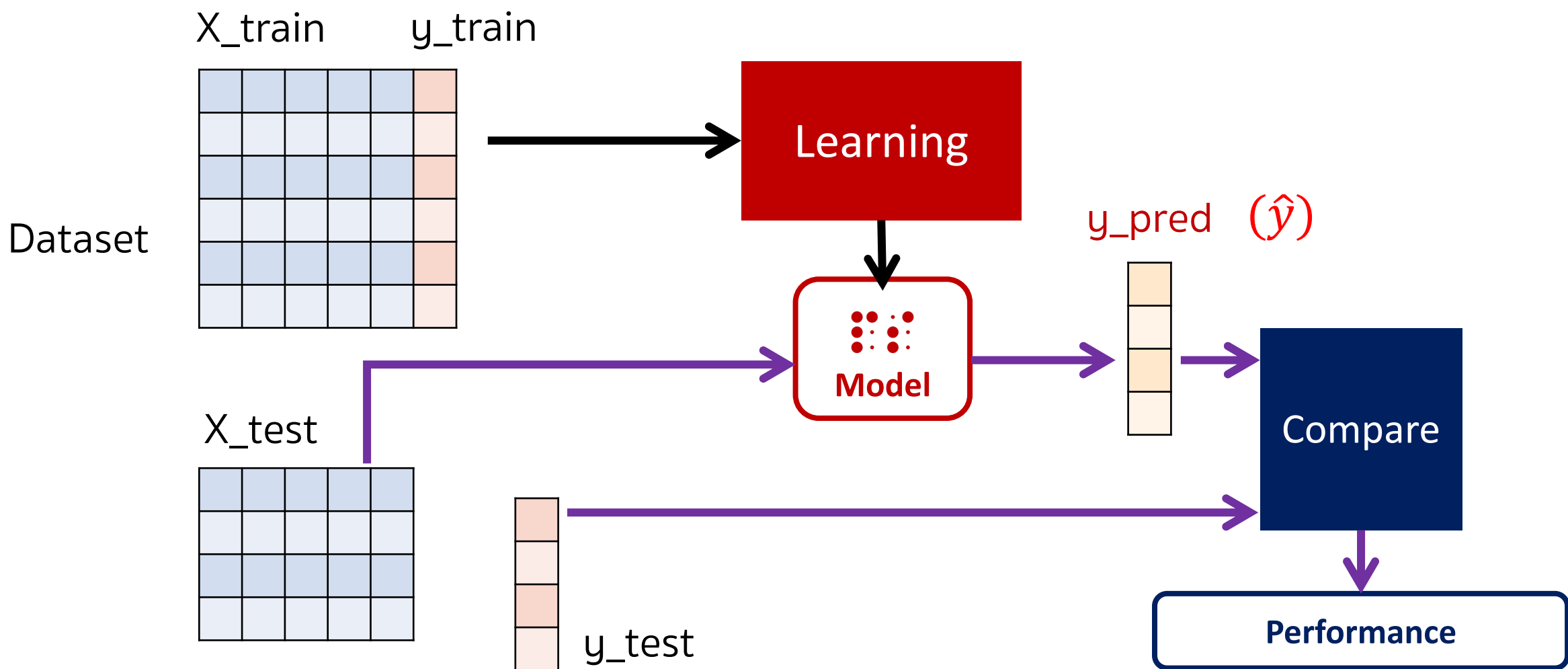


# SUMMARY

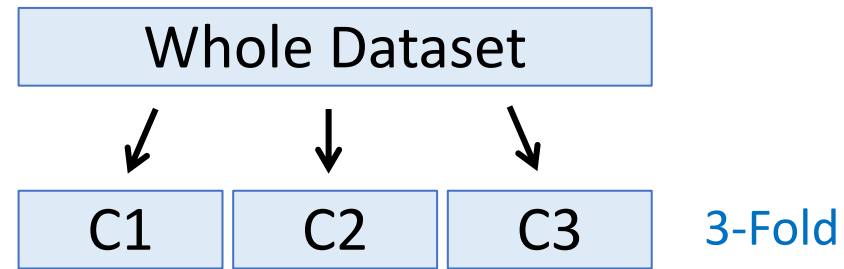
# Summary



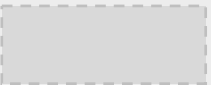
# Train-Test Split



# K-Fold Cross-Validation



3-Fold Cross-Validation

| Round          | Training Set                                                                                | Test Set | Evaluation               |
|----------------|---------------------------------------------------------------------------------------------|----------|--------------------------|
| 1              |  C2 C3   | C1       | e1                       |
| 2              | C1  C3   | C2       | e2                       |
| 3              | C1 C2  | C3       | e3                       |
| Average Result |                                                                                             |          | $\frac{e1 + e2 + e3}{3}$ |

# Python Ecosystem for AI

