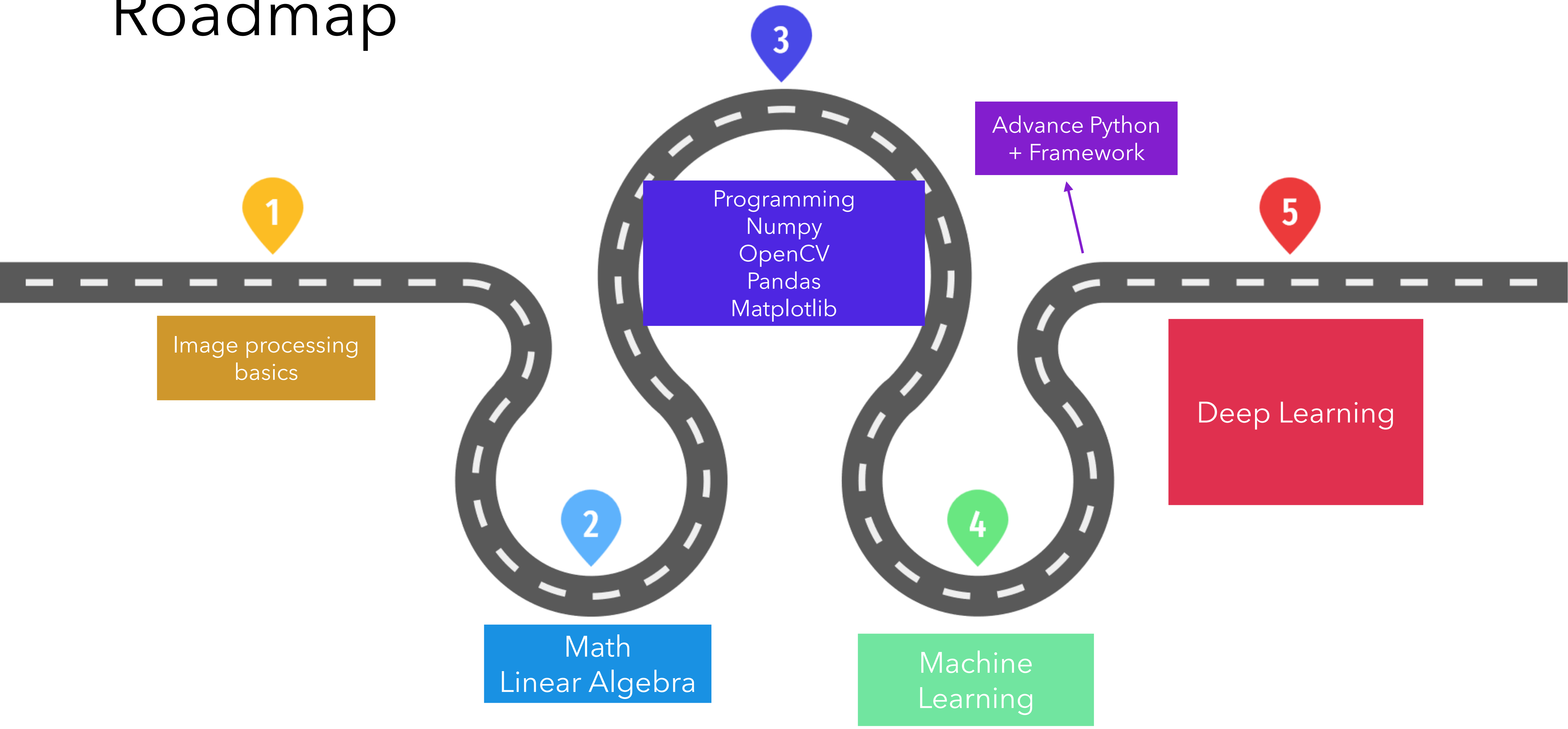
Computer Vision

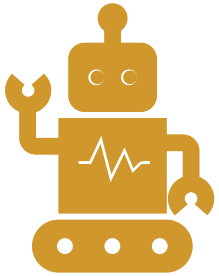
CVI620

Session 11
02/2025

Roadmap


1
Image processing basics
2
Math
Linear Algebra
3
Programming
Numpy
OpenCV
Pandas
Matplotlib
Advance Python
+ Framework
4
Machine Learning
5
Deep Learning

AI



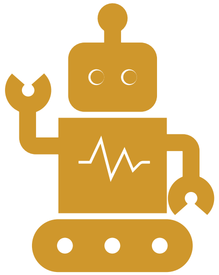
Classic AI (genetic algorithm, ant colony optimization, ...)

Machine Learning

Classic ML

Deep Learning

AI



Classic AI (genetic algorithm, ant colony optimization, …)

Our focus

Machine Learning

Classic ML

Deep Learning

Name some algorithms?

Name some algorithms?

binary search, insertion sort, Hanoi tower, breadth first search, …

ML

- Machine learning is also a set of algorithms
- But has some distinctive difference:
  - Data driven
  - Don’t have a closed-form formula
  - Results that are close to the best possible outcome, even if not exact

Example

Give a closed-form algorithm that predicts house prices?!

Example

Too many loops and conditions

```
if room_number > 4 and location == 3:
        return 3000
elif ...
```

Error prone without validation

ML’s Main Fuel

Instead of being manually programmed, ML models improve automatically by analyzing **da**

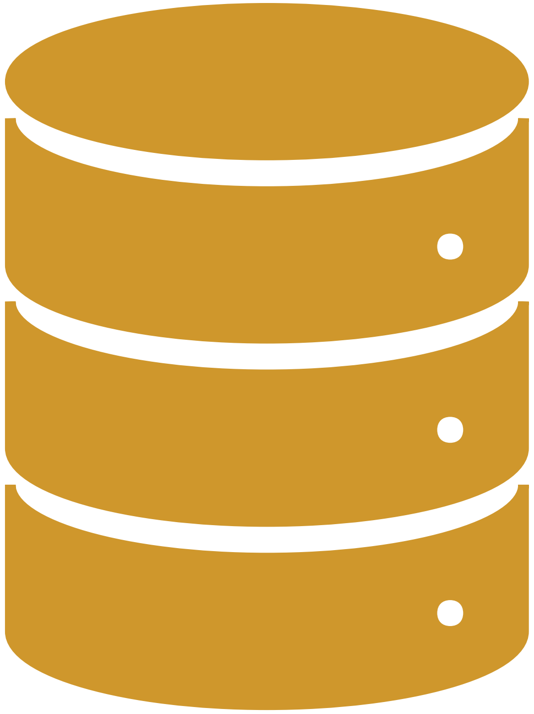
Example

Look at previously rented houses and give results.

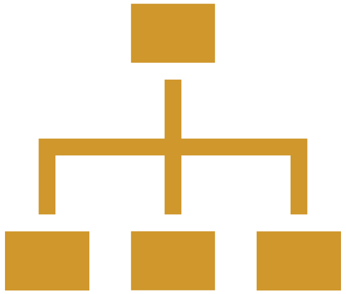
But still do it with an algorithm!

Data

- So before thinking about solving any problem with ML we should first have data:
  - Do we have any dataset?
  - If we don't have any dataset, how can we collect?
  - How much data we need?



Type of Data



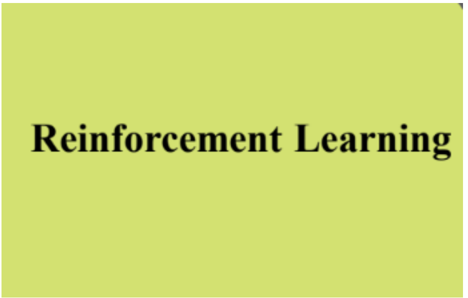
Structured

CSV, SQL, Excel, …

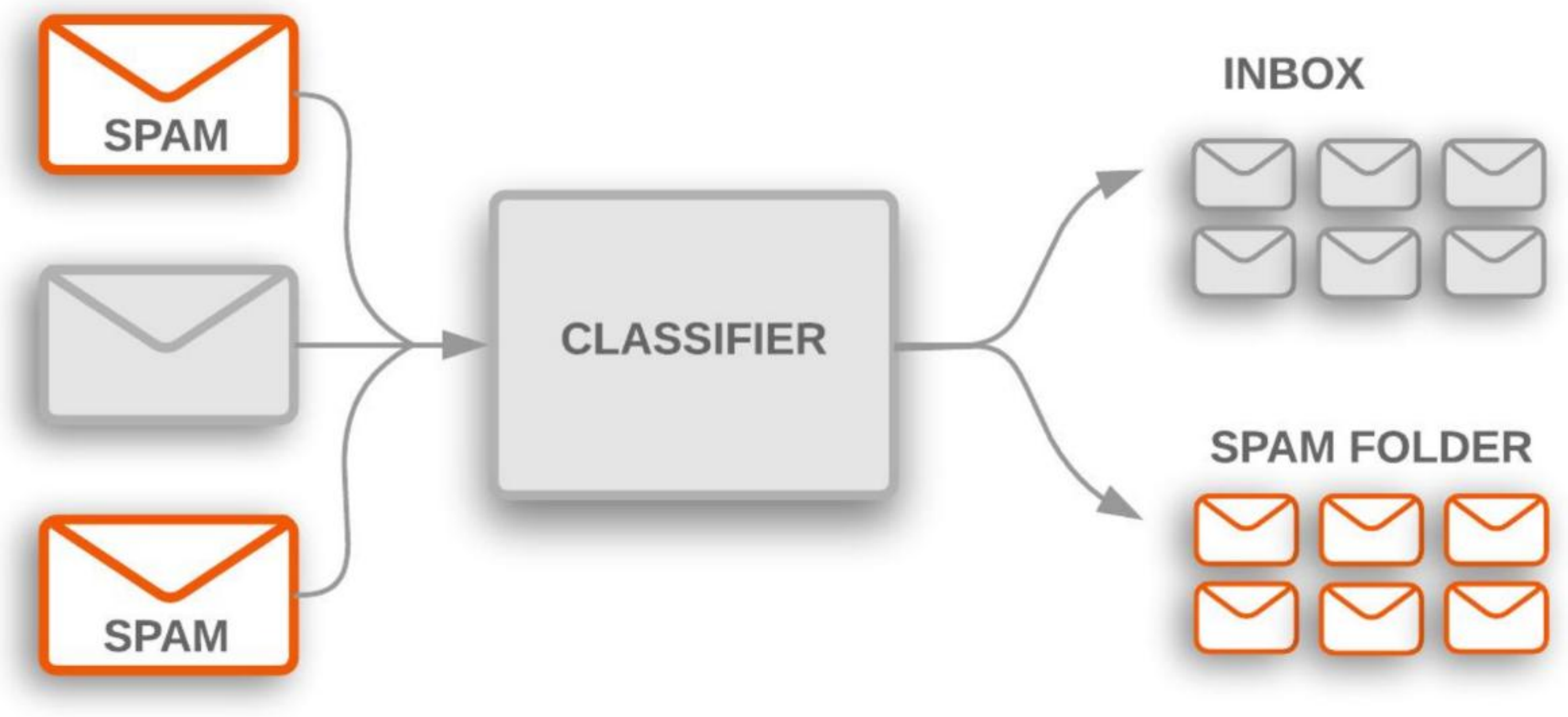
Unstructured

Images, videos, audio, …

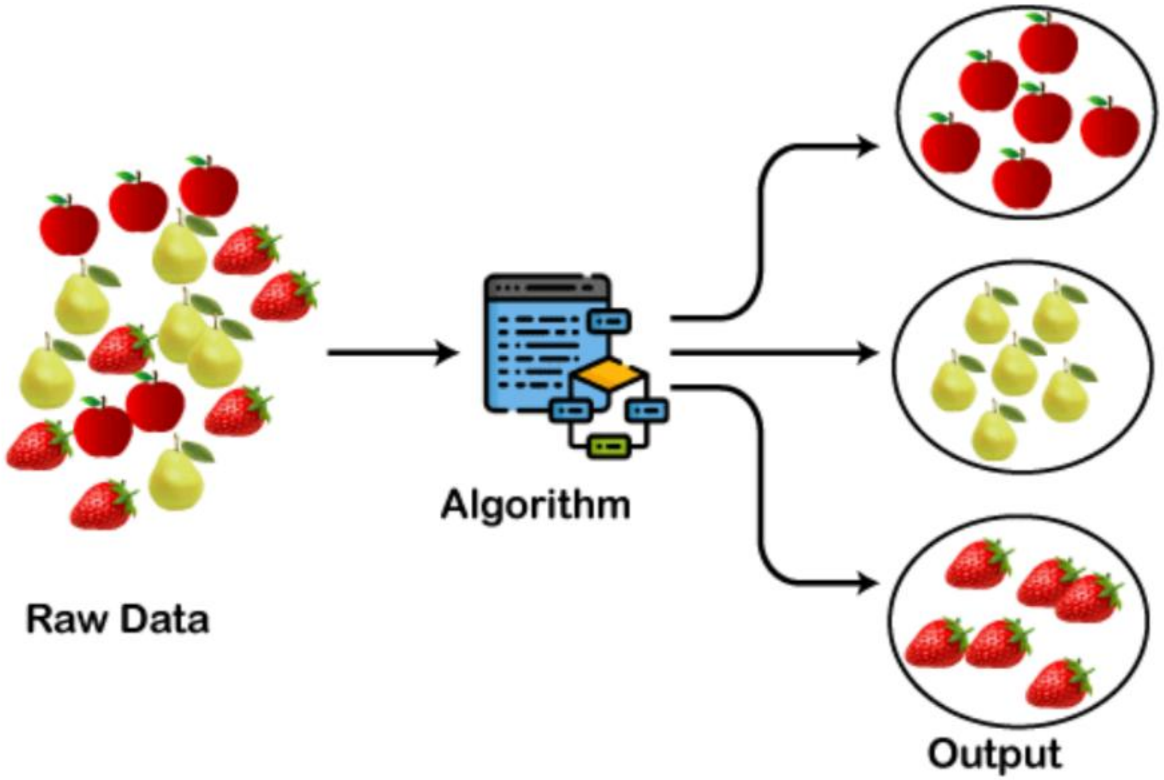
ML Algorithm Categorizations



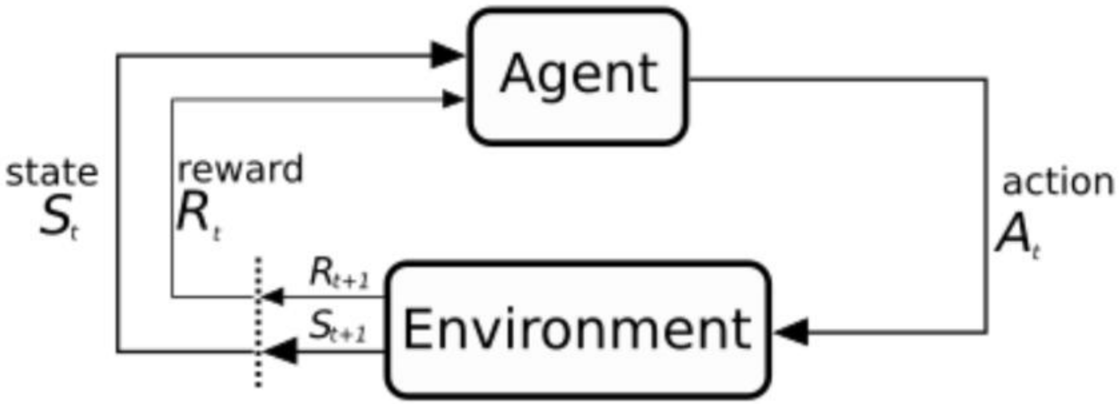
Supervised Learning



Unsupervised Learning

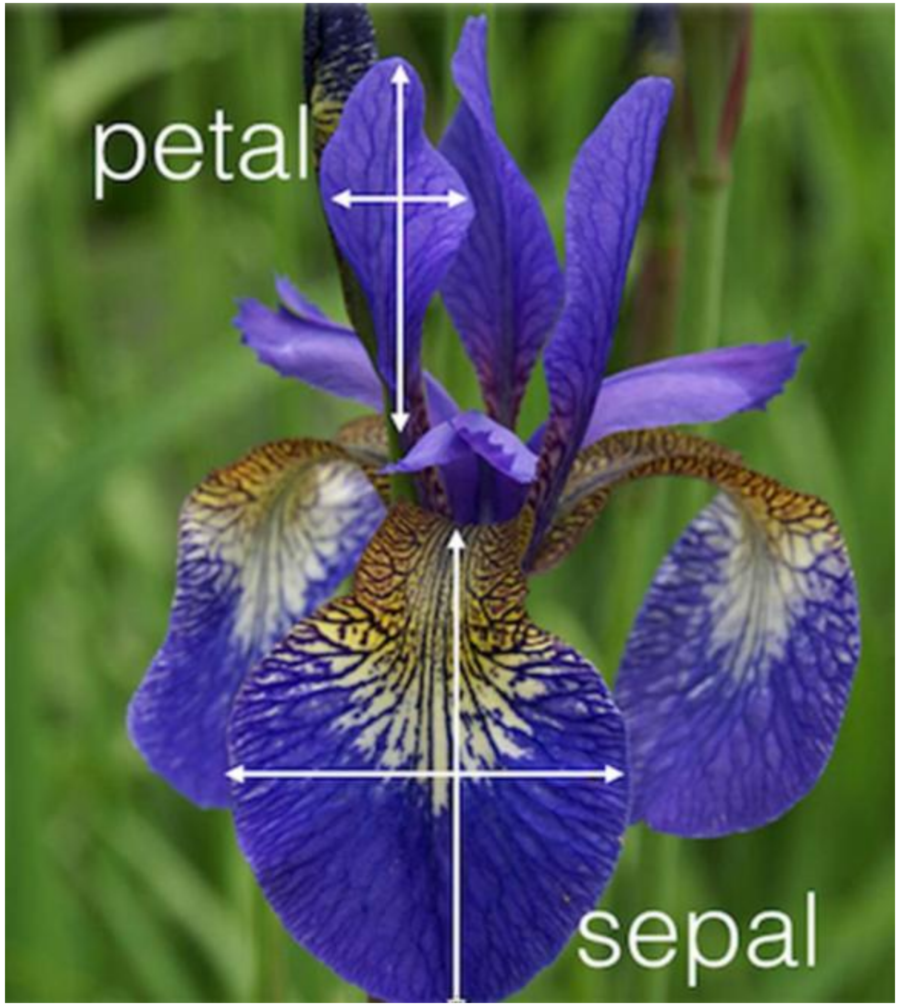


Supervised Learning



Iris Flowers Dataset

- 150 samples (rows)
- 3 classes (Virginia, Versicolor, Setosa), each with 50 samples
- Features (columns): height and width of petal and sepal



MNIST Dataset

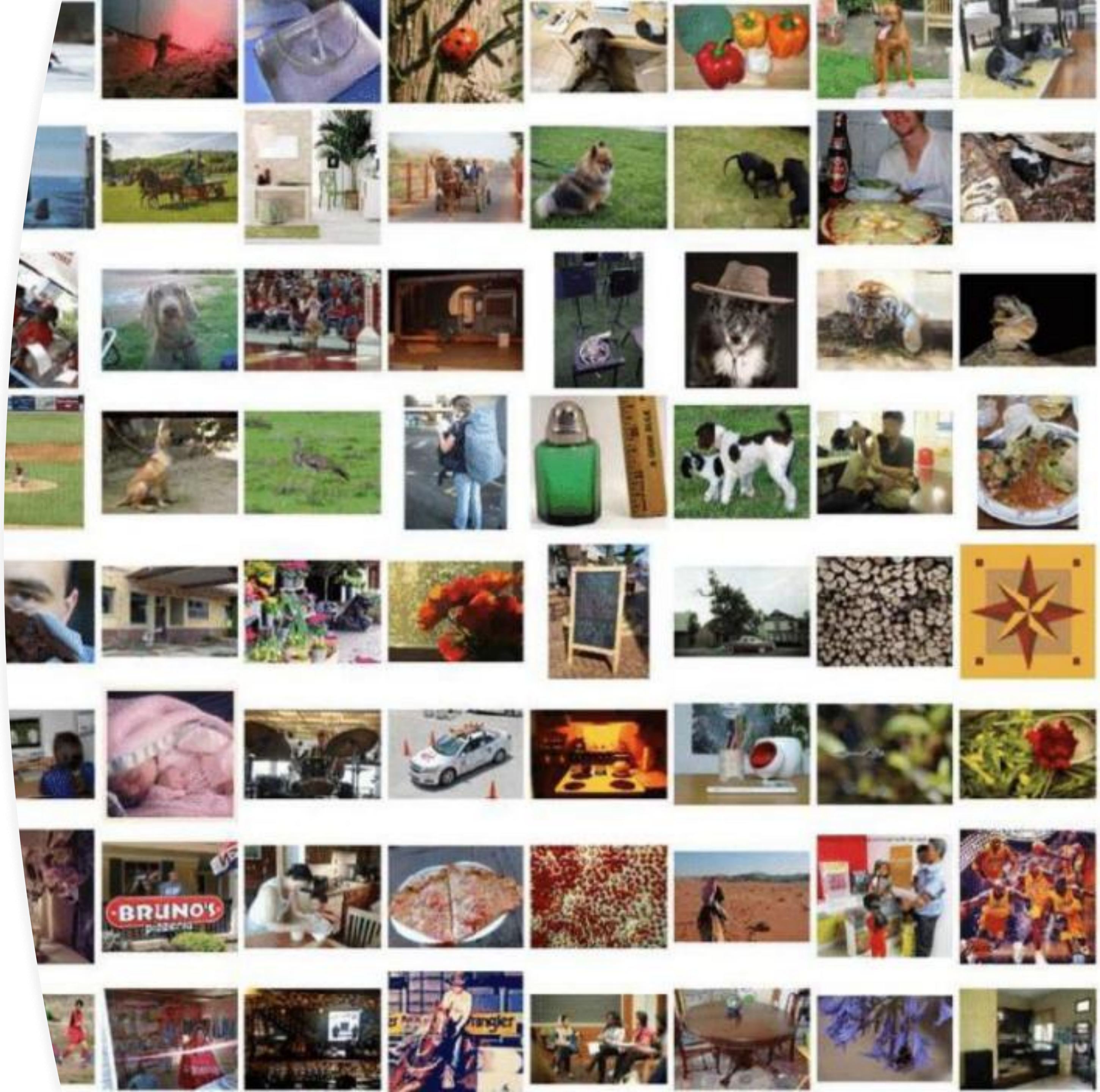
- 60,000 train data, 10,000 test data
- 28*28 one-channel images
- Classes: 1 to 10

SMILES Dataset

- 13,165 images
- 64*64
- Smile or not-smile

ImageNet Dataset

- More than 14 million images
- 3 channeled with varying sizes
- More than 22,000 classes



Pandas Library

Data manipulation and analysis
Inspired by SQL

First Algorithm

Case Study1

Case Study2

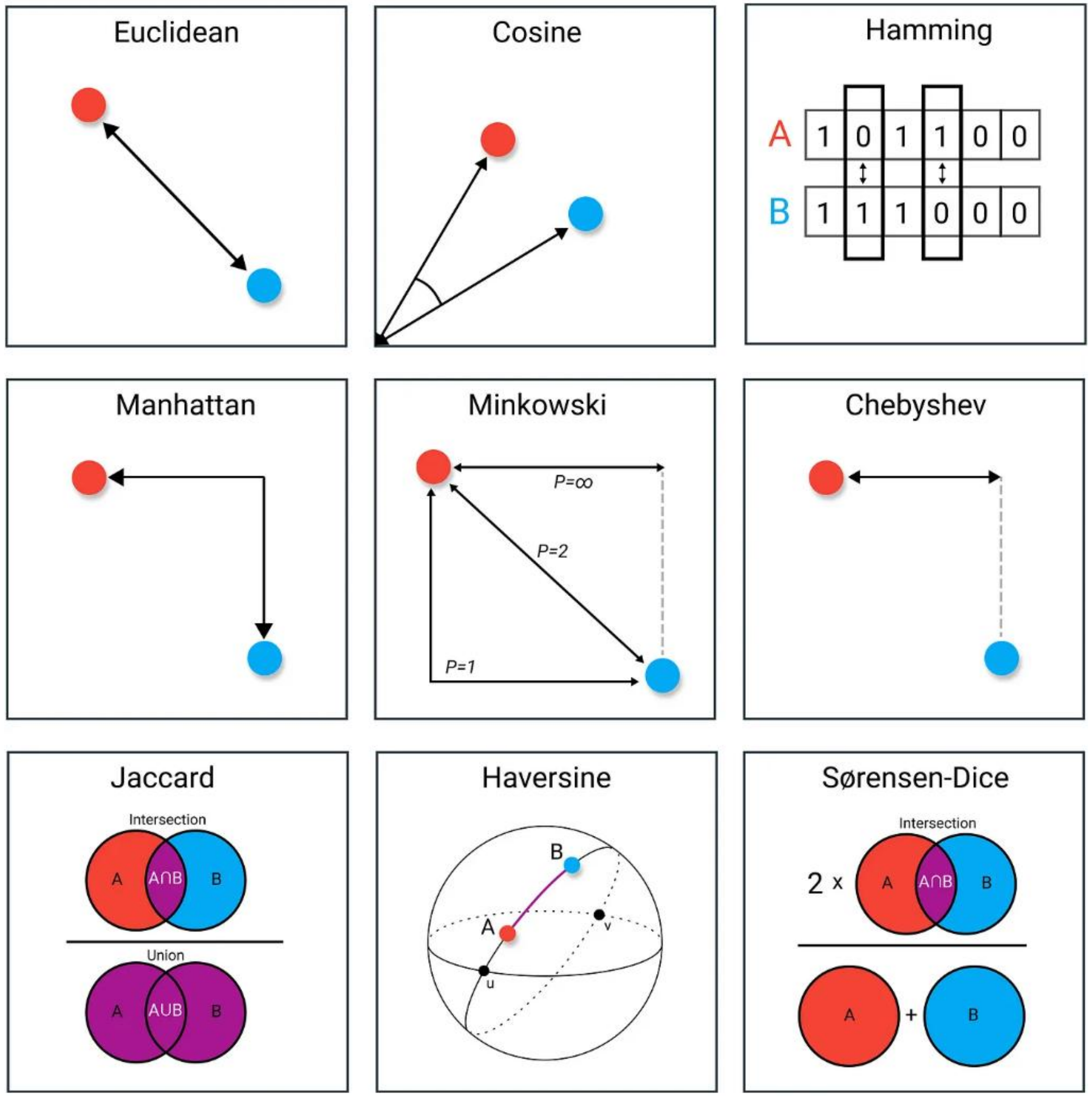
Weight(x2)	***Height(y2)***	***Class***
51	*167*	*Underweight*
62	*182*	*Normal*
69	*176*	*Normal*
64	*173*	*Normal*
65	*172*	*Normal*
56	*174*	*Underweight*
58	*169*	*Normal*
57	*173*	*Normal*
55	*170*	*Normal*

57 kg	**170 cm**	**?**

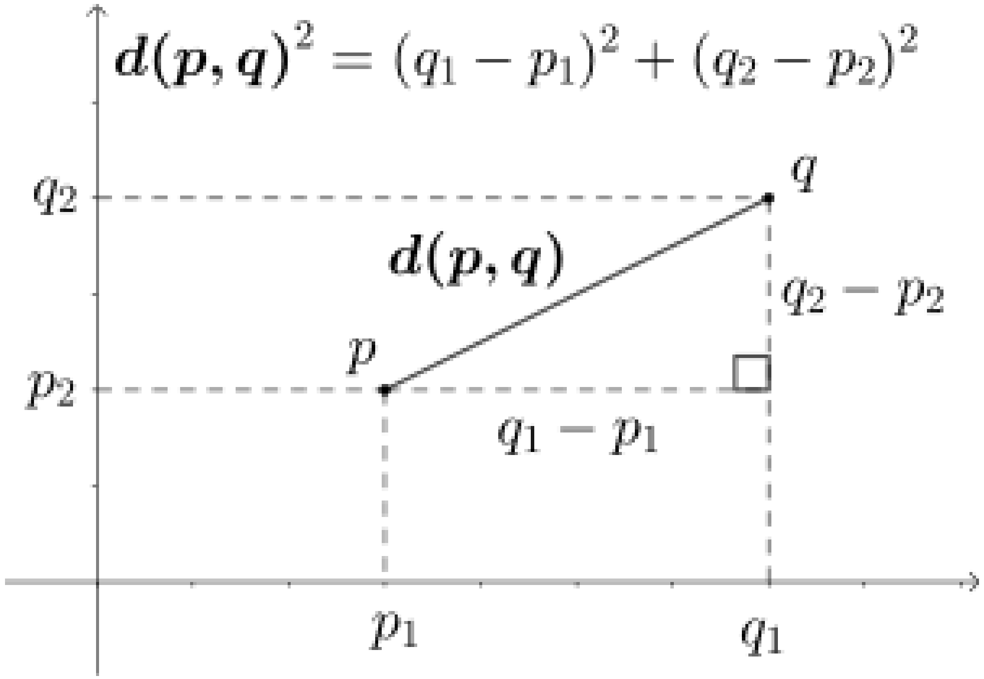
Similarity Metrics

- Distance based
- Statistical
- Probabilistic
- String-based

Distance Based Similarities



Euclidean Distance



Calculate Distance

Data: (x1, y1) = (57, 170)

Weight(x2)	Height(y2)	Class	Euclidean Distance
51	167	Underweight	6.7
62	182	Normal	13
69	176	Normal	13.4
64	173	Normal	7.6
65	172	Normal	8.2
56	174	Underweight	4.1
58	169	Normal	1.4
57	173	Normal	3
55	170	Normal	2

Closest Neighbors

Weight(x2)	Height(y2)	Class	Euclidean Distance
51	167	Underweight	6.7
62	182	Normal	13
69	176	Normal	13.4
64	173	Normal	7.6
65	172	Normal	8.2
56	174	Underweight	4.1
58	169	Normal	1.4
57	173	Normal	3
55	170	Normal	2

k = 3

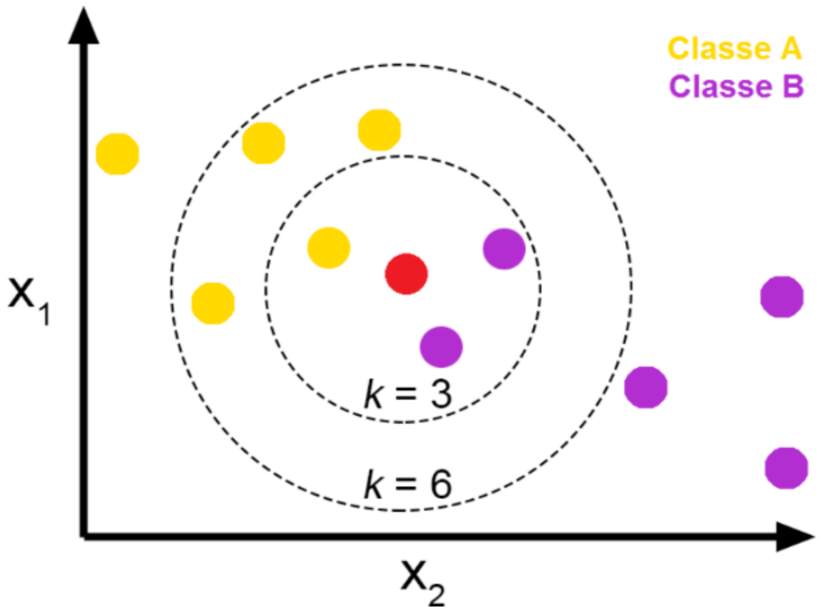
57 kg	**170 cm**	**?**

Major Vote

Class	Euclidean Distance
Underweight	6.7
Normal	13
Normal	13.4
Normal	7.6
Normal	8.2
Underweight	4.1
Normal	1.4
Normal	3
Normal	2

k = 3

KNN

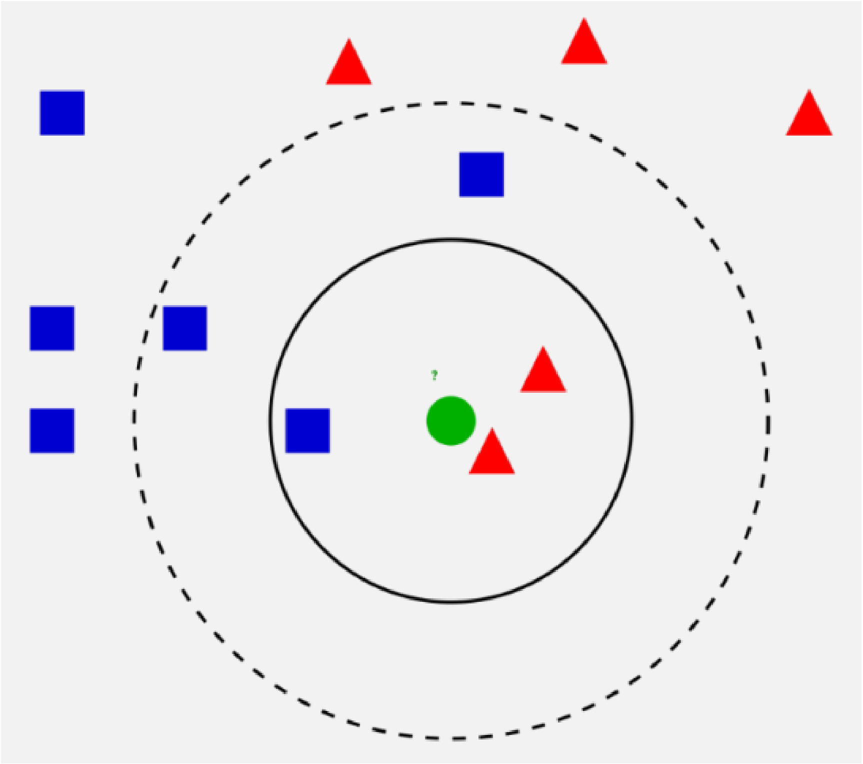


How to choose K?

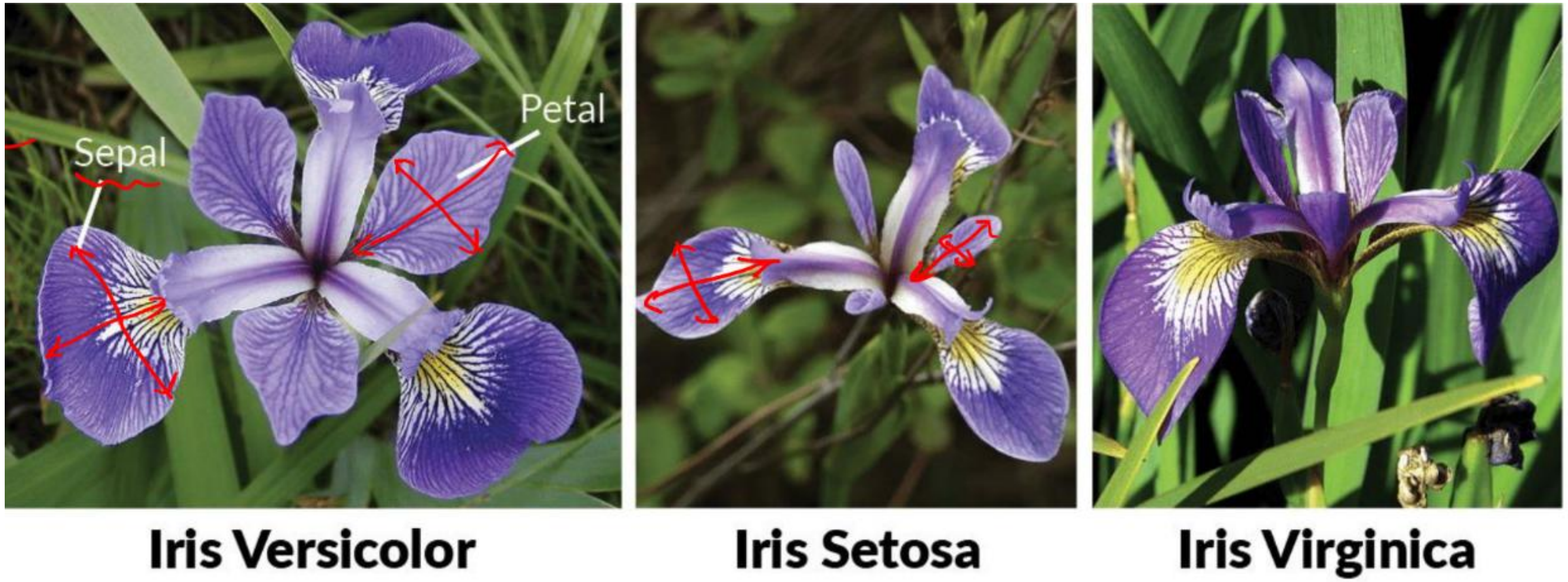
- Odd number
- Experiments
- Higher k → more accurate

KNN Algorithm Summary

- Save all the data
- Calculate the distance of the query data with all the data
- Consider k nearest samples
- Vote and label



Iris Data Classification



Steps to Solve ML Problems

Data and Preprocessing

ML Algorithm

Plot and Evaluate

Scikit-Learn Library

```python
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.neighbors import KNeighborsClassifier
from sklearn.preprocessing import LabelEncoder
from sklearn.metrics import accuracy_score

df = pd.read_csv("iris.data", header=None)
df.columns = ["sepal_length", "sepal_width", "petal_length", "petal_width", "target"]

X = df.drop(columns=["target"])
y = df["target"]

label_encoder = LabelEncoder()
y = label_encoder.fit_transform(y)

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)


knn = KNeighborsClassifier(n_neighbors=3)
knn.fit(X_train, y_train)


y_pred = knn.predict(X_test)
print("Accuracy:", accuracy_score(y_test, y_pred))
```