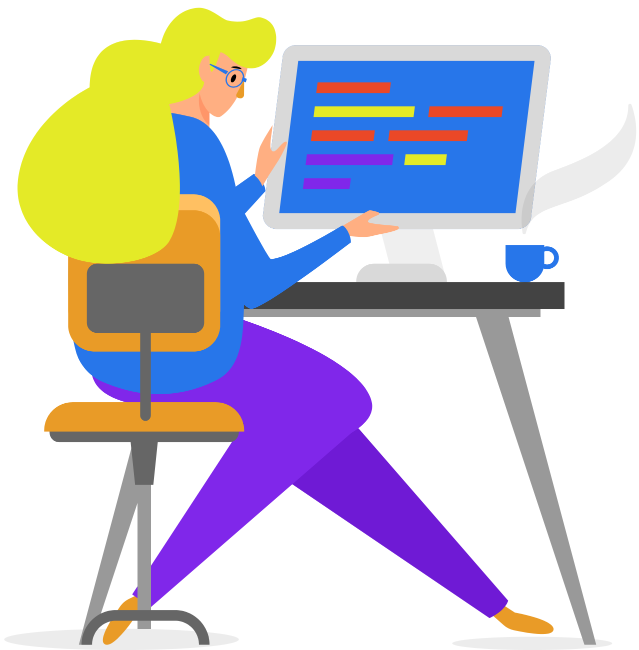
K-NEAREST NEIGHBOR ALGORITHM

INDICE

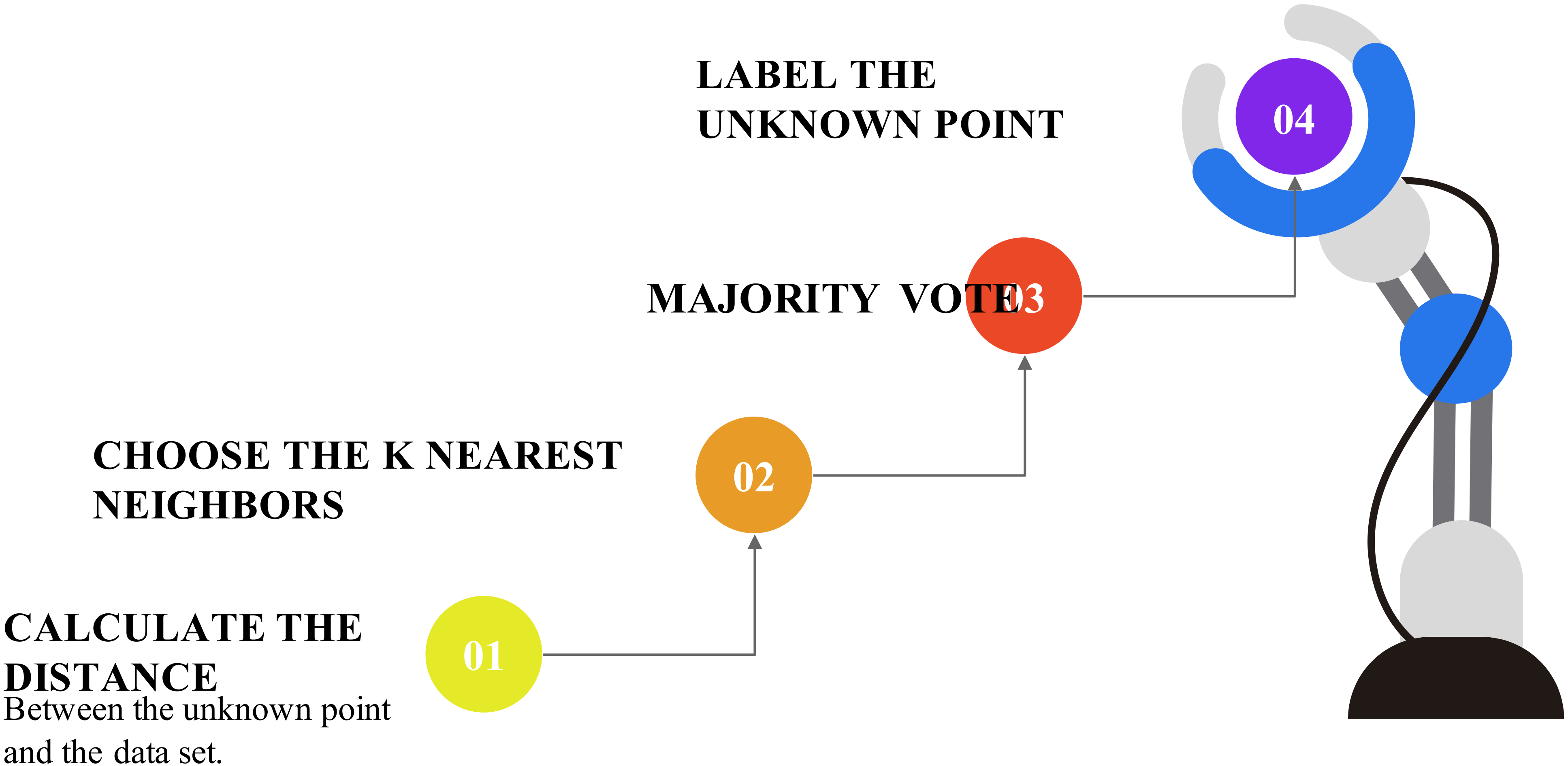


01

KNN ALGORITHM

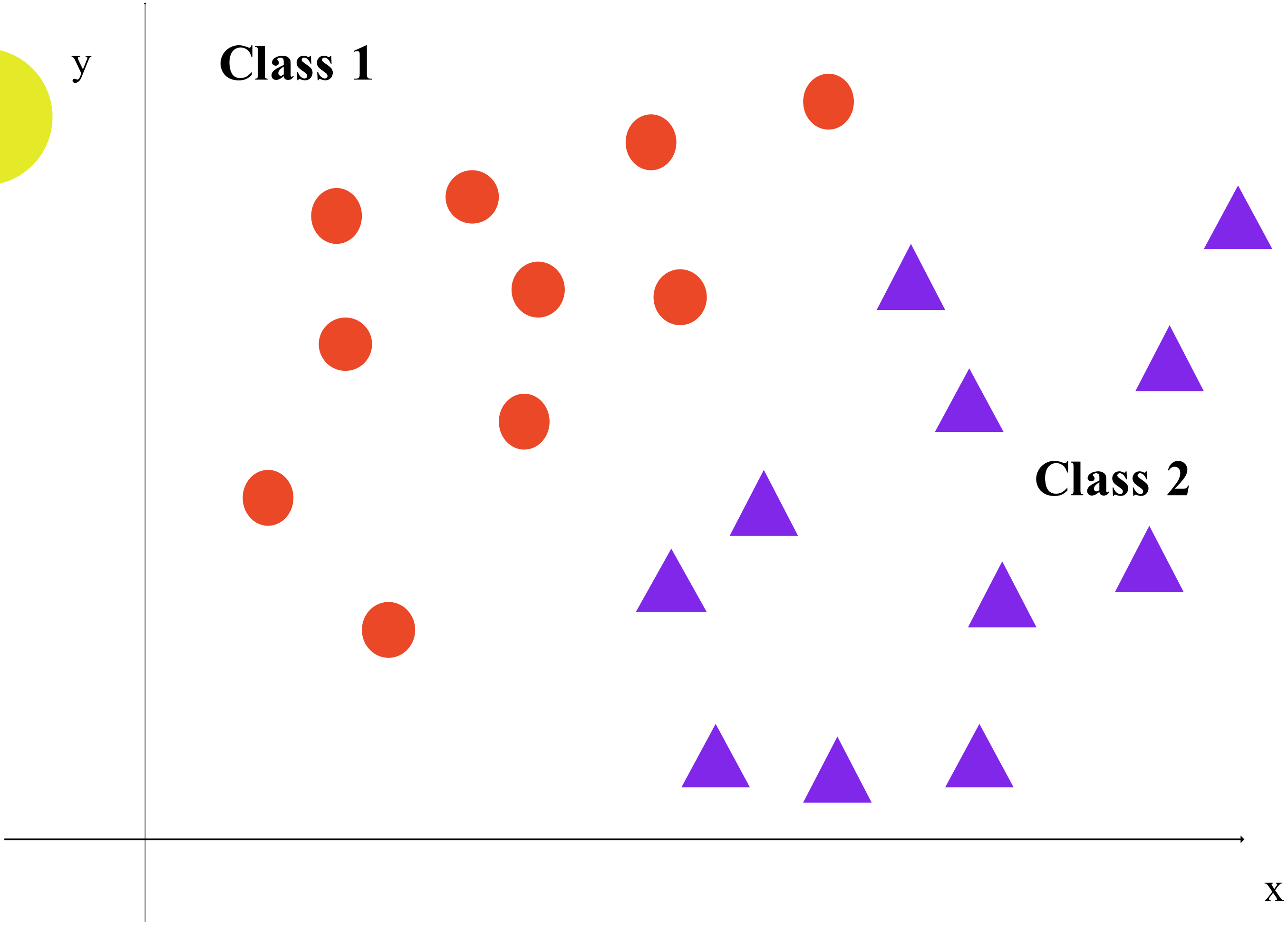
- SIMPLE AND SUPERVISED ALGORITHM
- USED IN CLASSIFICATION AND REGRESSION PROBLEMS
- EUCLIDEAN, MANHATTAN, MINKOWSKY, HAMMING DISTANCE
- USES PROXIMITY/SIMILARITY TO MAKE CLASSIFICATIONS

COMPUTATION



KNN WITH AN EXEMPLE

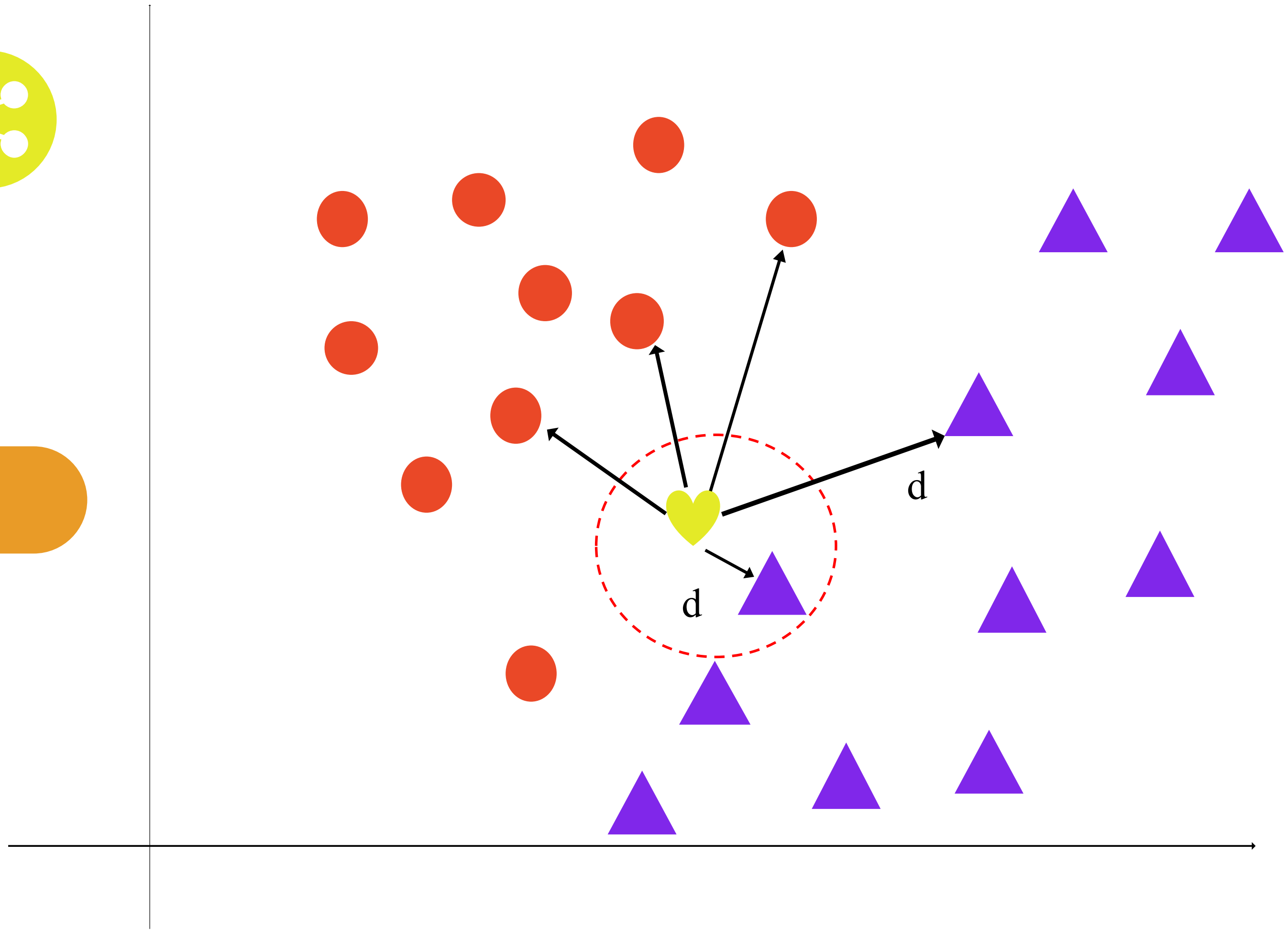
We choose a point



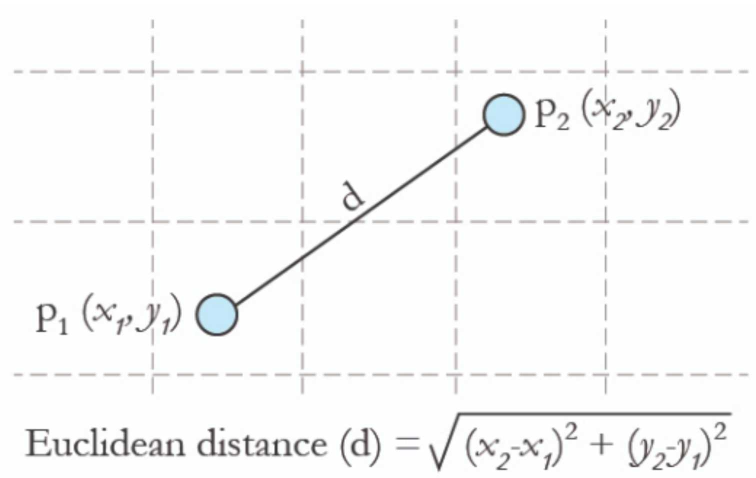
KNN WITH AN EXEMPLE

Calculate distances

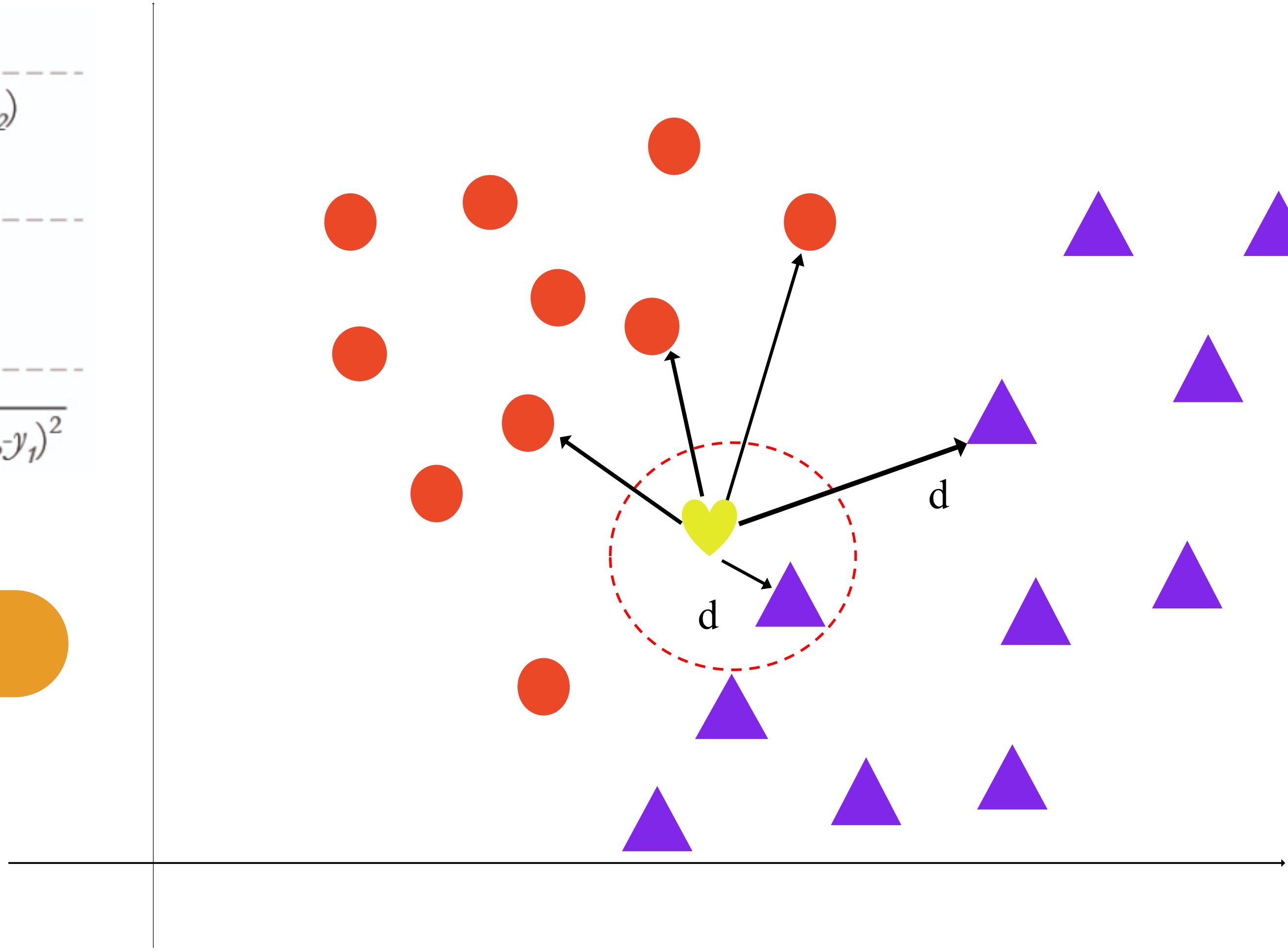
EUCLIDEAN DISTANCE



KNN WITH AN EXEMPLE



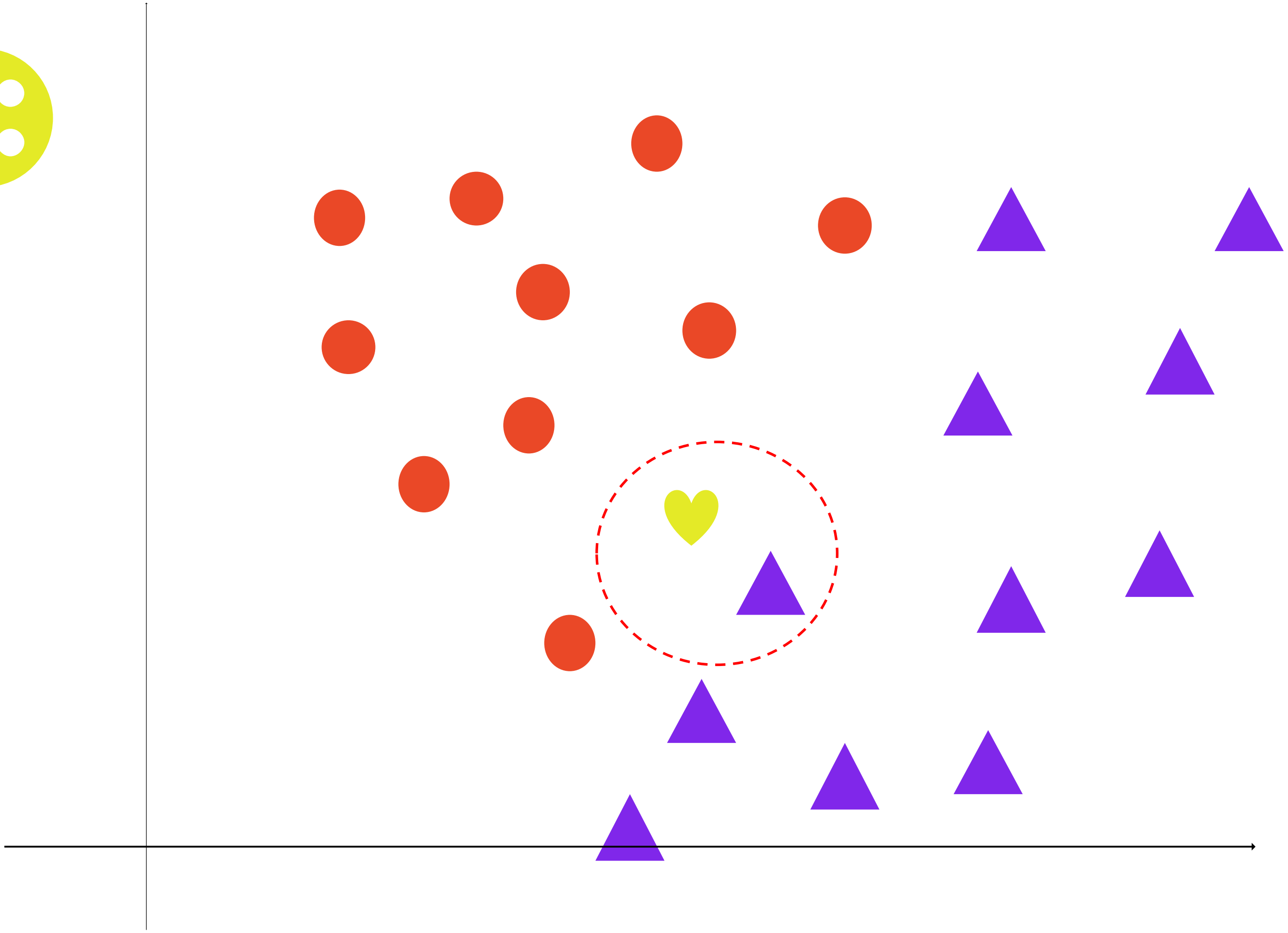
EUCLIDEAN DISTANCE



KNN WITH AN EXEMPLE

We choose the k

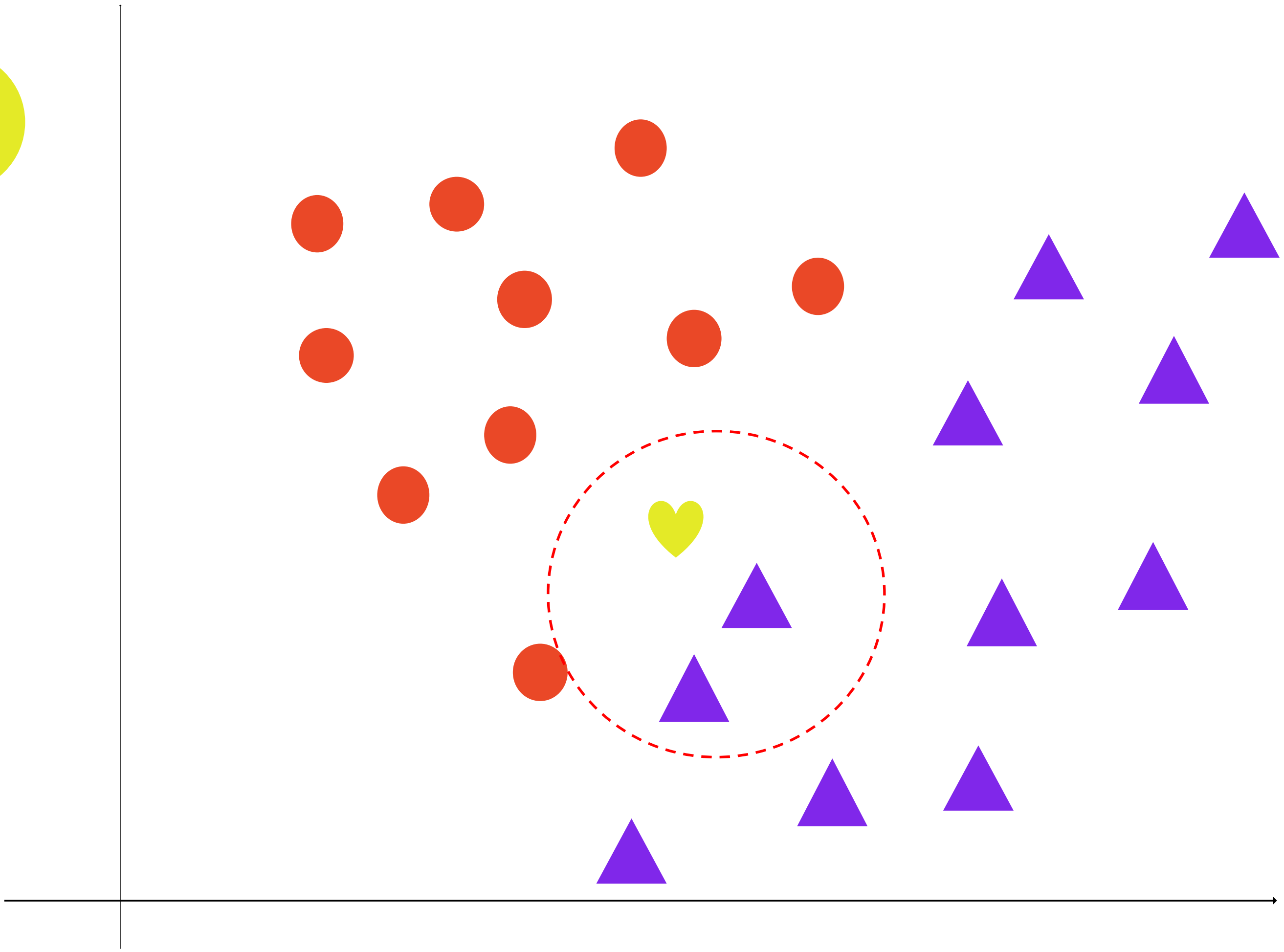
K=1



KNN WITH AN EXEMPLE

We choose the k

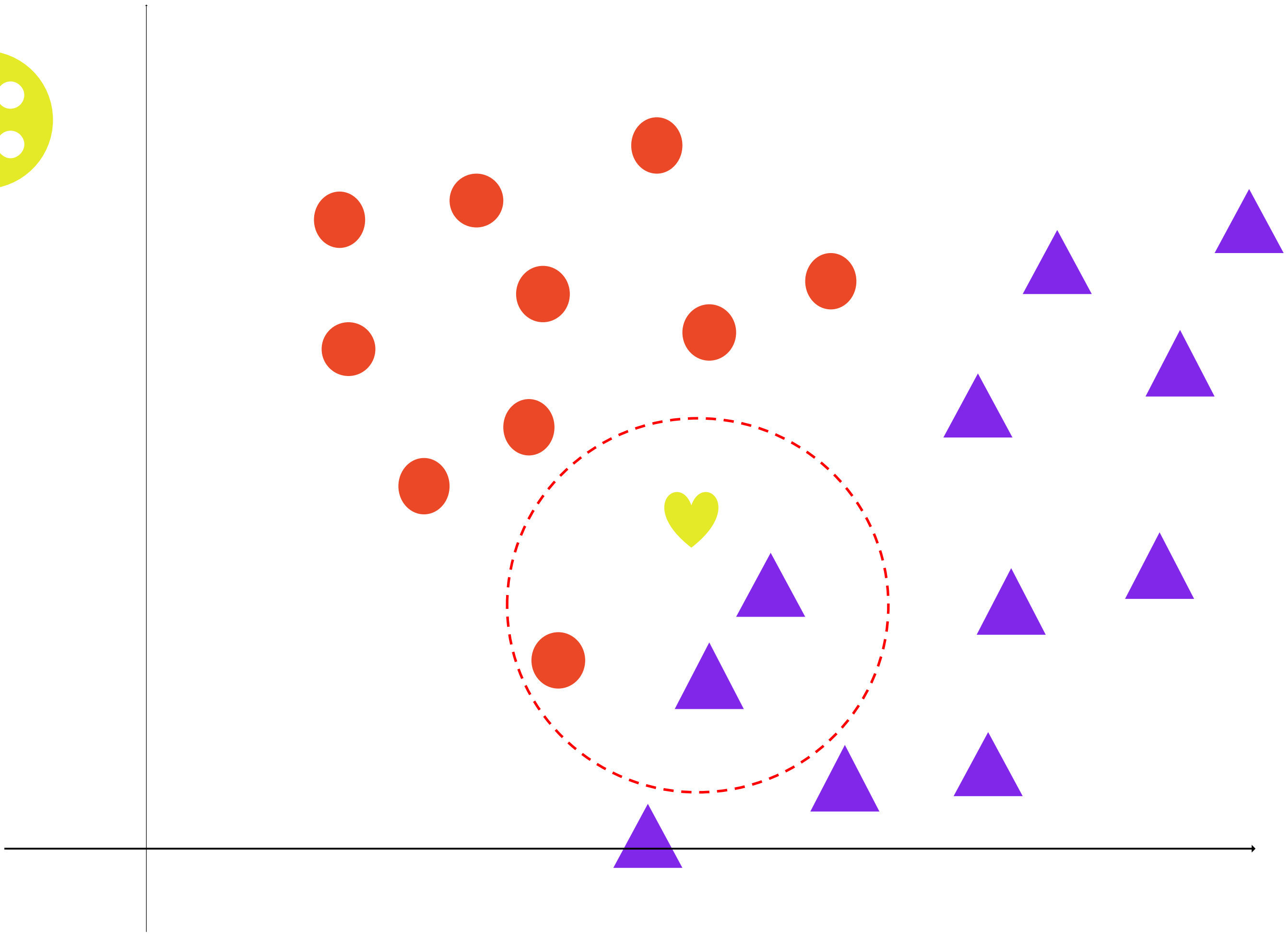
K=2



KNN WITH AN EXEMPLE

We choose the k

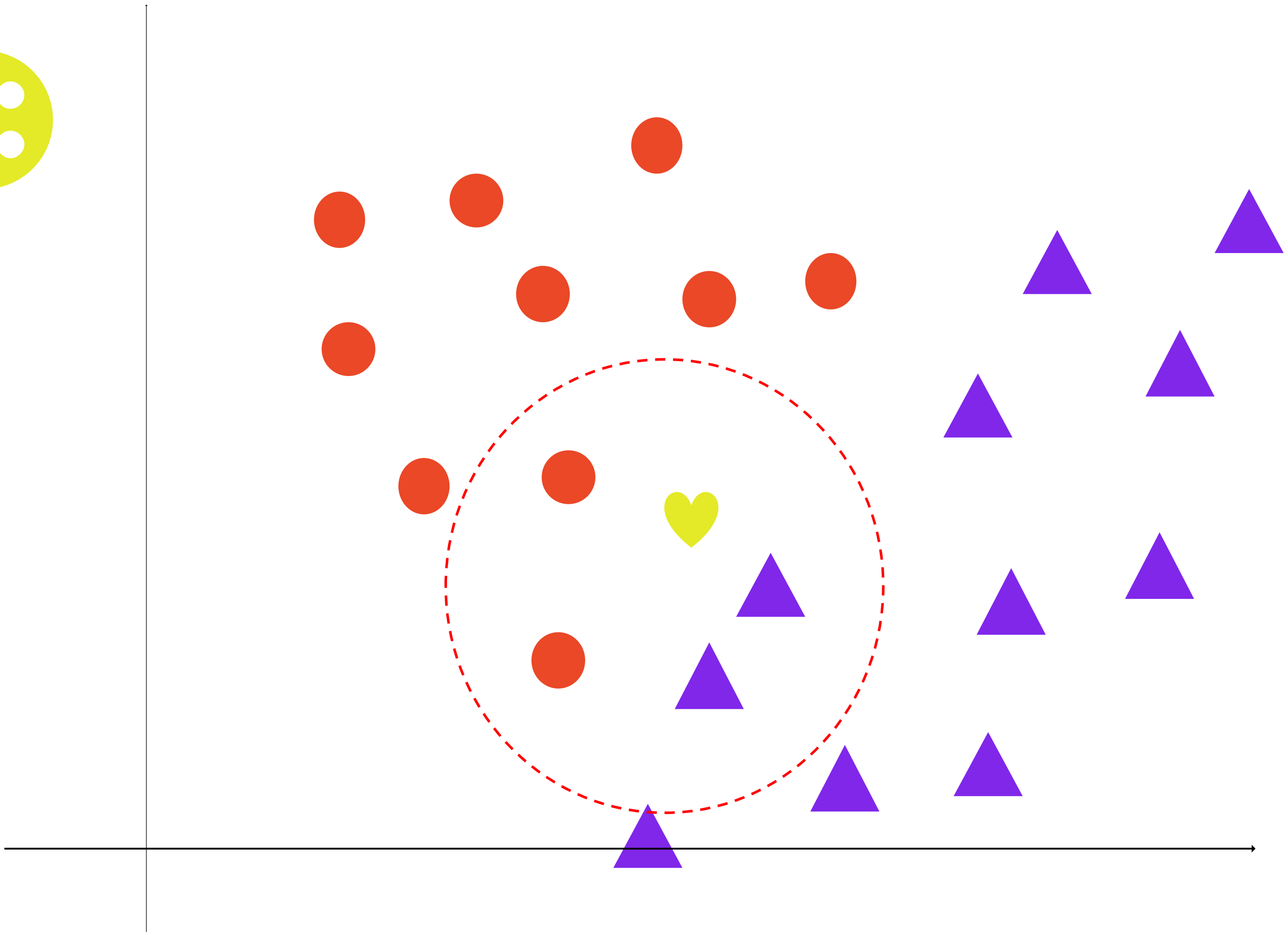
K=3



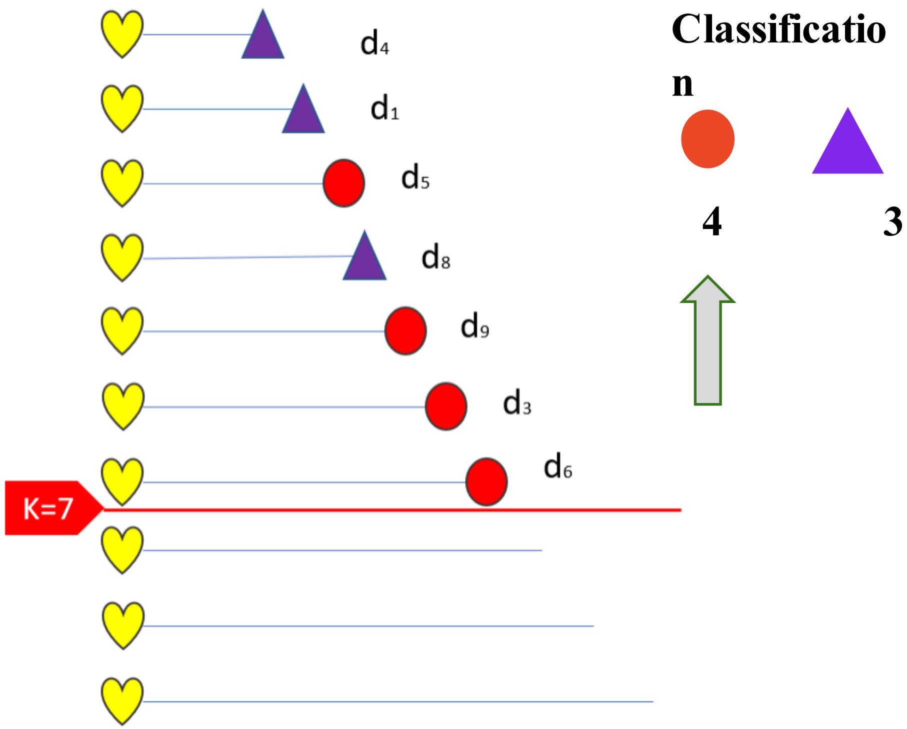
KNN WITH AN EXEMPLE

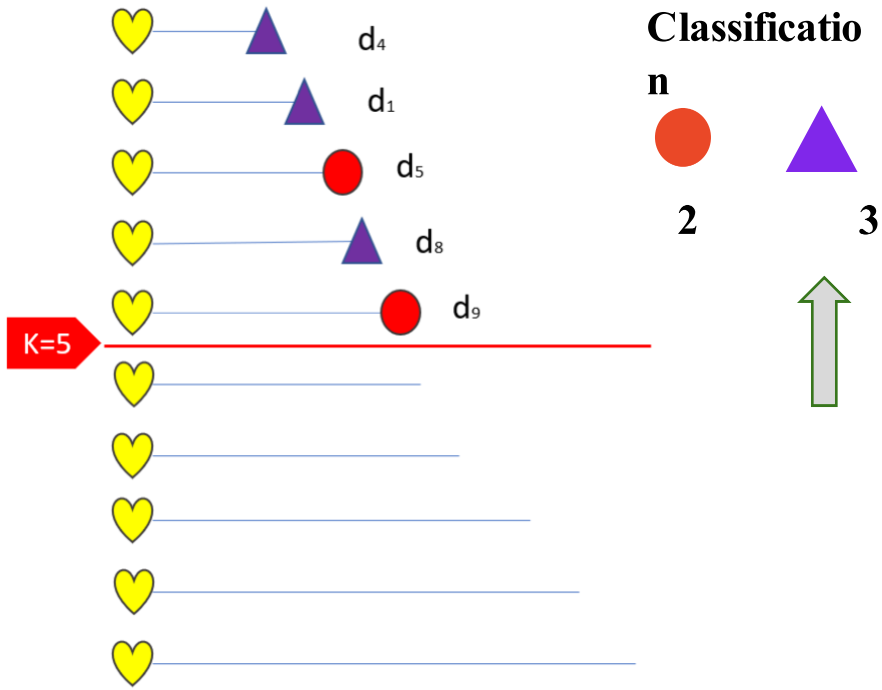
We choose the k

K=4



03 OOSING THE PARAMETER K





03

CHOOSING THE PARAMETER K

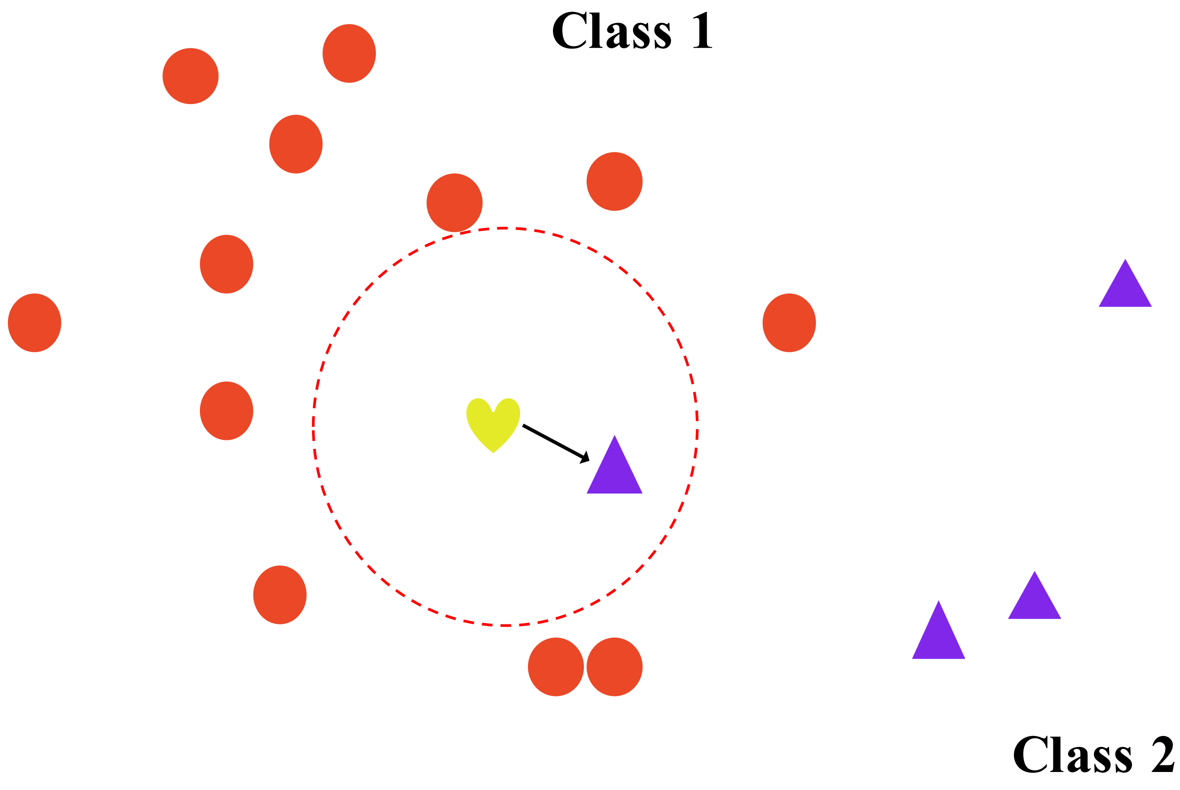
PARAMETER THAT REFERS TO THE NUMBER OF NEAREST NEIGHBOR TO INCLUDE

POSSIBILITIES:

- K ≈ 1, unstable predictions
- Odd value to avoid confusion

https://www.youtube.com/watch?v=0p0o5cmgLdE

K = 1



03 CHOOSING THE PARAMETER K

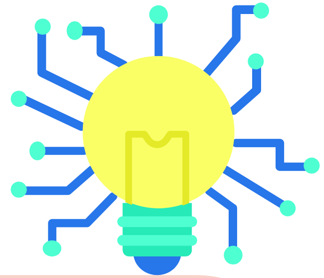
PARAMETER THAT REFERS TO THE NUMBER OF NEAREST NEIGHBOR TO INCLUDE

POSSIBILITIES:

+ K ≈ 1, unstable predictions
+ Odd value to avoid confusion

https://www.youtube.com/watch?v=0p0o5cmgLdE

04 NN ALGORITHM



Advantages

- Simple and easy to implement
- Fast: No training
- Versatile and adaptable

Disadvantages

- Slower when high dimensions or large datasets
- Need more storage
- Need to rescale dataset (normalization)
- Sensible to noise.

05 Biomedical applications

Genetics 01

Microarray gene expression analysis and clinical outcome prediction

https://www.nature.com/articles/tpj201056

Neural Networks 02

Classification of MRI brain images using k-nearest neighbor and artificial neural network

https://ieeexplore.ieee.org/abstract/document/5972341?casa_token=TwngIWjRiroAAAAA:qtRpoQV_h7-eLLZTHxWoyjaywvJHtOYEqpVrKRdxiPfJhAmJeu1MuLMt7aolMgEROdm-5_VdlVw

Cancer predictions 03

Lymph Node Metastasis in Gastric Cancerhttps://www.ncbi.nlm.nih.gov/pmc/articles/PMC3488413/

Cell Classification 05

Decide which type of cell a given feature is

https://www.youtube.com/watch?v=HVXime0nQeI

Heart Disease 04

Diagnosing Heart Disease Patients

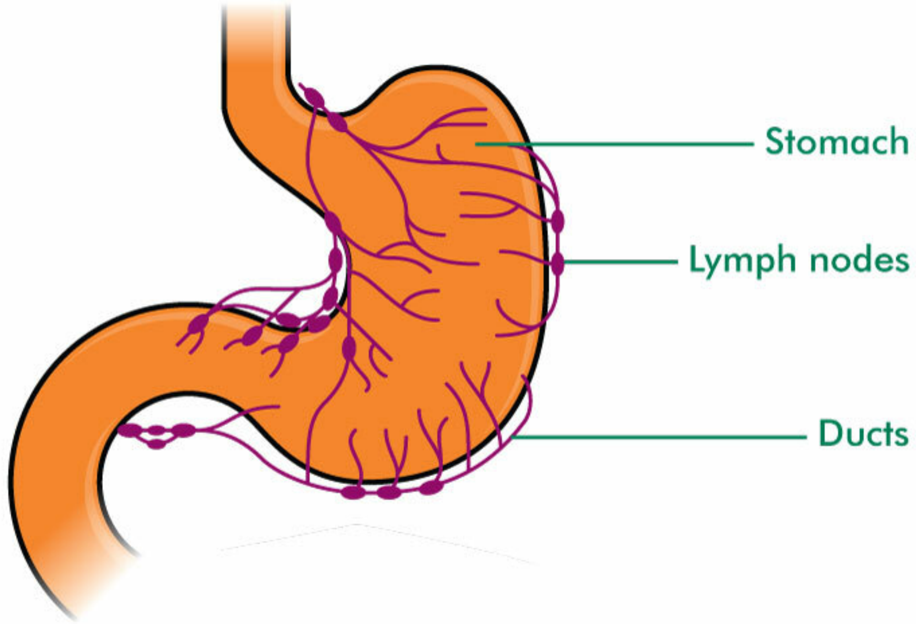
http://www.ijiet.org/papers/114-K0009.pdf

Biomedical applications: Classification of Lymph Node Metastasis in Gastric Cancer

05

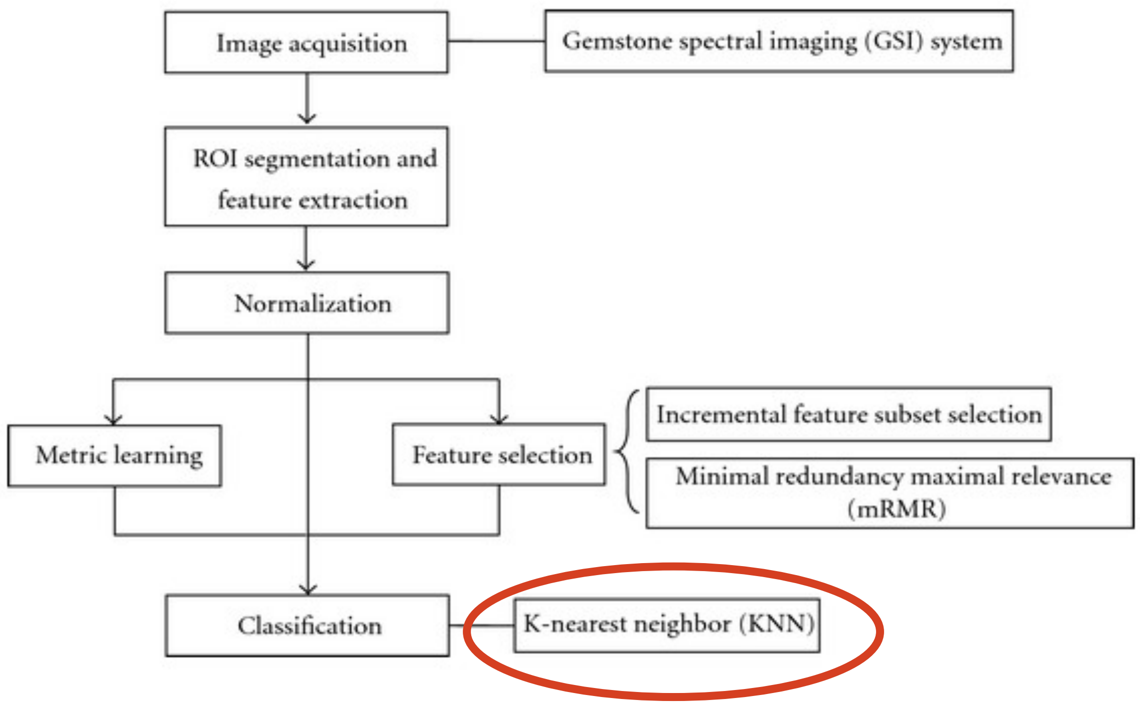
OBJECTIVE

classify lymph node metastasis from nonlymph node metastasis using KNN algorithm



Li, C., Zhang, S., Zhang, H., Pang, L., Lam, K., Hui, C., & Zhang, S. (2012). Using the K-nearest neighbor algorithm for the classification of lymph node metastasis in gastric cancer. *Computational and mathematical methods in medicine*, *2012*, 876545. https://doi.org/10.1155/2012/876545

05



CLASSES

POSITIVE
27 lymph node metastasis

NEGATIVE
11 nonlymph node metastasis

05

Table 4

Classification performance of the SFS-KNN algorithm with different neighborhood sizes.

Neighborhood size		***K* = 1**	***K* = 3**	***K* = 5**	***K* = 7**	***K* = 9**
Pre-norm	Selected features	14, 16	14, 31, 5, 15, 26, 4, 27, 21, 24, 9, 32, 2, 25, 8, 28, 3, 16	14, 31, 10, 36, 3, 25, 2	12, 31, 8, 29, 3, 15, 33, 1	12, 31, 23, 26, 3, 24, 30, 16
	Accuracy	88.29%	93.68%	93.29%	91.71%	92.24%
Norm	Selected features	12, 30	20, 15, 11, 30, 5	12, 30, 31, 33, 14	12, 19, 20, 30, 5, 18, 25, 17, 34, 3, 32, 15, 24	12, 19, 29, 30, 8, 34, 33, 25, 15, 6, 24, 7, 10, 20, 17
	Accuracy	93.95%	96.45%	96.58%	96.18%	97.89%

Table 5

Classification performance of mRMR-KNN (MIQ) with different neighborhood sizes.

	Neighborhood size	***K* = 1**	***K* = 3**	***K* = 5**	***K* = 7**	***K* = 9**
Prenorm	Sequence	14, 19, 5, 17, 23, 12, 3, 16, 18, 22, 1, 15, 4, 2, 30, 13, 21, 32, 1 33, 11, 34, 20, 35, 31, 25, 9, 29, 24, 8, 7, 26, 36, 27, 28, 6				
	Length	1	28	28	35	1
	Accuracy	87.50%	89.74%	89.08%	87.24%	81.71%
Norm	Sequence	15, 21, 3, 30, 17, 24, 12, 14, 23, 5, 16, 22, 2, 18, 27, 1, 20, 4, 3 25, 13, 19, 6, 28, 35, 26, 32, 7, 29, 34, 8, 31, 9, 11, 10, 36				
	Length	4	2	2	2	10
	Accuracy	90.00%	94.87%	94.87%	94.74%	95.66%

K=5 is the optimal neighborhood

06 Code Example

```python
import numpy as np
import pandas as pd
from matplotlib import pyplot as plt
from sklearn.datasets import load_breast_cancer
from sklearn.metrics import confusion_matrix
from sklearn.neighbors import KNeighborsClassifier
from sklearn.model_selection import train_test_split
import seaborn as sns
sns.set()

#BREAST CANCER DATASET 30 features
##The dataset classifies tumors into two categories (malignant, 0,  and benign, 1).
# We must encode categorical data for it to be interpreted by the model.

breast_cancer = load_breast_cancer()
X = pd.DataFrame(breast_cancer.data, columns=breast_cancer.feature_names)
X = X[['mean area', 'mean compactness']]
y = pd.Categorical.from_codes(breast_cancer.target, breast_cancer.target_names)
y = pd.get_dummies(y, drop_first=True)

#We need to put aside data to verify whether our model does a good job at classifying the data.
# By default, train_test_split sets aside 25% of the samples in the original dataset for testing.
X_train, X_test, y_train, y_test = train_test_split(X, y, random_state=1)

#The sklearn library has provided a layer of abstraction on top of Python.
# Therefore, it’s sufficient to create an instance of KNeighborsClassifier.
#Set at k=5 nearest neighbors. We chose Euclidean distance for determining
#the proximity between neighboring points.
knn = KNeighborsClassifier(n_neighbors=5, metric='euclidean')
knn.fit(X_train, y_train)

#Using our newly trained model, we predict whether a tumor is benign or not given its
#mean compactness and area.
y_pred = knn.predict(X_test)


#We visually compare the predictions made by our model with the samples inside the testing set.
```

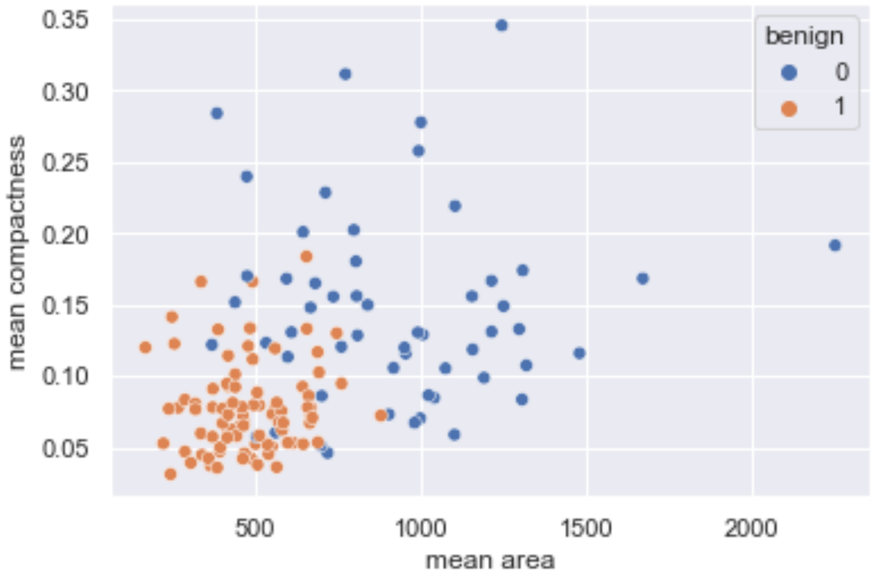
```python
sns.scatterplot(
    x='mean area',
    y='mean compactness',
    hue='benign',
    data=X_test.join(y_test, how='outer')
)


plt.scatterplot(
    X_test['mean area'],
    X_test['mean compactness'],
    c=y_pred,
    cmap='coolwarm',
    alpha=0.7
)

#Another way of evaluating our model is to compute the confusion matrix.
#The numbers on the diagonal of the confusion matrix correspond to correct predictions whereas
#the others imply false positives and false negatives.
foraccuracy=confusion_matrix(y_test, y_pred)

#Given our confusion matrix, our model has an accuracy of 121/143 = 84.6%
print(foraccuracy)
```

https://towardsdatascience.com/k-nearest-neighbor-python-2fccc47d2a55

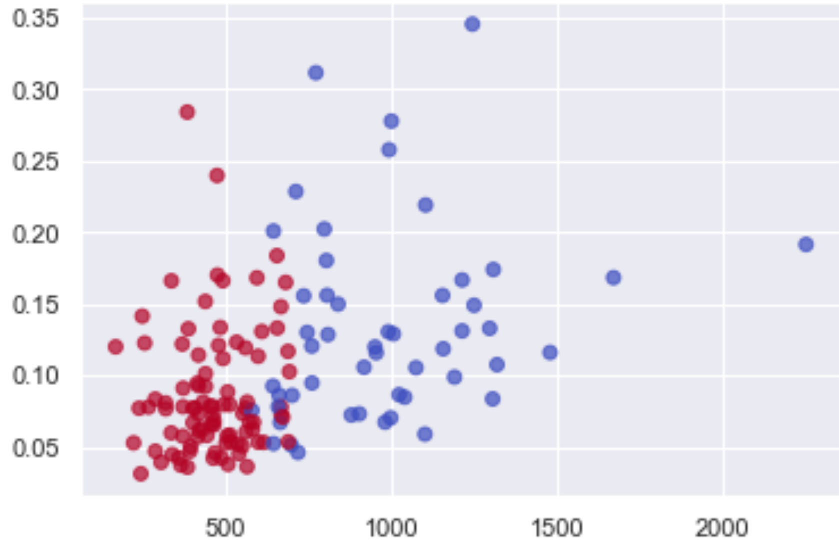
06 Code Example

K=5

Our model



Samples inside the testing set



https://towardsdatascience.com/k-nearest-neighbor-python-2fccc47d2a55

06 Code Example

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

[[TP FP],
[FN TN]]

k=1

[[39 16]
 [14 74]]

Accuracy= 79.02%

K=5

```
[[42 13]
 [ 9 79]]
```

Accuracy= 84.6%

K=10

[[42 13]
 [11 77]]

Accuracy= 83.22%

https://towardsdatascience.com/k-nearest-neighbor-python-2fccc47d2a55

THE END