AI VIET NAM
@aivietnam.edu.vn

KNN

(Basic, Advanced Concepts and Its Applications)

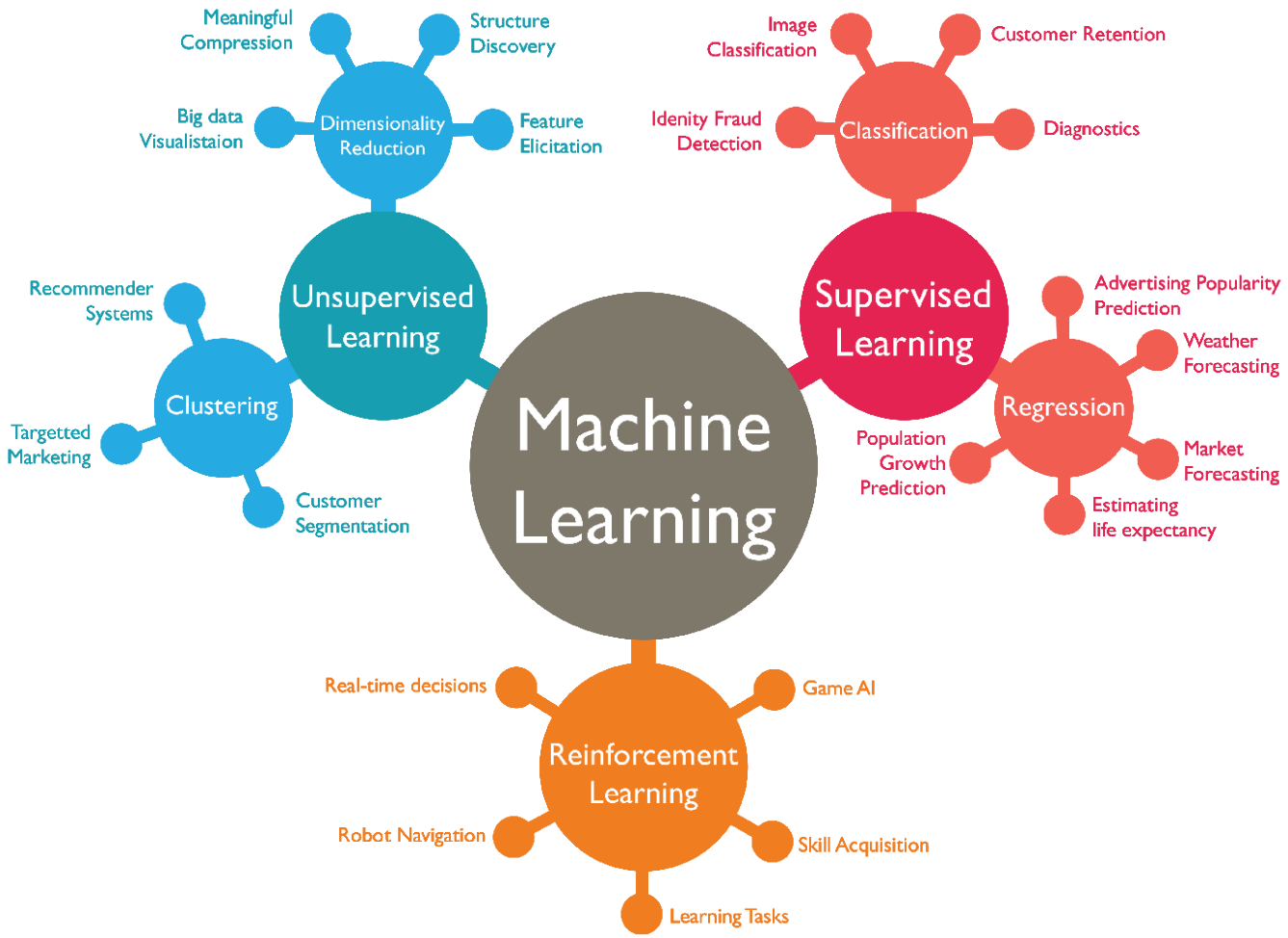
Vinh Dinh Nguyen
PhD in Computer Science

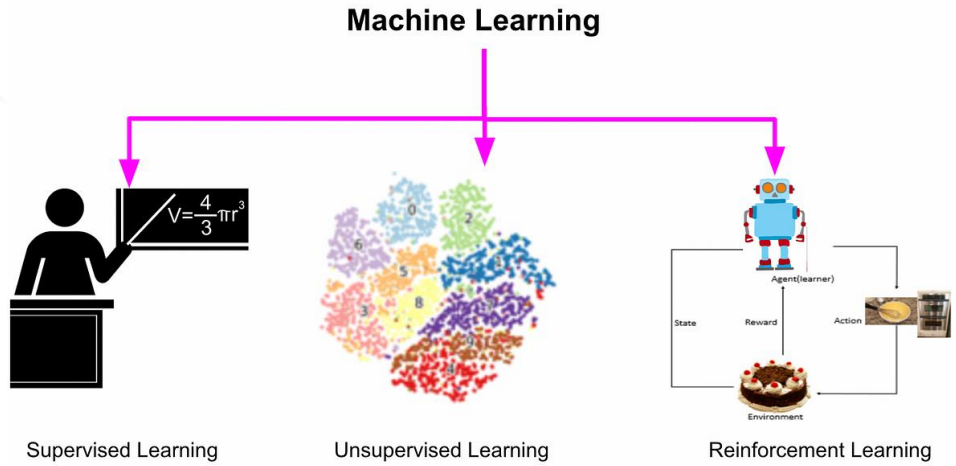
Outline

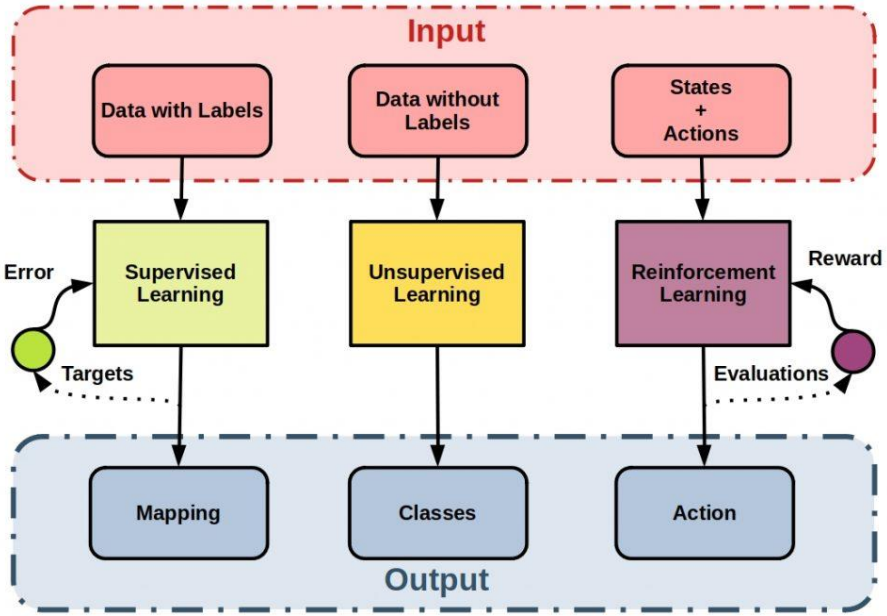
- Machine Learning: Review
- KNN Motivation
- KNN for Classification
- How to Select K in KNN
- KNN for Regression
- KNN with K-D Tree
- Data Science Application: Case study

AI VIET NAM
@aivietnam.edu.vn

Machine Learning

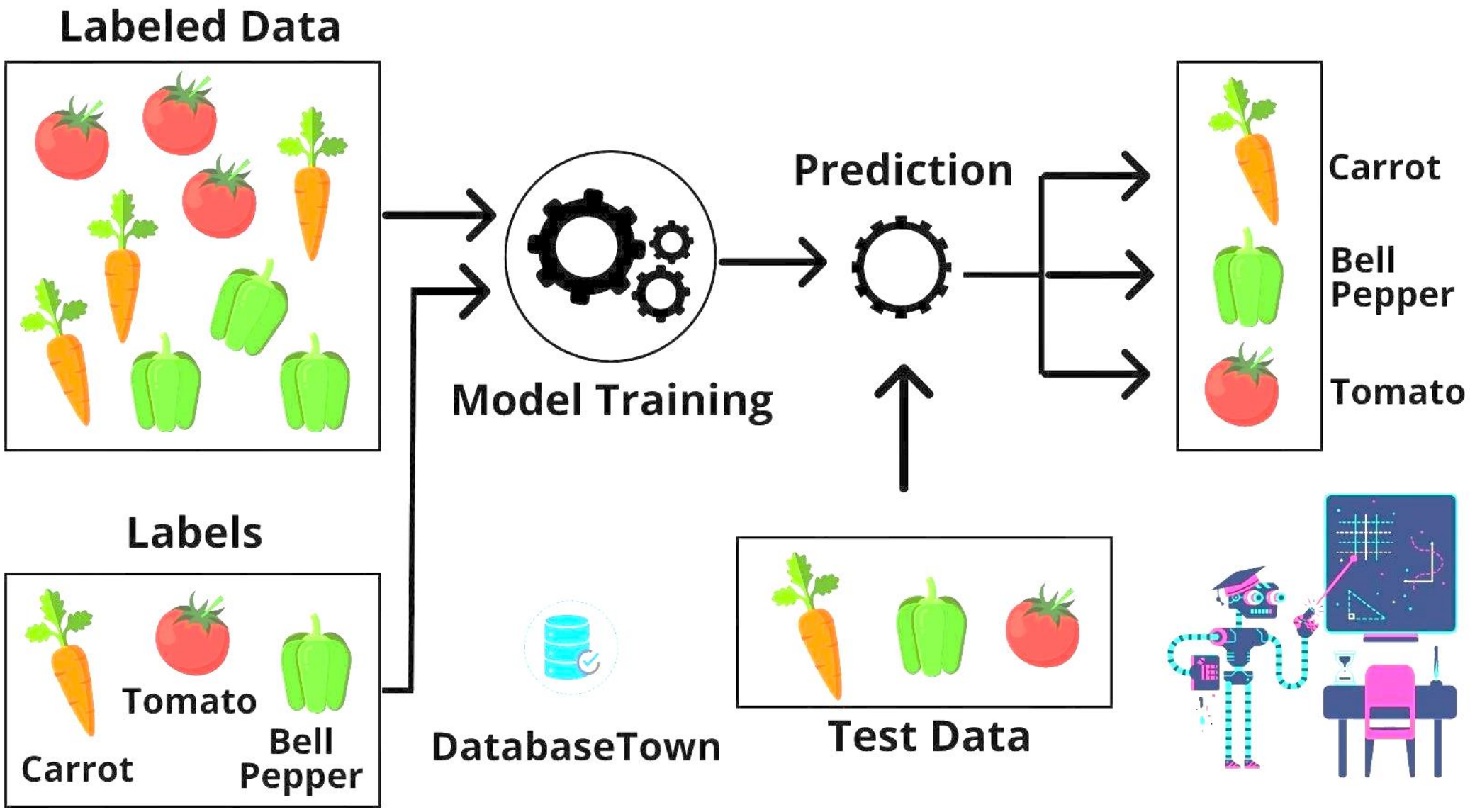






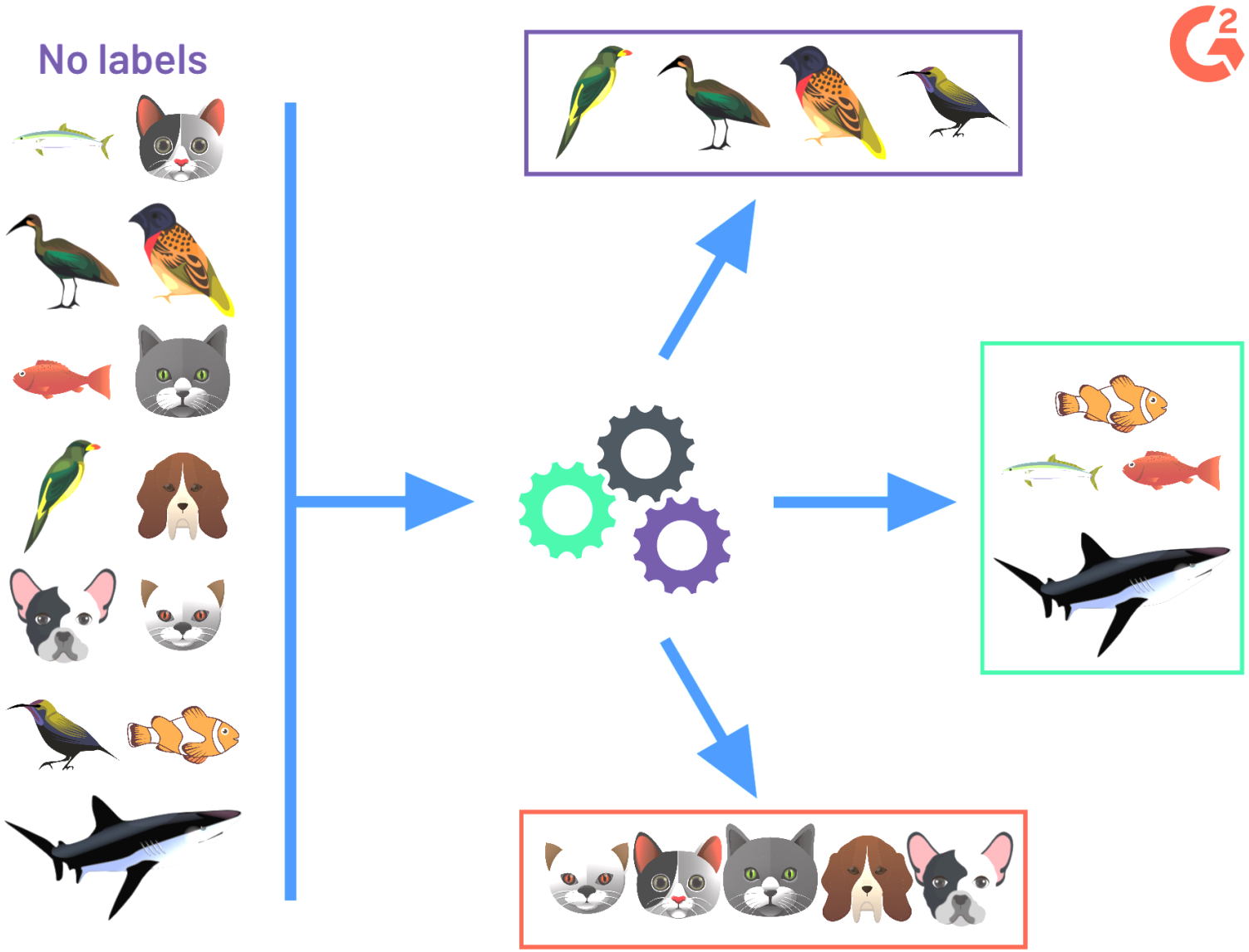
AI VIET NAM
@aivietnam.edu.vn

Supervised Learning

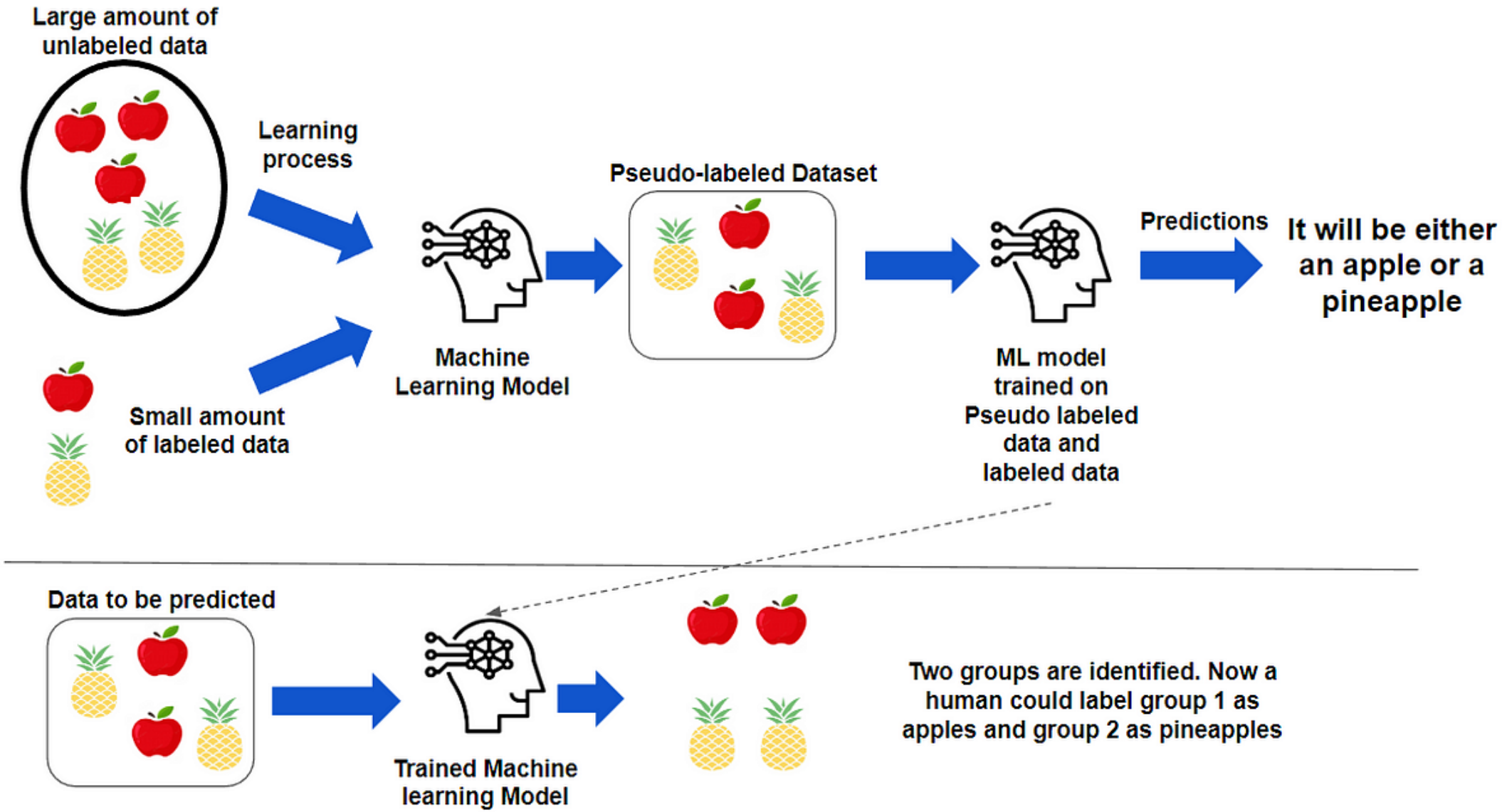


AI VIET NAM
@aivietnam.edu.vn

Unsupervised Learning

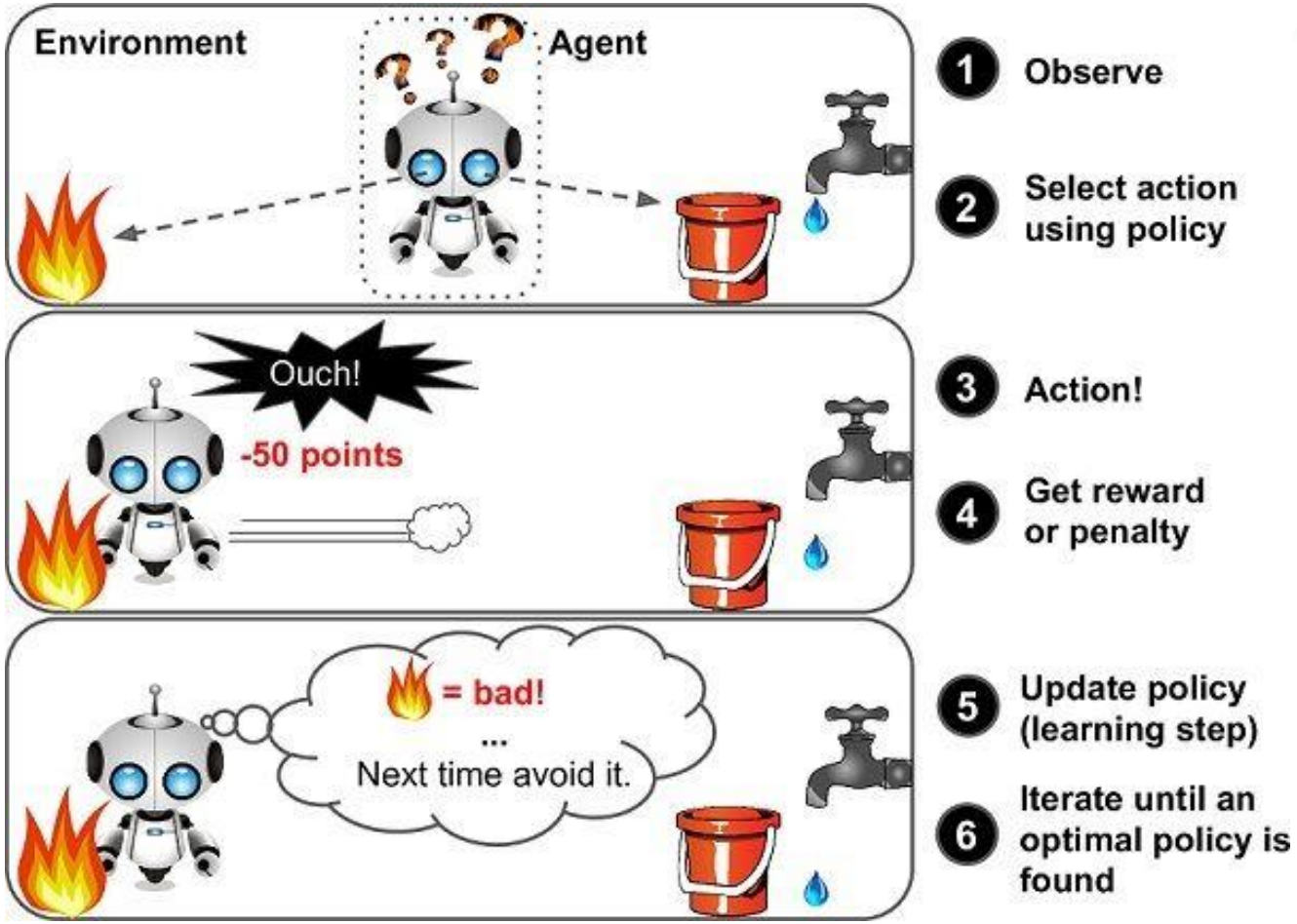


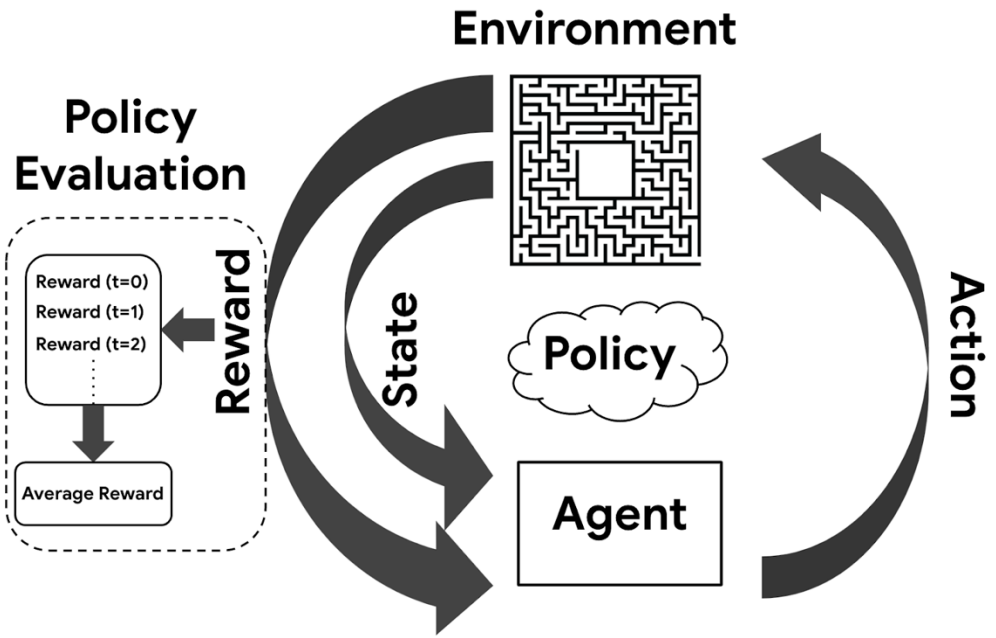
Semi-supervised Learning



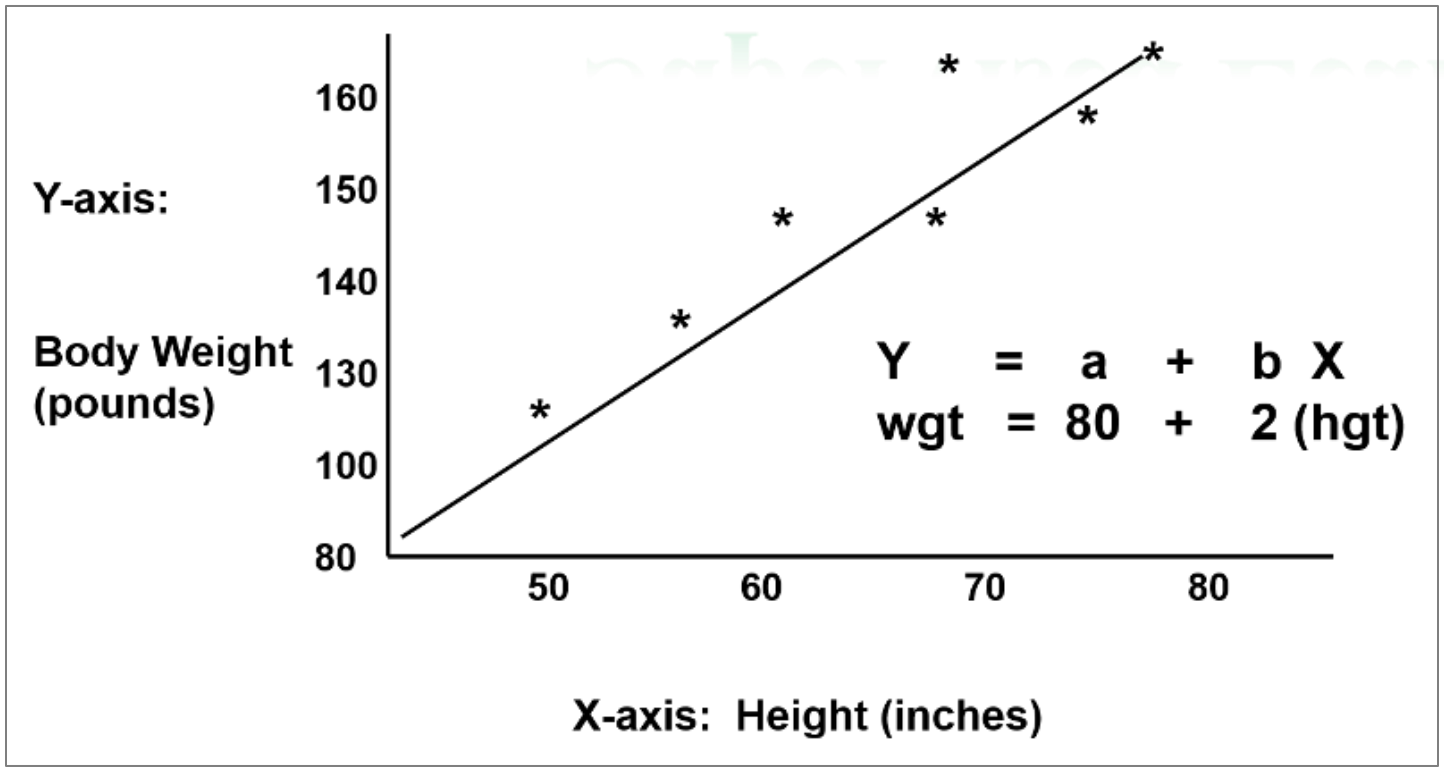
AI VIET NAM
@aivietnam.edu.vn

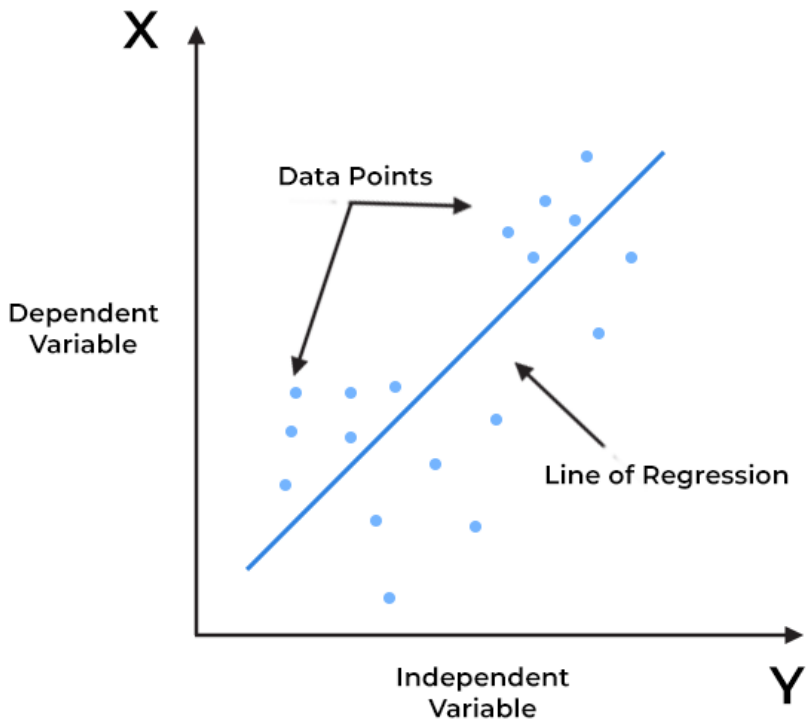
Reinforcement Learning





Supervised Learning: Review





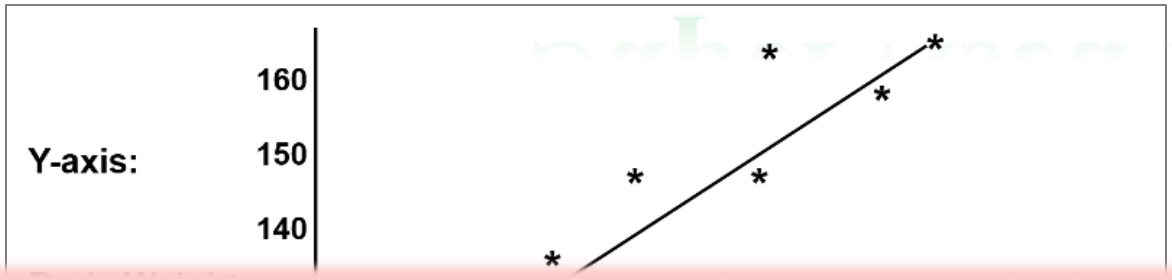
We assume that there is a true underlying relationship between the response Y and the predictors X:

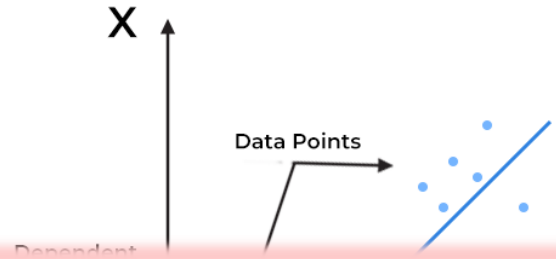
$$Y = f(X) + \epsilon$$

Our goal in supervised machine learning is to estimate f from training data.

AI VIET NAM
@aivietnam.edu.vn

Supervised Learning Review

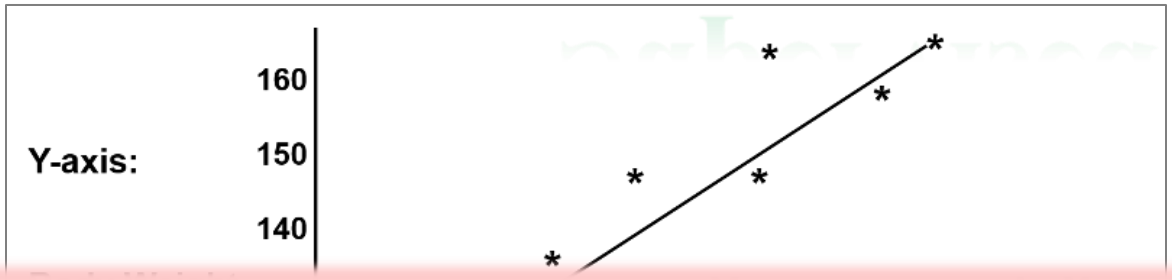


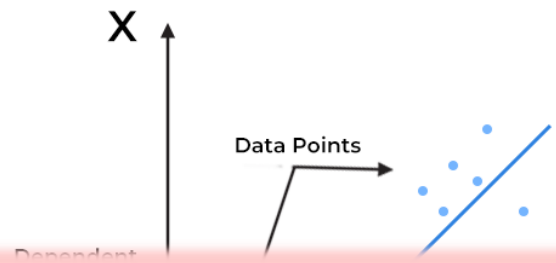


This is a Parametric Approach

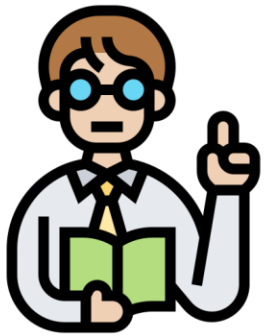
- Identifying the function: The first step is to identify the function such as linear or non-linear function.
- Identifying the parameters of the function in case this is linear function

Supervised Learning Review





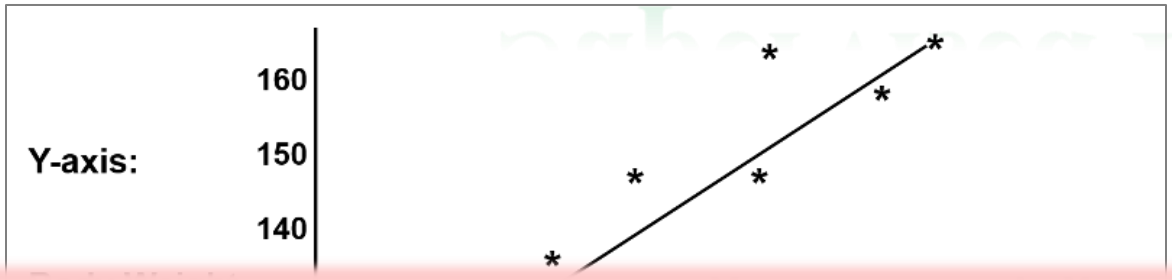
This is a Eager Learning Approach

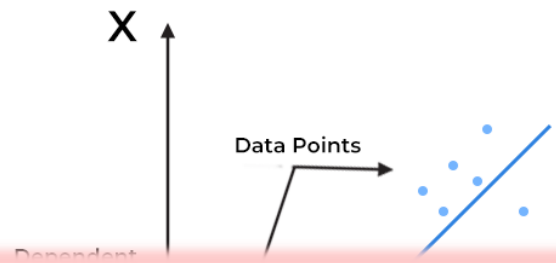


It takes long time learning and less time classifying data

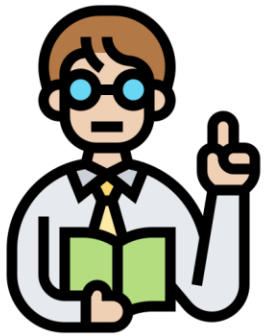
AI VIET NAM
@aivietnam.edu.vn

Supervised Learning Review





This is a Model-based Learning Approach



It takes long time learning and less time classifying data

Outline

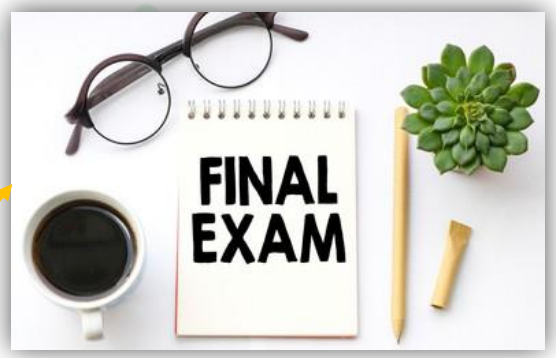
- Machine Learning: Review
- KNN Motivation
- KNN for Classification
- How to Select K in KNN
- KNN for Regression
- KNN with K-D Tree
- Data science application: Case study

Lazy Motivation

ANY APPROACHES DOES NOT MAKE ANY ASSUMPTION ON UNDERLYING DATA

- Just store dataset without learning from it
- Start classifying data when it receive Test data
- It takes less time learning

Lazy Motivation



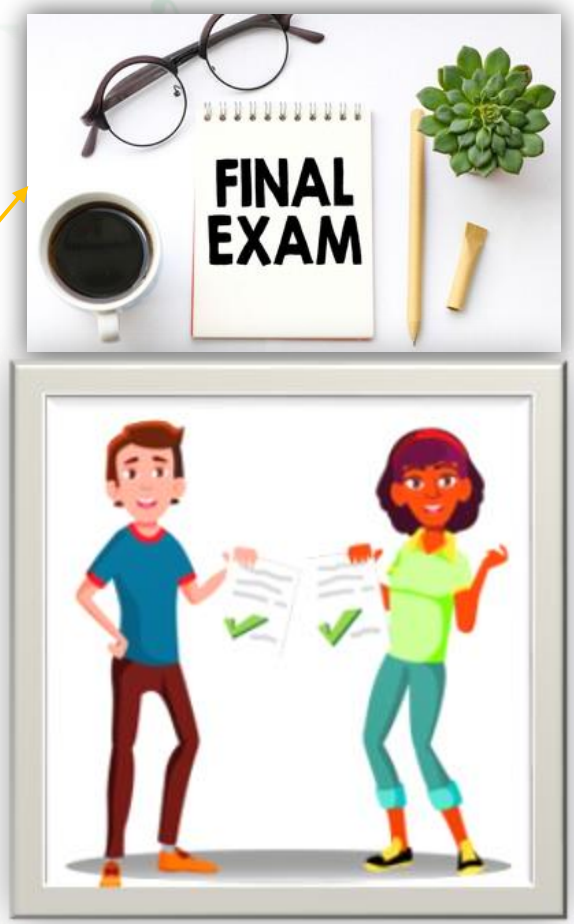
Study Hard

Laziness

Two students are going to take the final exam

Lazy Motivation

Study hard

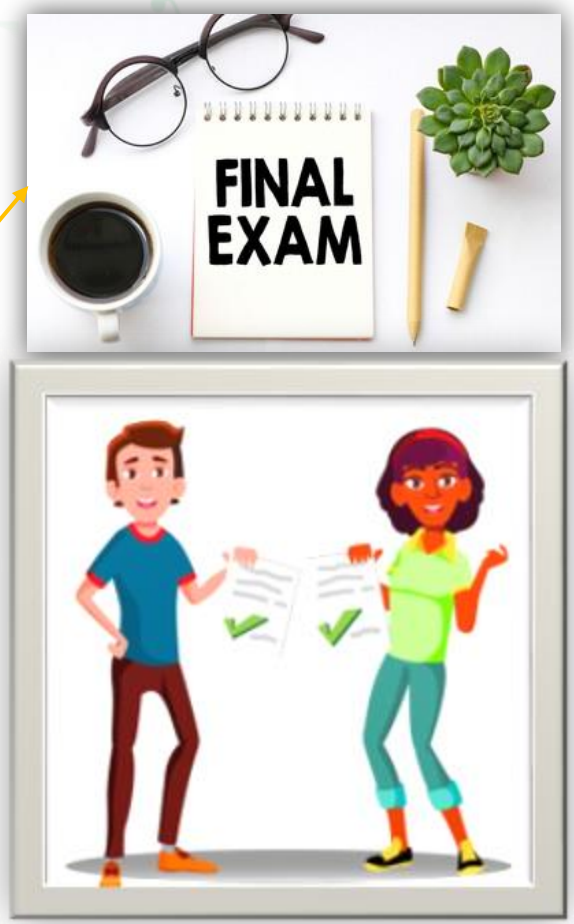


Laziness

Two students are get good results

Lazy Motivation

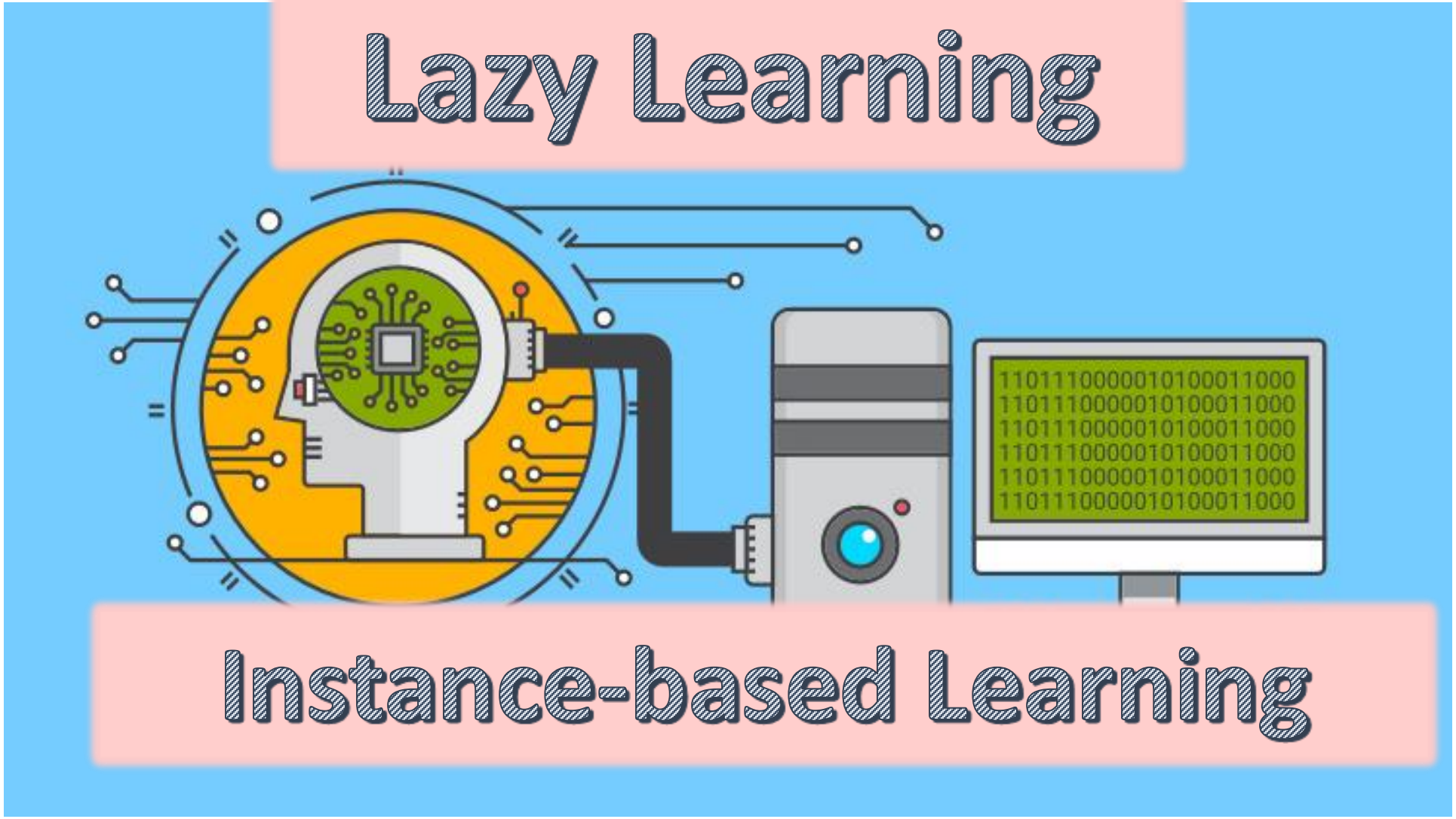
Study hard



Laziness

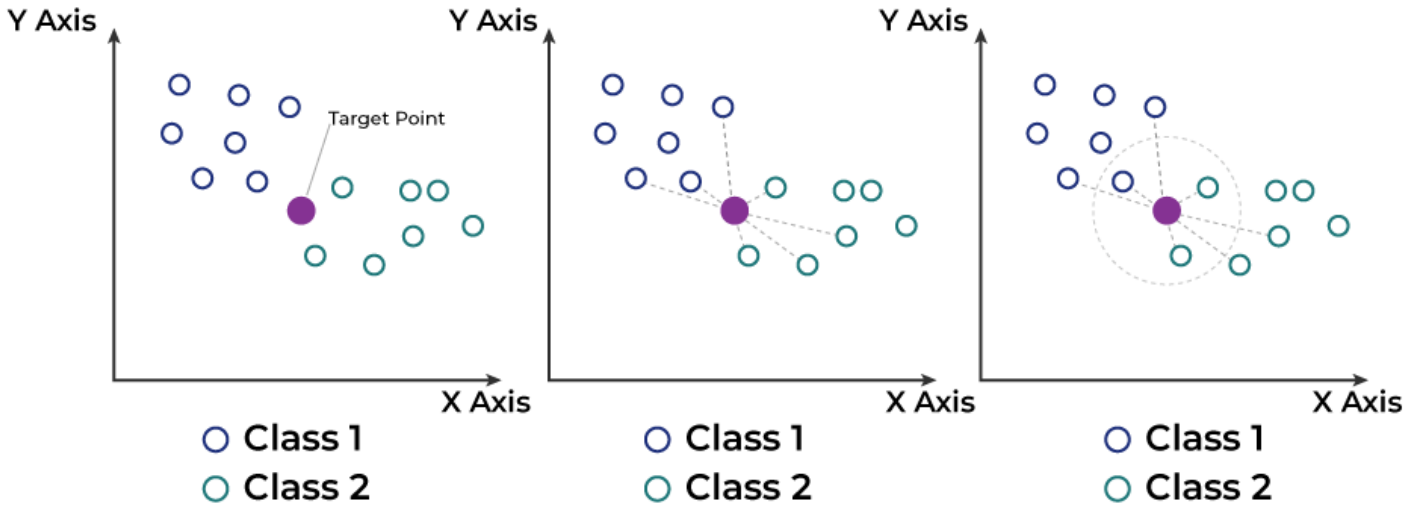
What is happening Here?

Lazy Motivation

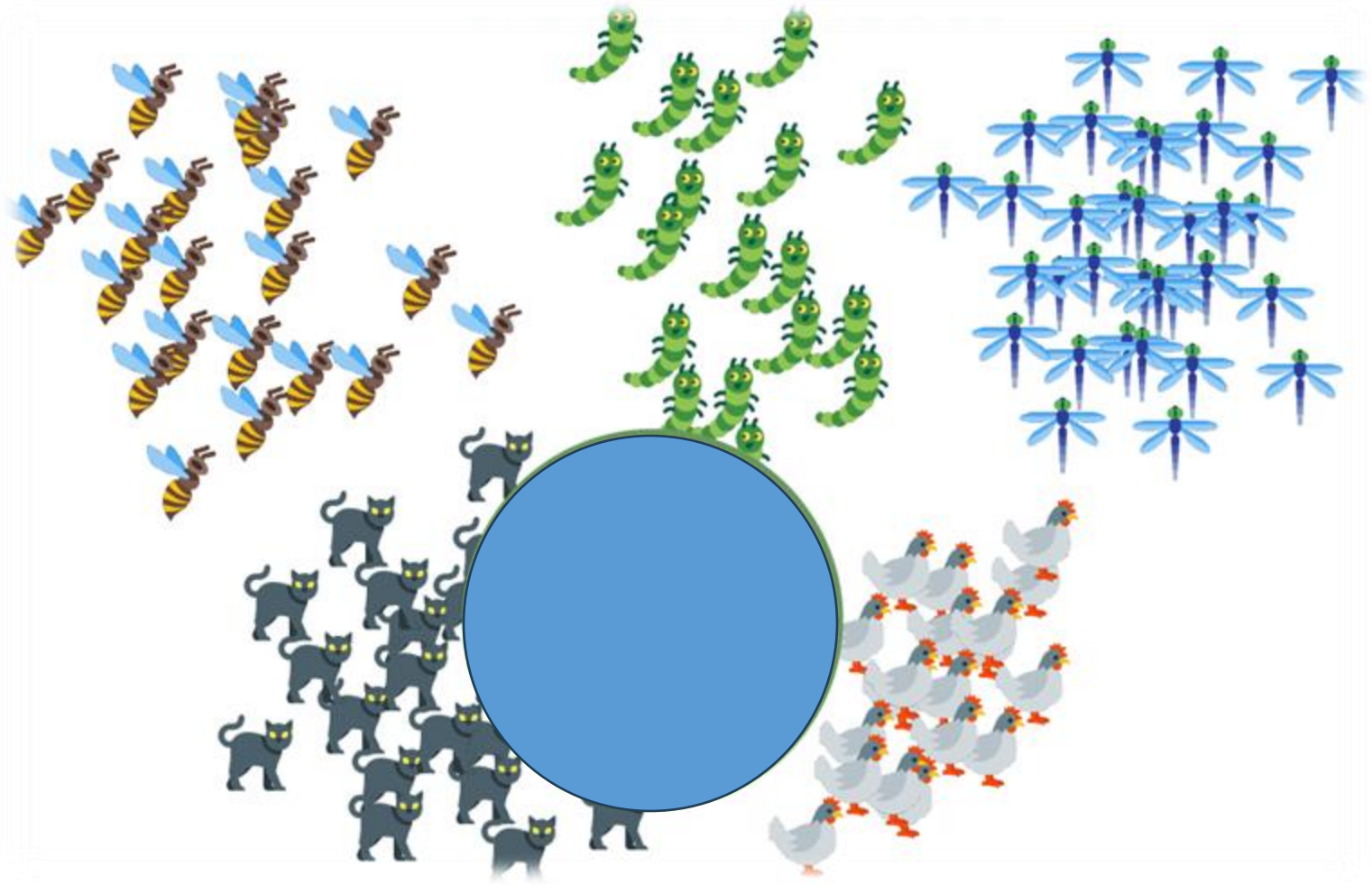


Lazy Learning, Instance-based Learning

- One of the most simplest algorithms

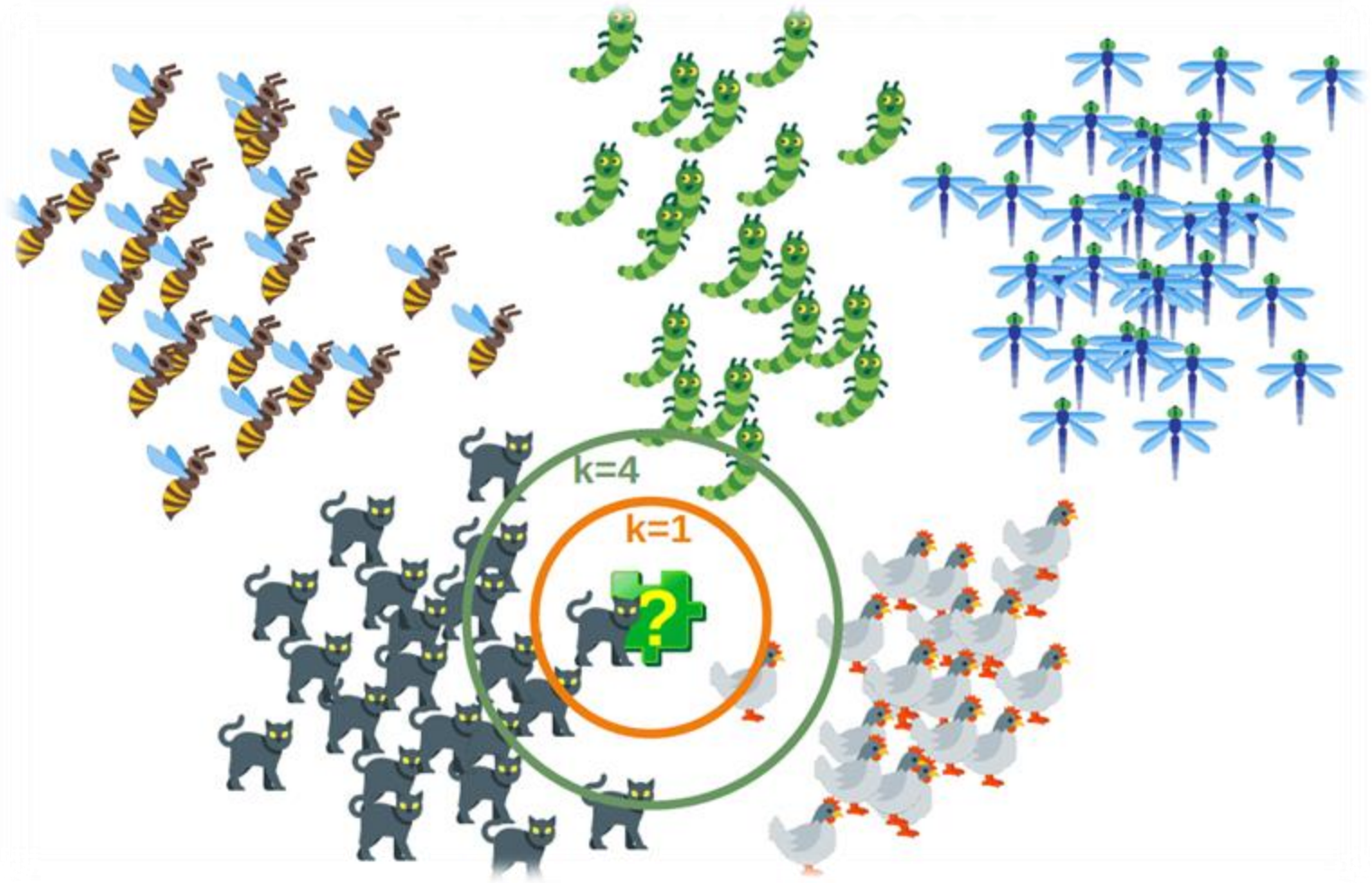


Motivation



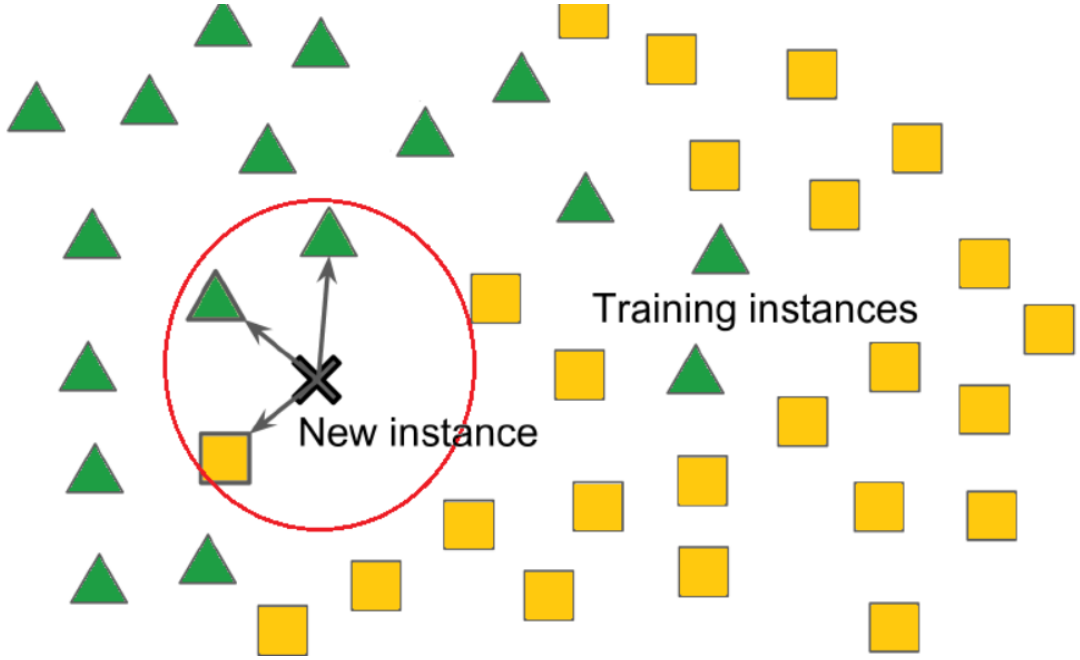
Given dataset with known categories

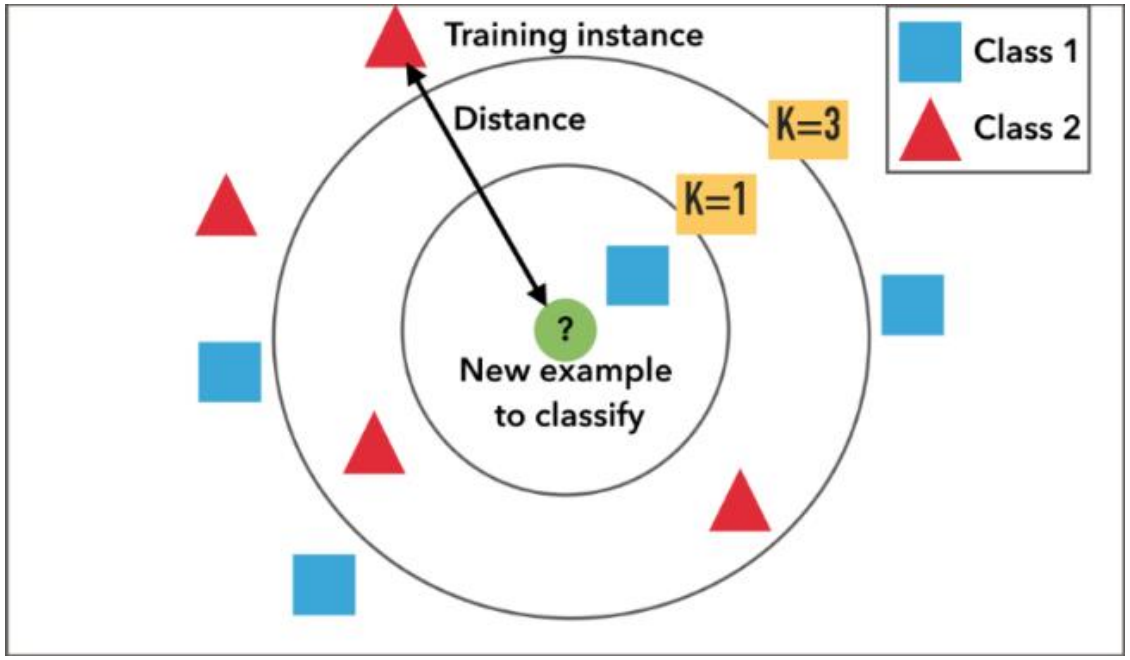
Motivation



What is a category of a new image? → **Look for K nearest neighbors**

Motivation





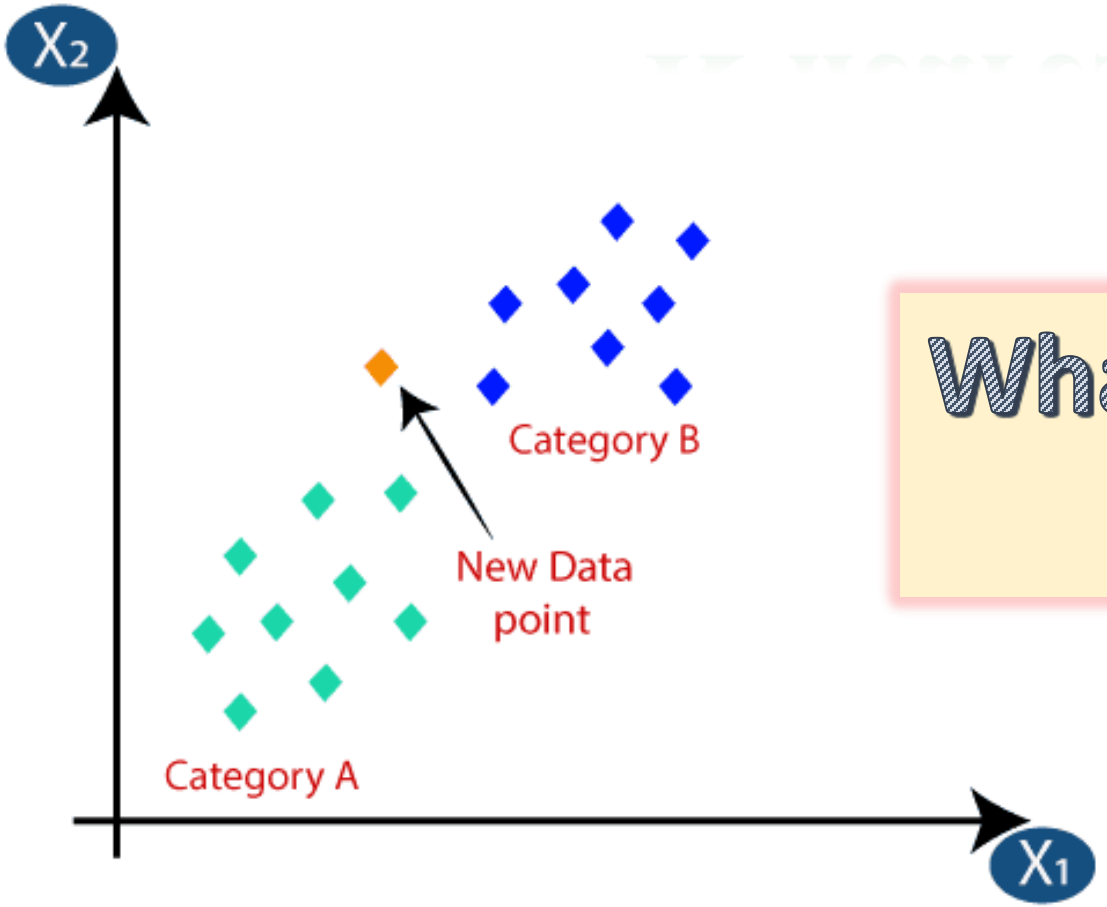
What is a category of a new image? → **Look for K-nearest neighbors**

Outline

- Machine Learning: Review
- KNN Motivation
- KNN for Classification
- How to Select K in KNN
- KNN for Regression
- KNN with K-D Tree
- Data science application: Case study

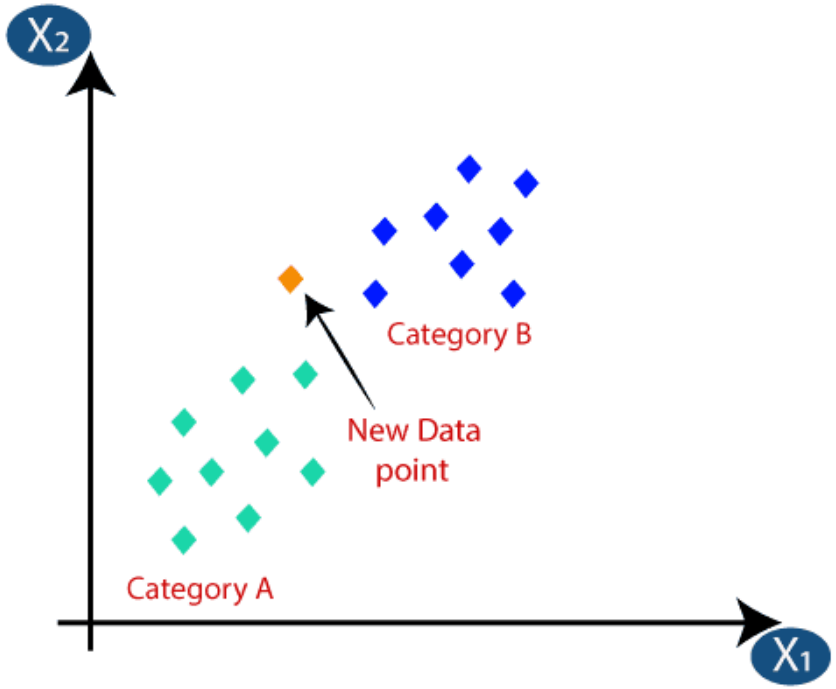
AI VIET NAM
@aivietnam.edu.vn

K-nearest Neighbor



What is the class label of a new data point?

K-nearest Neighbor



STEP 1

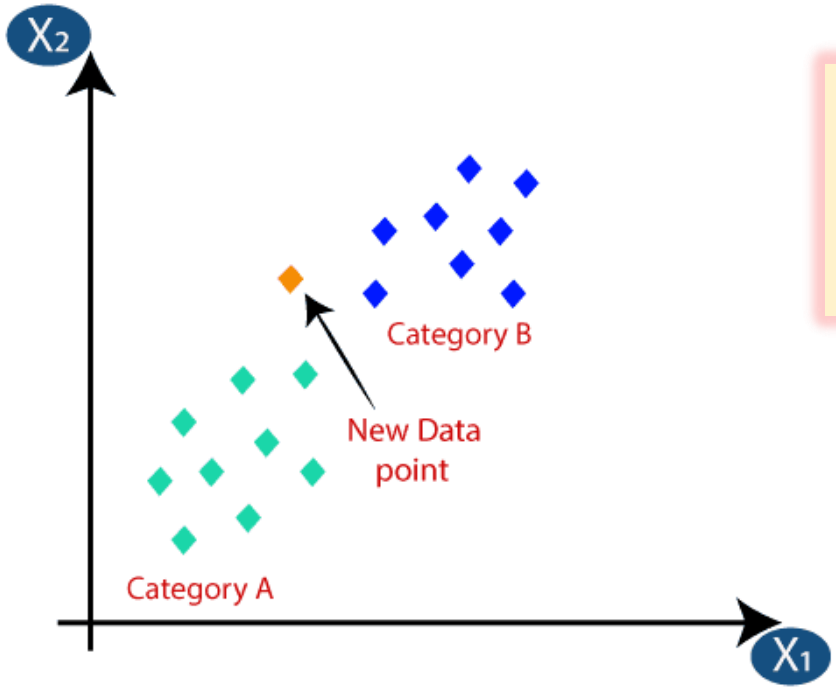
Select K-Nearest neighbors

K = 5

How to find 5 nearest neighbors

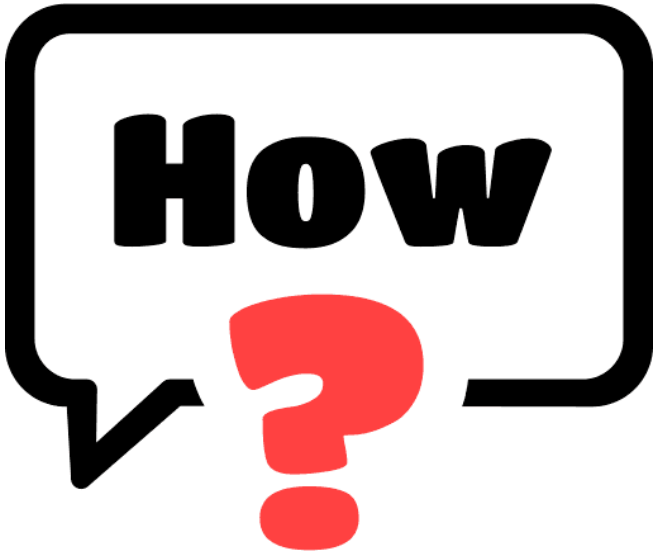
AI VIET NAM
@aivietnam.edu.vn

K-nearest Neighbor



STEP 2

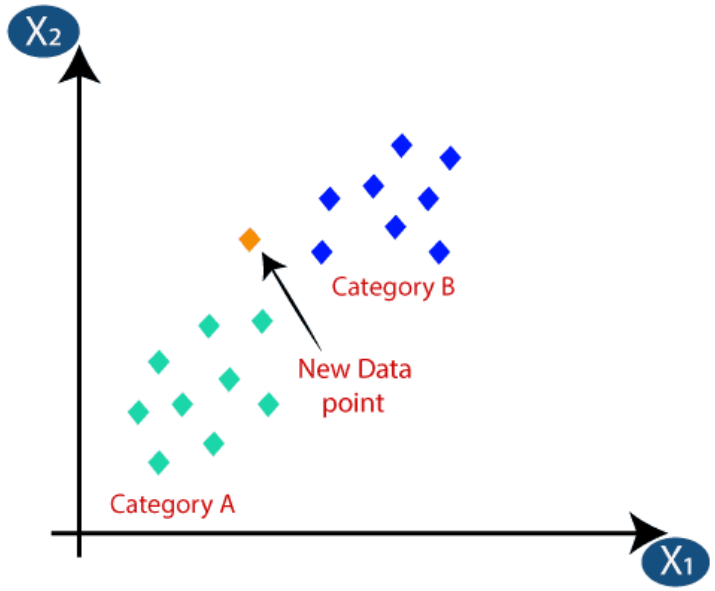
Calculate the similiar feature (distance) between test point and all data point

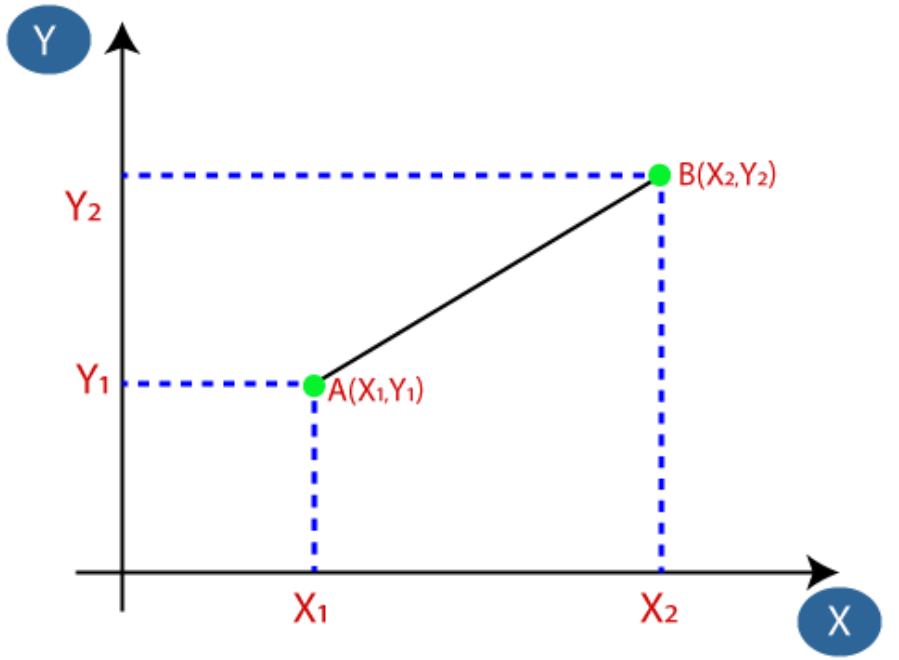


AI VIET NAM
@aivietnam.edu.vn

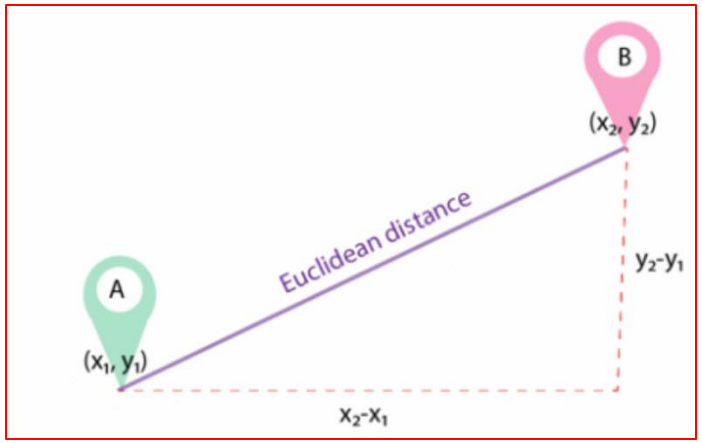
K-nearest Neighbor

STEP 2: Euclidean Distance





Euclidean Distance between A_1 and B_2 = $\sqrt{(X_2-X_1)^2+(Y_2-Y_1)^2}$

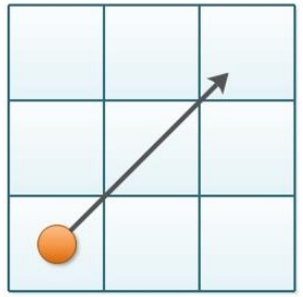


AI VIET NAM
@aivietnam.edu.vn

K-nearest Neighbor

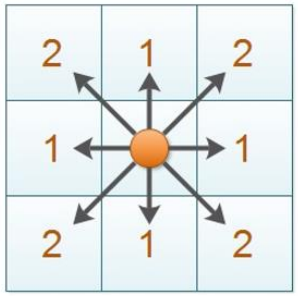
STEP 2: Other Distances

Euclidean Distance



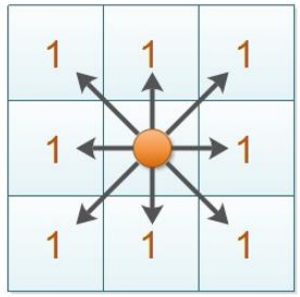
$$\sqrt{(x_1 - x_2)^2 + (y_1 - y_2)^2}$$

Manhattan Distance

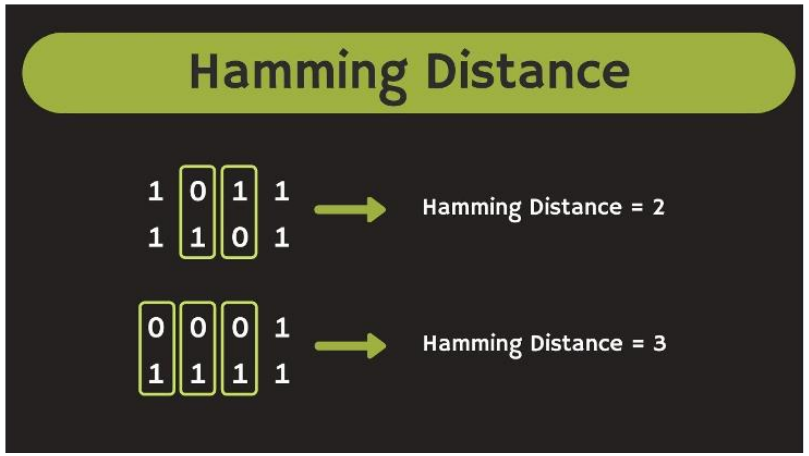


$$|x_1 - x_2| + |y_1 - y_2|$$

Chebyshev Distance

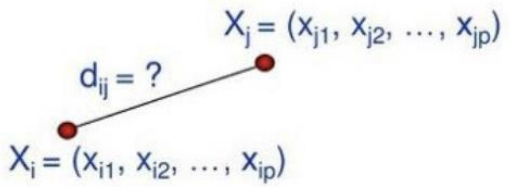


$$\max(|x_1 - x_2|, |y_1 - y_2|)$$



K-nearest Neighbor

STEP 2: Other Distances



- **Minkowski distance**

$$d(i,j) = \sqrt[q]{|x_{i1} - x_{j1}|^q + |x_{i2} - x_{j2}|^q + \ldots + |x_{ip} - x_{jp}|^q}$$

1st dimension 2nd dimension pth dimension

- **Euclidean distance**

$q = 2$

$$d(i,j) = \sqrt{|x_{i1} - x_{j1}|^2 + |x_{i2} - x_{j2}|^2 + \ldots + |x_{ip} - x_{jp}|^2}$$

- **Manhattan distance**

$q = 1$

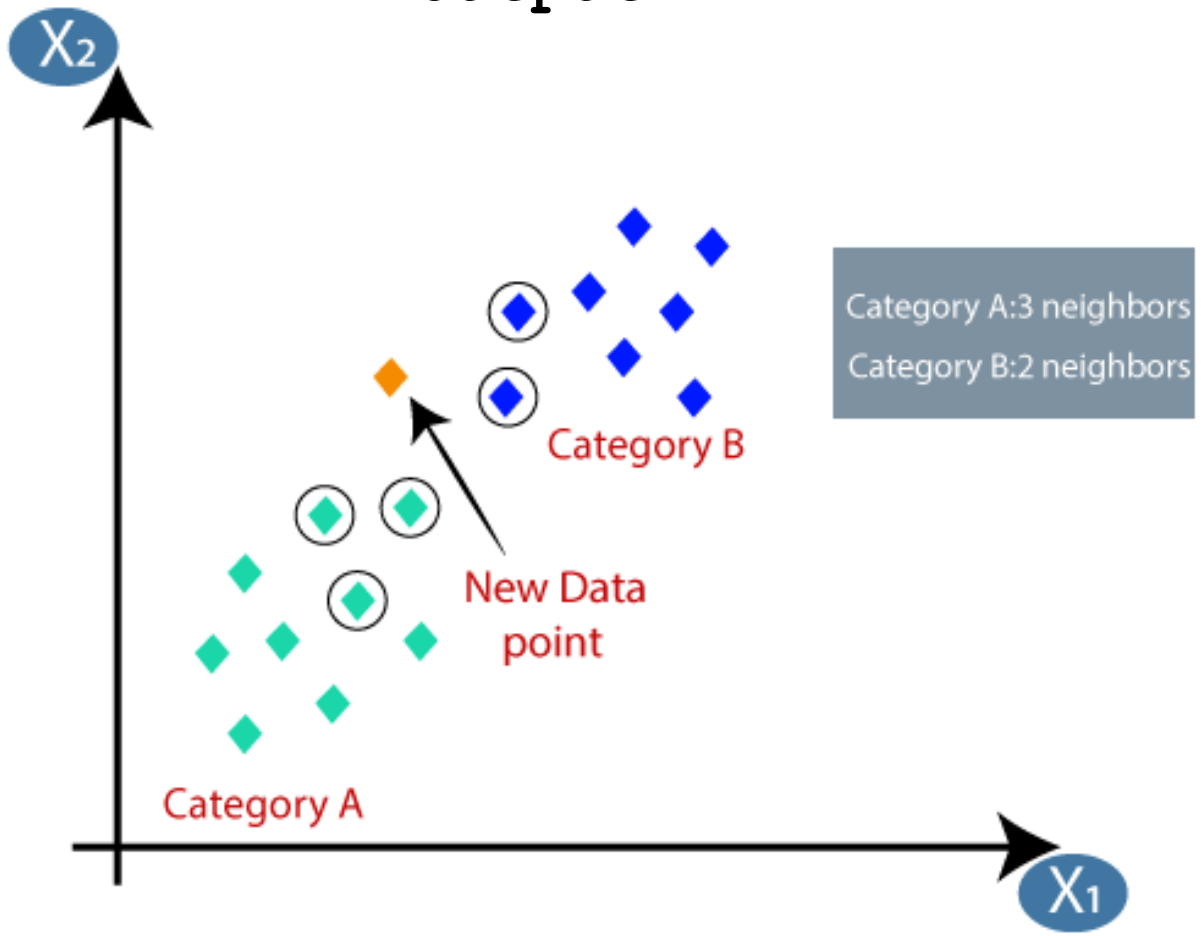
$$d(i,j) = |x_{i1} - x_{j1}| + |x_{i2} - x_{j2}| + \ldots + |x_{ip} - x_{jp}|$$

K-nearest Neighbor

STEP 3: Findout 5 nearest neighbors based on distance metrics

K-nearest Neighbor

STEP 4: Voting the label and predict an output



Iris Flower Classification

Id	SepalLengthCm	SepalWidthCm	PetalLengthCm	PetalWidthCm	Species
1	5.1	3.5	1.4	0.2	Iris-setosa
2	4.9	3.0	1.4	0.2	Iris-setosa
3	4.7	3.2	1.3	0.2	Iris-setosa
4	4.6	3.1	1.5	0.2	Iris-setosa
5	5.0	3.6	1.4	0.2	Iris-setosa
6	5.4	3.9	1.7	0.4	Iris-setosa
7	4.6	3.4	1.4	0.3	Iris-setosa
8	5.0	3.4	1.5	0.2	Iris-setosa
9	4.4	2.9	1.4	0.2	Iris-setosa
10	4.9	3.1	1.5	0.1	Iris-setosa

Setosa

Versicolor

Virginica

Iris Flower Classification

```
[ ]  file = open("/content/drive/MyDrive/AI-VN-Programming/Iris.csv", "r")
     dataset = csv.reader(file)
     dataset = np.array(list(dataset))
     dataset = np.delete(dataset, 0, 0)
     dataset = np.delete(dataset, 0, 1)
     file.close()

     trainingSet = dataset[:149]
     testingSet = dataset[149:]
```

Load dataset

Euclidean distance computation

```
def computeDistance(dataPoint1, dataPoint2):
    result = 0
    for i in range(4):
        result += (float(dataPoint1[i]) - float(dataPoint2[i]))**2
    return math.sqrt(result)
```

Iris Flower Classification

Compute and return k nearest neighbors

```python
def computeKnearestNeighbor(trainingSet, item, k):
    distances = []
    for dataPoint in trainingSet:
        distances.append(
            {
                "label": dataPoint[-1],
                "value": computeDistance(item, dataPoint)
            }
        )
    distances.sort(key=lambda x: x["value"])
    labels = [item["label"] for item in distances]
    return labels[:k]
```

Iris Flower Classification

```python
def voteTheDistances(array):
    labels = set(array)
    result = ""
    maxOccur = 0
    for label in labels:
        num = array.count(label)
        if(num > maxOccur):
            maxOccur = num
            result = label

    return result
```

Voting the distance to find the predicted result

Iris Flower Classification

```
k = 5
# print(testingSet)
for item in testingSet:
  knn = computeKnearestNeighbor(trainingSet, item, k)
  result = voteTheDistances(knn)
  print("GT = ", item[-1], ", Prediction: =", result)
```

```
GT =  Iris-virginica , Prediction: = Iris-virginica
```

Testing Phase

Outline

- Machine Learning: Review
- KNN Motivation
- KNN for Classification
- How to Select K in KNN
- KNN for Regression
- KNN with K-D Tree
- Data science application: Case study

HOW TO SELECT K IN THE K-NN ?

How to select *k* in K-NN

```
from sklearn.metrics import accuracy_score, classification_report
print(round(accuracy_score(y_test, y_pred),2))
```

```
0.97
```

Accuracy

Classification Report

```
from sklearn.metrics import accuracy_score, classification_report
print(classification_report(y_test, y_pred))
```

```
                precision    recall  f1-score   support

    Iris-setosa       1.00      1.00      1.00        11
Iris-versicolor       1.00      0.92      0.96        13
 Iris-virginica       0.86      1.00      0.92         6

      accuracy                           0.97        30
     macro avg       0.95      0.97      0.96        30
  weighted avg       0.97      0.97      0.97        30
```

Error Analysis

- Cancer classification => Train Logistic Regression => You got 1% error on the the test test => 99% accuracy.
- How about the case if only 0.5% of patient have cancer on the Test set.

Simple Algorithm

```
# Không quan tâm x là gì
def predictCancer(x):
  y = 0
 return y
```

What is the accuracy?

Logistic Regression

```
# import the class
from sklearn.linear_model import LogisticRegression

# instantiate the model (using the default parameters)
logreg = LogisticRegression(random_state=16)

# fit the model with data
logreg.fit(X_train, y_train)

def predictCancer(x):
  y_pred = logreg.predict(x)
 return y_pred
```

Accuracy: 99%

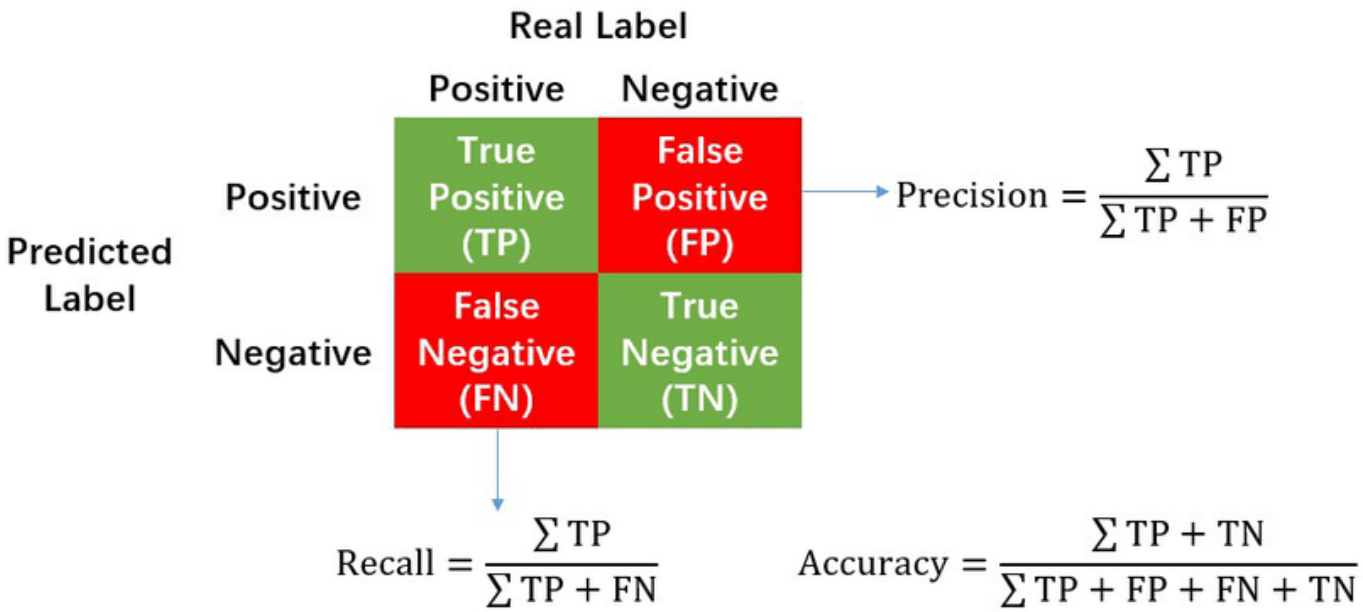
Error Analysis

Precision:

Trong tất cả các bệnh nhân mà chương trình dự đoán bị cancer, tỉ lệ cancer thật sự là bao nhiêu?

Recall:

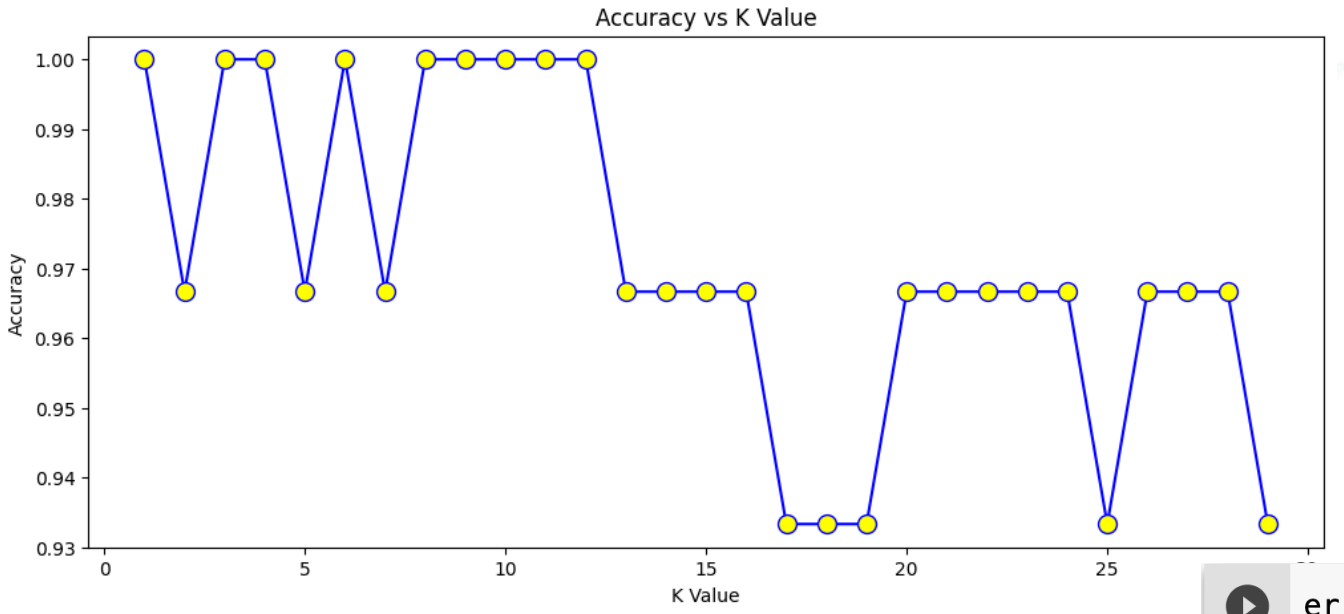
Trong tất cả các bệnh nhân thật sự bị cancer, chương trình dự đoán đúng được bao nhiêu trường hợp?



Error Analysis

	Precision (P)	Recall (R)
Algorithm 1	0.5	0.4
Algorithm 2	0.7	0.1
Algorithm 3	0.02	1.0

How to select *k* in K-NN



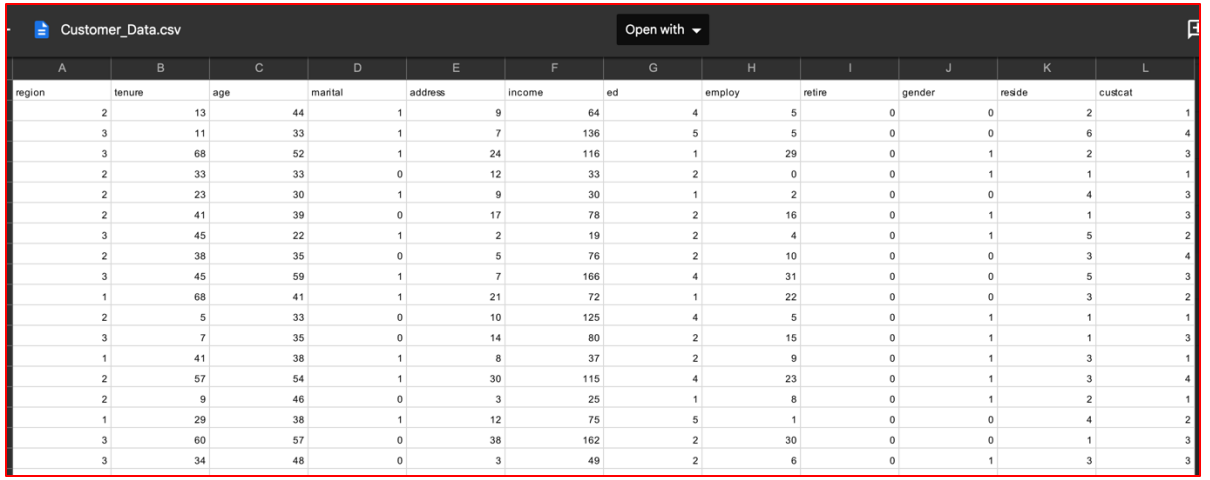
Bạn có thắc mắc gì chỗ này không?

```
error = []
# Calculating accuracy for K-values between 1 and 30
for i in range(1, 30):
    knn = KNeighborsClassifier(n_neighbors=i)
    knn.fit(X_train, y_train)
    pred_i = knn.predict(X_test)
    error.append(accuracy_score(pred_i,y_test))

plt.figure(figsize=(12, 5))
plt.plot(range(1, 30), error, color='blue', marker='o',
        markerfacecolor='yellow', markersize=10)
plt.title('Accuracy vs K Value')
plt.xlabel('K Value')
plt.ylabel('Accuracy')
```

How to select *k* in K-NN

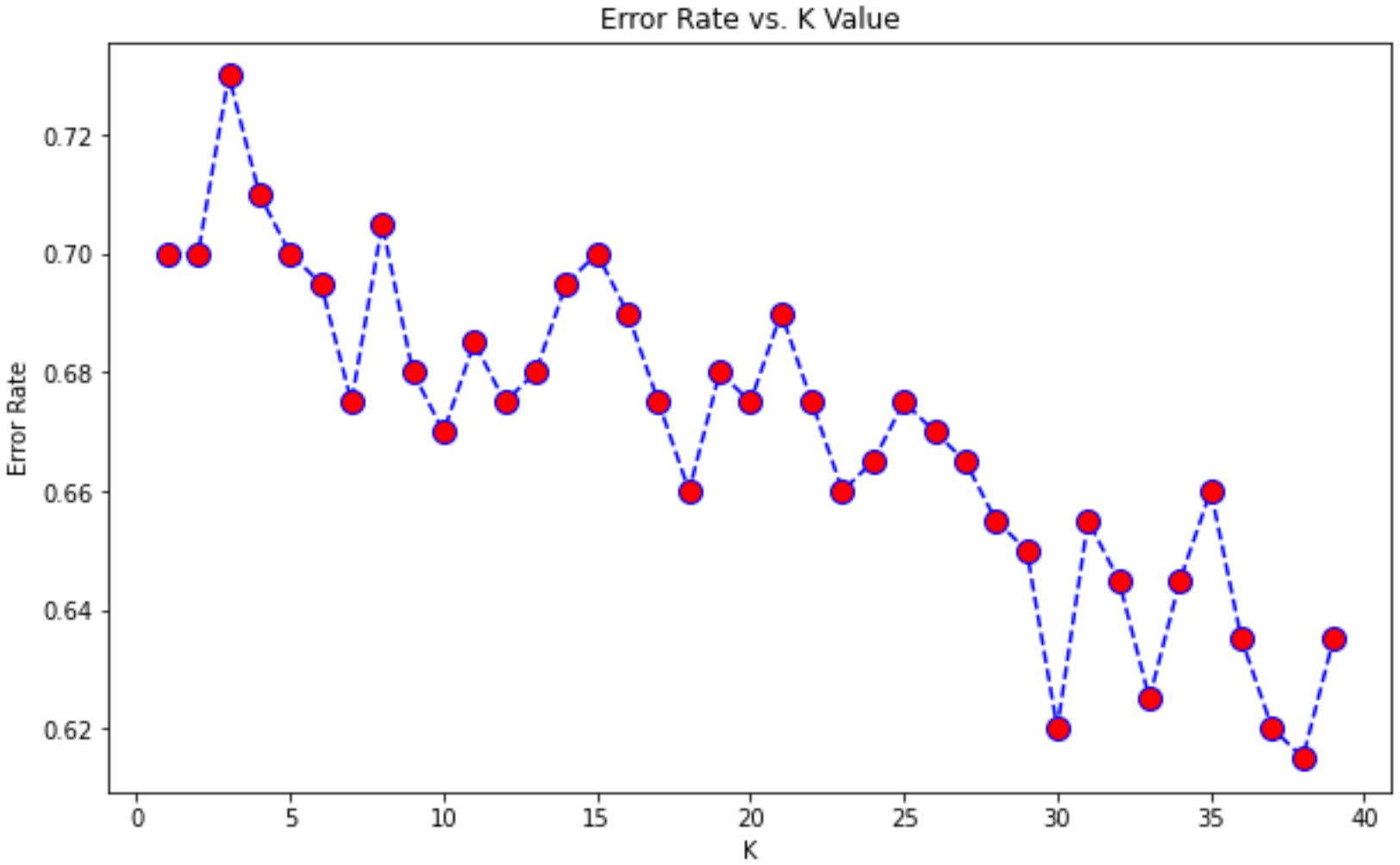
	region	tenure	age	marital	address	income	ed	employ	retire	gender	reside	custcat
0	2	13	44	1	9	64.0	4	5	0.0	0	2	1
1	3	11	33	1	7	136.0	5	5	0.0	0	6	4
2	3	68	52	1	24	116.0	1	29	0.0	1	2	3
3	2	33	33	0	12	33.0	2	0	0.0	1	1	1
4	2	23	30	1	9	30.0	1	2	0.0	0	4	3

- 1- Basic Service
- 2- E-Service
- 3- Plus Service
- 4- Total Service

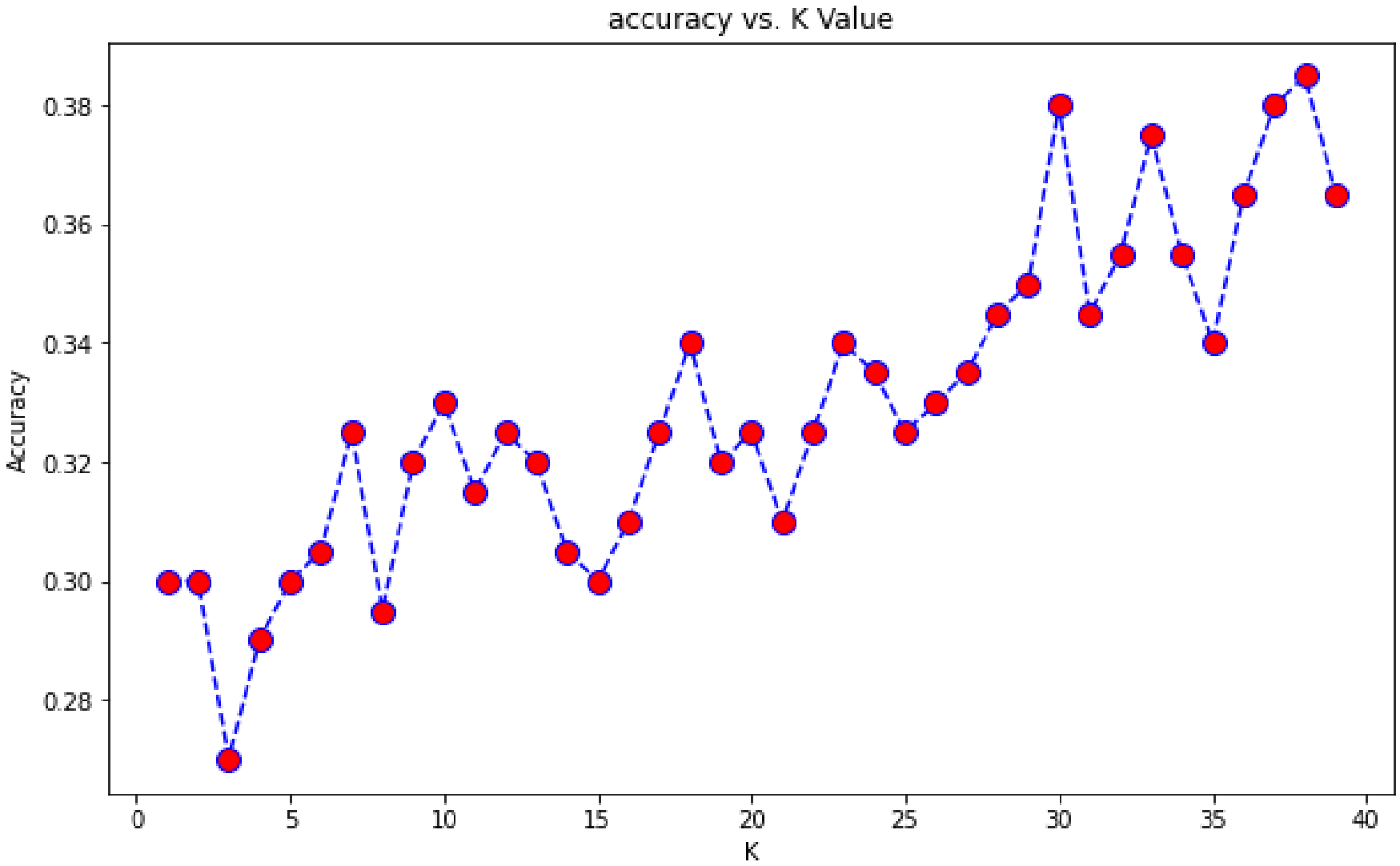


region	tenure	age	marital	address	income	ed	employ	retire	gender	reside	custcat
2	13	44	1	9	64	4	5	0	0	2	1
3	11	33	1	7	136	5	5	0	0	6	4
3	68	52	1	24	116	1	29	0	1	2	3
2	33	33	0	12	33	2	0	0	1	1	1
2	23	30	1	9	30	1	2	0	0	4	3
2	41	39	0	17	78	2	16	0	1	1	3
3	45	22	1	2	19	2	4	0	1	5	2
2	38	35	0	5	76	2	10	0	0	3	4
3	45	59	1	7	166	4	31	0	0	5	3
1	68	41	1	21	72	1	22	0	0	3	2
2	5	33	0	10	125	4	5	0	1	1	1
3	7	35	0	14	80	2	15	0	1	1	3
1	41	38	1	8	37	2	9	0	1	3	1
2	57	54	1	30	115	4	23	0	1	3	4
2	9	46	0	3	25	1	8	0	1	2	1
1	29	38	1	12	75	5	1	0	0	4	2
3	60	57	0	38	162	2	30	0	0	1	3
3	34	48	0	3	49	2	6	0	1	3	3

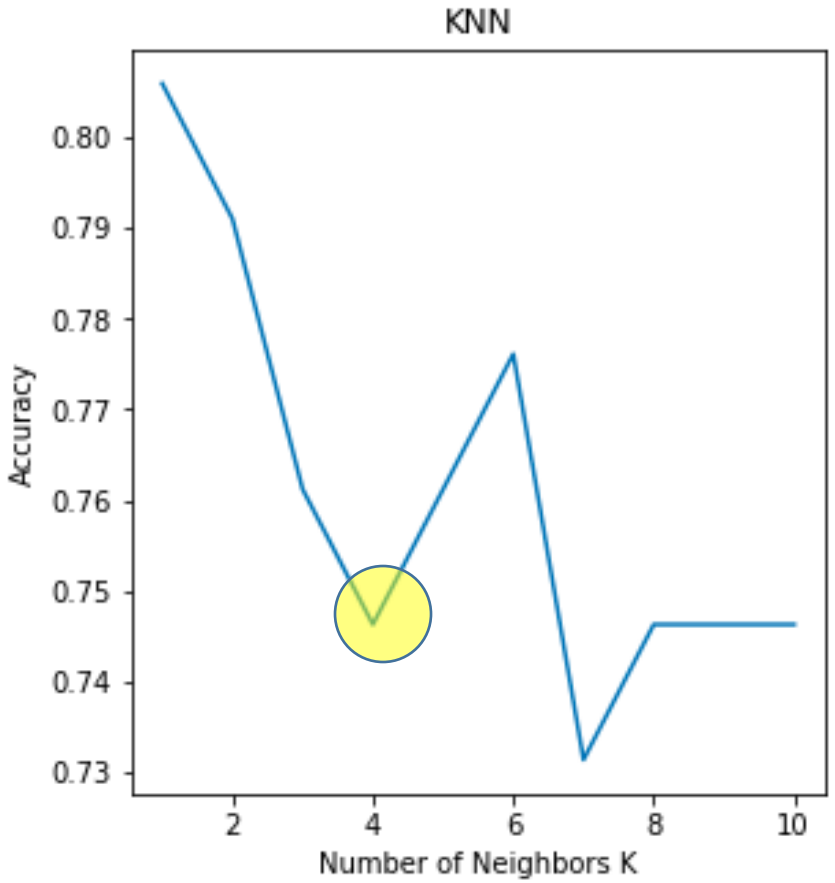
How to select *k* in K-NN

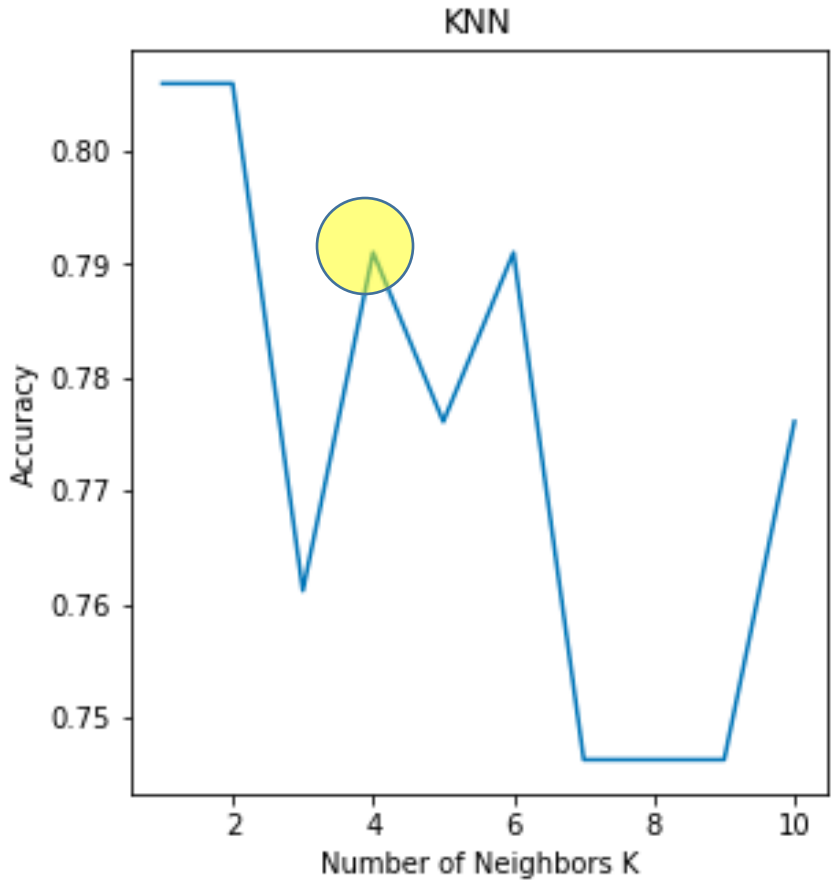


How to select *k* in K-NN



K-NN with Bruce-force





Did you find any difference between the two graphs?
'Brute-force' algorithm, same dataset, same distance metric

KNN Review

1.Load the data
2.Initialize the value of k
3.For getting the predicted class, iterate from 1 to total number of training data points
4.Calculate the distance between test data and each row of training dataset. Distance metrics: Euclidean distance, Manhattan distance, Minkowski distance, Chebyshev, cosine, Hamming Distance etc.
5.Sort the calculated distances in ascending order based on distance values
6.Get top k rows from the sorted array
7.Get the most frequent class of these rows
8.Return the predicted class

Which step might cause the problem? And Why?

K-NN with Bruce-force

Modified

```
scores1 = []
neighbors = list(range(1,10))
for i in neighbors:
    pred = knn_modified(X_train,y_train,X_test,k=i)
    accuracy =  accuracy_score(y_test, pred)
    scores1.append(accuracy)
```

Sklearn

```
neighbors = list(range(1,10))
scores = []
for k in neighbors:
    knn = KNeighborsClassifier(n_neighbors=k,
                               algorithm='brute',
                               metric='euclidean')
    pred1 = knn.fit(X_train, y_train)
    pred = pred1.predict(X_test)
    accuracy =  accuracy_score(y_test, pred)
    scores.append(accuracy)
```

Result

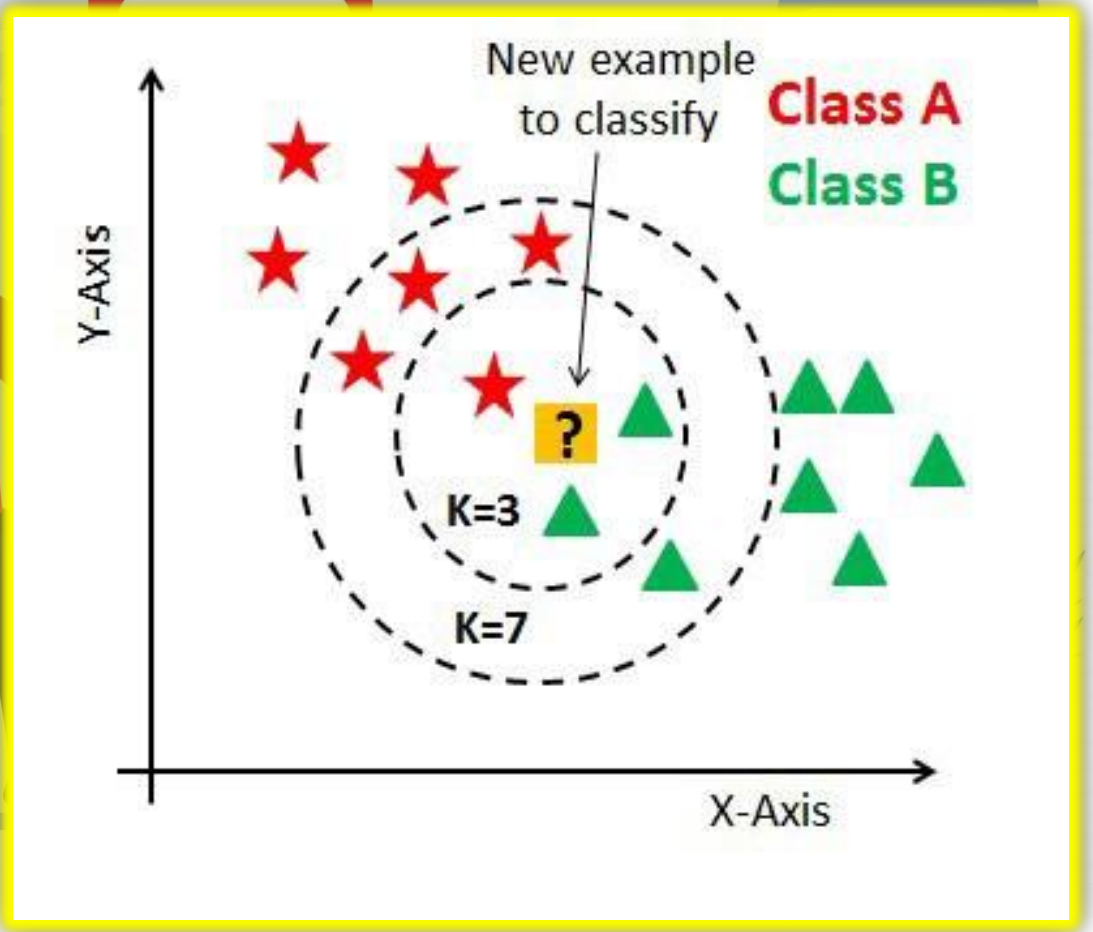
```
print("\t\t KNN in SKlearn  \t Modified KNN")
for i in range(0,4):
    print("k = ",i+1, "\t\t     ",round(scores[i],3),"\t\t    ",round(scores1[i],3))
```

```
                KNN in SKlearn          Modified KNN
k =  1              0.806                   0.806
k =  2              0.791                   0.806
k =  3              0.761                   0.761
k =  4              0.746                   0.791
```

In case of Odd k values, it takes the majority. For even k rows, majority classes are selected. If it happens to have two or more classes having majority. Those two or more major class distances will go to the distance metric loop again and check which class has the lowest distance metric and that class is chosen as the majority class.

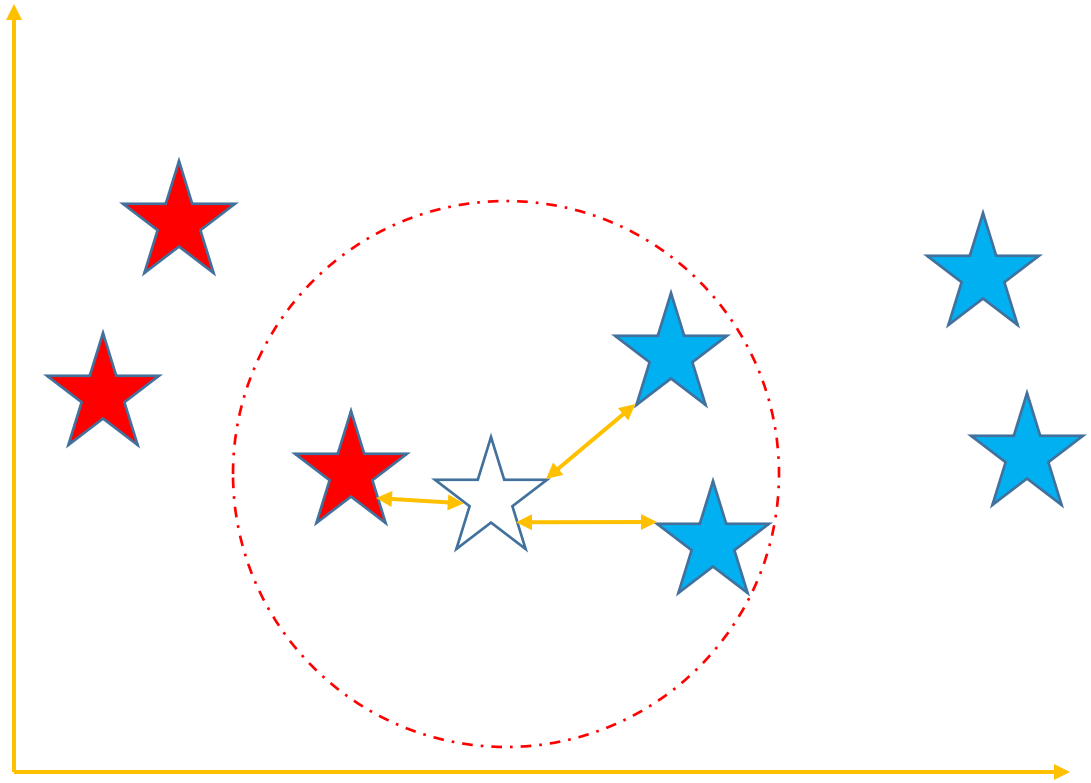
Discussion

WEIGHT IN K-NEAREST NEIGHBOR?

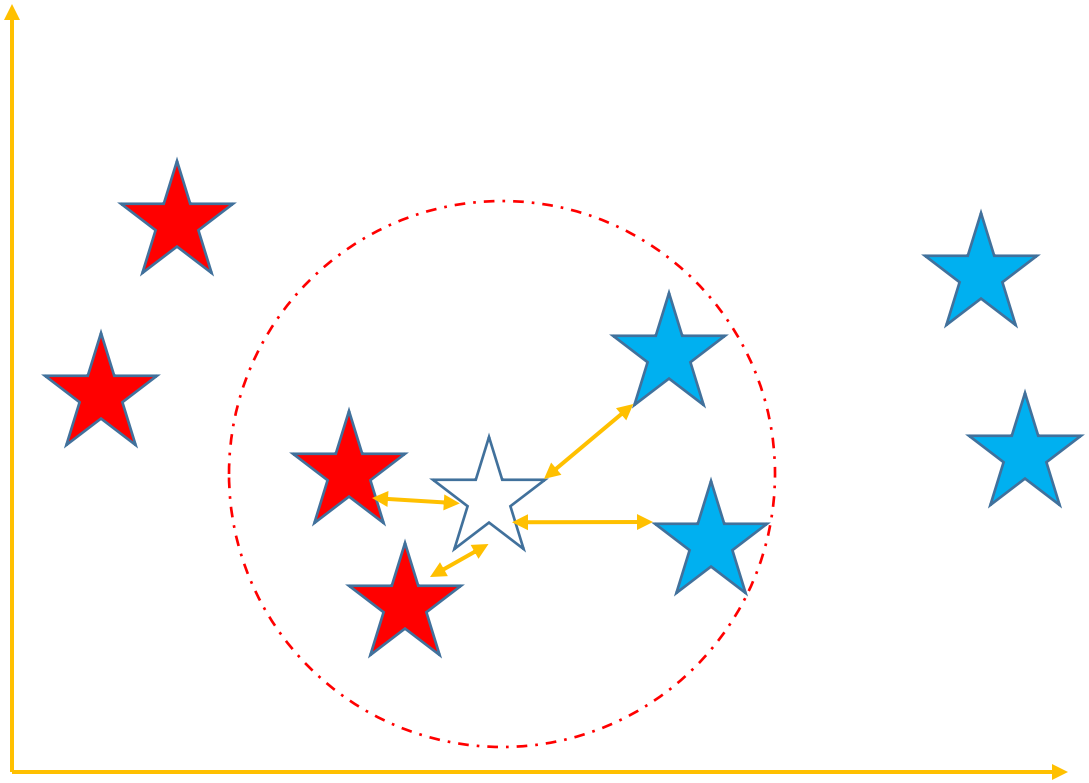


Discussion

WEIGHT IN K-NEAREST NEIGHBOR?

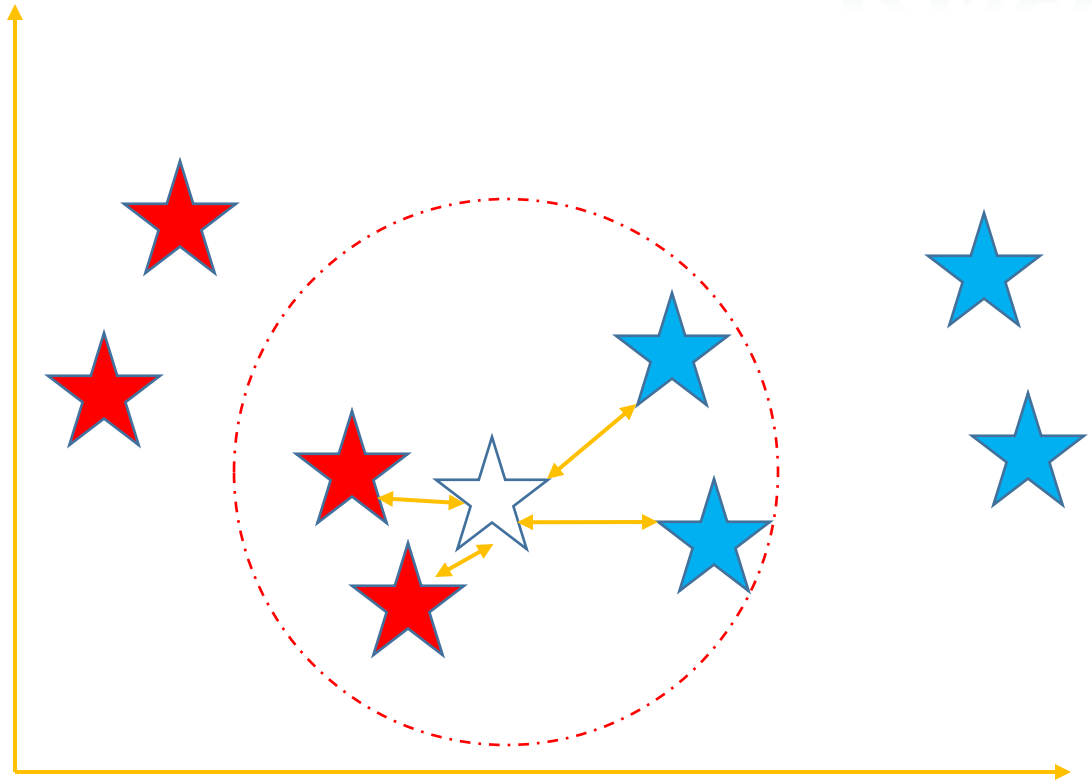


Even K



Odd K

Discussion



UNIFORM WEIGHTS
DISTANCE WEIGHTS
USER-DEFINED WEIGHTS

Discussion

Uniform weight

```
[171] classifier = KNeighborsClassifier(n_neighbors=8, weights = 'uniform', p = 2)
      classifier.fit(X_train, y_train)
      y_pred = classifier.predict(X_test)
```

```
print("accuracy %.2f%%"%(100*accuracy_score(y_test,y_pred)))
print(classification_report(y_test,y_pred))
```

```
accuracy 96.00%
              precision    recall  f1-score   support

           0       1.00      1.00      1.00        19
           1       0.91      1.00      0.95        30
           2       1.00      0.88      0.94        26

    accuracy                           0.96        75
   macro avg       0.97      0.96      0.96        75
weighted avg       0.96      0.96      0.96        75
```

Discussion

Distance weight

```
classifier = KNeighborsClassifier(n_neighbors=8, p = 2, weights="distance")
classifier.fit(X_train, y_train)
y_pred = classifier.predict(X_test)
```

```
print("accuracy %.2f%%"%(100*accuracy_score(y_test,y_pred)))
print(classification_report(y_test,y_pred))
```

```
accuracy 97.33%
              precision    recall  f1-score   support

           0       1.00      1.00      1.00        19
           1       0.97      0.97      0.97        30
           2       0.96      0.96      0.96        26

    accuracy                           0.97        75
   macro avg       0.98      0.98      0.98        75
weighted avg       0.97      0.97      0.97        75
```

Discussion

Customize weight

```
[175] def customizeWeight(distances):
        sigma = 0.5
        return np.exp(-distances**2/sigma)
```

```
[176] classifier = KNeighborsClassifier(n_neighbors=4, p = 2, weights=customizeWeight)
      classifier.fit(X_train, y_train)
      y_pred = classifier.predict(X_test)
```

```
print("accuracy %.2f%%"%(100*accuracy_score(y_test,y_pred)))
print(classification_report(y_test,y_pred))
```

```
accuracy 96.00%
              precision    recall  f1-score   support

           0       1.00      1.00      1.00        19
           1       0.97      0.93      0.95        30
           2       0.93      0.96      0.94        26

    accuracy                           0.96        75
   macro avg       0.96      0.96      0.96        75
weighted avg       0.96      0.96      0.96        75
```

Discussion

Classification Report

```
[171] classifier = KNeighborsClassifier(n_neighbors=8, weights = 'uniform', p = 2)
      classifier.fit(X_train, y_train)
      y_pred = classifier.predict(X_test)
```

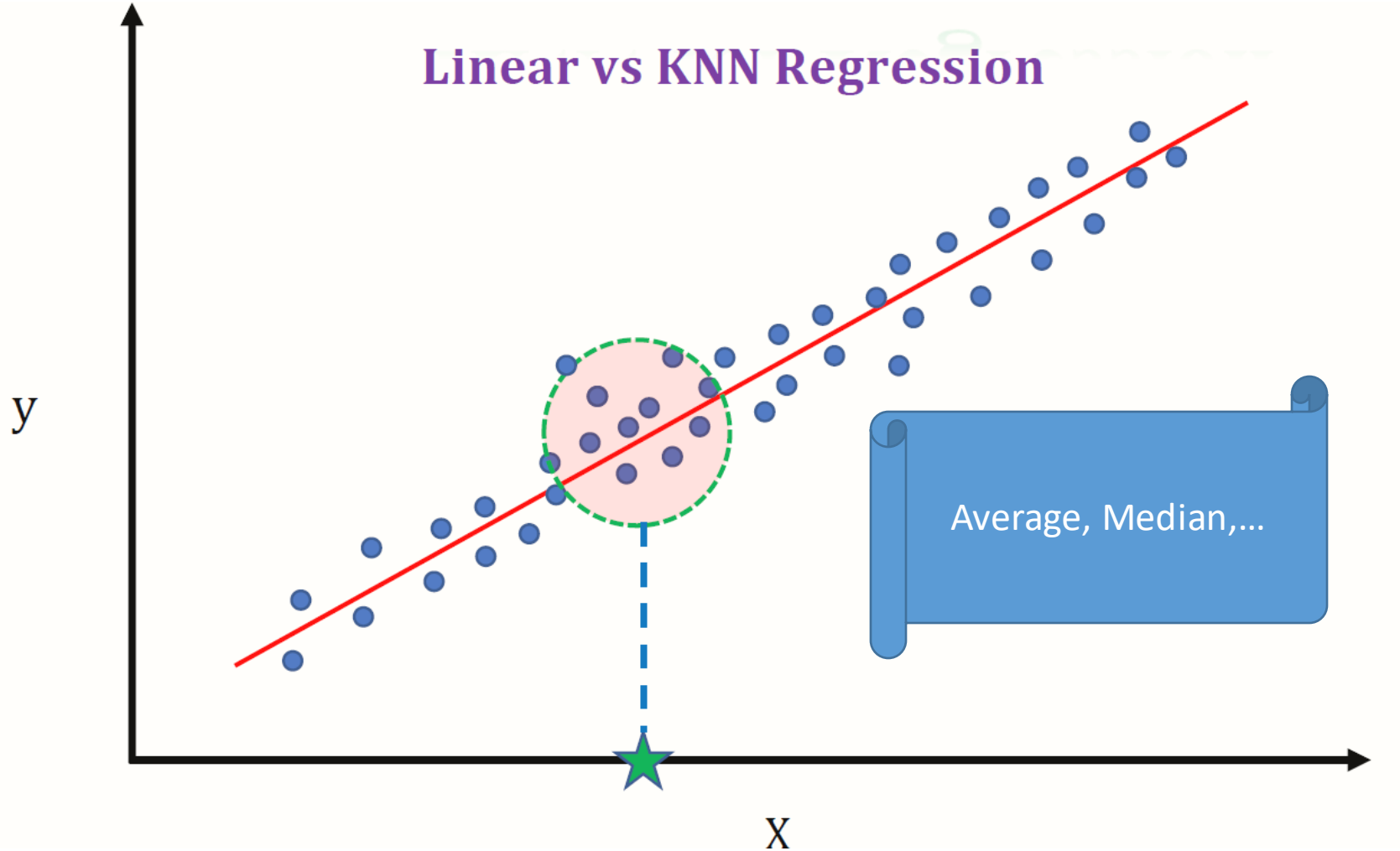
```
print("accuracy %.2f%%"%(100*accuracy_score(y_test,y_pred)))
print(classification_report(y_test,y_pred))
```

```
accuracy 96.00%
              precision    recall  f1-score   support

           0       1.00      1.00      1.00        19
           1       0.91      1.00      0.95        30
           2       1.00      0.88      0.94        26

    accuracy                           0.96        75
   macro avg       0.97      0.96      0.96        75
weighted avg       0.96      0.96      0.96        75
```

Outline

- Machine Learning: Review
- KNN Motivation
- KNN for Classification
- How to Select K in KNN
- KNN for Regression
- KNN with K-D Tree
- Data science application: Case study

KNN for Regression



KNN for Regression

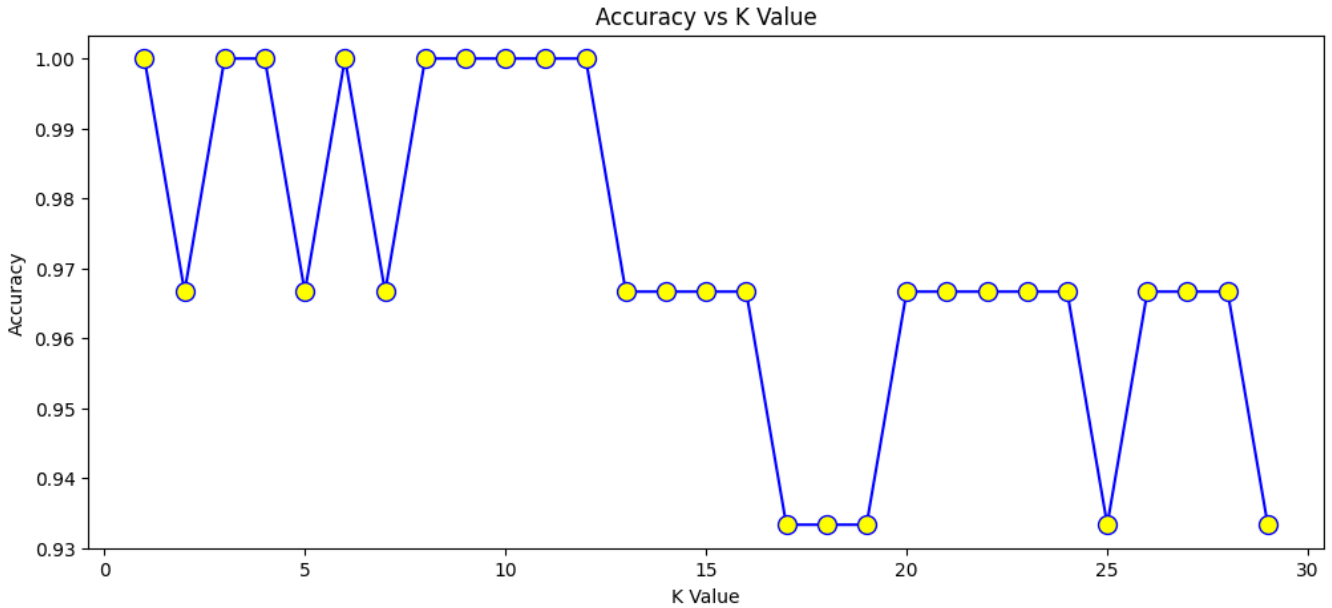
Choose a value for K

Calculate the distance

Find the K nearest neighbors

Calculate the prediction

Issue Review



Bạn có thắc mắc gì chỗ này không?

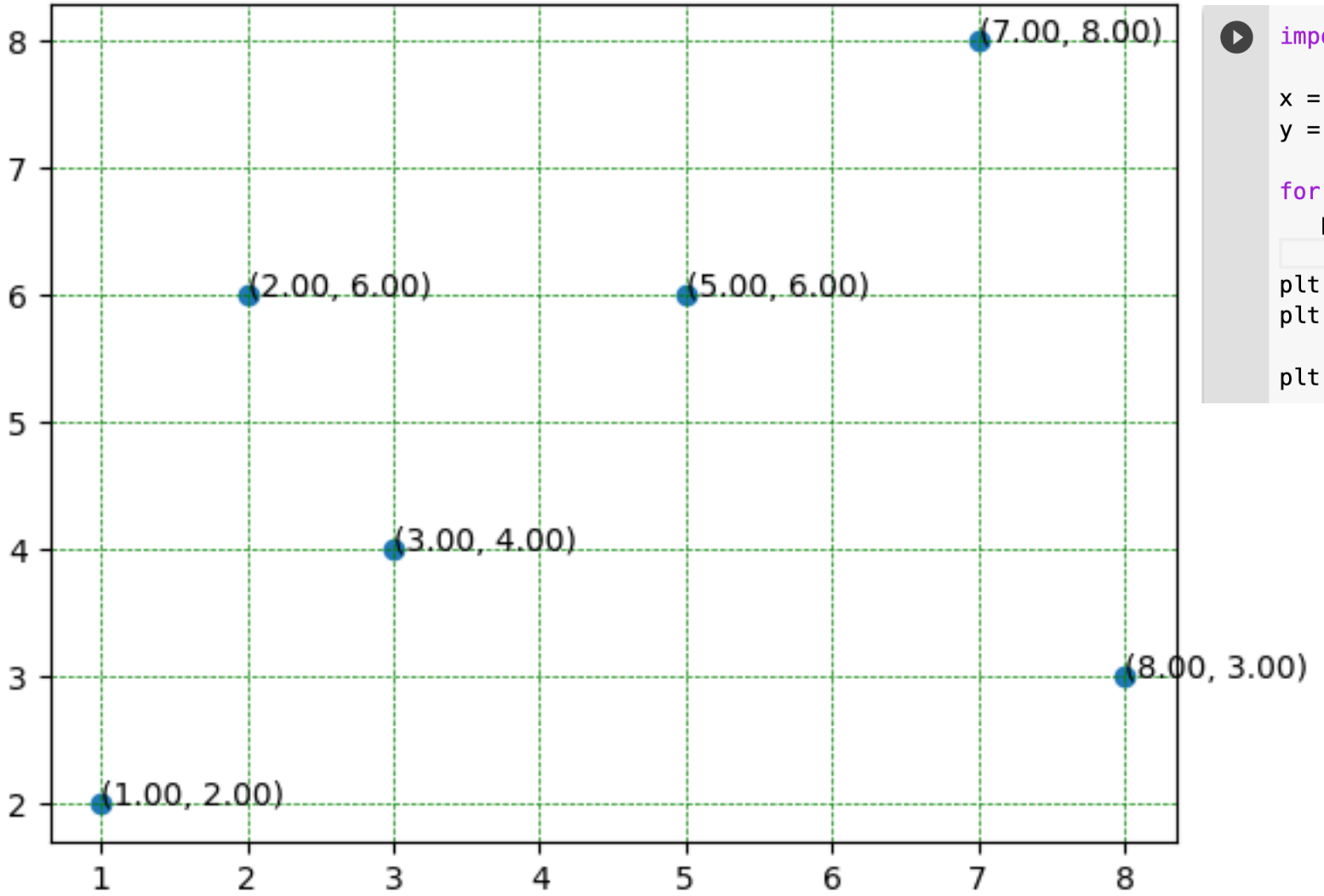
```
error = []
# Calculating accuracy for K-values between 1 and 30
for i in range(1, 30):
    knn = KNeighborsClassifier(n_neighbors=i)
    knn.fit(X_train, y_train)
    pred_i = knn.predict(X_test)
    error.append(accuracy_score(pred_i,y_test))

plt.figure(figsize=(12, 5))
plt.plot(range(1, 30), error, color='blue', marker='o',
         markerfacecolor='yellow', markersize=10)
plt.title('Accuracy vs K Value')
plt.xlabel('K Value')
plt.ylabel('Accuracy')
```

Outline

- Machine Learning: Review
- KNN Motivation
- KNN for Classification
- How to Select K in KNN
- KNN for Regression
- KNN with K-D Tree
- Data science application: Case study

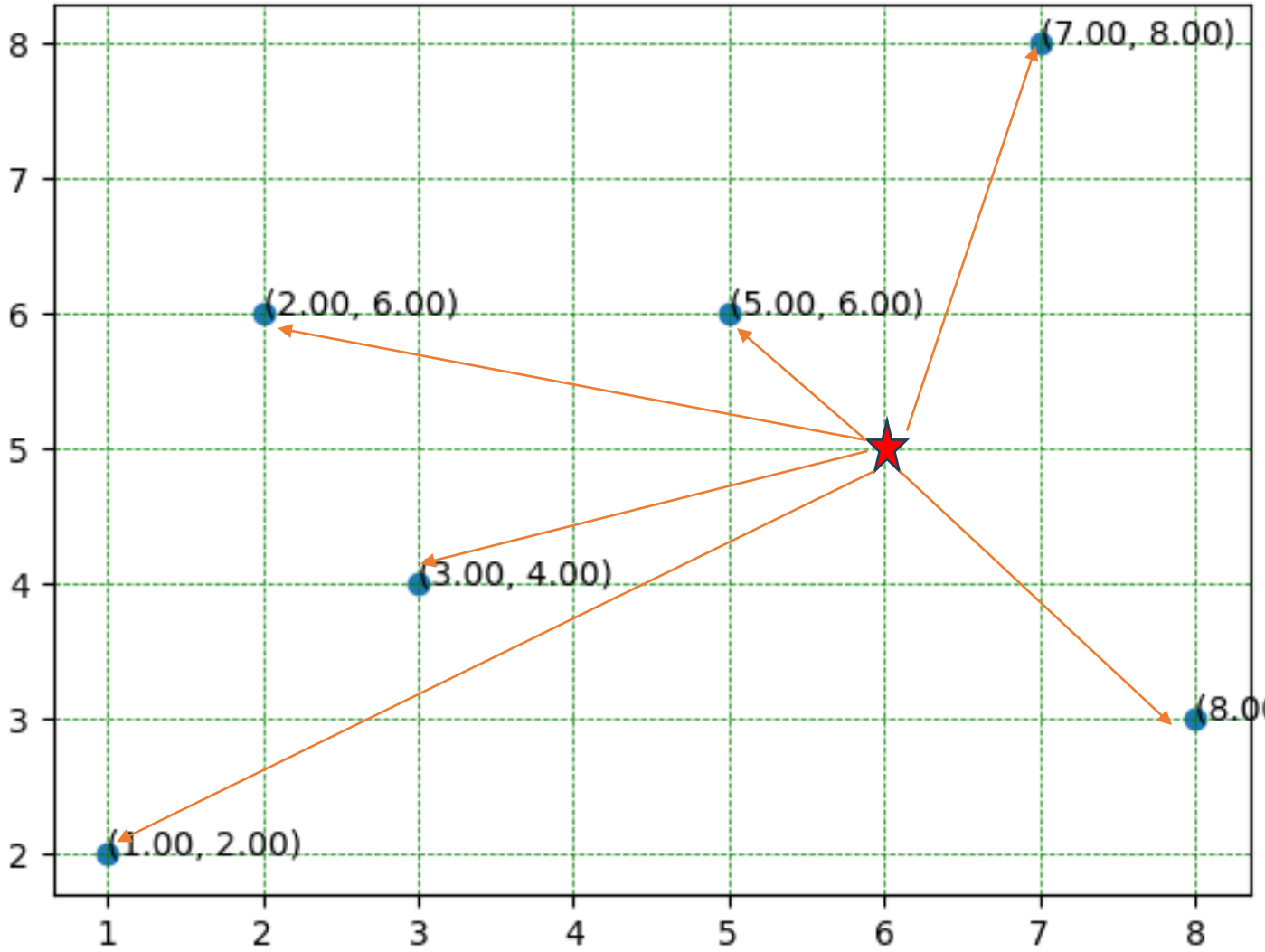
AI VIET NAM
@aivietnam.edu.vn

KNN with Brute Force




```python
import matplotlib.pyplot as plt

x = [1, 2, 3, 5, 7, 8]
y = [2, 6, 4, 6, 8, 3]

for xy in zip(x, y):
    plt.annotate('(%.2f, %.2f)' % xy, xy=xy)

plt.scatter(x, y)
plt.grid(color = 'green', linestyle = '--', linewidth = 0.5)

plt.show()
```

KNN with Brute Force




```
import matplotlib.pyplot as plt

x = [1, 2, 3, 5, 7, 8]
y = [2, 6, 4, 6, 8, 3]

for xy in zip(x, y):
    plt.annotate('(%.2f, %.2f)' % xy, xy=xy)

plt.scatter(x, y)
plt.grid(color = 'green', linestyle = '--', linewidth = 0.5)

plt.show()
```

New datapoint

1. Find distance of each point in the dataset
2. The point with lowest distance would be nears

Outline

- Machine Learning: Review
- KNN Motivation
- KNN for Classification
- How to Select K in KNN
- KNN for Regression
- KNN with K-D Tree
- Data science application: Case study

K-D Tree Implementation Solution

Algorithm 1: K-D Tree Construction

Function *kdtree(pointList, depth)*:

Input: a list of points, *pointList*, and an integer, *depth* ;

Output: a k-d tree rooted at the median point of *pointList* ;

```
/* Select axis based on depth so that axis cycles
   through all valid values                          */
```

let $axis := depth \mod k$;

```
/* Sort point list and choose median as pivot element
   */
```

select median by *axis* from *pointList*;

```
/* Create node and construct subtree                 */
```

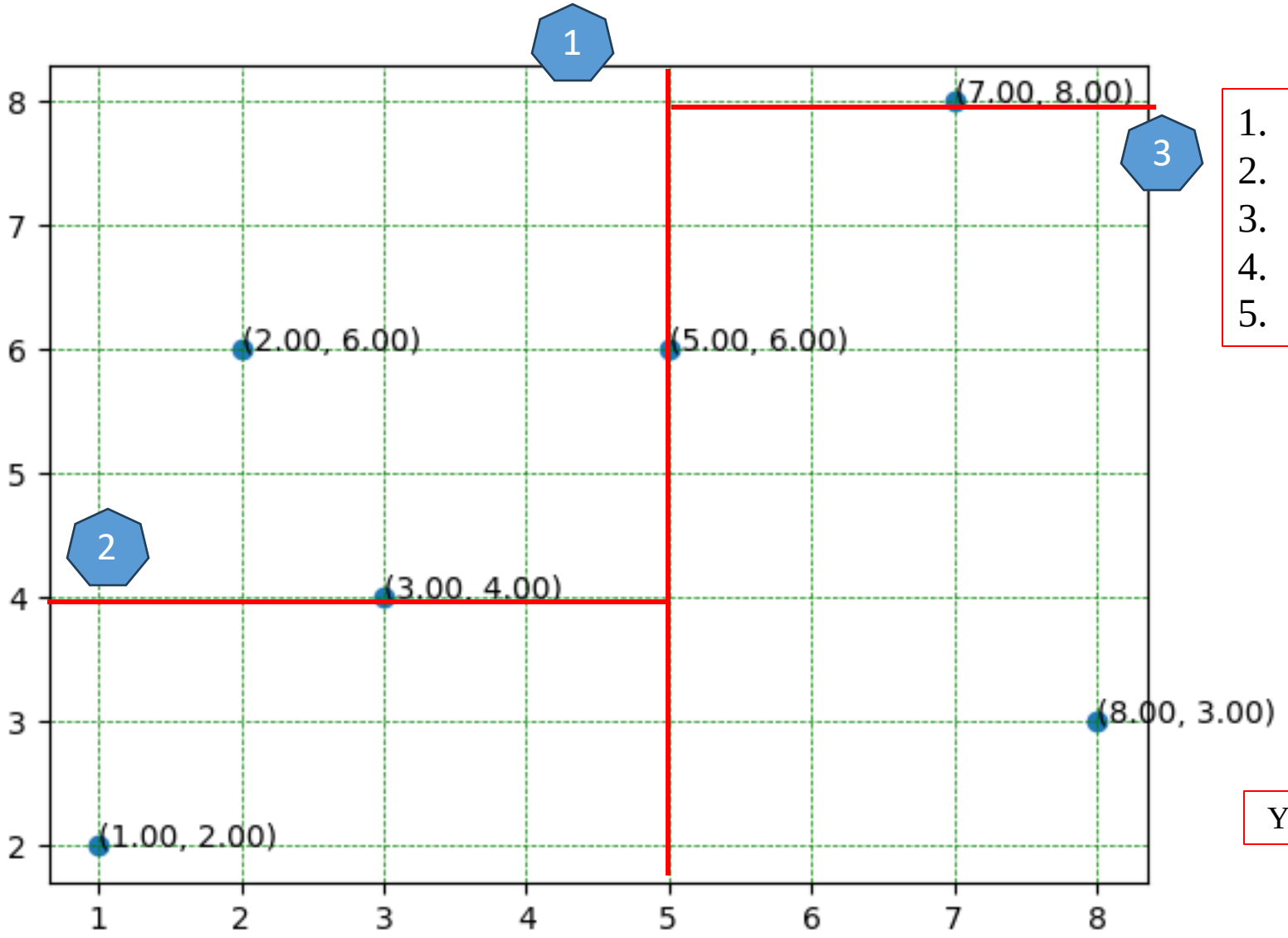
let *node.location* := *median*;

let *node.leftChild* := kdtree(points in *pointList* before *median*, $depth + 1$);

let *node.rightChild* := kdtree(points in *pointList* after *median*, $depth + 1$);

return *node*;

K-D Tree Implementation Solution



1. Pick any one feature at random
2. Find median
3. Split dataset in approximate equal halves
4. Pick next feature and repeat step #2,3
5. Continue until all data points are partitioned

X = 1, 2, 3, 5, 7, 8; median = 5

1

Left

Right

Y = 2, 4, 6; median = 4

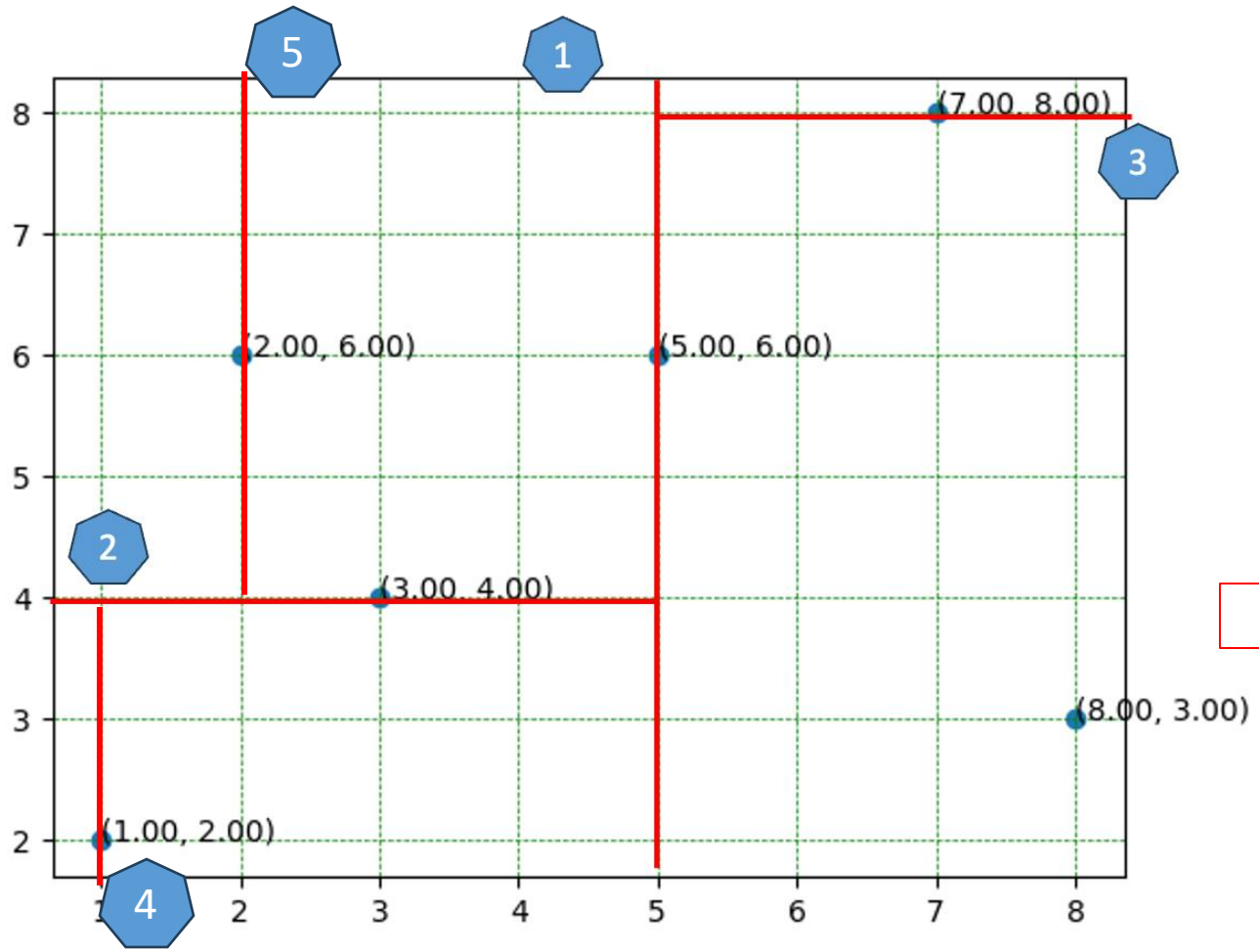
Y = 3, 8; median = 8

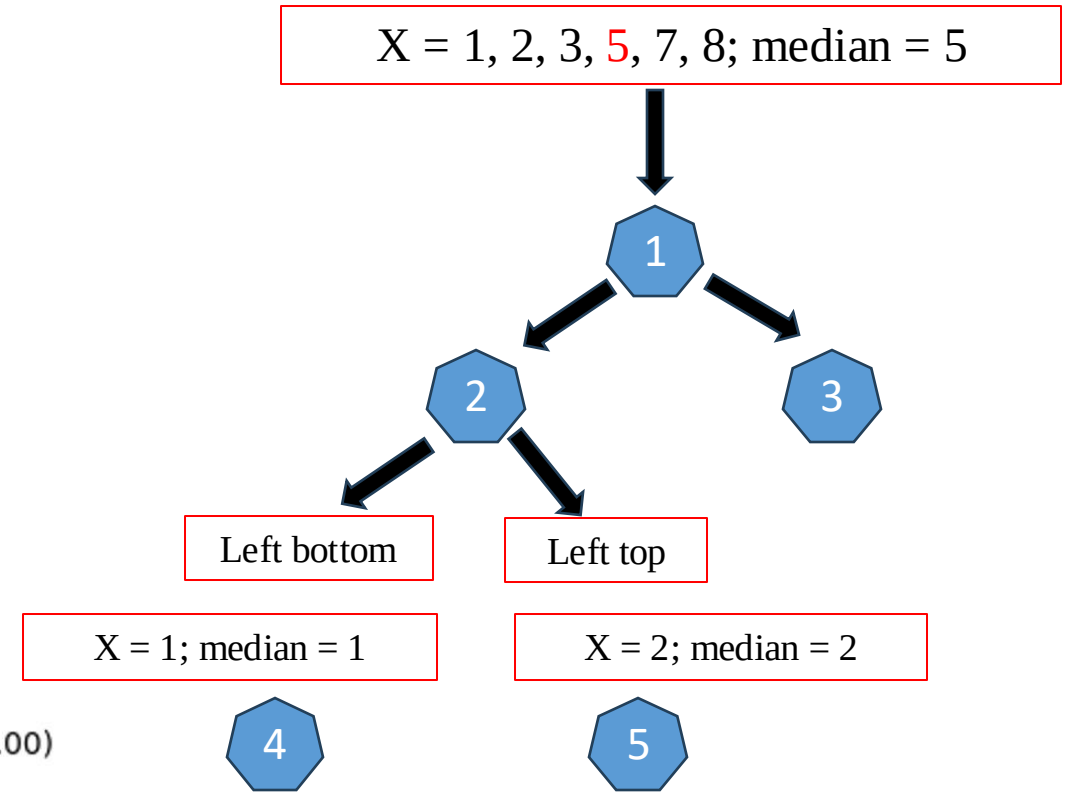
2

3

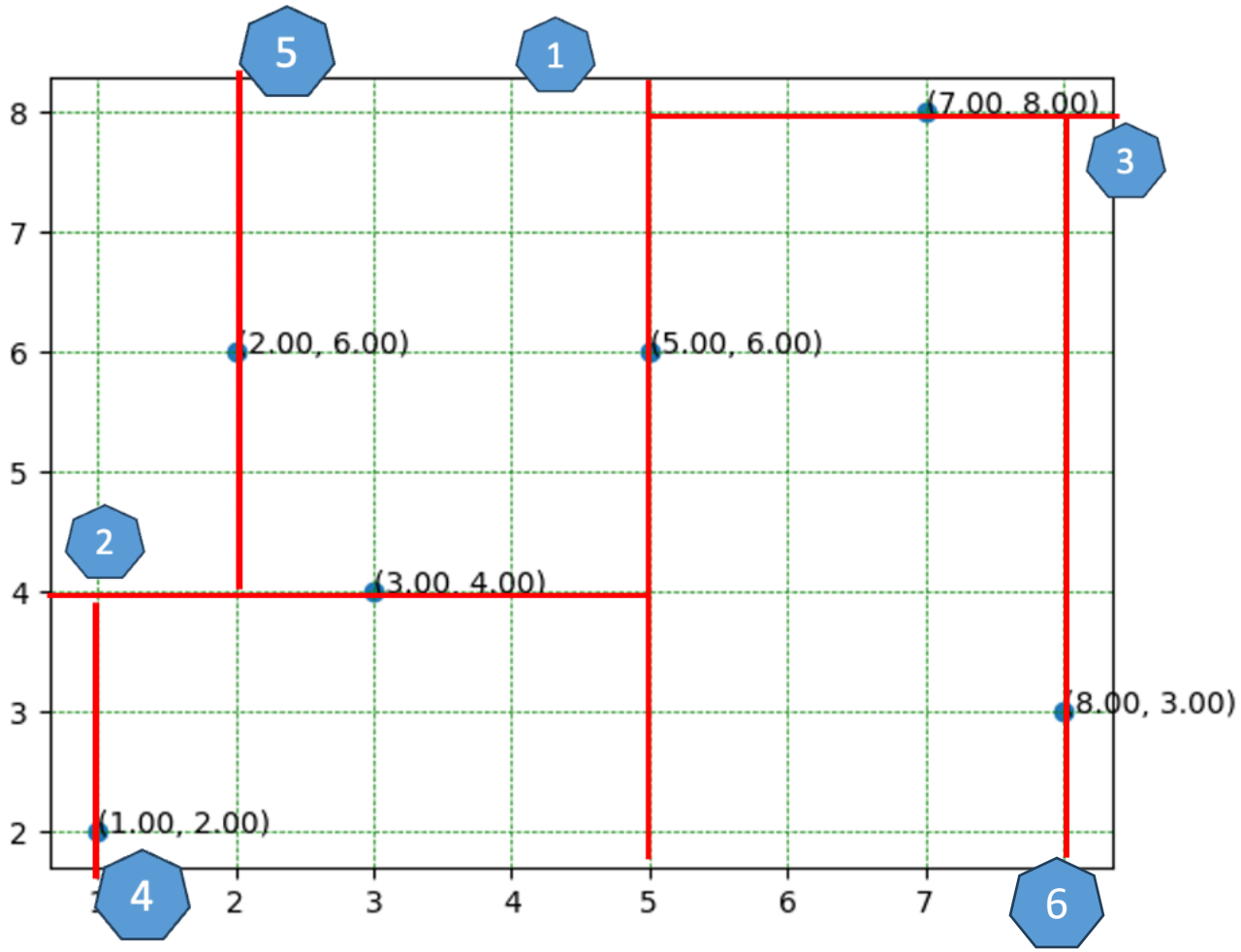
AI VIET NAM
@aivietnam.edu.vn

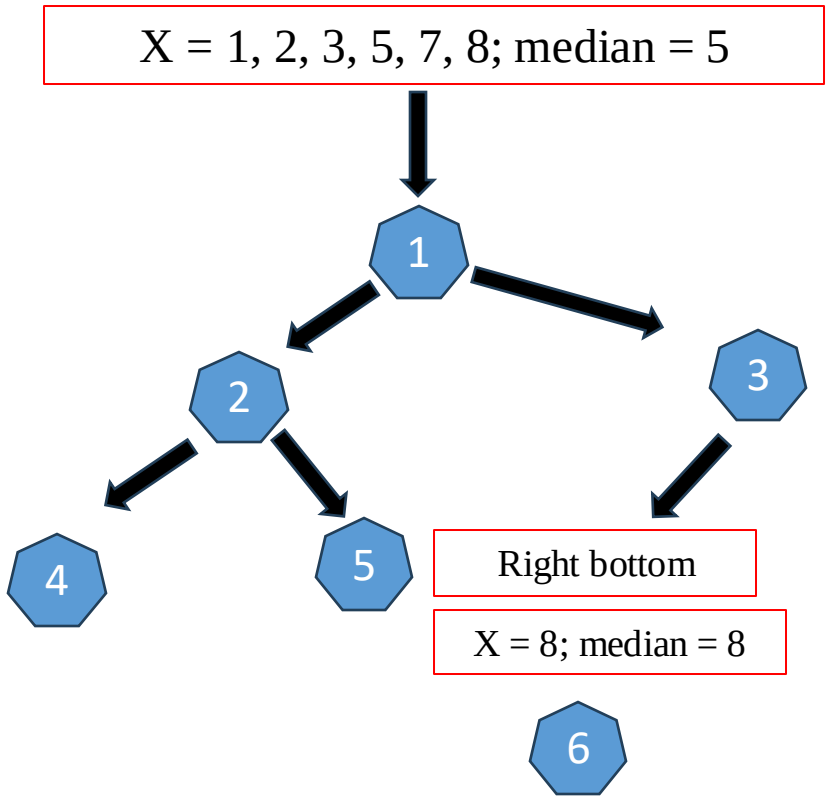
K-D Tree Implementation Solution





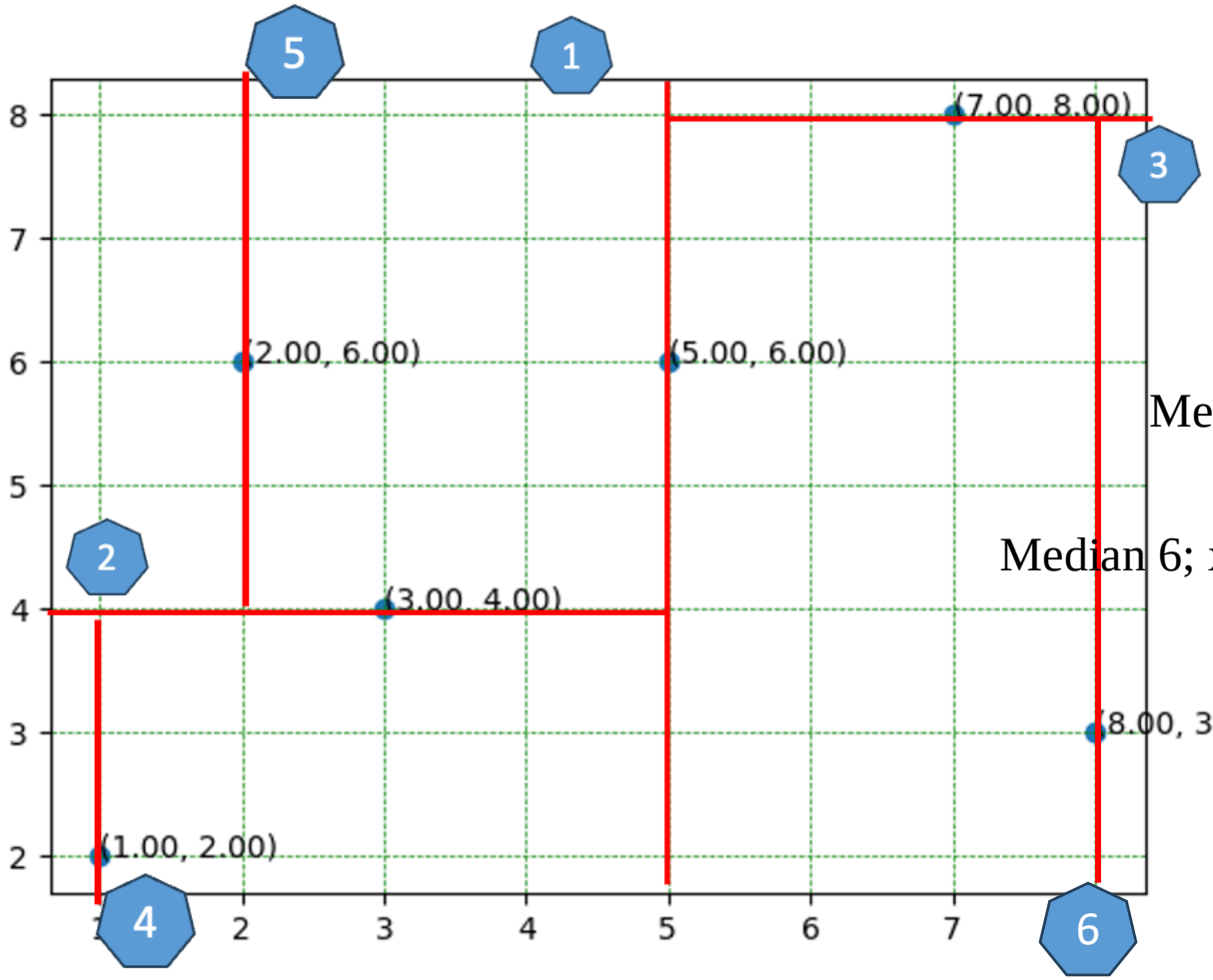
K-D Tree Implementation Solution

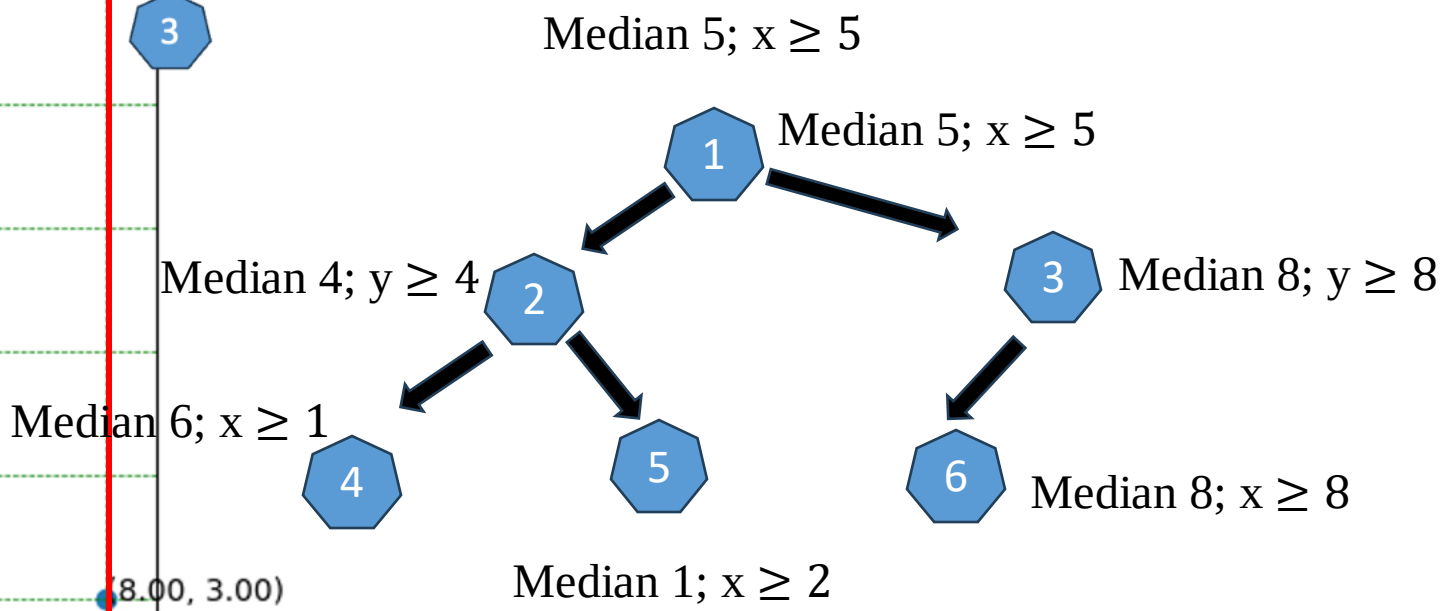




AI VIET NAM
@aivietnam.edu.vn

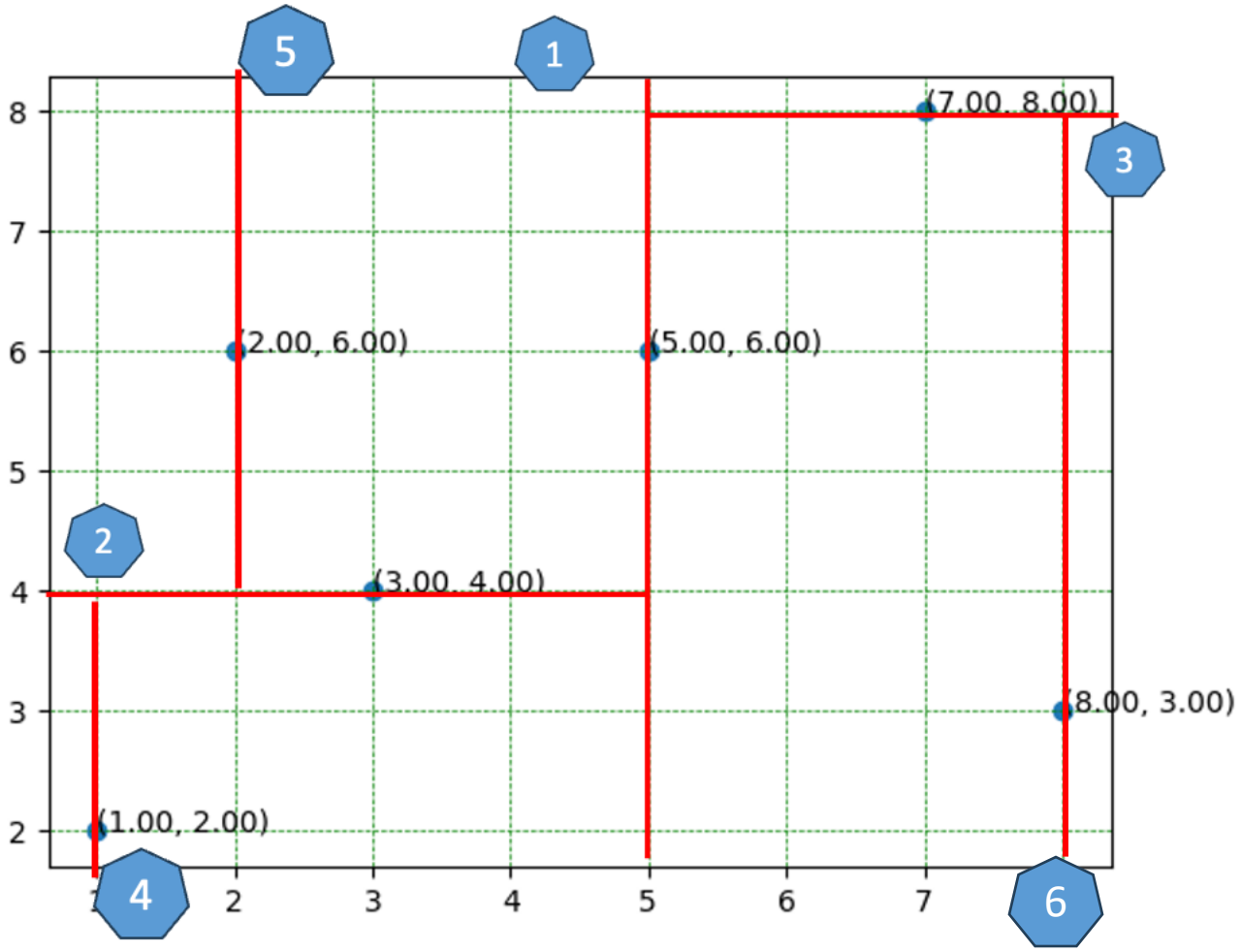
K-D Tree Implementation Solution

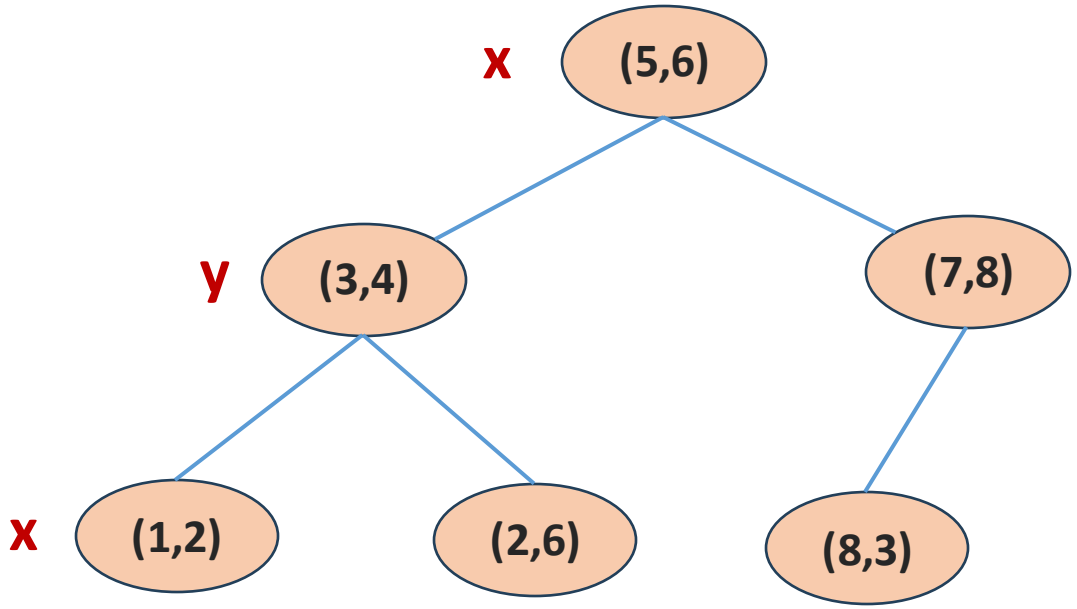




★ **New datapoint**

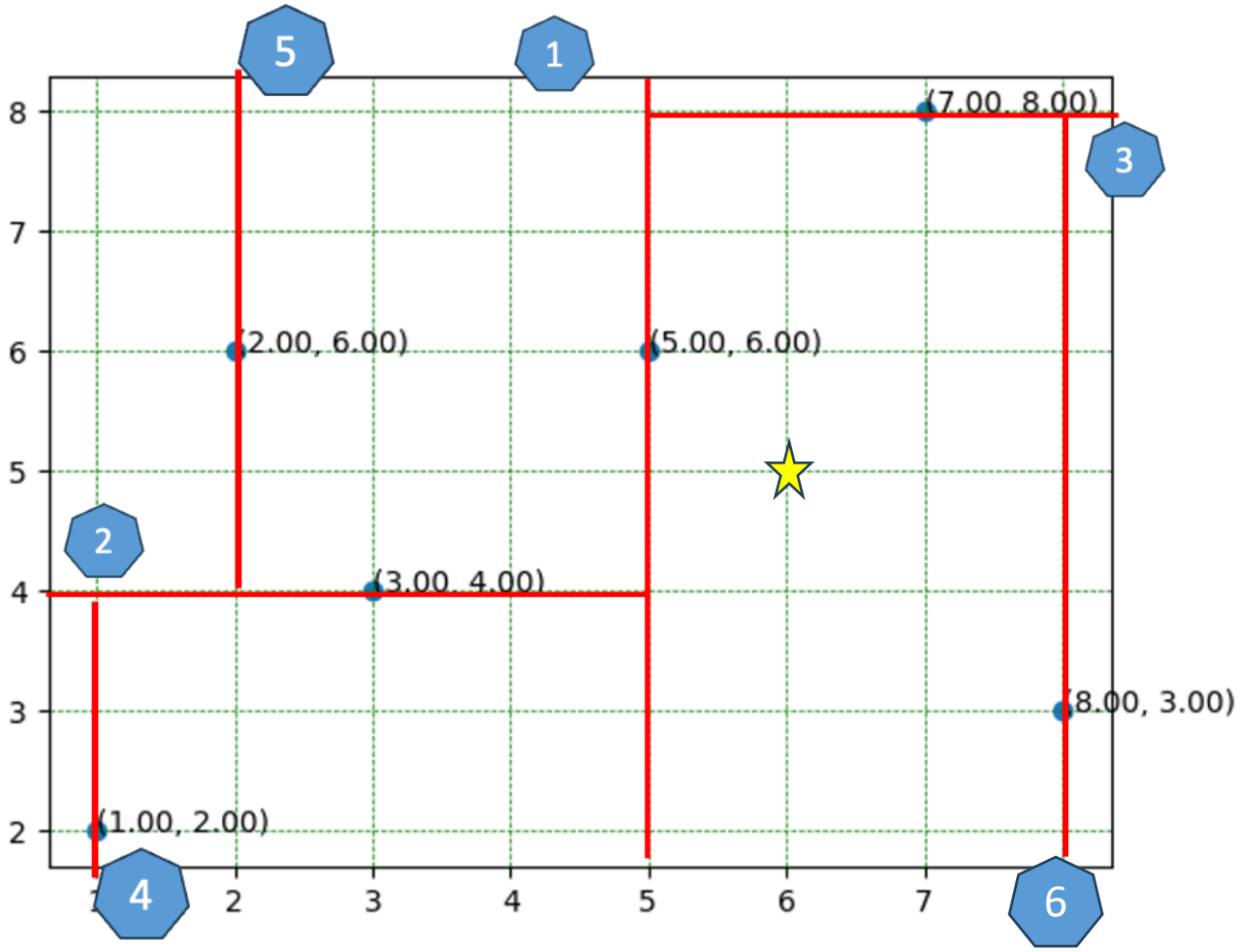
K-D Tree Implementation Solution

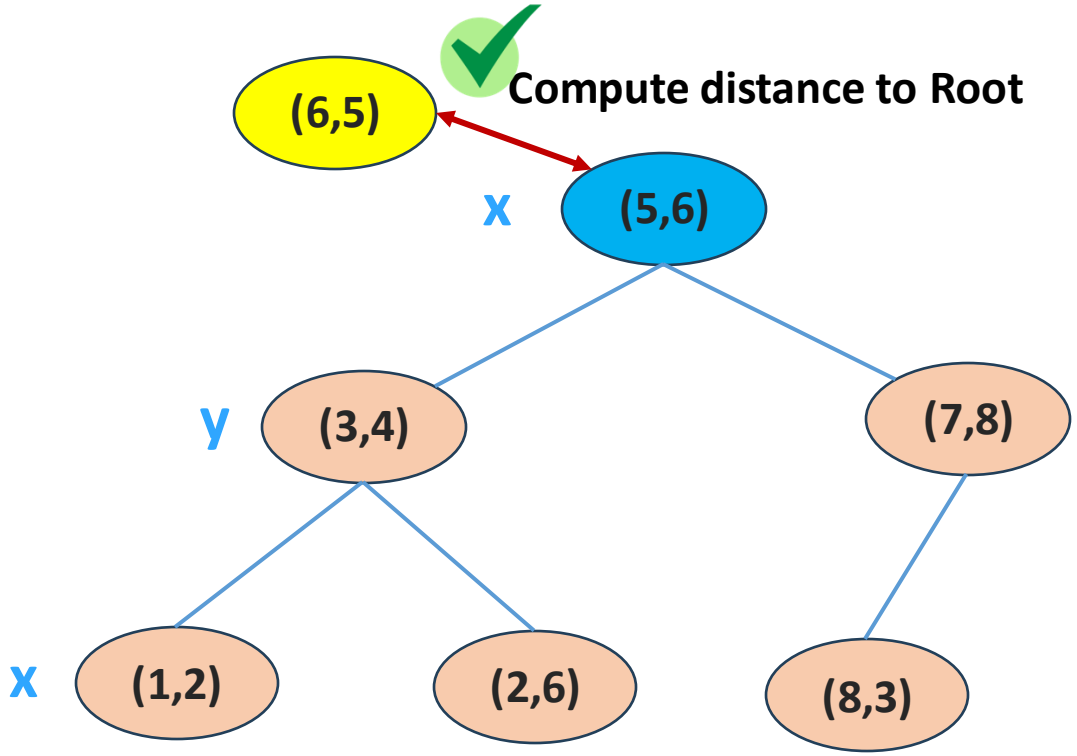




AI VIET NAM
@aivietnam.edu.vn

K-D Tree Implementation Solution





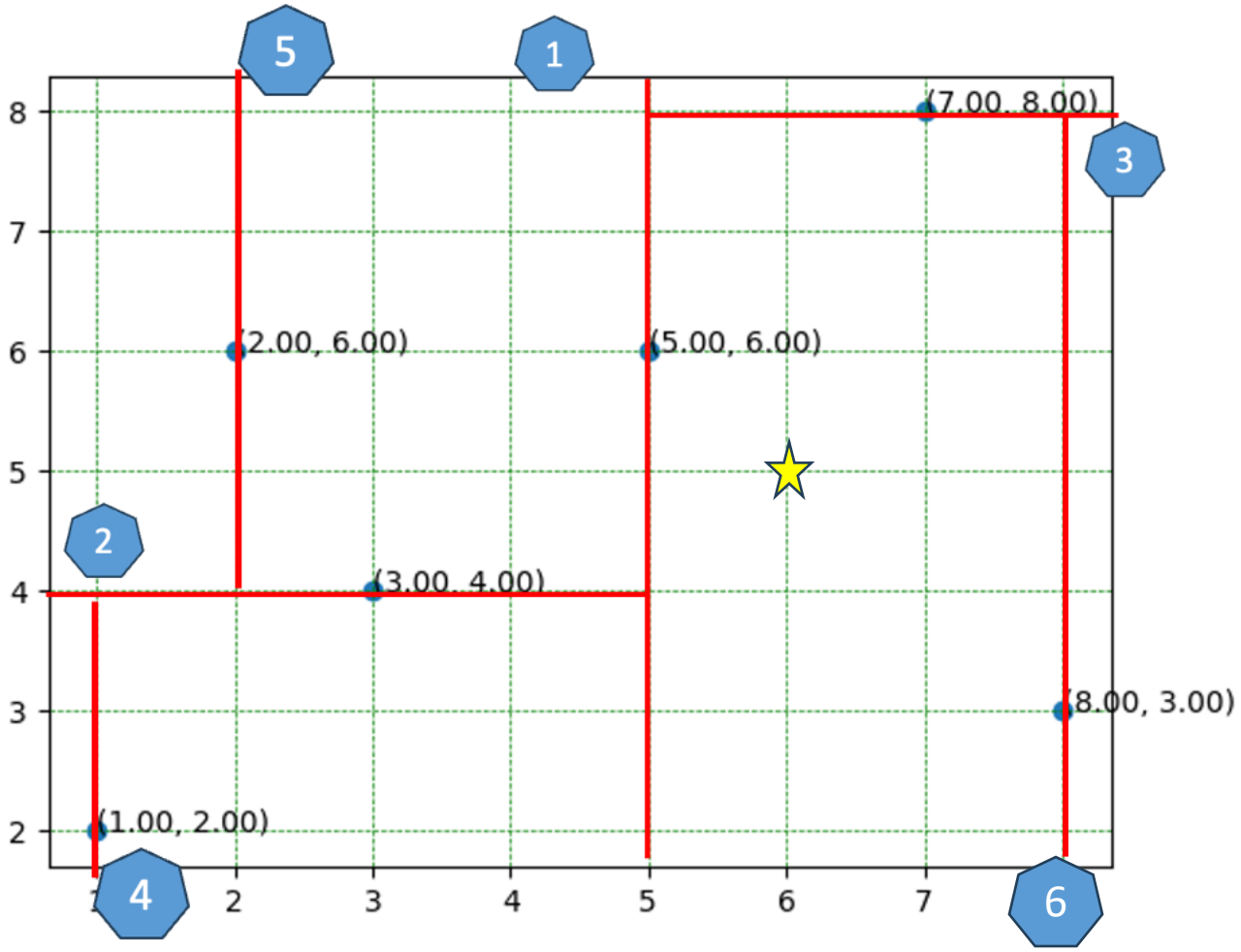
New datapoint

Best distance = 1.41

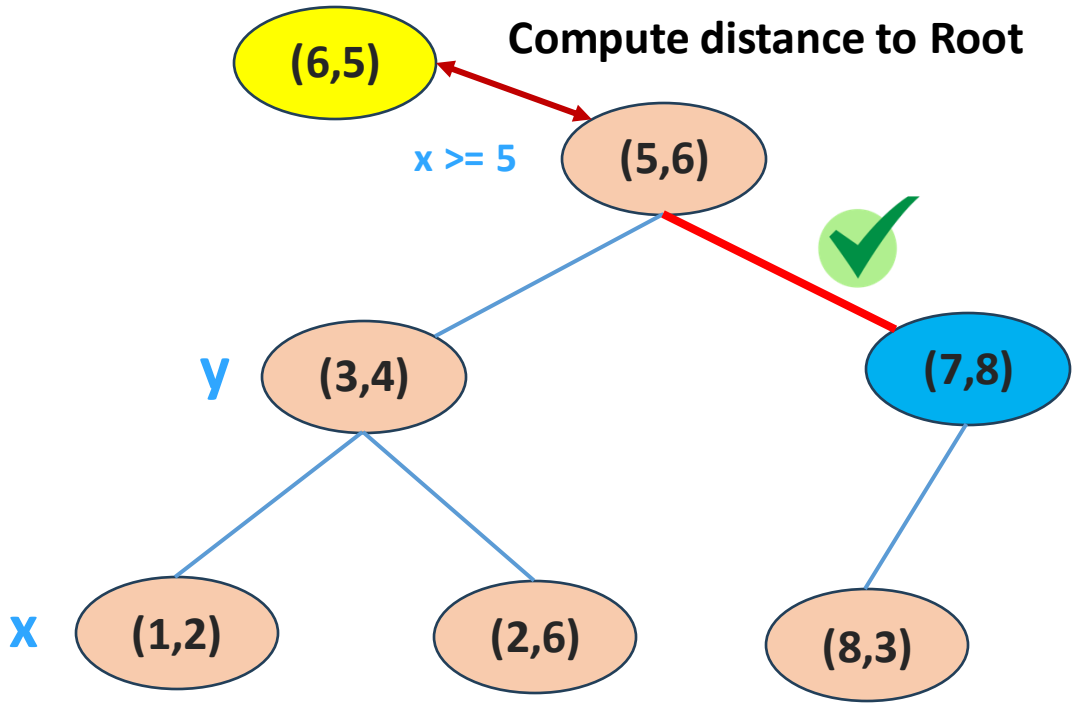
Best label = (5,6)

AI VIET NAM
@aivietnam.edu.vn

K-D Tree Implementation Solution



New datapoint



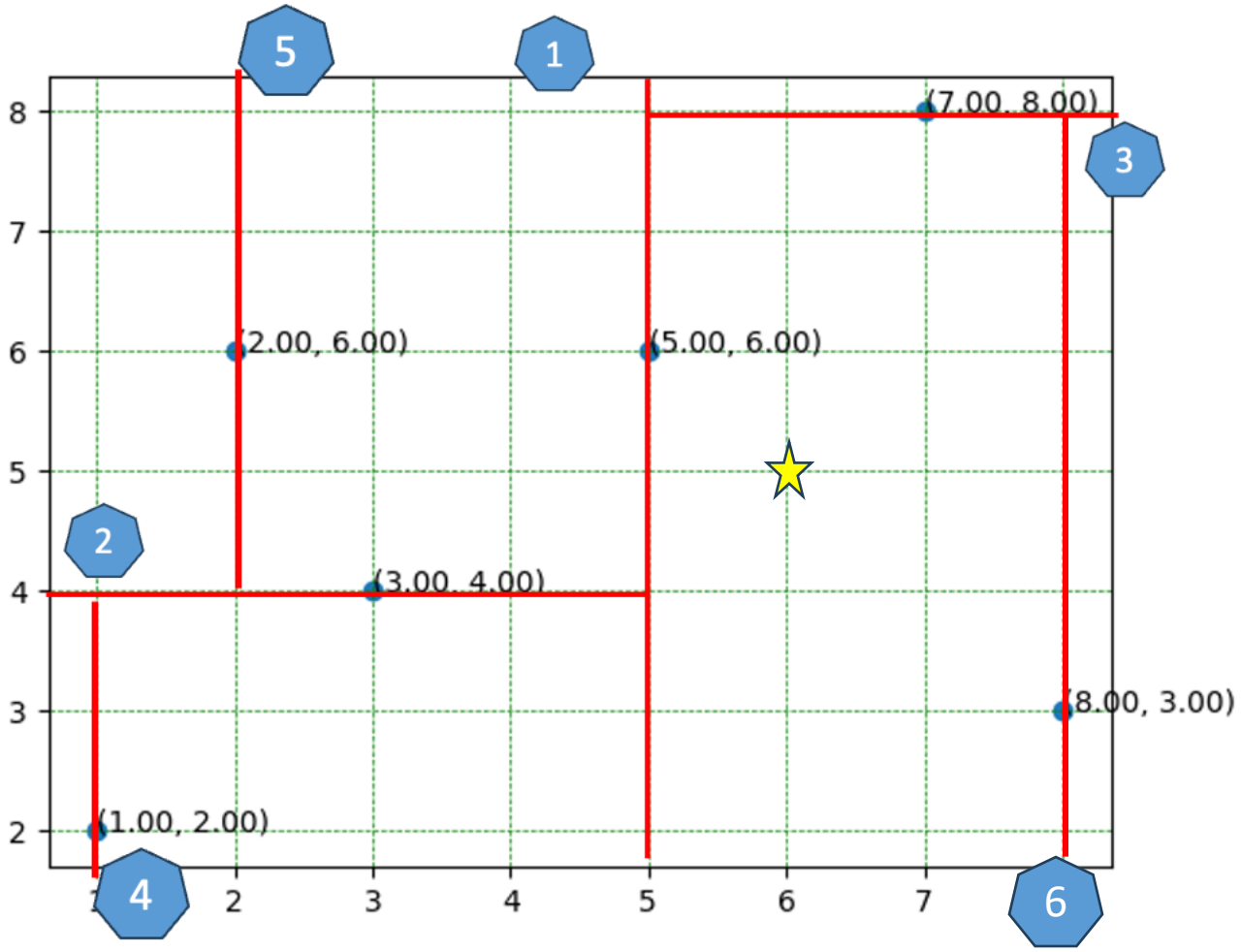
Best distance = 1.41
Best label = (5,6)

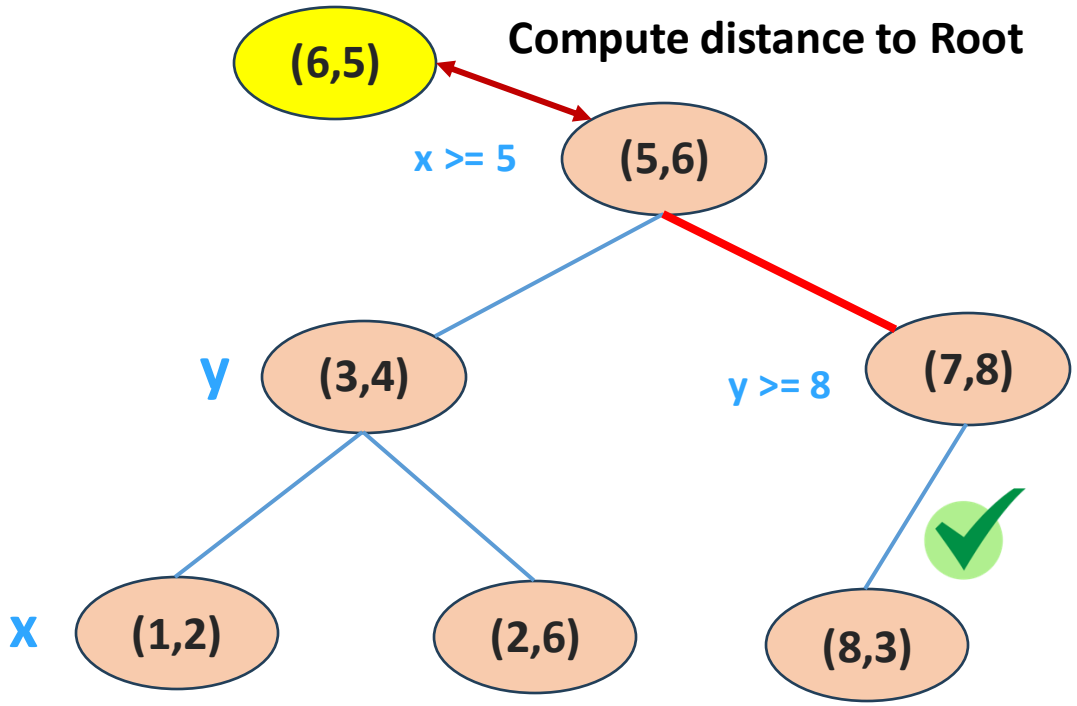
Curent distance = 3.16
Curent label = (5,6)

Not update the best distance

AI VIET NAM
@aivietnam.edu.vn

K-D Tree Implementation Solution





(6,5)

The closest points is (5,6)

Best distance = 1.41

Best label = (5,6)

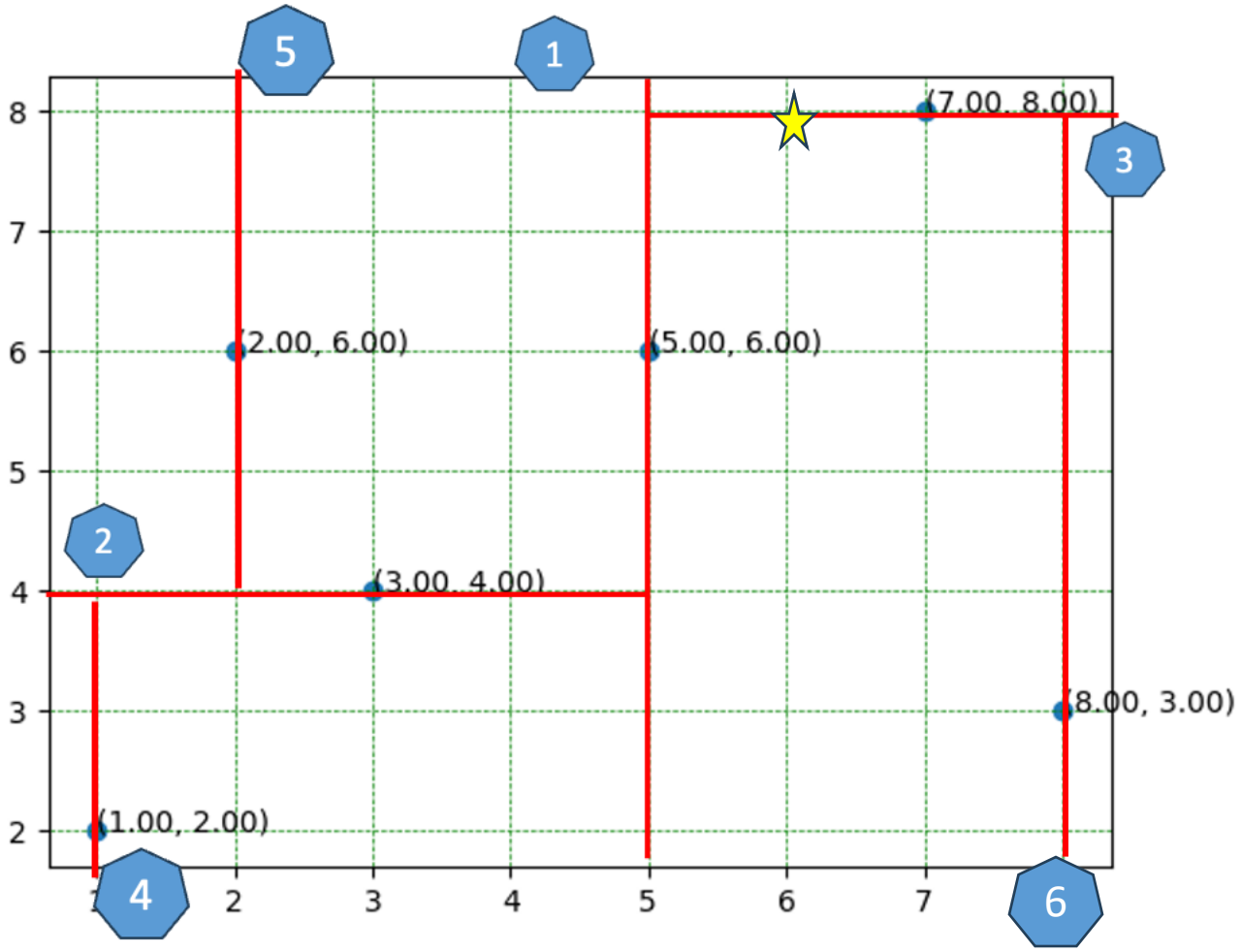
Curent distance = 2.82

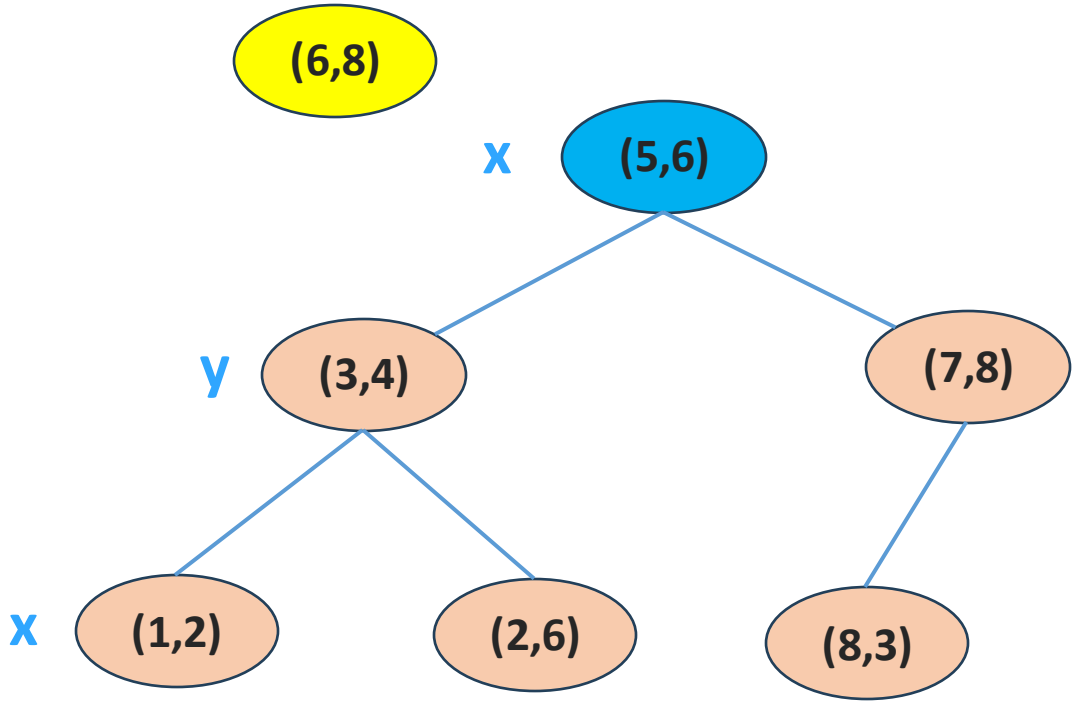
Curent label = (8,3)

Not update the best distance

AI VIET NAM
@aivietnam.edu.vn

K-D Tree Implementation Solution





★ **New datapoint**

What is the nearest neighbor?

Best distance = 1.41

st label = (5,6)

Outline

- Machine Learning: Review
- KNN Motivation
- KNN for Classification
- How to Select K in KNN
- KNN for Regression
- KNN with K-D Tree
- Data science application: Case study

Data Science: Case Study

A real-life example

Most of data science is oriented towards the **Train, Test, Predict paradigm**. Who doesn't want to guess the future!

But there are some cases where other implementations are needed like unsupervised classification or discovering patterns in existing data. In other words, how to take advantage of the **data which is already owned**.

Data Science: Case Study

Here's the story:

A client of ours needed a way to find similar items (neighbours) for a given entity, according to a fixed number of parameters. Practically, the dataset is composed of votes from Human Resources Professionals who could attribute up to 5 skills to an arbitrary amount of World universities. It means that Edouard from HR could vote for MIT as a good institution for Digitalisation, Oxford for Internationality and La Sorbonne for Soft Skills.

AI VIET NAM
@aivietnam.edu.vn

KDTree to Detect Similarities

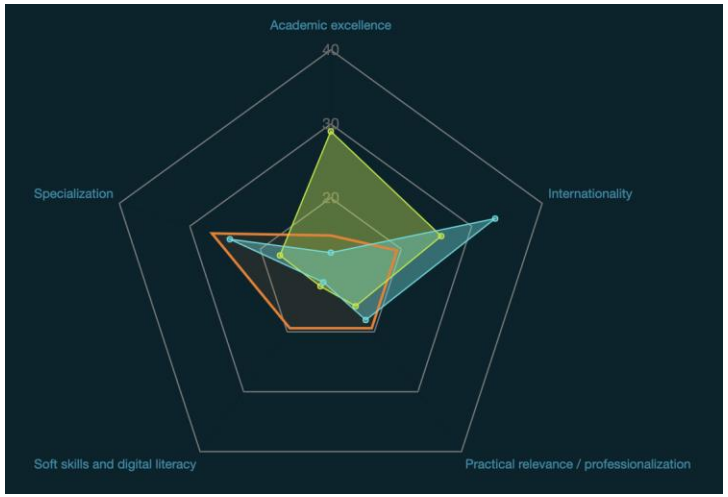
It seemed interesting to search for universities that would have been voted the same way, maybe to compare their actions and study what they were doing good and what wrong.

index	SUID	univid	response
0	538.0	5c9e7a6834cda4f3677e662b	Internationality
1	2285.0	5c9e373c34cda4f3677e14ef	Practical relevance / professionalization
2	2285.0	5c9e7d8334cda4f3677e6b4f	Internationality
3	2285.0	5e84b8a01ead1a77be3bd3a8	Soft skills and digital literacy
4	2285.0	5c9e471734cda4f3677e2ed7	Internationality

Each row corresponds to a vote where:

- **SUID** is the Unique ID of the voter
- **univid**: the unique ID of the institution
- **response**: the voted skill

the user #538 voted "Internationality" as an important skill for University #5c9e7a6834cda4f3677e662b.



Voted skills for three universities

KDTree to Detect Similarities

Our next step consists in grouping by institution and skill (response)

```
skills = df.groupby(["univid", "response"])["response"].count().reset_index(name='value')
skills.head()
```

	univid	response	value
0	5c9e345f34cda4f3677e1047	Academic excellence	10
1	5c9e345f34cda4f3677e1047	Internationality	24
2	5c9e345f34cda4f3677e1047	Practical relevance / professionalization	10
3	5c9e345f34cda4f3677e1047	Soft skills and digital literacy	7
4	5c9e345f34cda4f3677e1047	Specialization	18

Even if a **lot more readable**, this format is useless for our goal. It is difficult to distinguish between institutions

AI VIET NAM
@aivietnam.edu.vn

KDTree to Detect Similarities

Our next step consists in grouping by institution and skill (response)

```
[5] univSkills = skills.pivot_table(values="value", columns="response", index=["univid"])
    univSkills.head()
```

univid	response	Academic excellence	Internationality	Practical relevance / professionalization	Soft skills and digital literacy	Specialization
5c9e345f34cda4f3677e1047		10.0	24.0	10.0	7.0	18.0
5c9e349434cda4f3677e109f		5.0	10.0	12.0	9.0	15.0
5c9e34a834cda4f3677e10bd		8.0	10.0	8.0	6.0	8.0
5c9e34ae34cda4f3677e10c7		1.0	17.0	17.0	14.0	17.0
5c9e34d534cda4f3677e1107		10.0	11.0	15.0	10.0	24.0

Our data is almost ready for processing, but we still have an issue?

```
univSkills.sum(axis=1).head()
```

univid	0
5c9e345f34cda4f3677e1047	69.0
5c9e349434cda4f3677e109f	51.0
5c9e34a834cda4f3677e10bd	40.0
5c9e34ae34cda4f3677e10c7	66.0
5c9e34d534cda4f3677e1107	70.0

KDTree to Detect Similarities

Our next step consists in grouping by institution and skill (response)

```
[10] univSkills = univSkills.div(univSkills.sum(axis=1), axis=0)
     univSkills.fillna(0, inplace=True )
     univSkills.head()
```

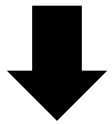
response univid	Academic excellence	Internationality	Practical relevance / professionalization	Soft skills and digital literacy	Specialization
5c9e345f34cda4f3677e1047	0.144928	0.347826	0.144928	0.101449	0.260870
5c9e349434cda4f3677e109f	0.098039	0.196078	0.235294	0.176471	0.294118
5c9e34a834cda4f3677e10bd	0.200000	0.250000	0.200000	0.150000	0.200000
5c9e34ae34cda4f3677e10c7	0.015152	0.257576	0.257576	0.212121	0.257576
5c9e34d534cda4f3677e1107	0.142857	0.157143	0.214286	0.142857	0.342857

AI VIET NAM
@aivietnam.edu.vn

Processing Data To Find Neighbors

Given the voting of a specific university:

Academic excellence	Internationality	Practical relevance / professionalization	Soft skills and digital literacy	Specialization
0.047059	0.258824	0.152941	0.247059	0.294118



Data Mining Information

```
[13] tree = KDTree(univSkills)
     dist, ind = tree.query(univSkills[9:10], k=5)
     univSkills.iloc[ind.tolist()[0]]
```

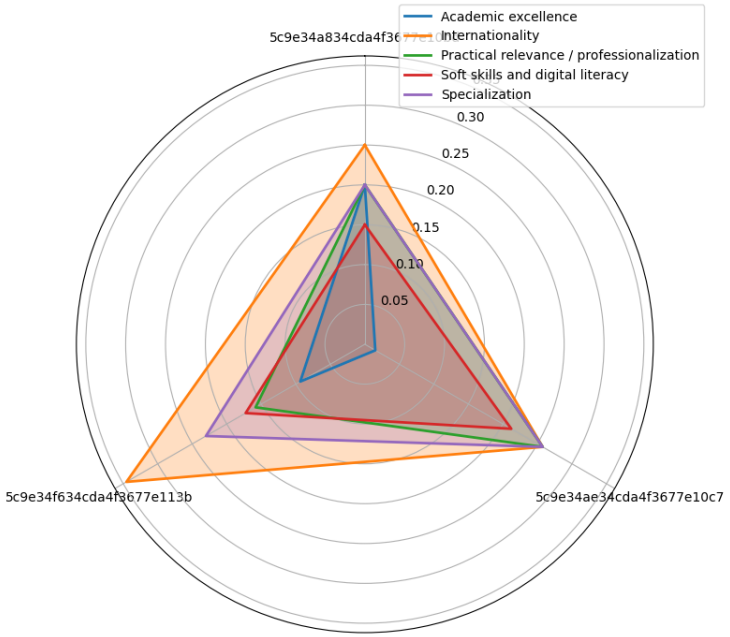
univid	response	Academic excellence	Internationality	Practical relevance / professionalization	Soft skills and digital literacy	Specialization
5c9e35b234cda4f3677e126b		0.047059	0.258824	0.152941	0.247059	0.294118
5c9e5b5e34cda4f3677e42fb		0.078125	0.270833	0.135417	0.229167	0.286458
5c9e3fc034cda4f3677e22e3		0.087500	0.250000	0.150000	0.225000	0.287500
5e8c879823d6684db014ea30		0.074074	0.268519	0.166667	0.212963	0.277778
5c9e7c4034cda4f3677e6937		0.075472	0.264151	0.169811	0.207547	0.283019

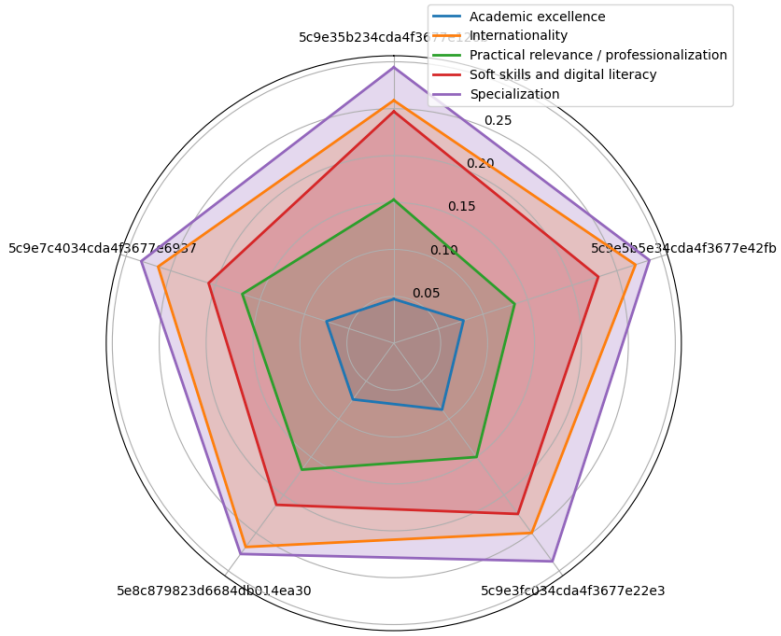
AI VIET NAM
@aivietnam.edu.vn

Processing Data To Find Neighbors

```
[13] tree = KDTree(univSkills)
     dist, ind = tree.query(univSkills[9:10], k=5)
     univSkills.iloc[ind.tolist()[0]]
```

univid	response	Academic excellence	Internationality	Practical relevance / professionalization	Soft skills and digital literacy	Specialization
5c9e35b234cda4f3677e126b		0.047059	0.258824	0.152941	0.247059	0.294118
5c9e5b5e34cda4f3677e42fb		0.078125	0.270833	0.135417	0.229167	0.286458
5c9e3fc034cda4f3677e22e3		0.087500	0.250000	0.150000	0.225000	0.287500
5e8c879823d6684db014ea30		0.074074	0.268519	0.166667	0.212963	0.277778
5c9e7c4034cda4f3677e6937		0.075472	0.264151	0.169811	0.207547	0.283019






Thank
you