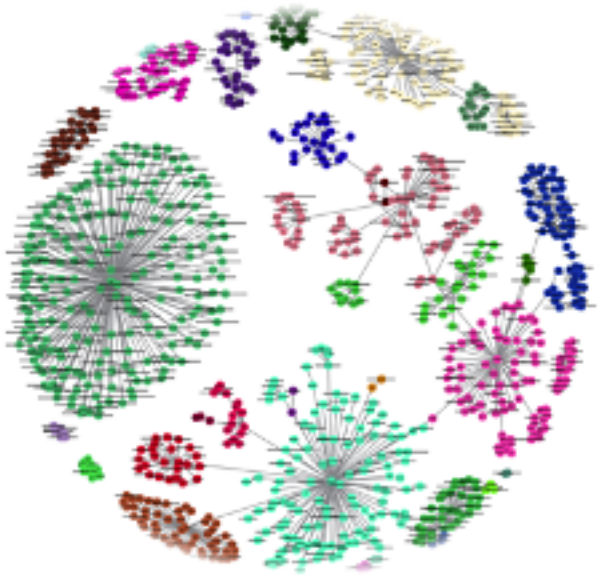
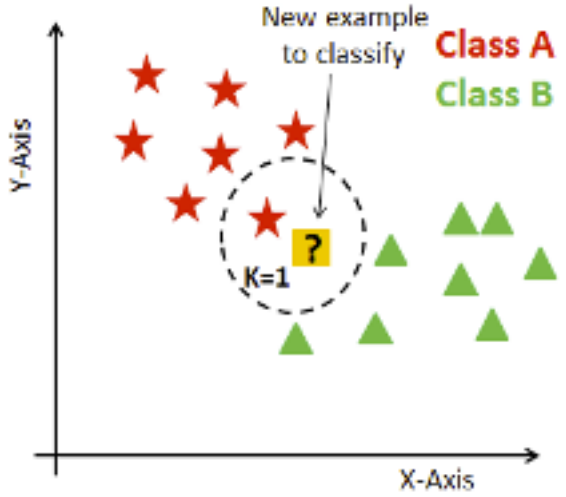
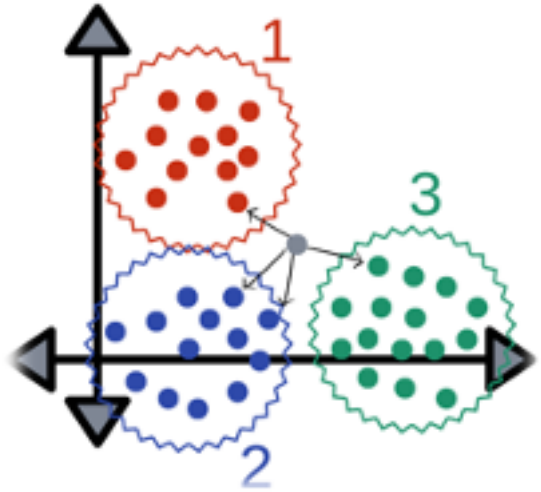
AI VIETNAM
All-in-One Course

K-Nearest Neighbors







Vinh Dinh Nguyen
PhD in Computer Science

Year 2023

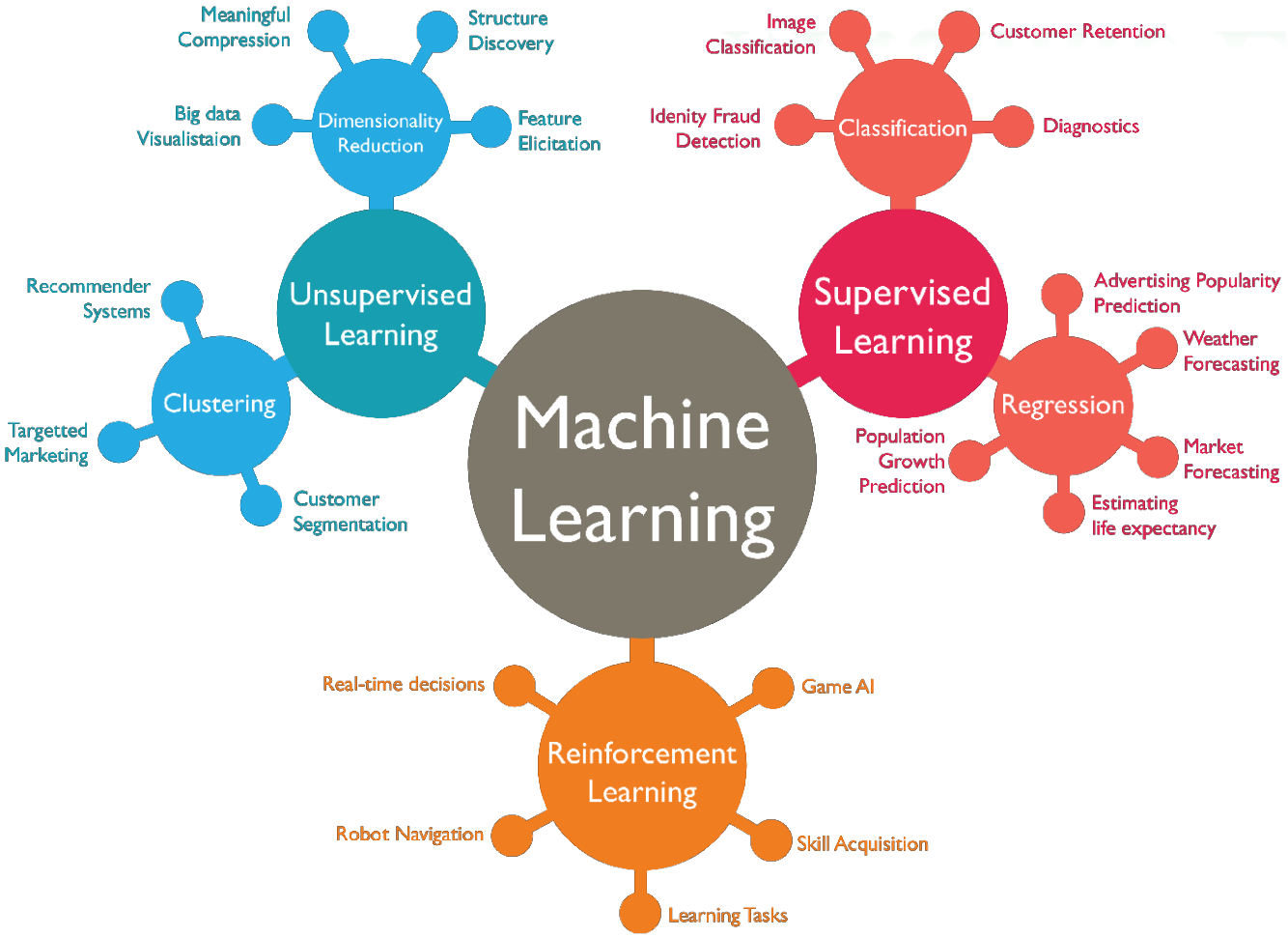
Outline

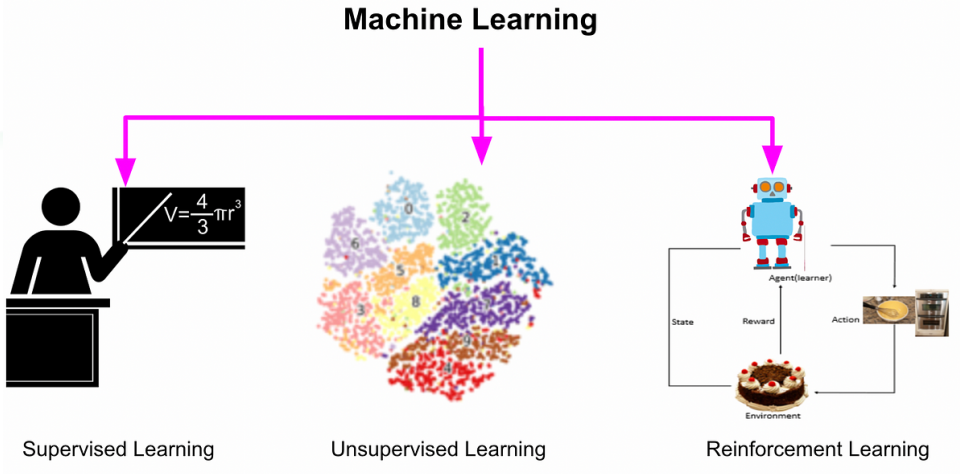
- Overview Machine Learning
- KNN Motivation
- KNN for Classification
- How to Select k in KNN
- KNN for Regression
- KNN with Brute force
- KNN with K-D Tree
- KNN with Ball Tree

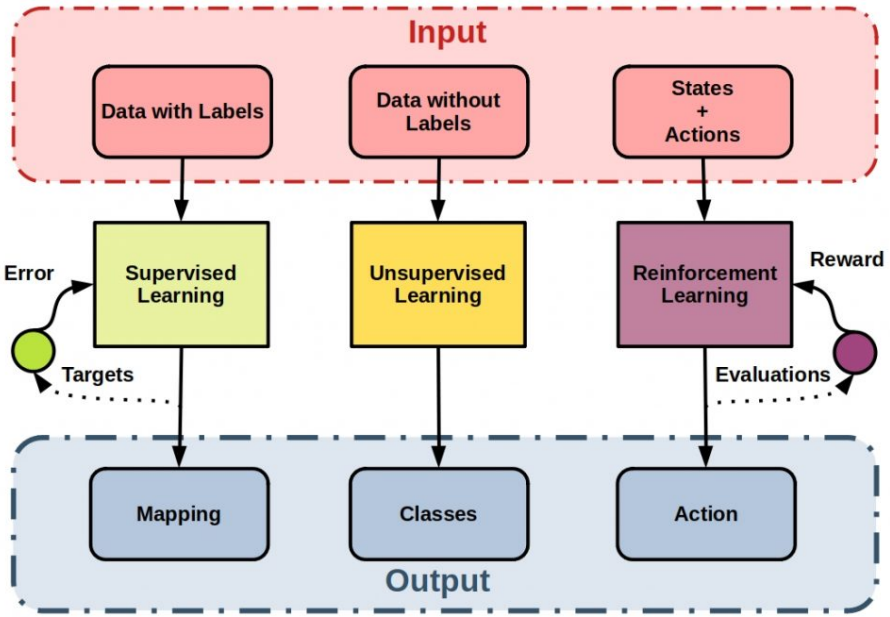
Outline

- Overview Machine Learning
- KNN Motivation
- KNN for Classification
- How to Select k in KNN
- KNN for Regression
- KNN with Brute force
- KNN with K-D Tree
- KNN with Ball Tree

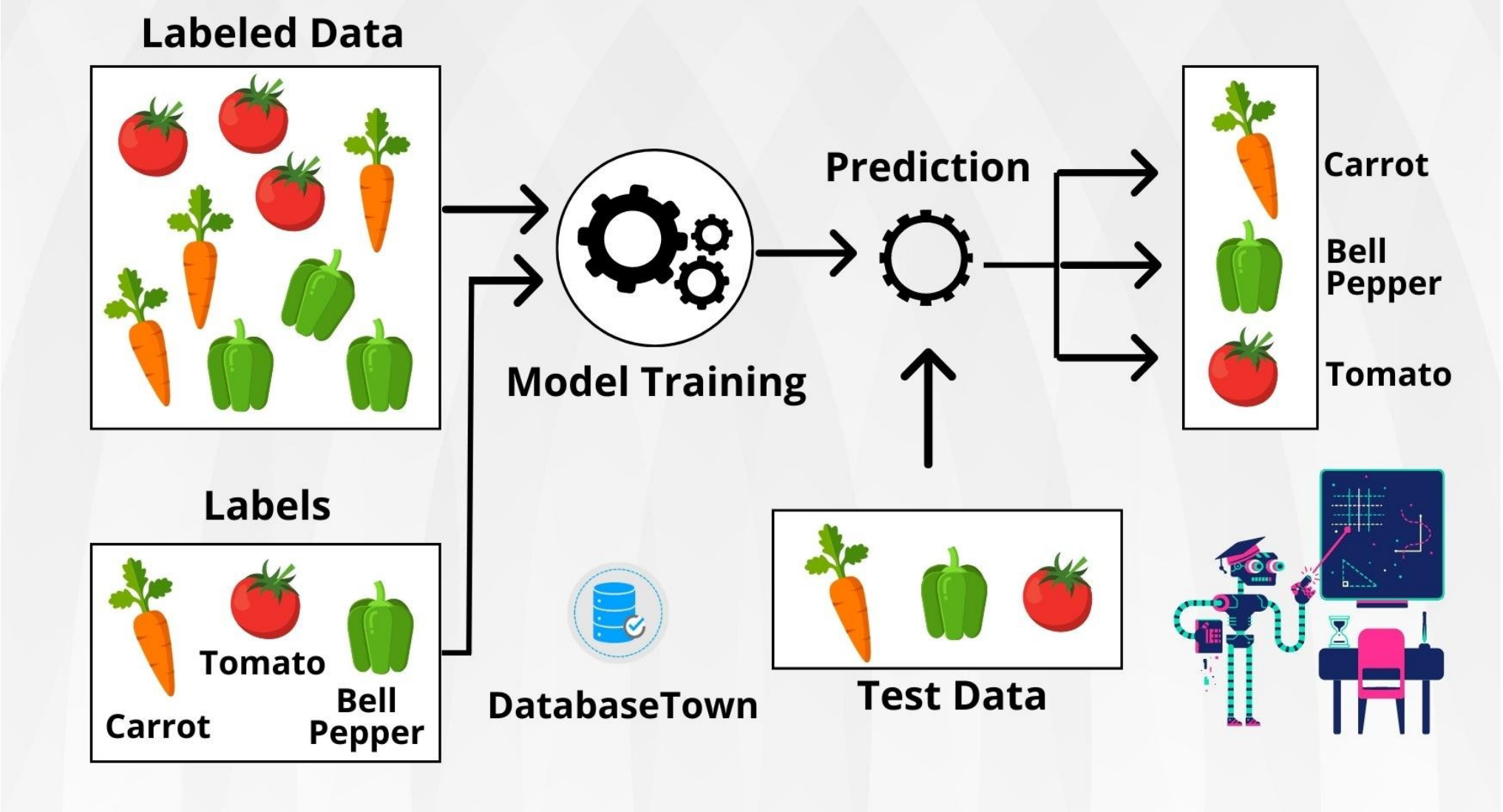
Machine Learning



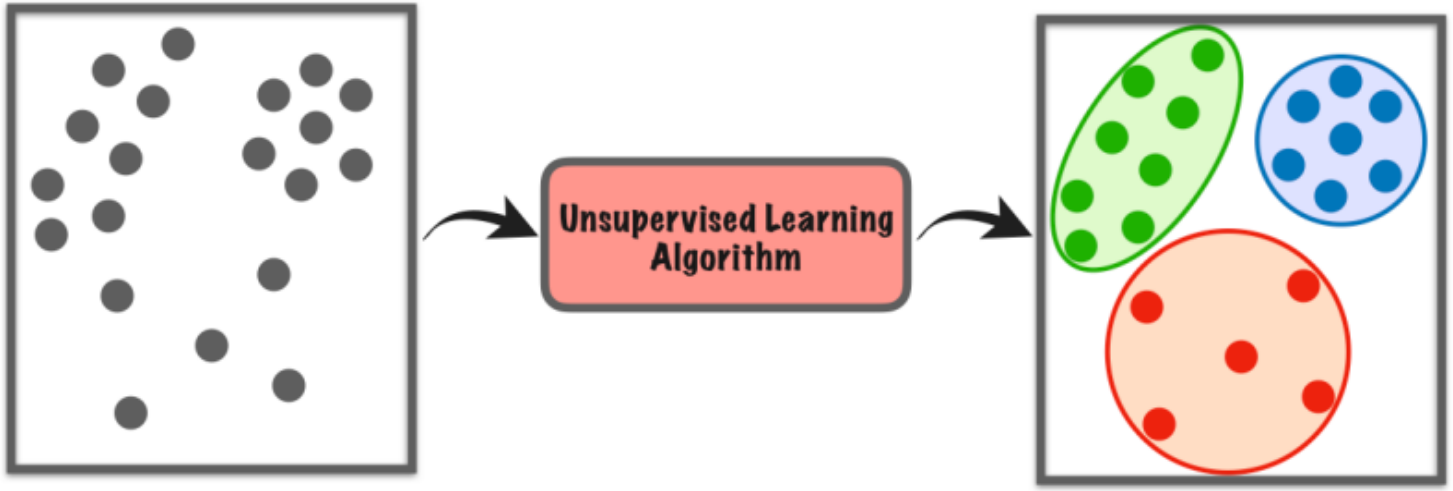


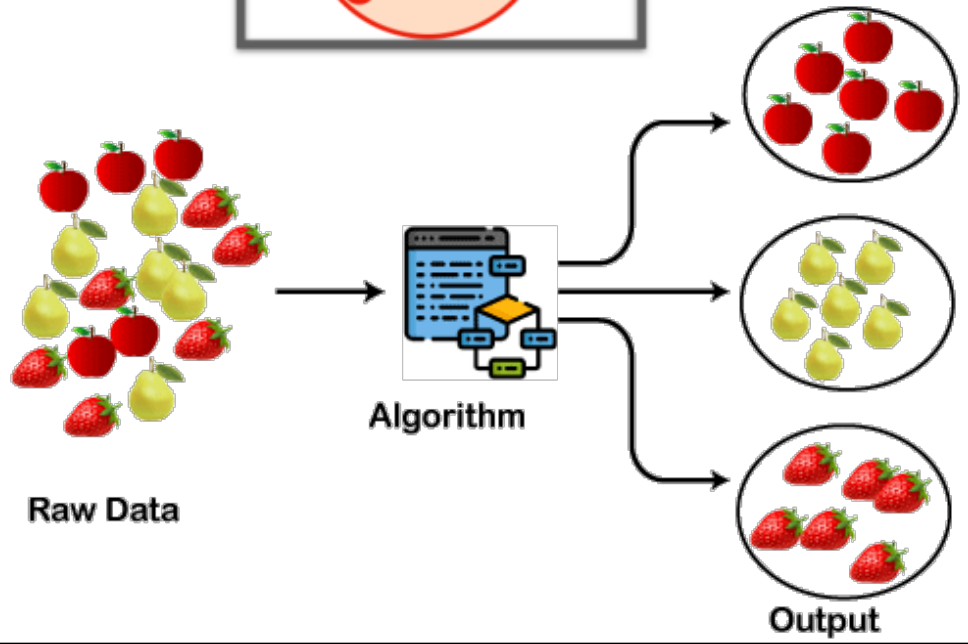


Supervised Learning

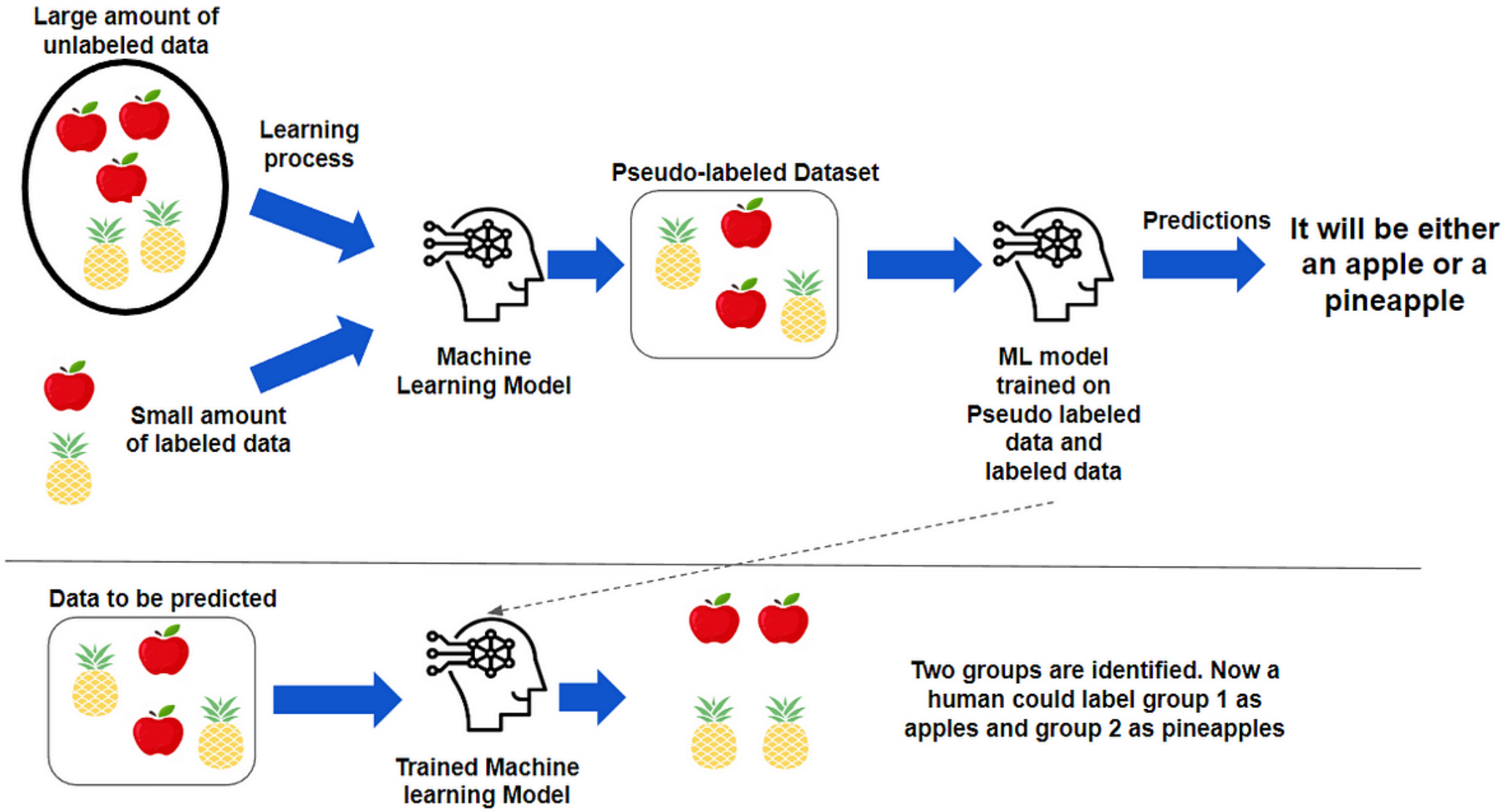


Unsupervised Learning

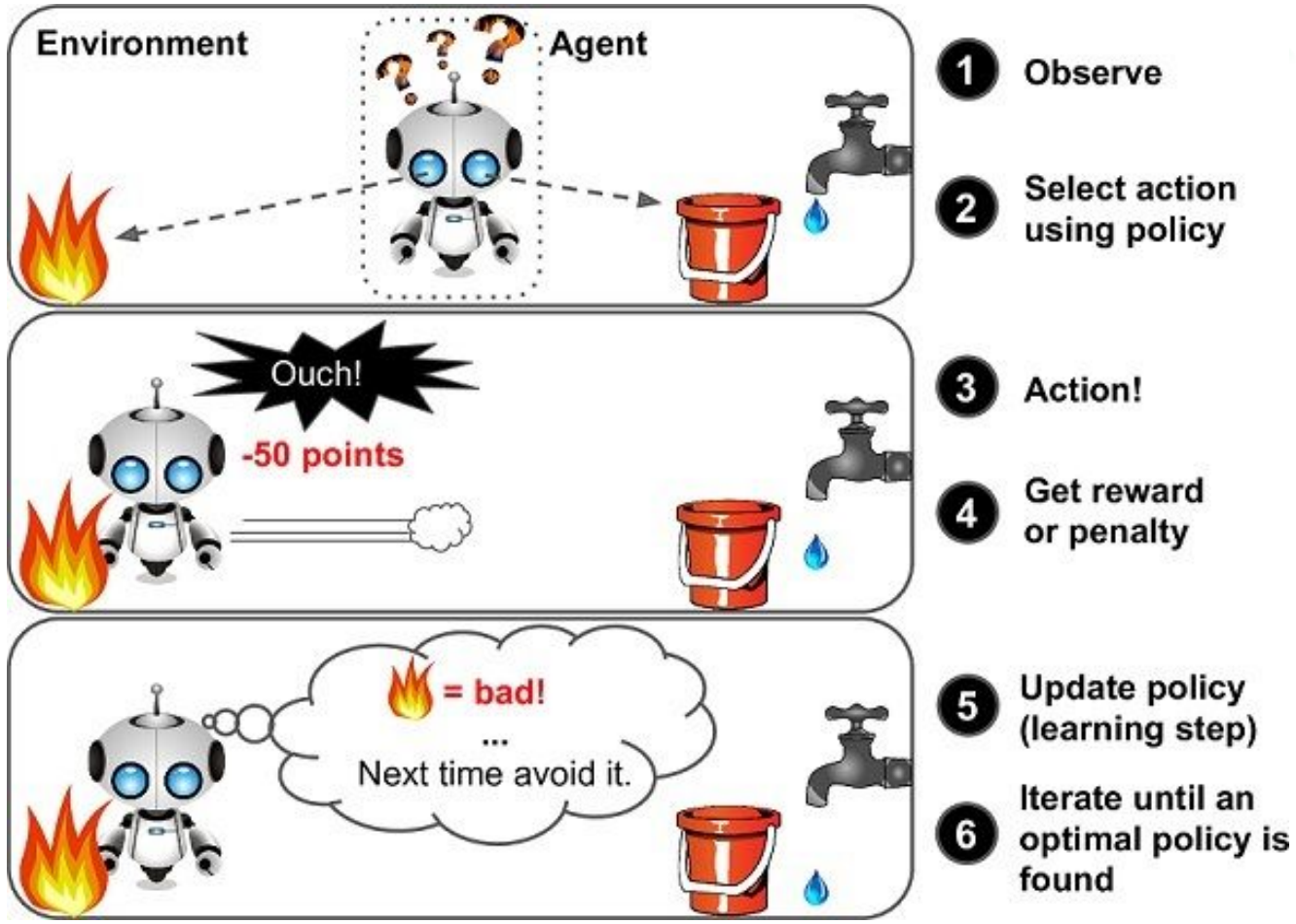


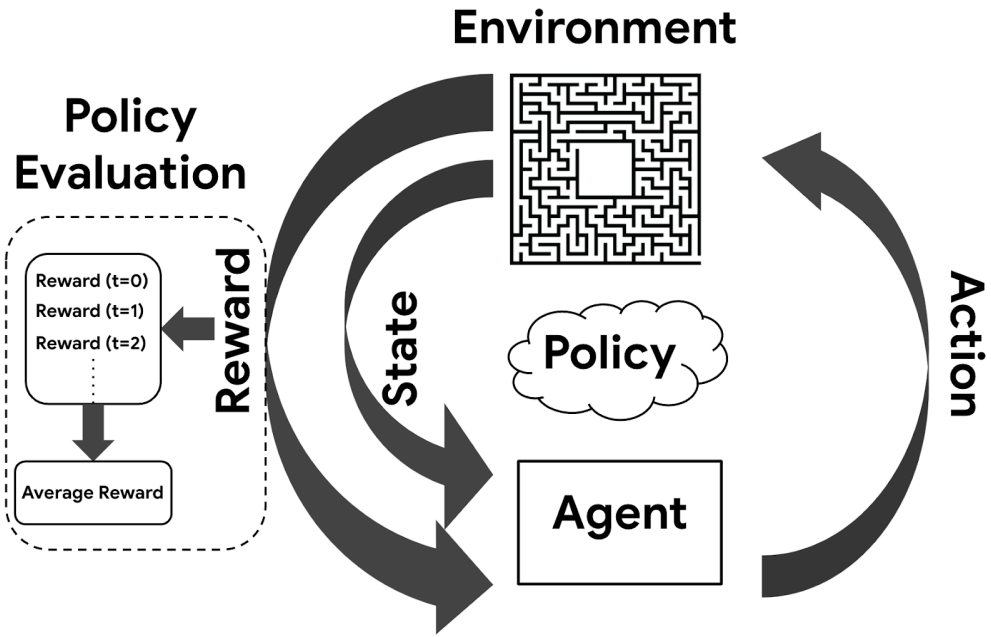


Semi-supervised Learning

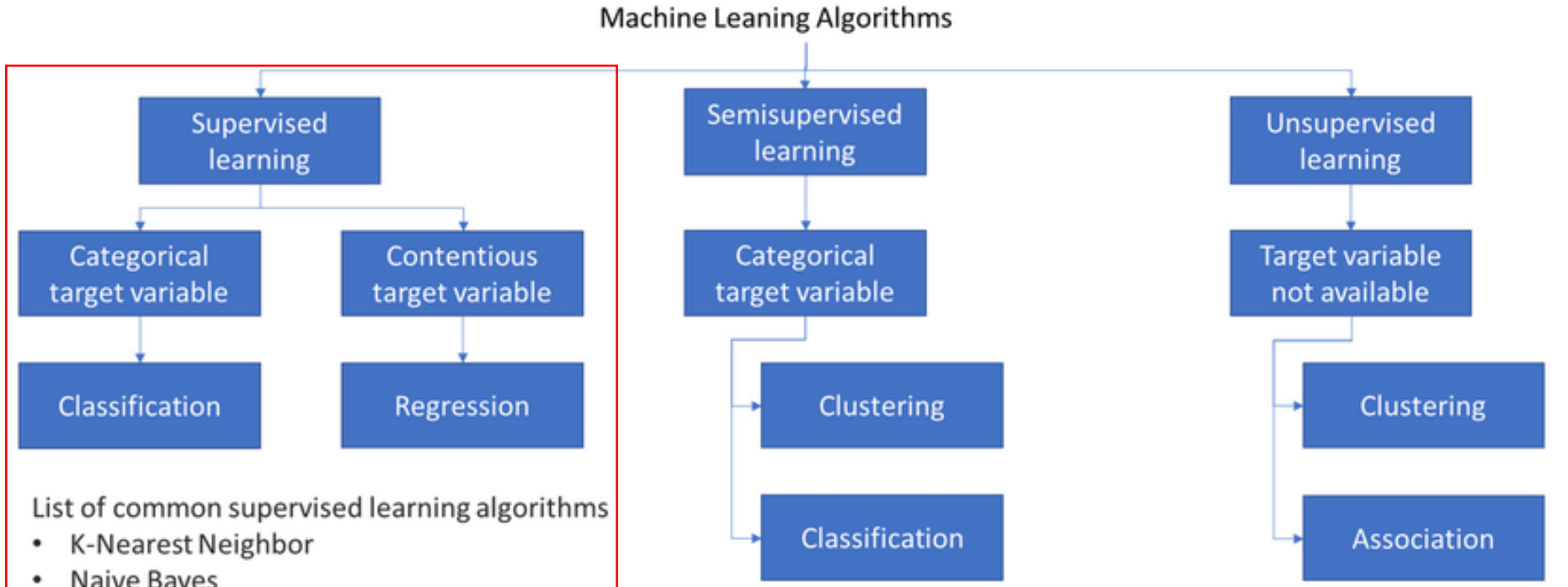


Semi-supervised Learning





Machine Learning

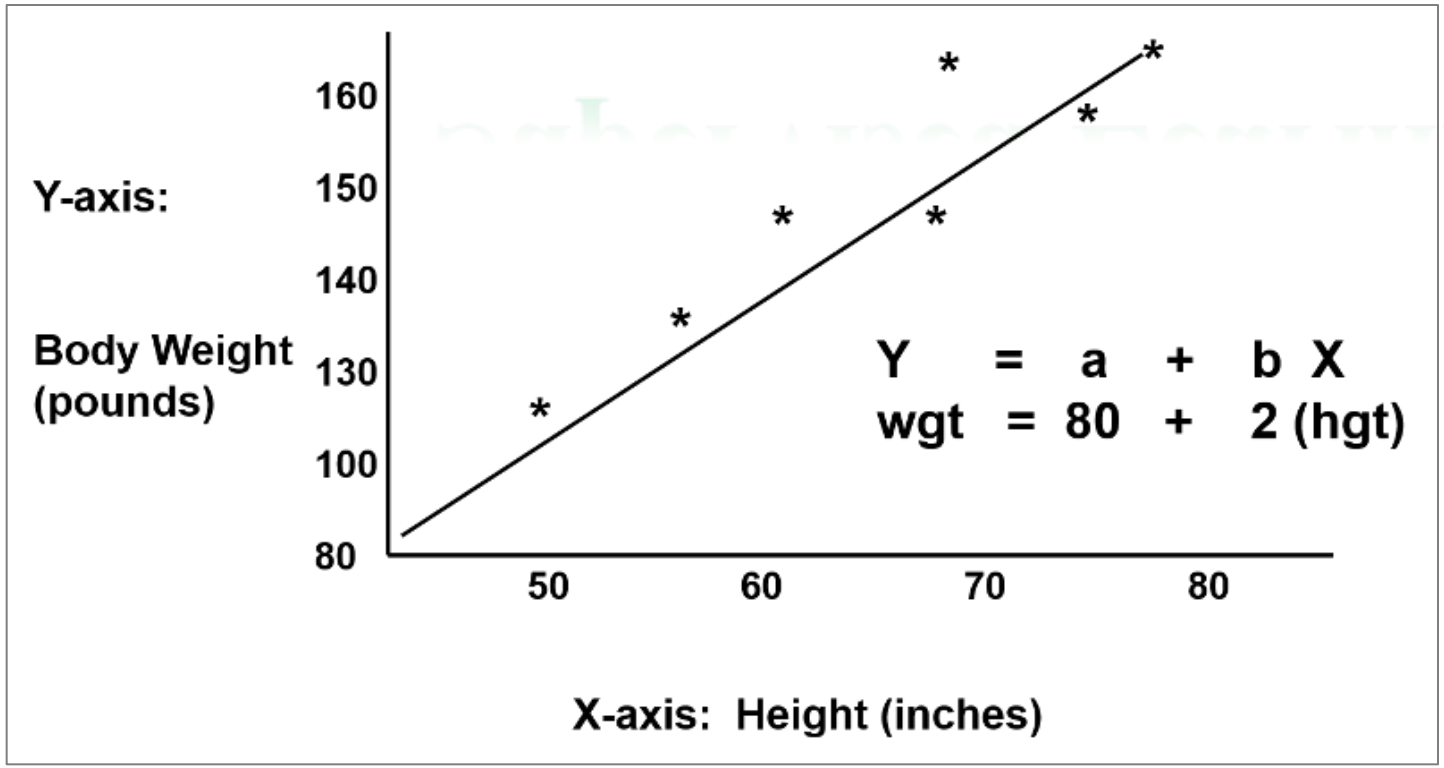


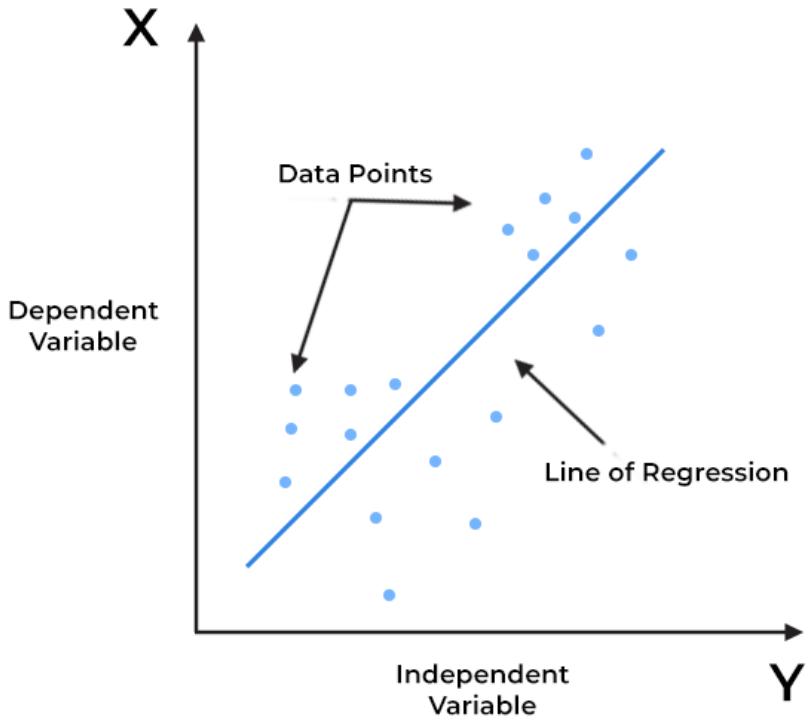
List of common supervised learning algorithms
- K-Nearest Neighbor
- Naive Bayes
- Decision Trees
- Linear Regression
- Support Vector Machines (SVM)
- Neural Networks
- Classification and Regression Trees
- Gradient Boosted Regression Tree
- Perceptron Back-Propagation
- Random Forest

List of common semi supervised learning algorithms:
- Linear Regression
- Logistic Regression

List of common unsupervised learning algorithms:
- k-means clustering and classification
- Association Rules

Supervised Learning Review



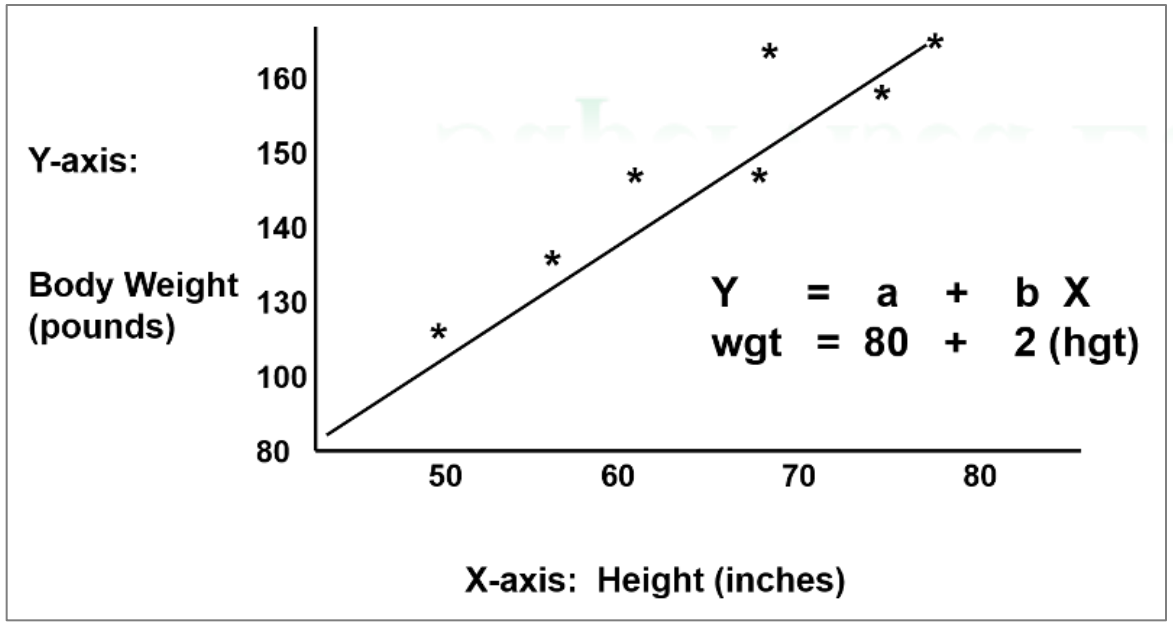


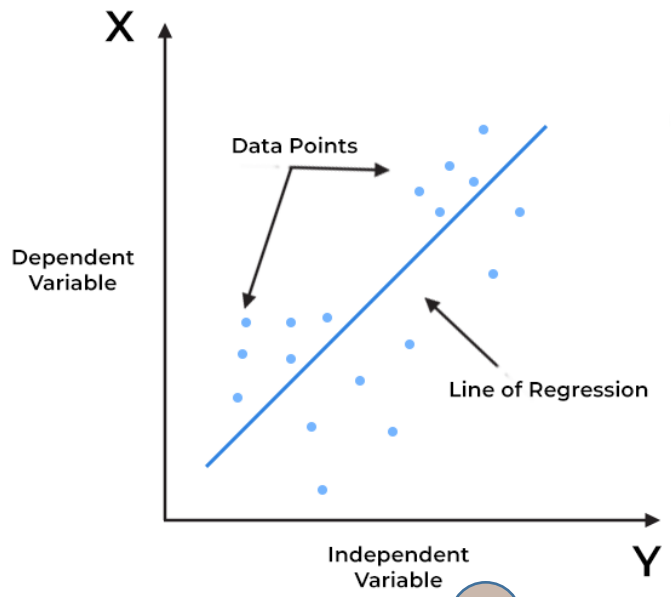
We assume that there is a true underlying relationship between the response Y and the predictors X:

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

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

Supervised Learning Review

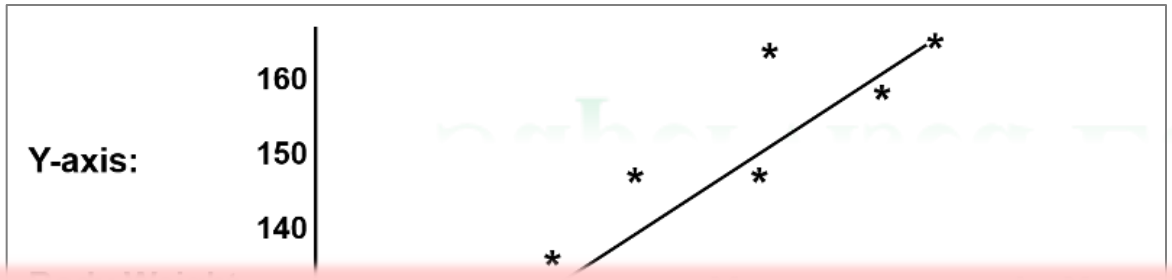


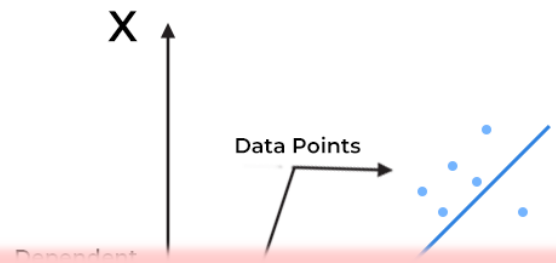


- Identifying the function: The first step is to identify the function such as linear or non-linear function.

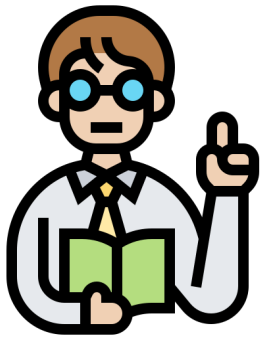
- Identifying the parameters of the function in case this is linear function

Supervised Learning Review





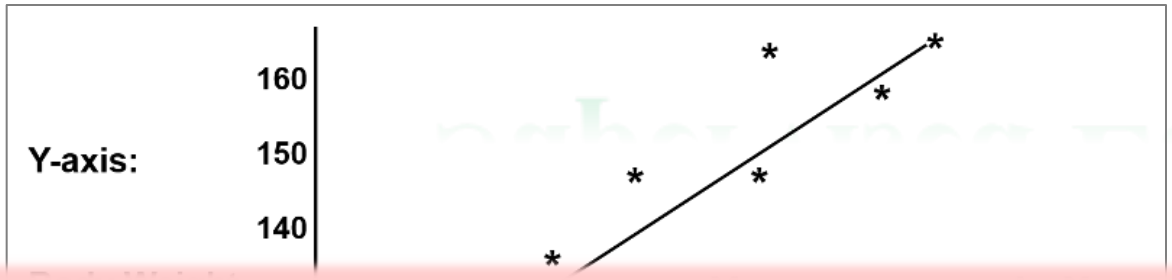
This is a Parametric Approach

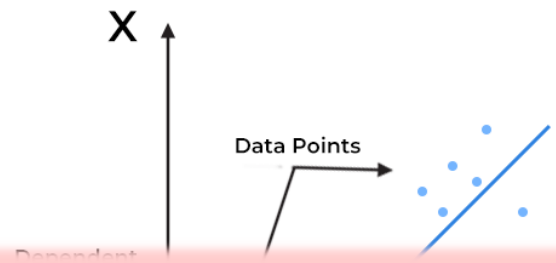


identify the function such as linear or non-linear function.

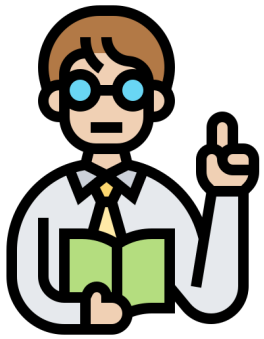
•Identifying the parameters of the function in case this is linear function

Supervised Learning Review



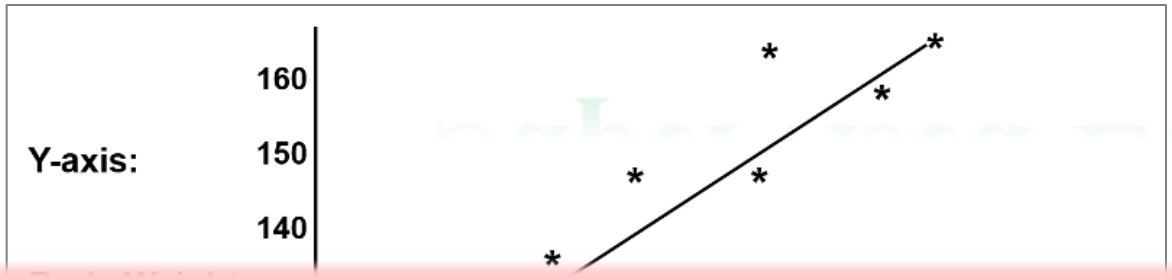


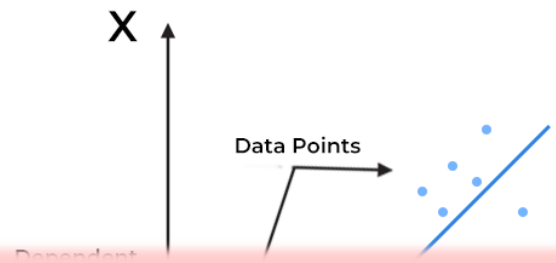
This is a Eager Learning Approach



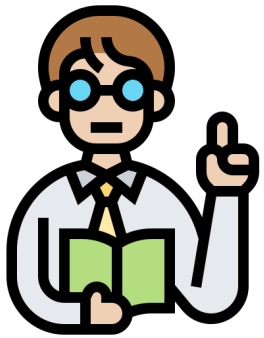
It takes long time learning and less time classifying data

Supervised Learning Review





This is a Model-based Learning Approach



It takes long time learning and less time classifying data

Outline

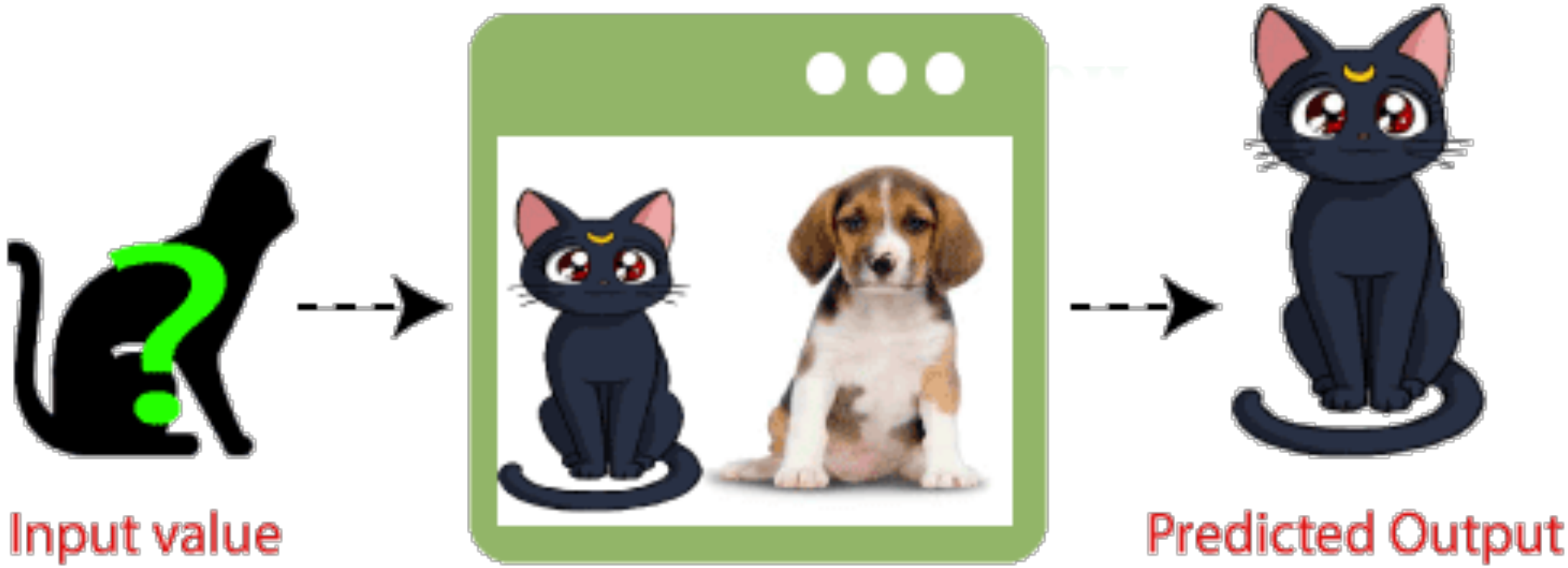
- Overview Machine Learning
- KNN Motivation
- KNN for Classification
- How to Select k in KNN
- KNN for Regression
- KNN with Brute force
- KNN with K-D Tree
- KNN with Ball Tree

Lazy Motivation

ANY APPROACHES DOES NOT MAKE ANY ASSUMPTION ON UNDERLYING DATA

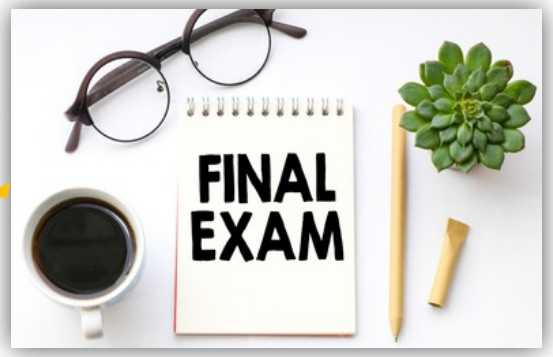
- Just store dataset without learning from it
- Start classifying data when it receive Test data
- It takes less time learning

Lazy Motivation



Measure the similar feature (distance)

Lazy Motivation



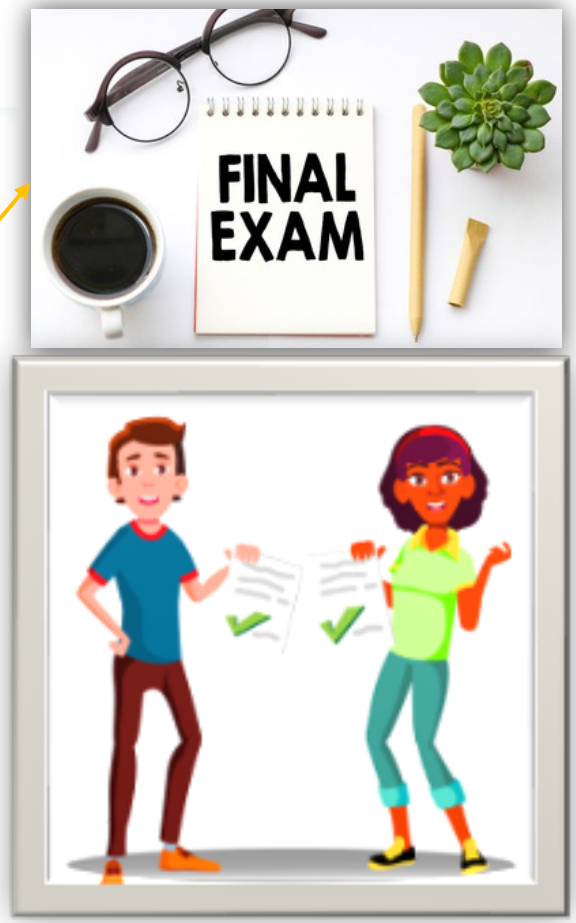
Study Hard

Laziness

Two students are going to take the final exam

Lazy Motivation

Study hard

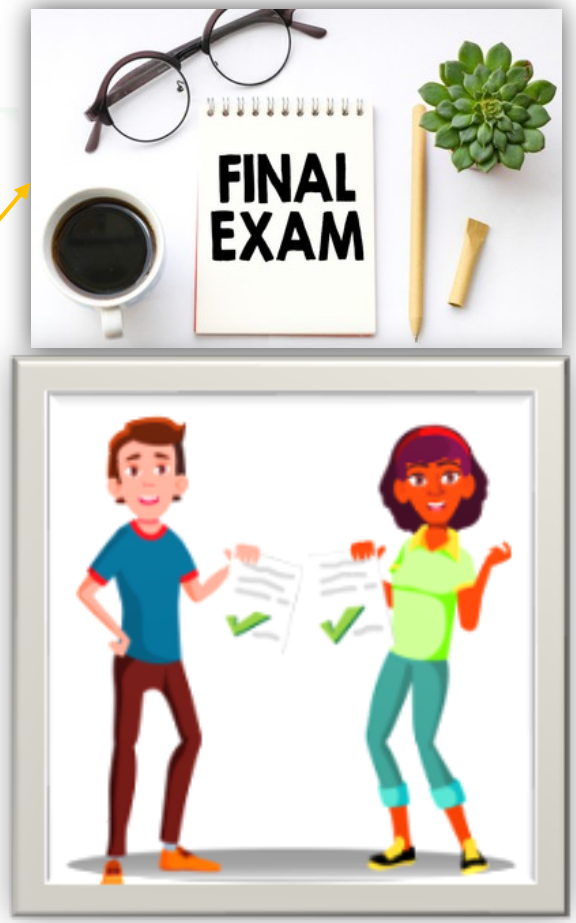


Laziness

Two students are get good results

Lazy Motivation

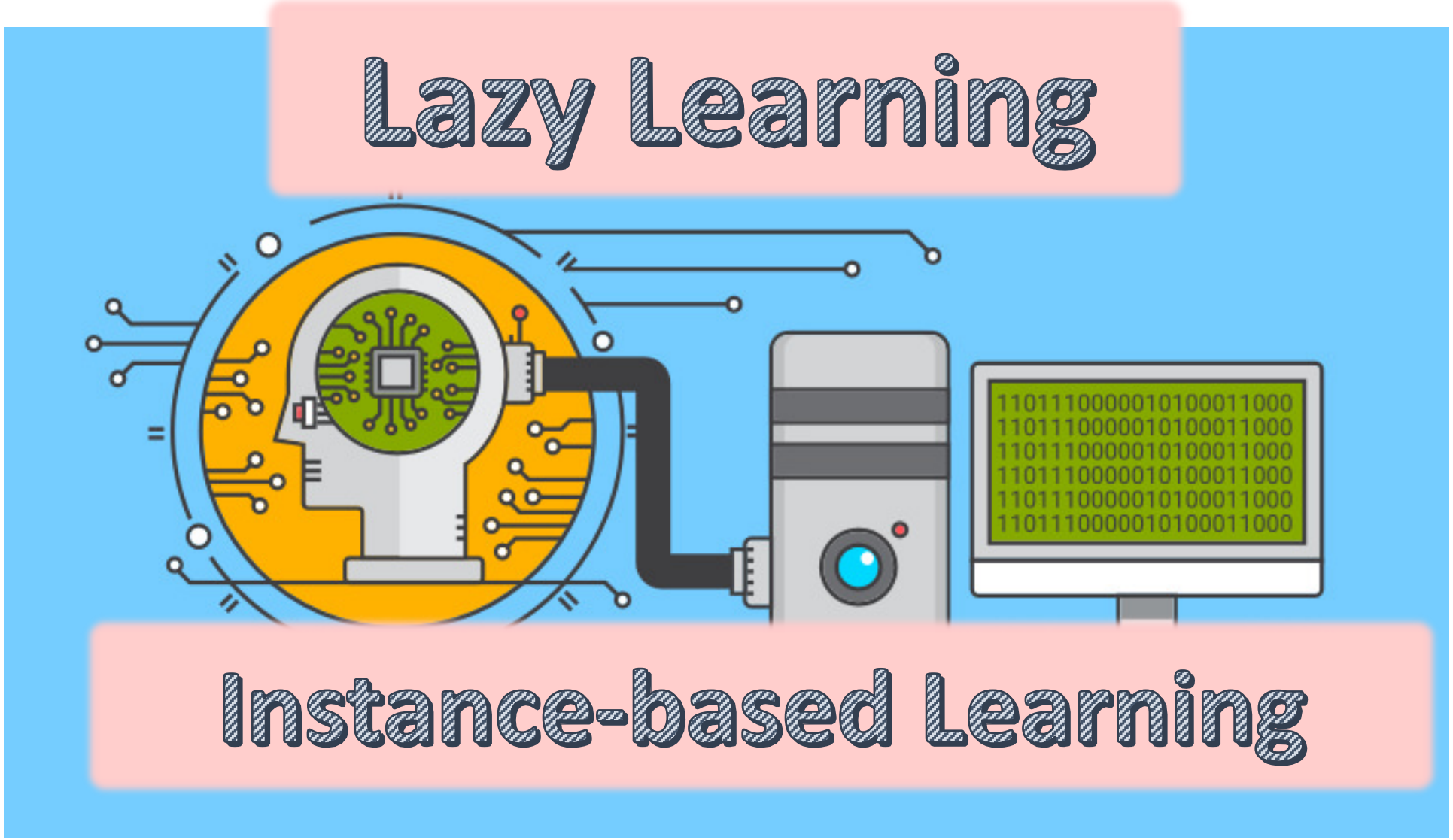
Study hard



Laziness

What is happening Here?

Lazy Motivation



Lazy Motivation

ANY APPROACHES DOES NOT MAKE ANY ASSUMPTION ON UNDERLYING DATA?

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

Lazy Learning, Instance-based Learning

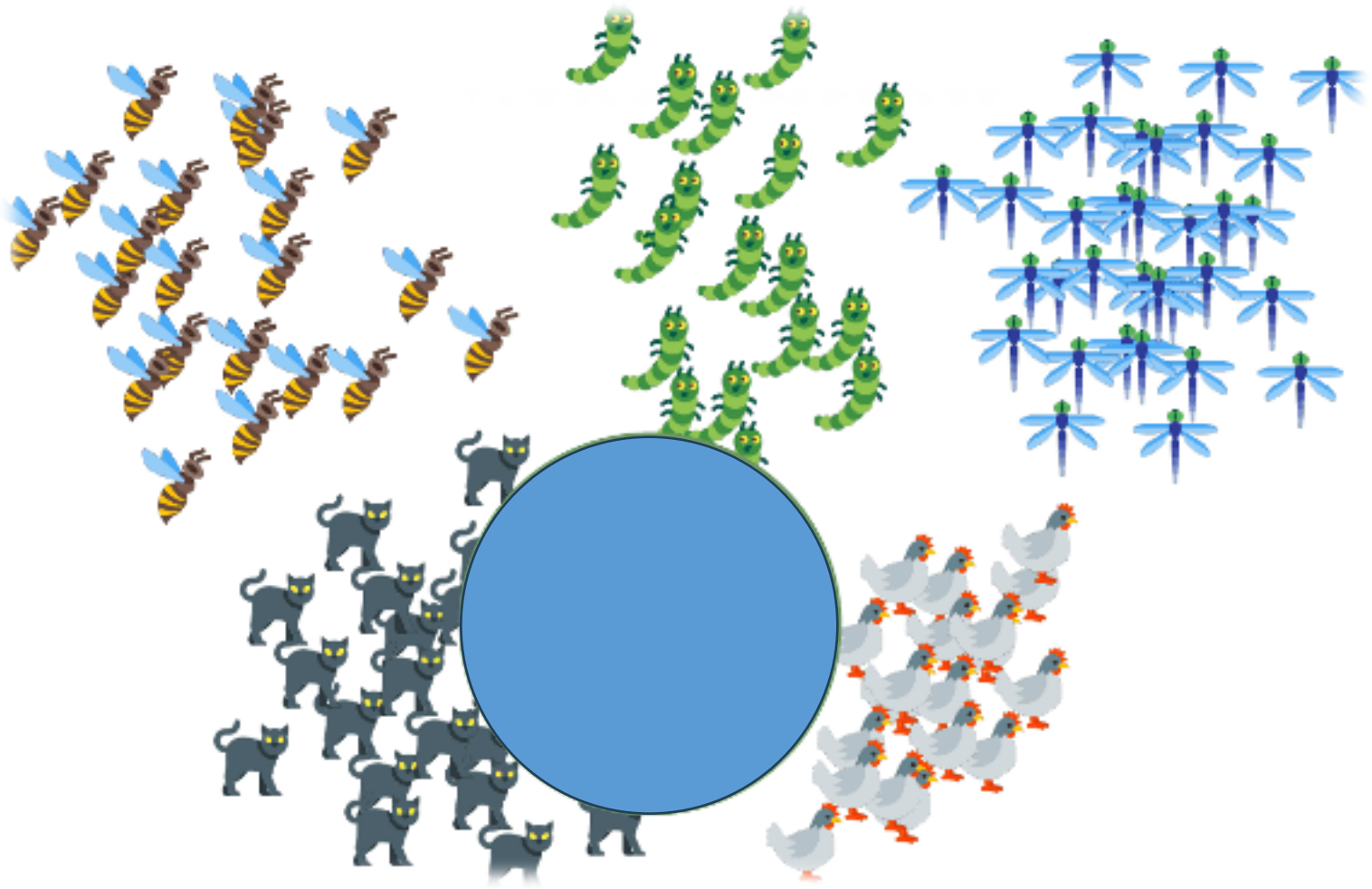
- One of the most simplest algorithms

Outline

- Overview Machine Learning
- KNN Motivation
- KNN for Classification
- How to Select k in KNN
- KNN for Regression
- KNN with Brute force
- KNN with K-D Tree
- KNN with Ball Tree

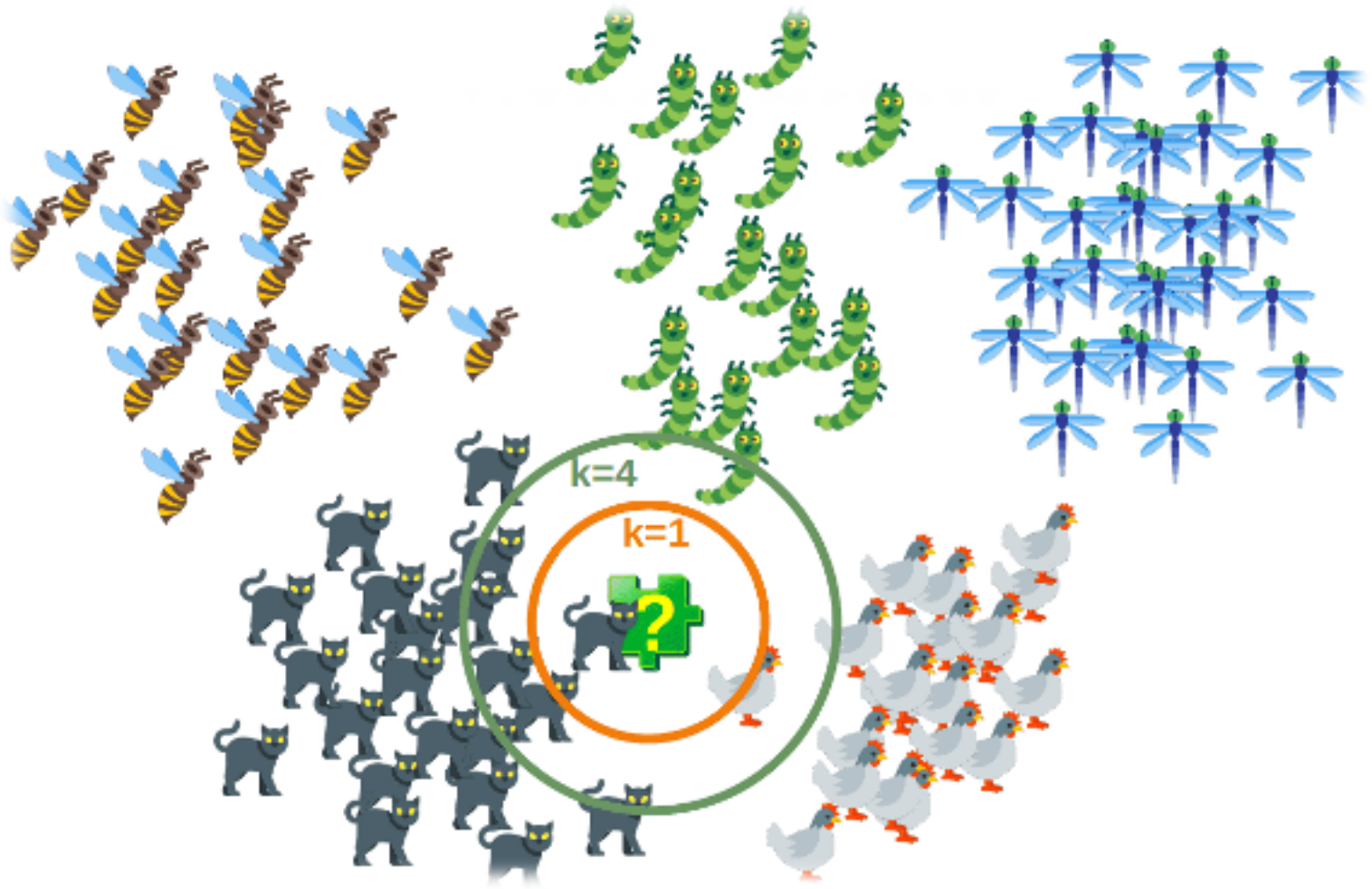
KNN FOR CLASSIFICATION

Motivation



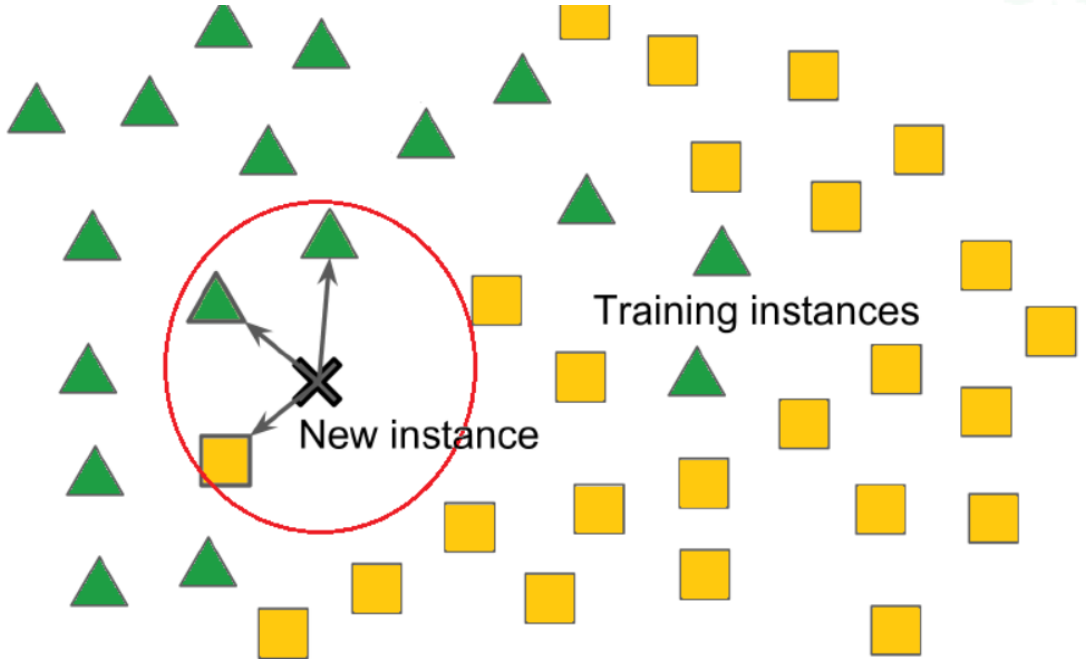
Given dataset with known categories

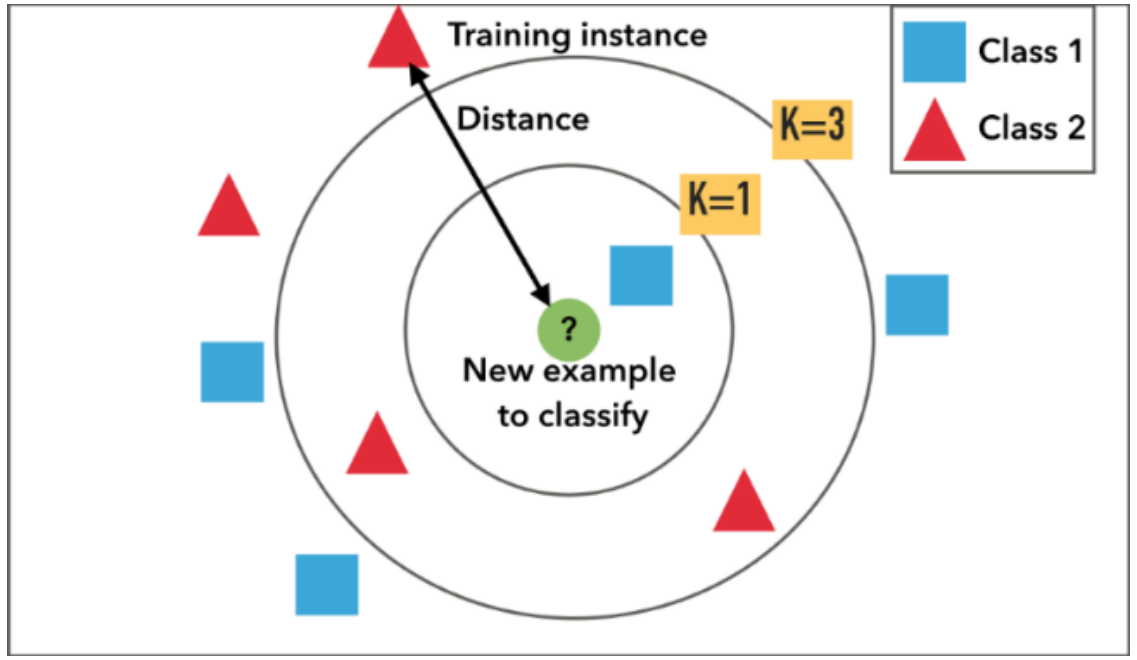
Motivation



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

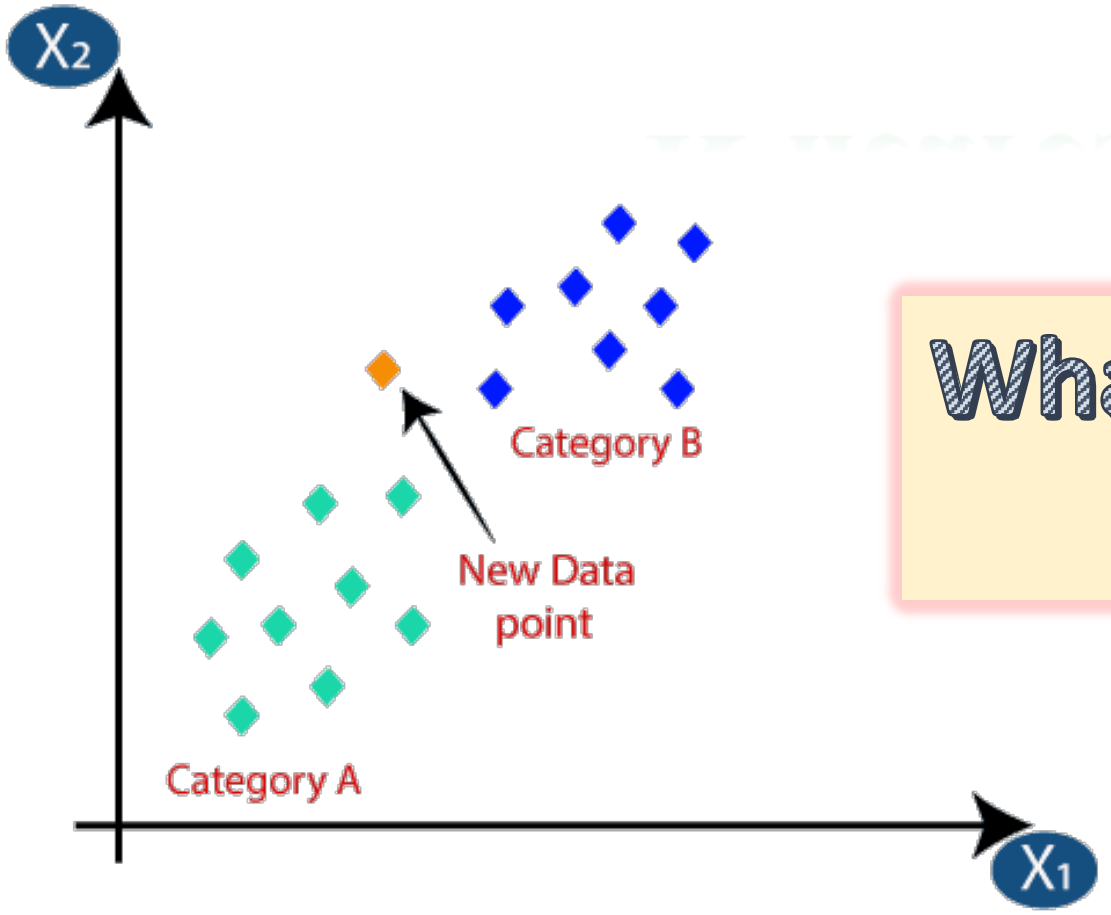
Motivation





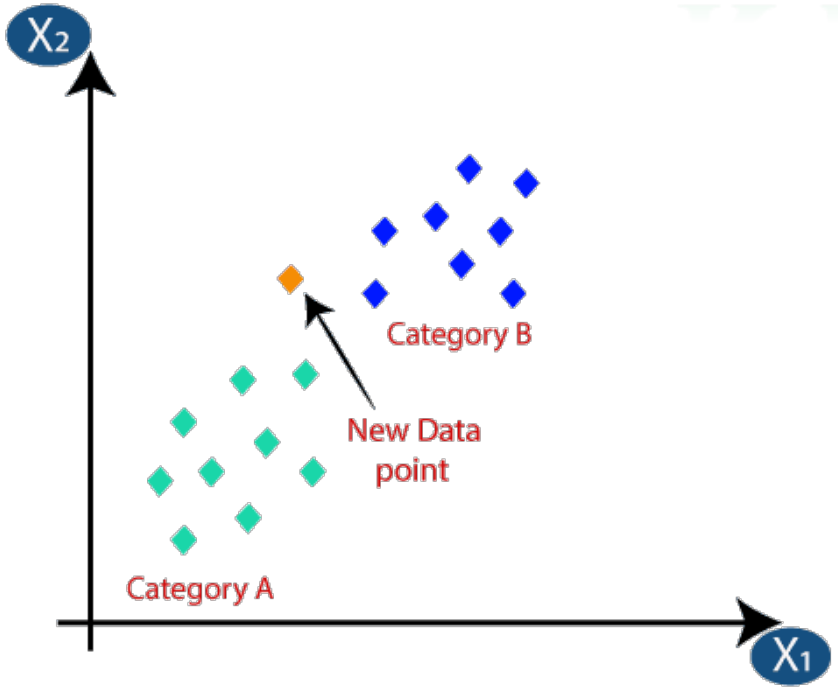
What is a category of a new image? → **Look for K-nearest neighbors**

K-nearest Neighbor



What is the class label of a new data point?

K-nearest Neighbor



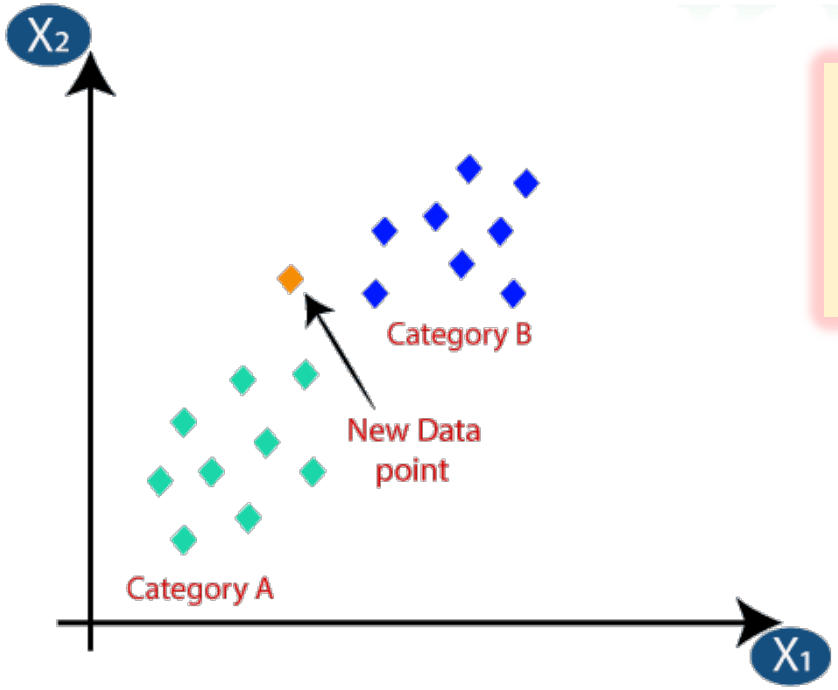
STEP 1

Select K-Nearest neighbors

K = 5

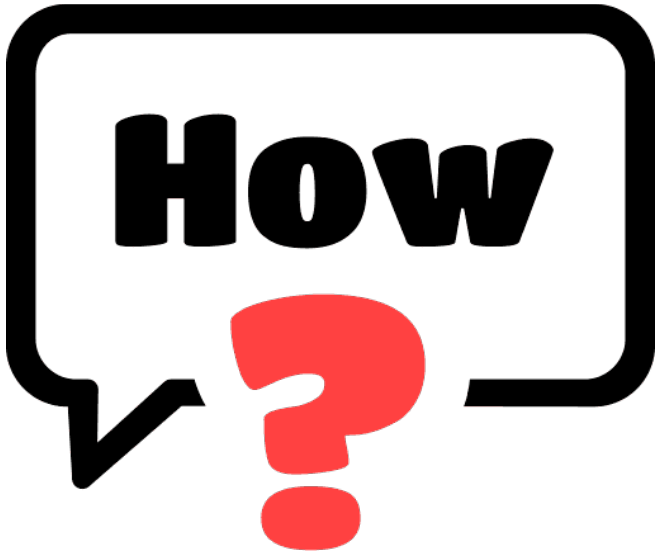
How to find 5 nearest neighbors

K-nearest Neighbor



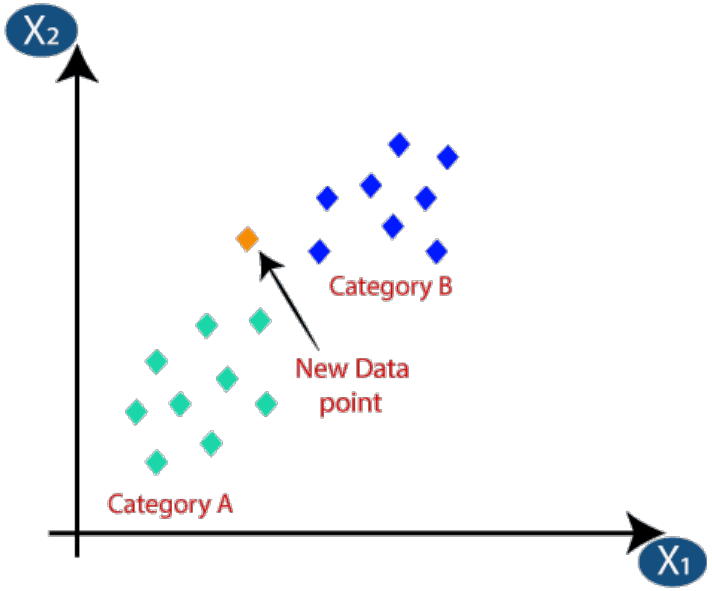
STEP 2

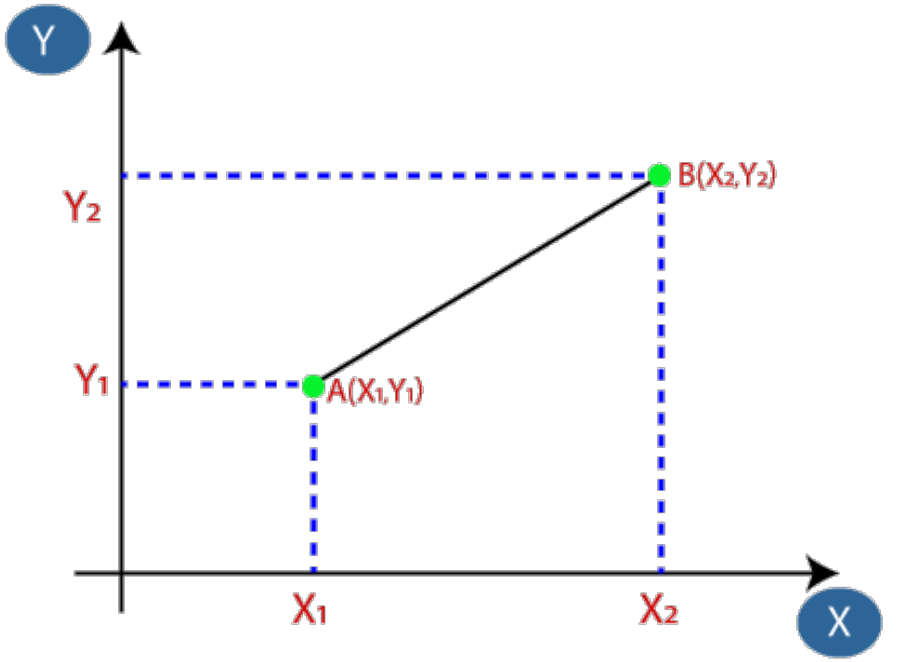
Calculate the similiar feature (distance) between test point and all data point



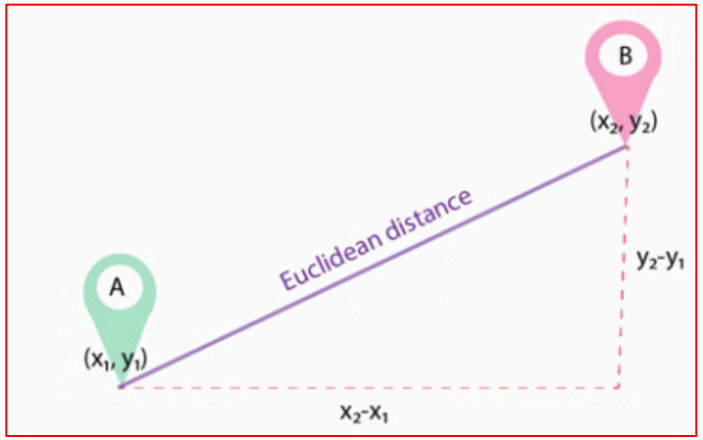
K-nearest Neighbor

STEP 2: Euclidean Distance





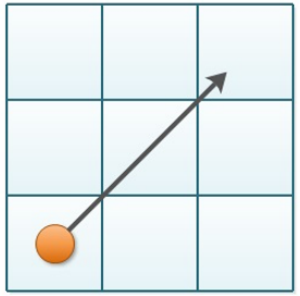
Euclidean Distance between A_1 and $B_2 = \sqrt{(X_2-X_1)^2+(Y_2-Y_1)^2}$



K-nearest Neighbor

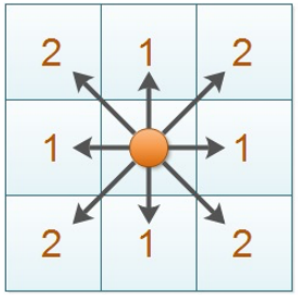
STEP 2: Other Distances

Euclidean Distance



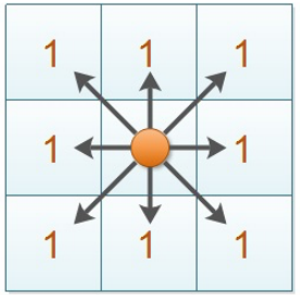
$$\sqrt{(x_1 - x_2)^2 + (y_1 - y_2)^2}$$

Manhattan Distance

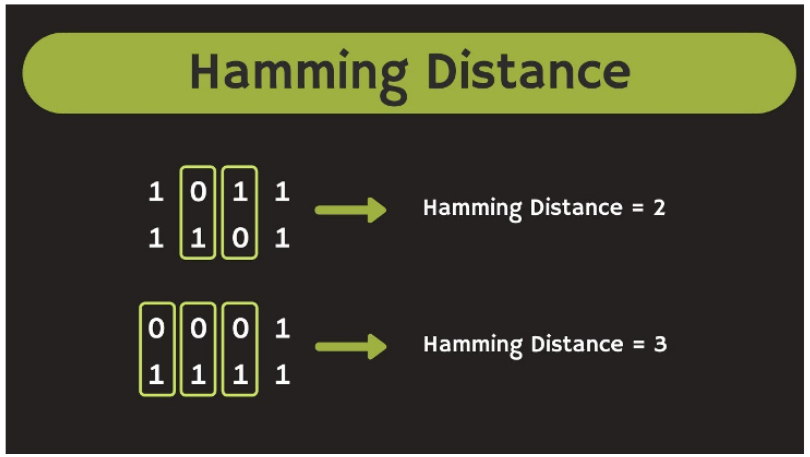


$$|x_1 - x_2| + |y_1 - y_2|$$

Chebyshev Distance

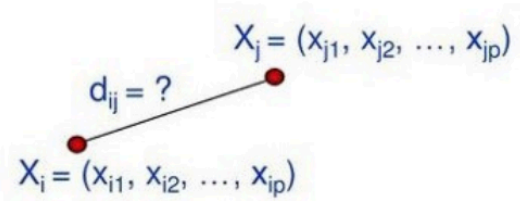


$$\max(|x_1 - x_2|, |y_1 - y_2|)$$



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

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

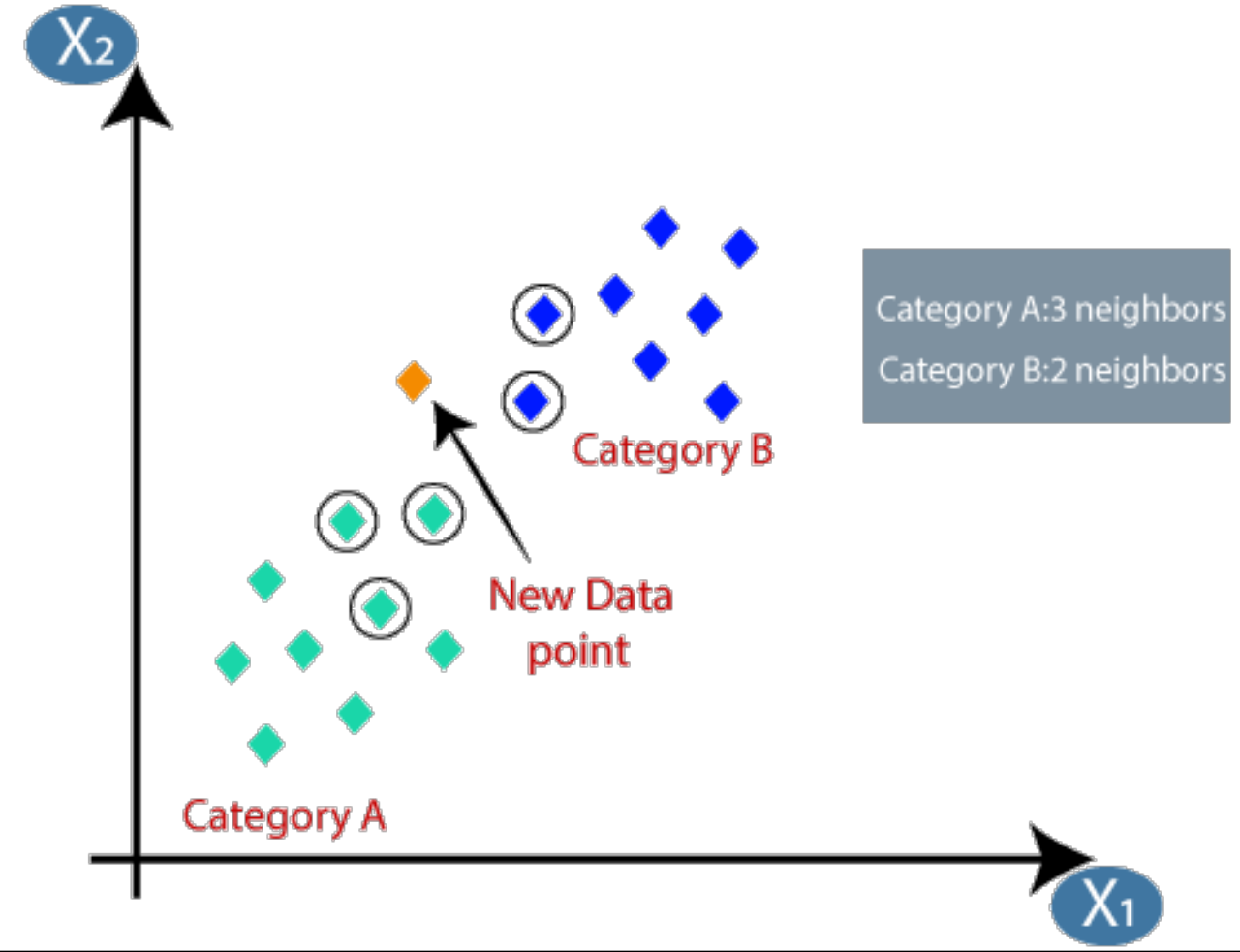
- **Manhattan distance**

q = 1

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

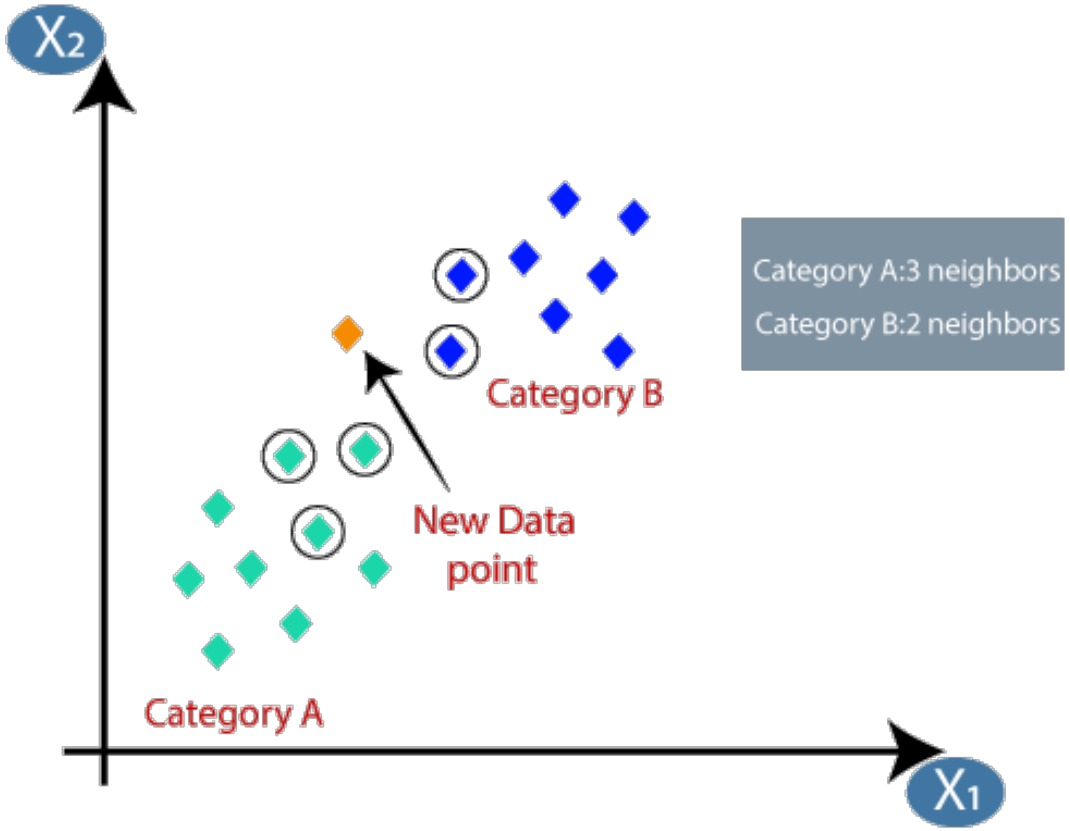
K-nearest Neighbor

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

```
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

- Overview Machine Learning
- KNN Motivation
- KNN for Classification
- How to Select k in KNN
- KNN for Regression
- KNN with Brute force
- KNN with K-D Tree
- KNN with Ball Tree

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%

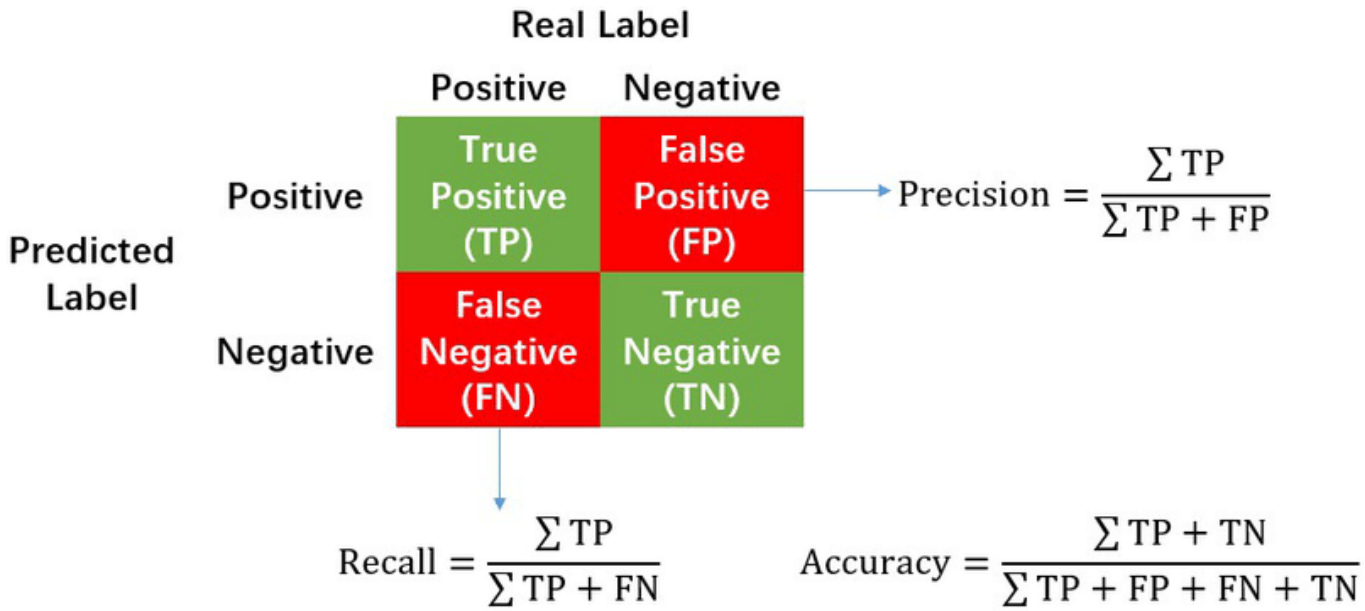
Error Analysis

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?

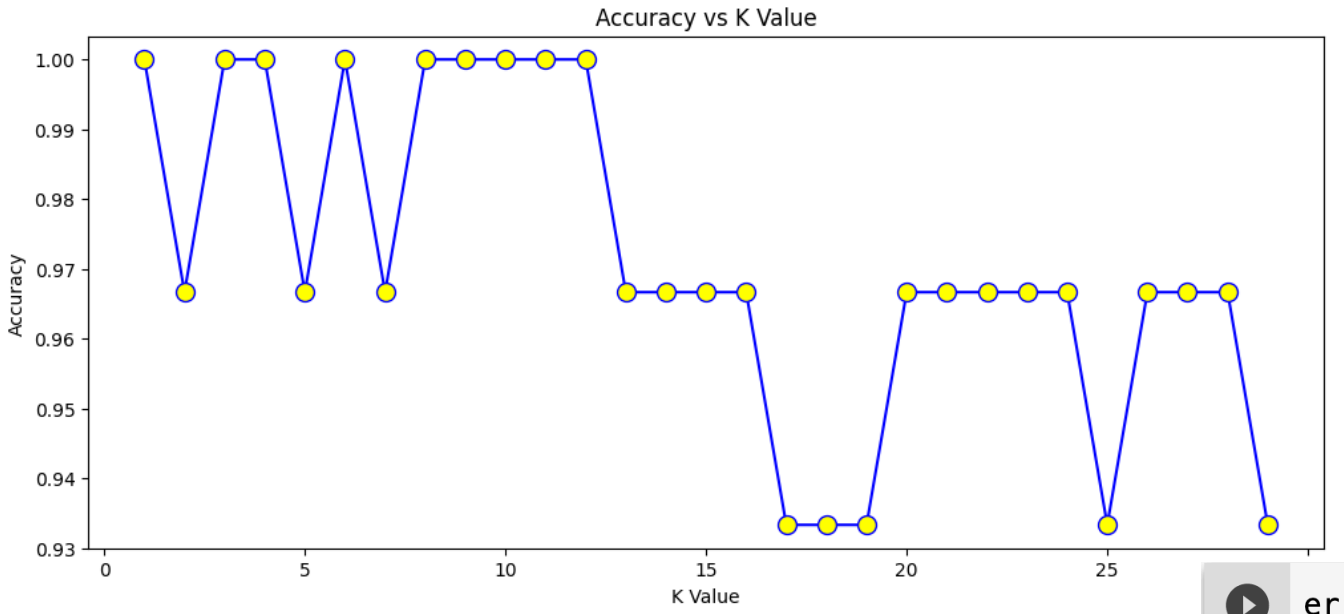


Error Analysis

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

AI VIETNAM
All-in-One Course

How to select *k* in K-NN



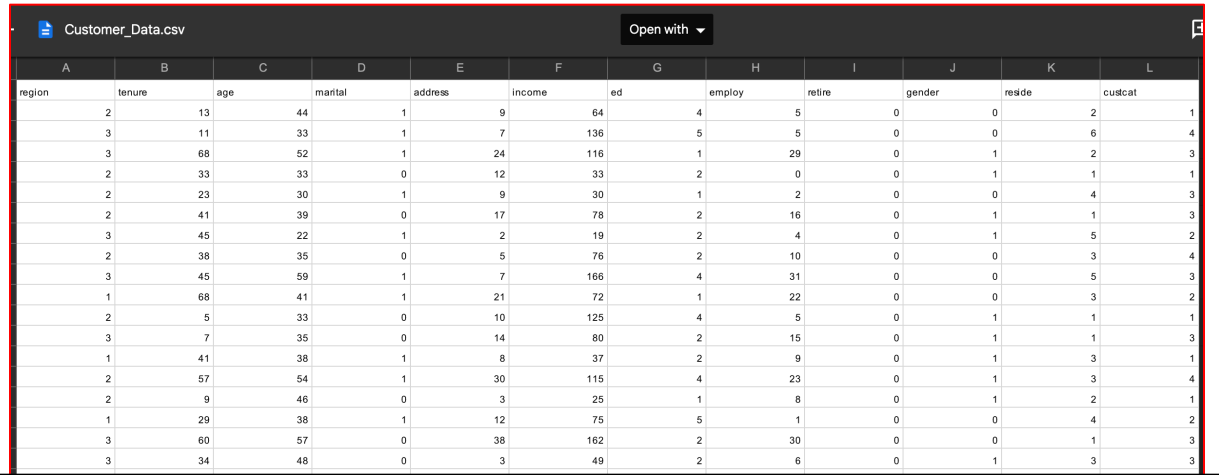
Bạn có thắc mắc gì chỗ này không?

```
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

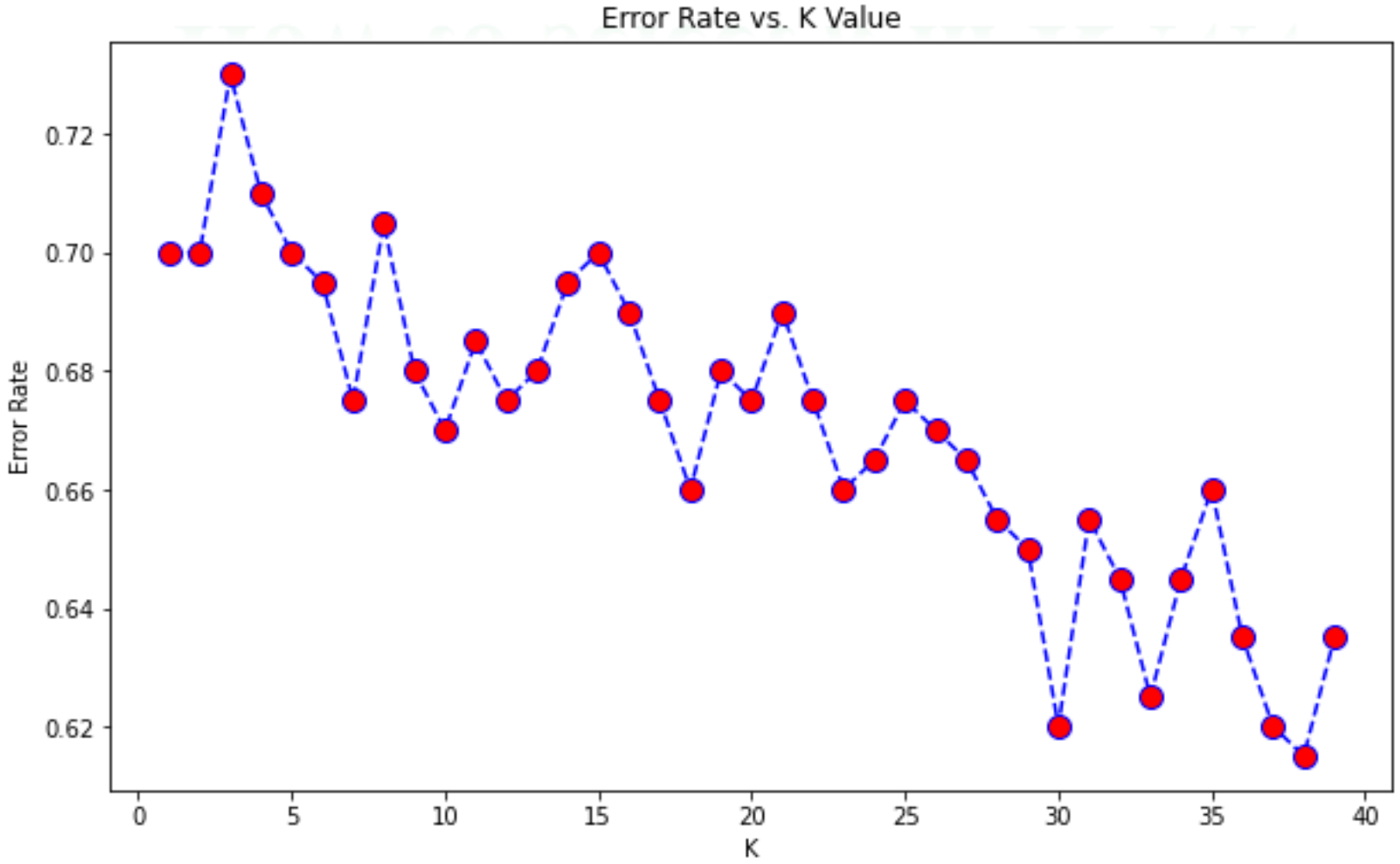
- 1- Basic Service
- 2- E-Service
- 3- Plus Service
- 4- Total Service



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

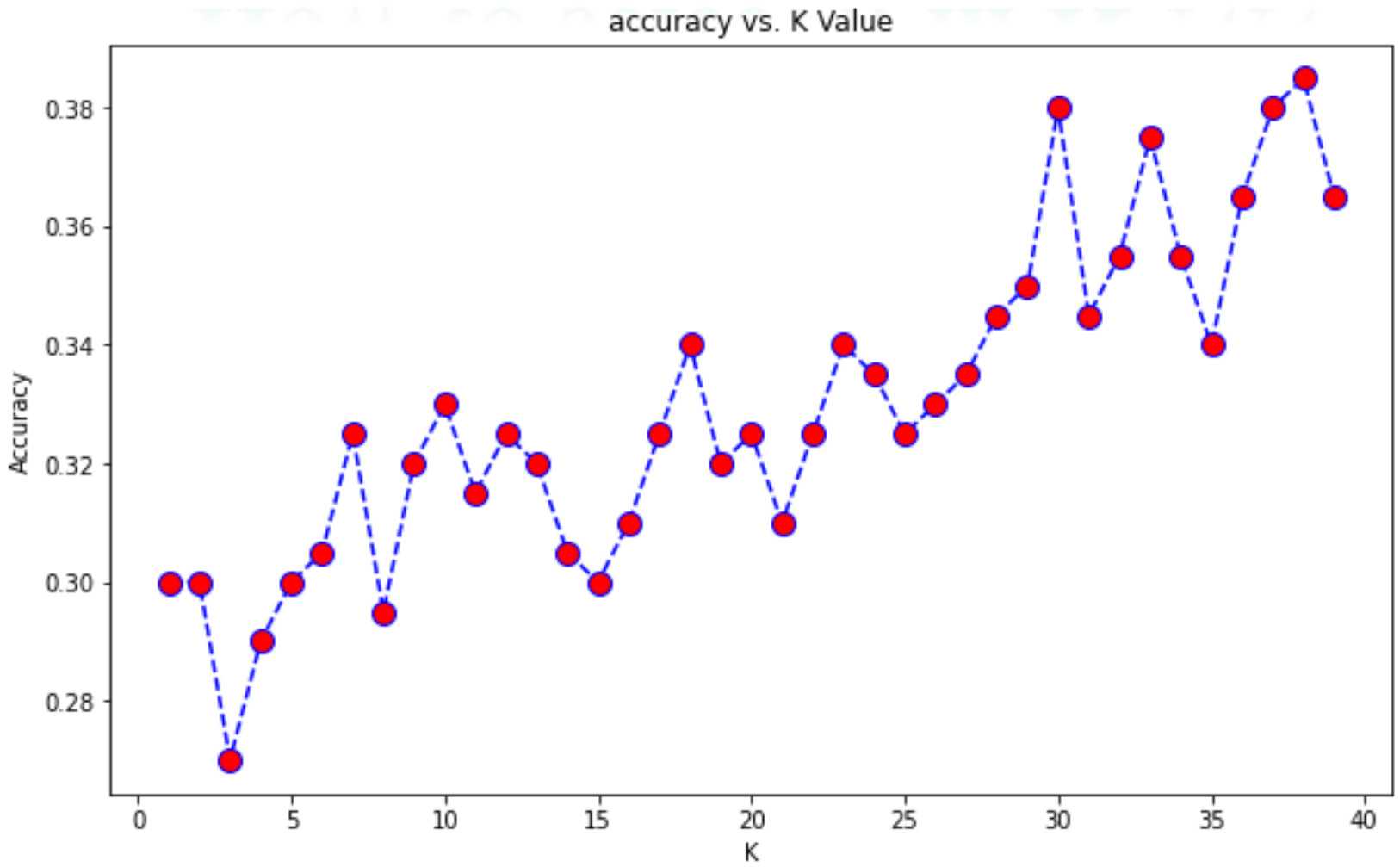
49

How to select *k* in K-NN



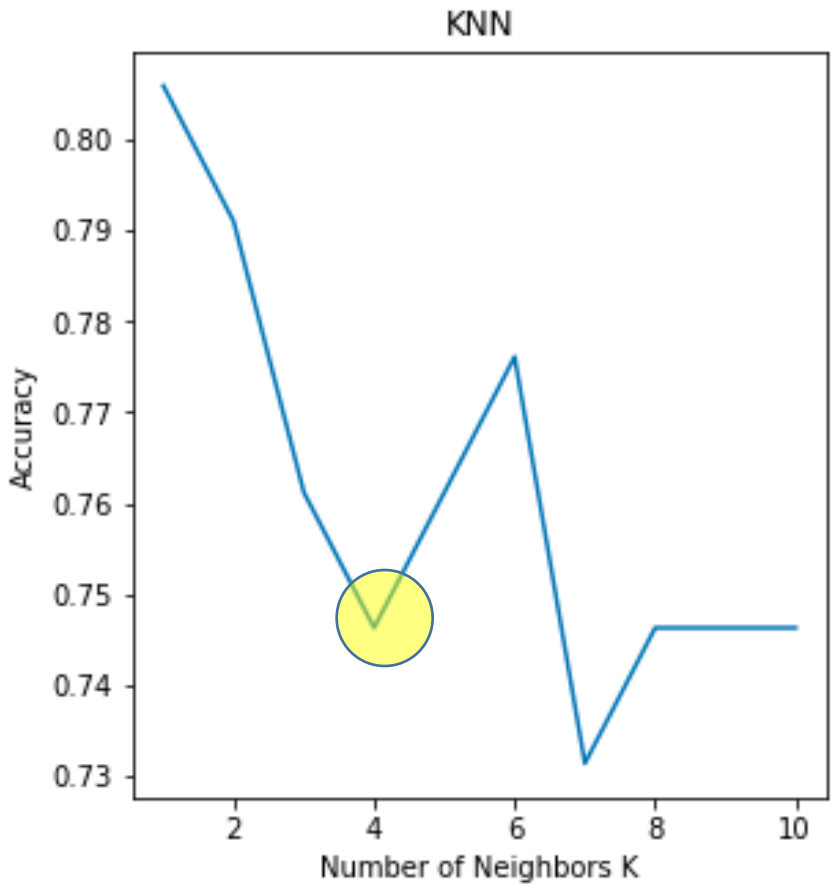
50

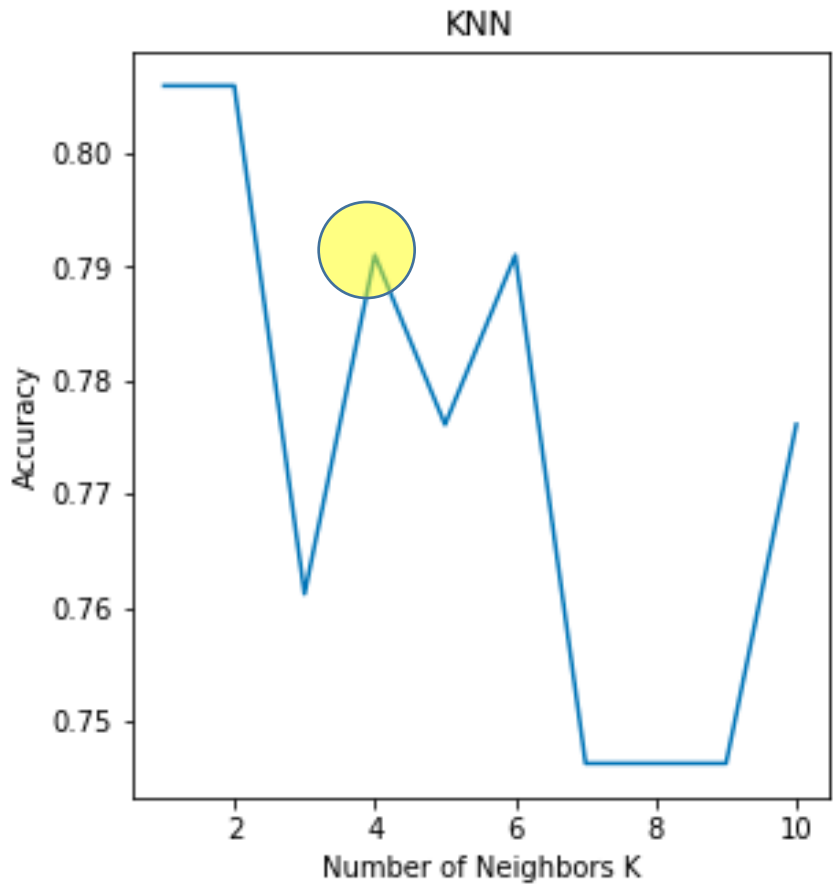
How to select *k* in K-NN



51

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)
```

Sklearn Library

Result

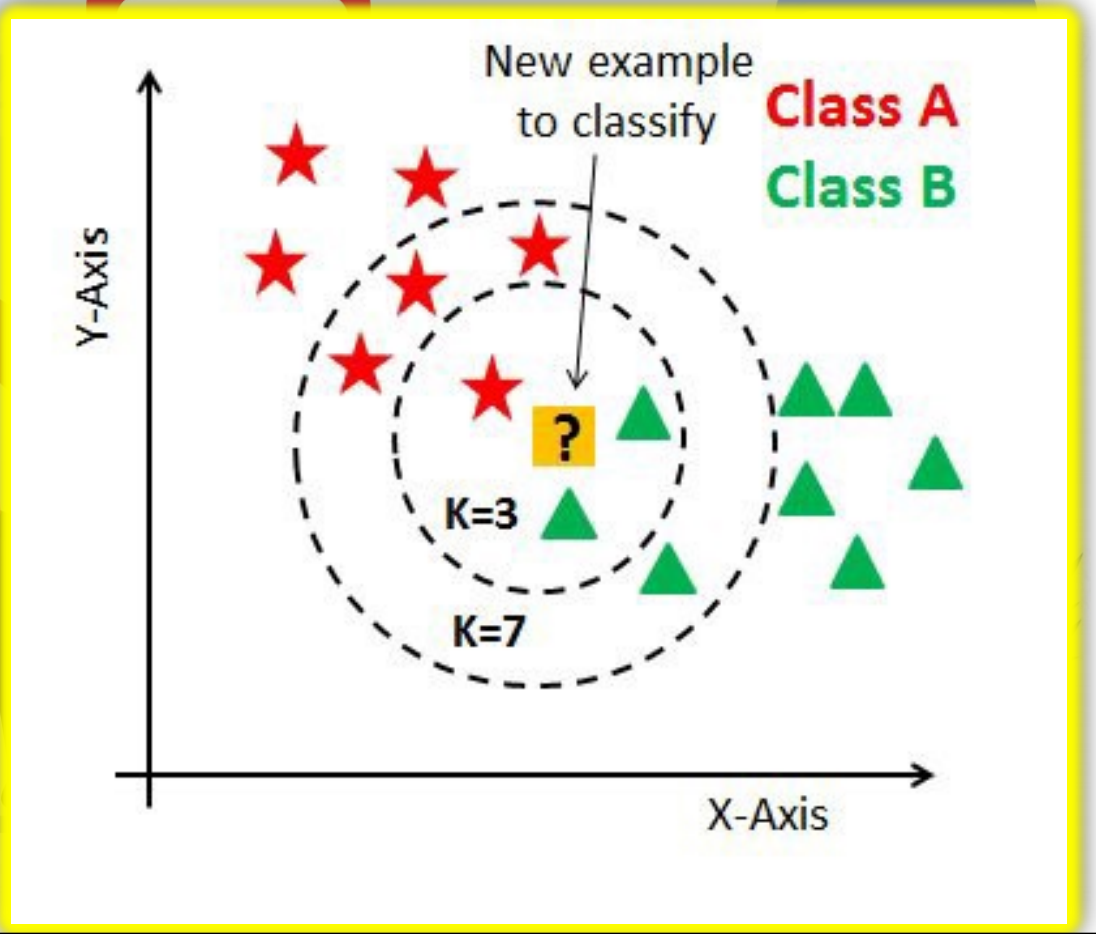
```
print("\t\t KNN in SKlearn  \t Modified KNN")
for i in range(0,4):
    print("k = ",i+1, "\t\t     ",round(scores[i],3),"\t\t    ",round(scores1[i],3))
```

```
                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

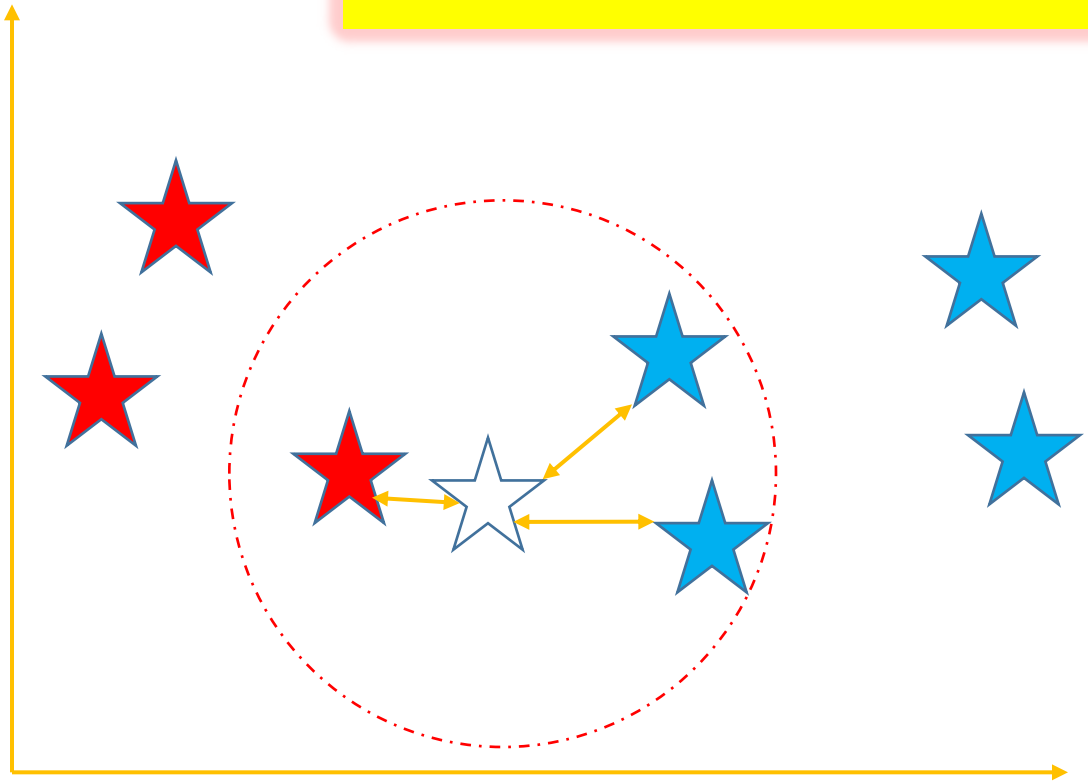
WEIGHT IN K-NEAREST NEIGHBOR?



55

Discussion

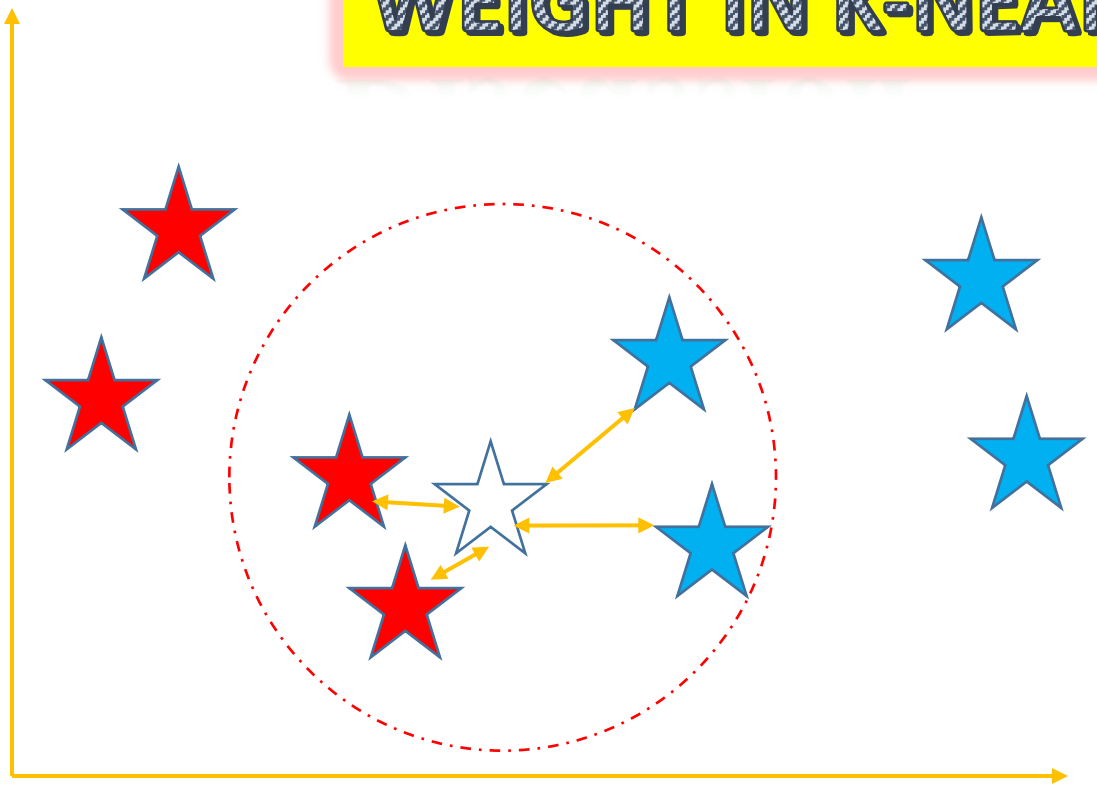
WEIGHT IN K-NEAREST NEIGHBOR?



56

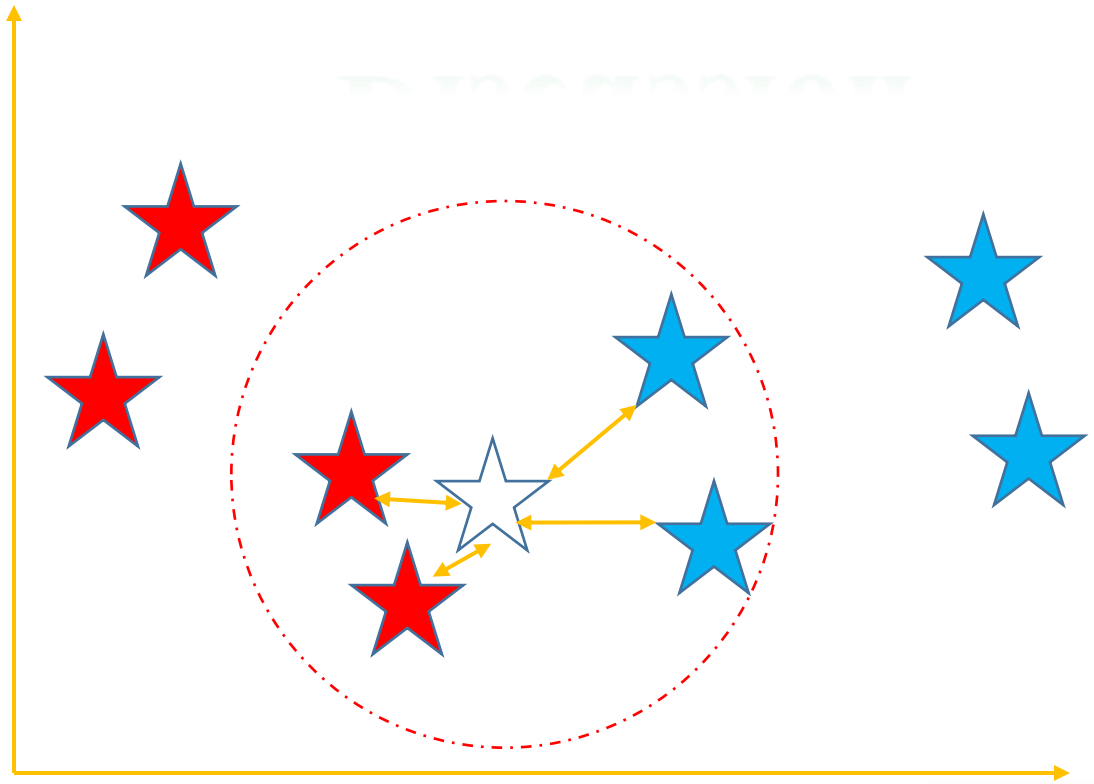
Discussion

WEIGHT IN K-NEAREST NEIGHBOR?



57

Discussion



UNIFORM WEIGHTS
DISTANCE WEIGHTS
USER-DEFINED WEIGHTS

58

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
```


59

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
```


60

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
```


61

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
```


62

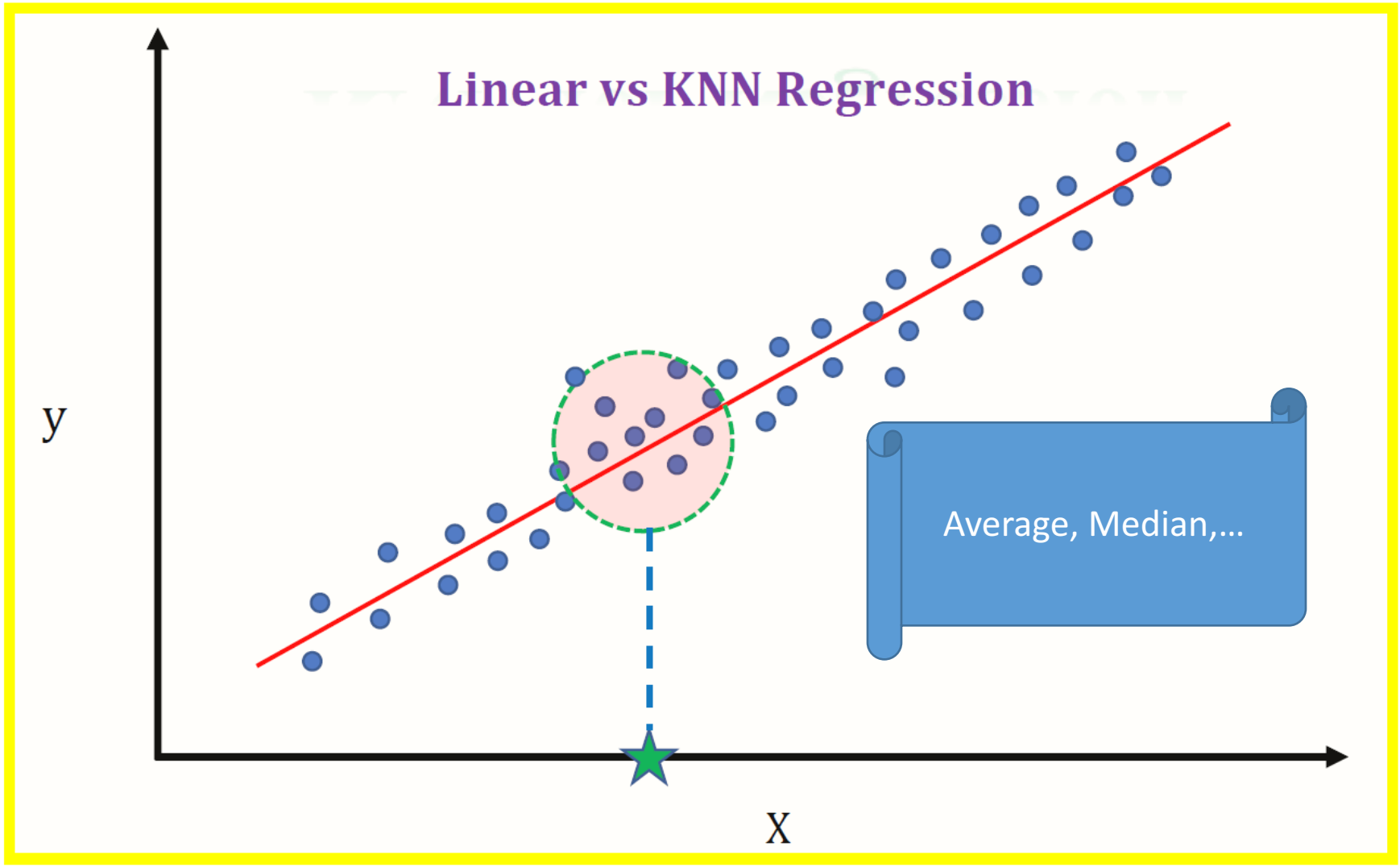
Outline

- Overview Machine Learning
- KNN Motivation
- KNN for Classification
- How to Select k in KNN
- KNN for Regression
- KNN with Brute force
- KNN with K-D Tree
- KNN with Ball Tree

KNN FOR REGRESSION

64

KNN for Regression



65

KNN for Regression

Choose a value for K

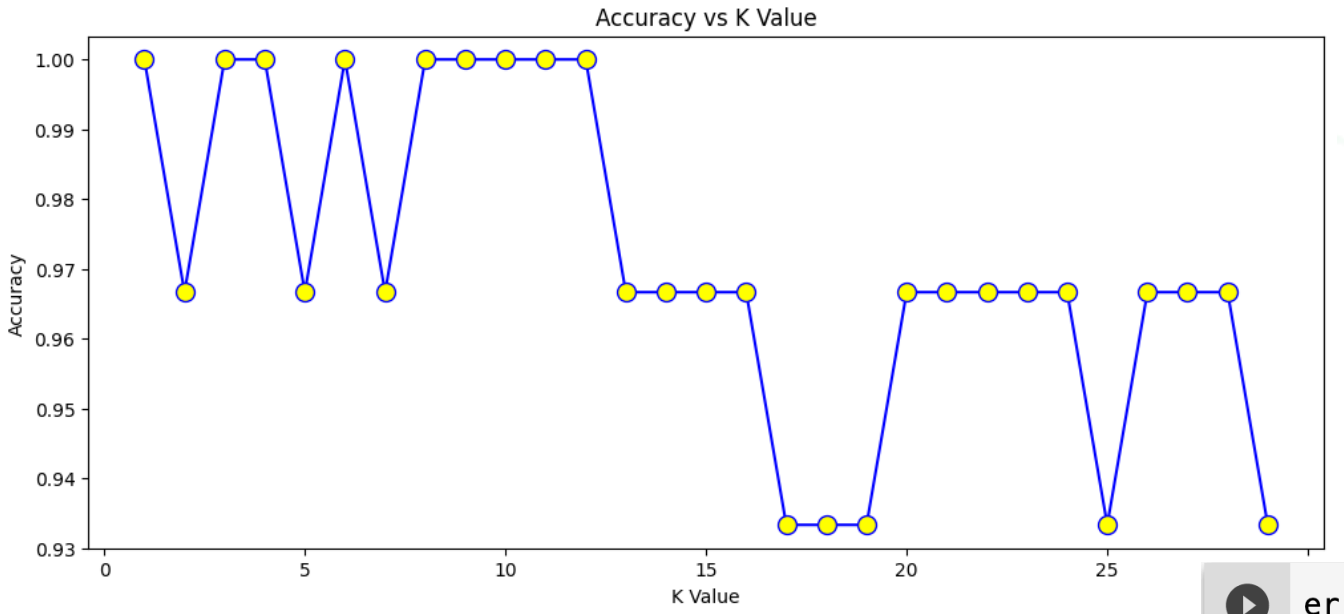
Calculate the distance

Find the K nearest neighbors

Calculate the prediction

66

Issue Review



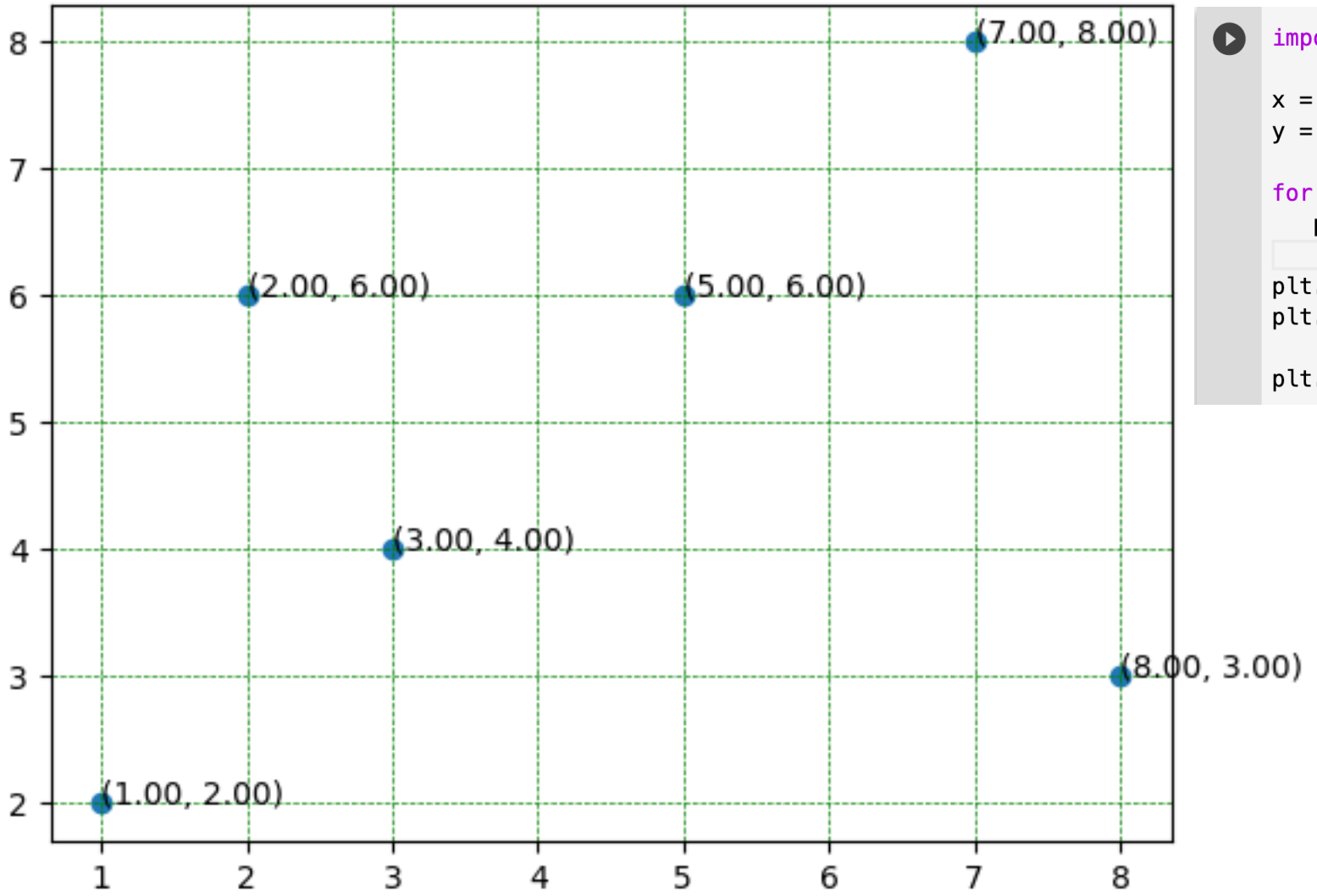
Bạn có thắc mắc gì chỗ này không?

```
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

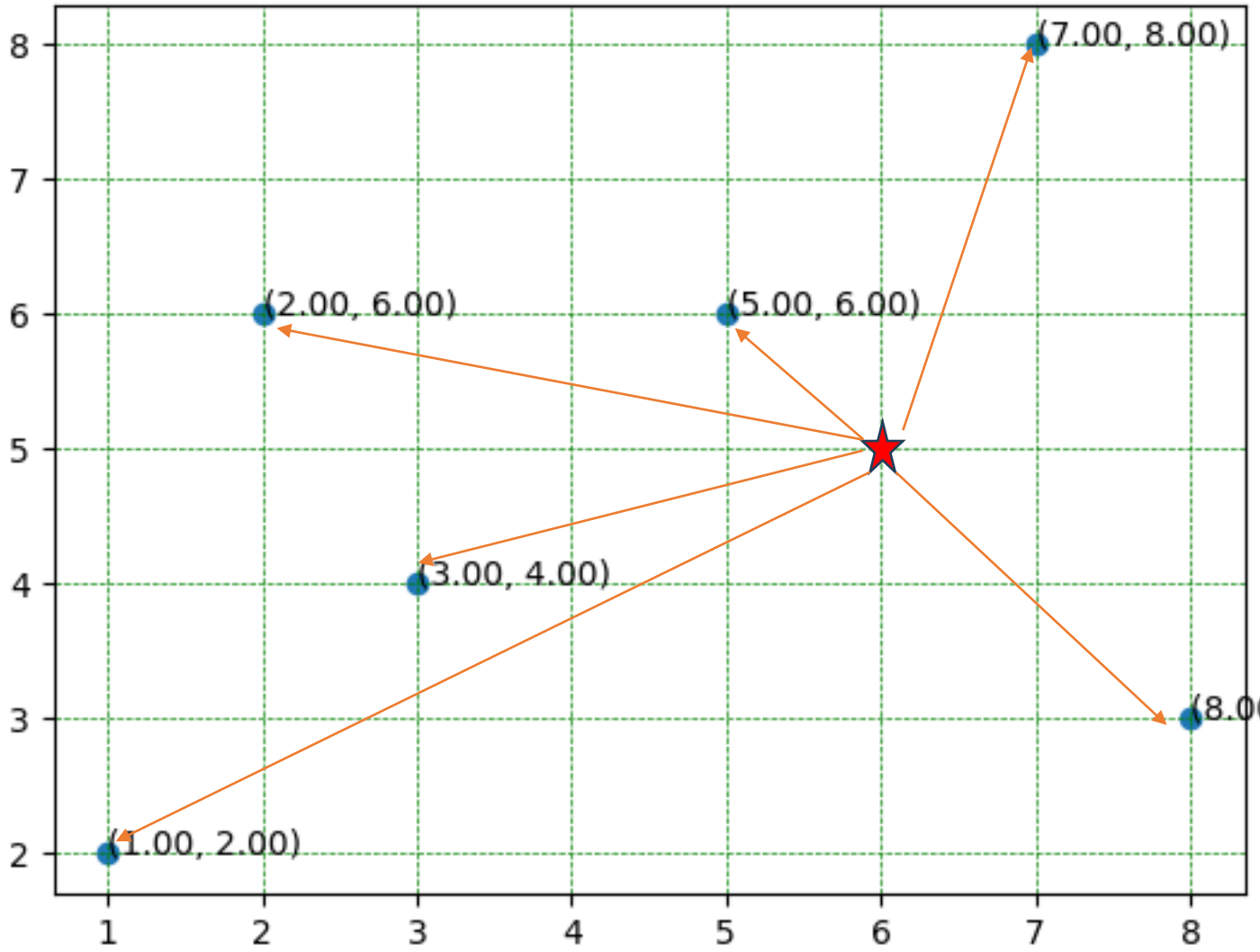
- Overview Machine Learning
- KNN Motivation
- KNN for Classification
- How to Select k in KNN
- KNN for Regression
- KNN with Brute force
- KNN with K-D Tree
- KNN with Ball Tree

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()
```

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()
```

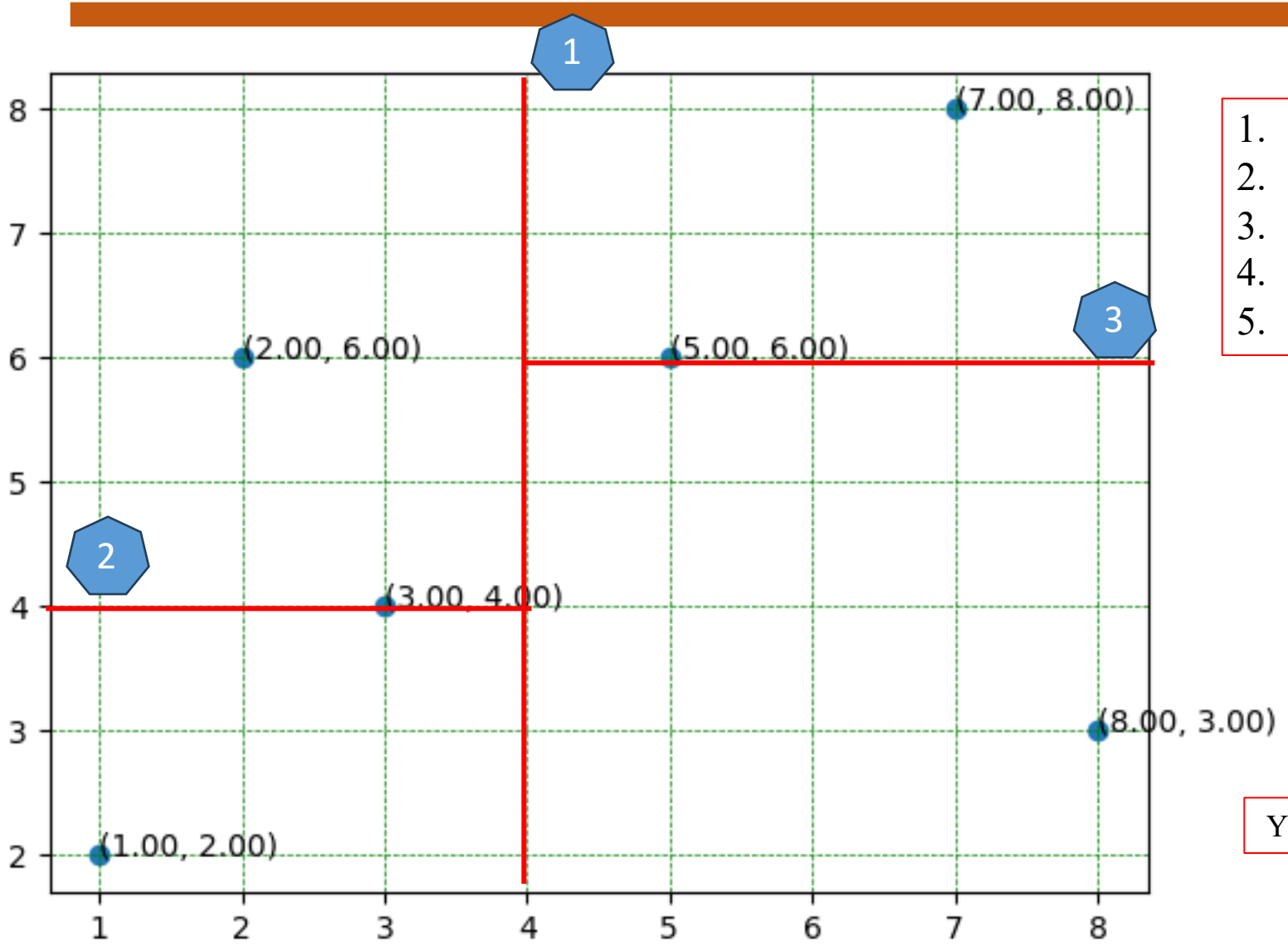
★ **New datapoint**

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

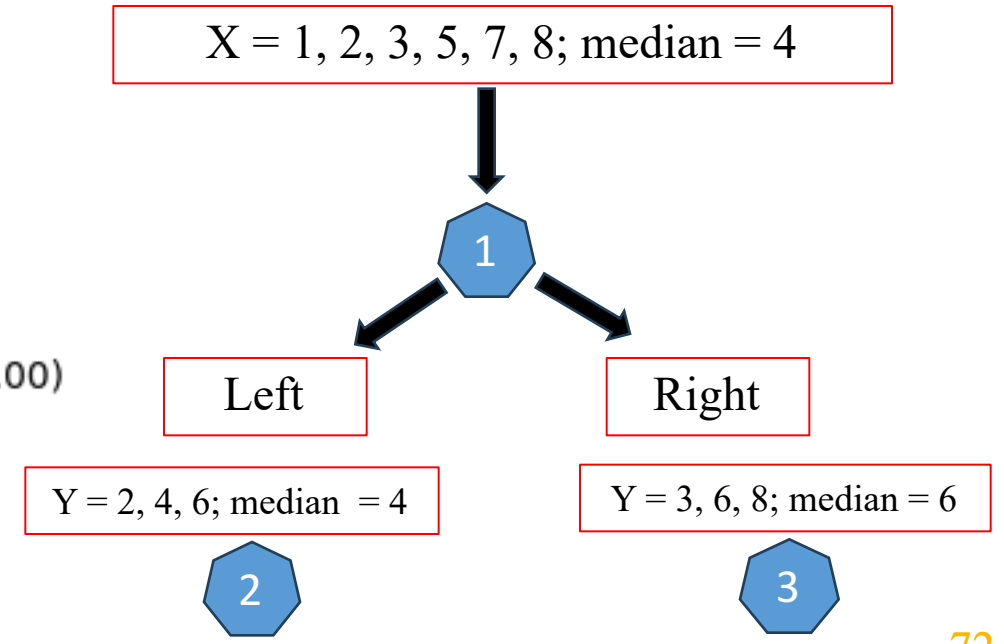
Outline

- Overview Machine Learning
- KNN Motivation
- KNN for Classification
- How to Select k in KNN
- KNN for Regression
- KNN with Brute force
- KNN with K-D Tree
- KNN with Ball Tree

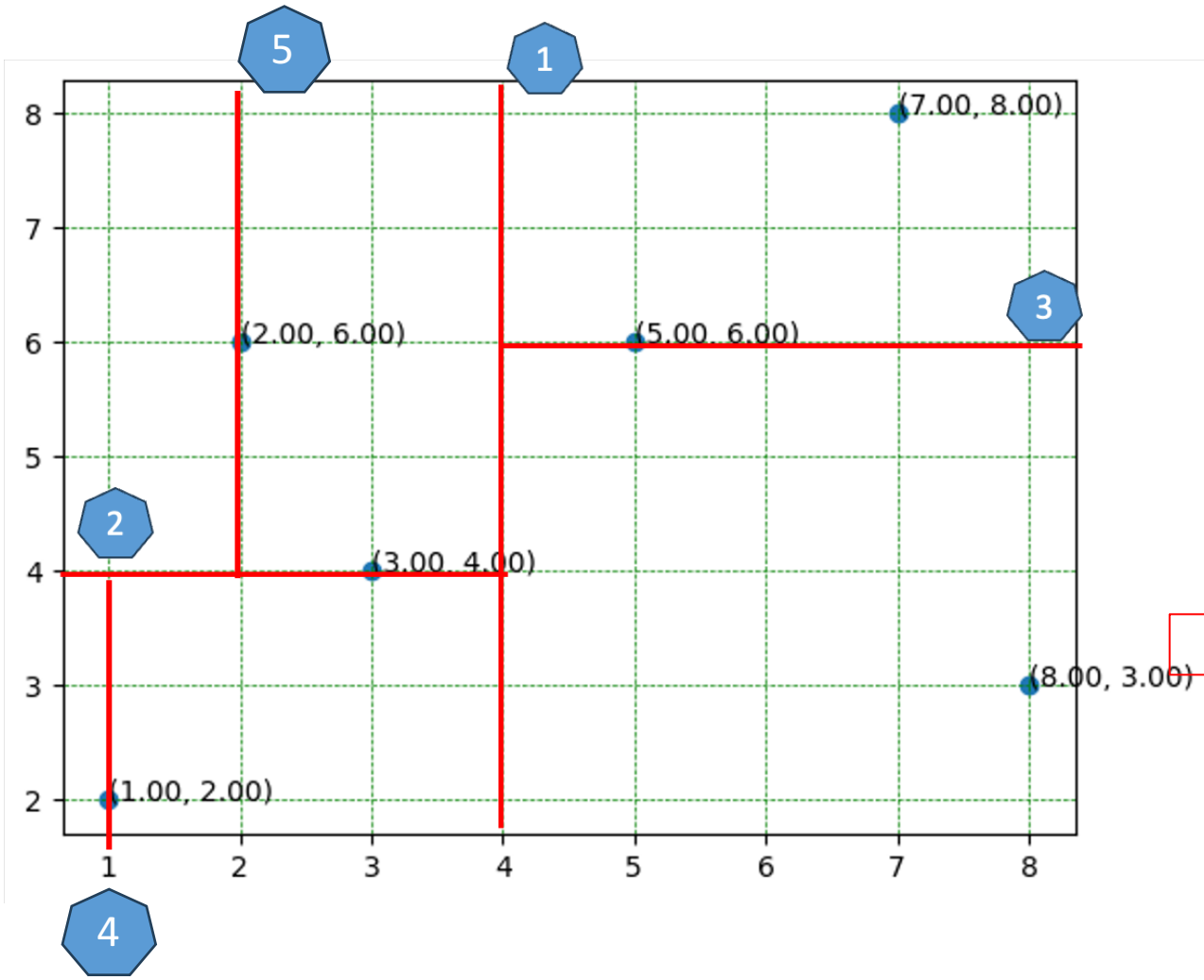
KNN with K-D Tree

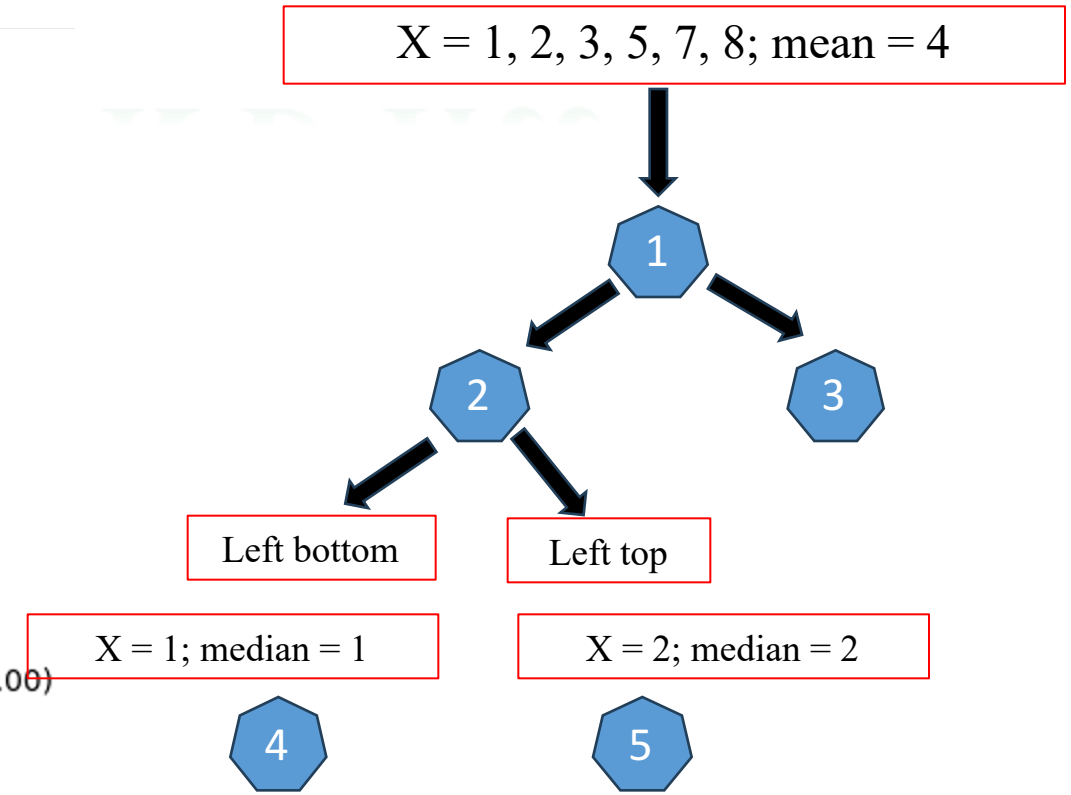


1. Pick any one feature at random
2. Find median
3. Split dataset in approximate equal halves
4. Pick next feature and repeat step #2,3
5. Continue until all data points are partitioned

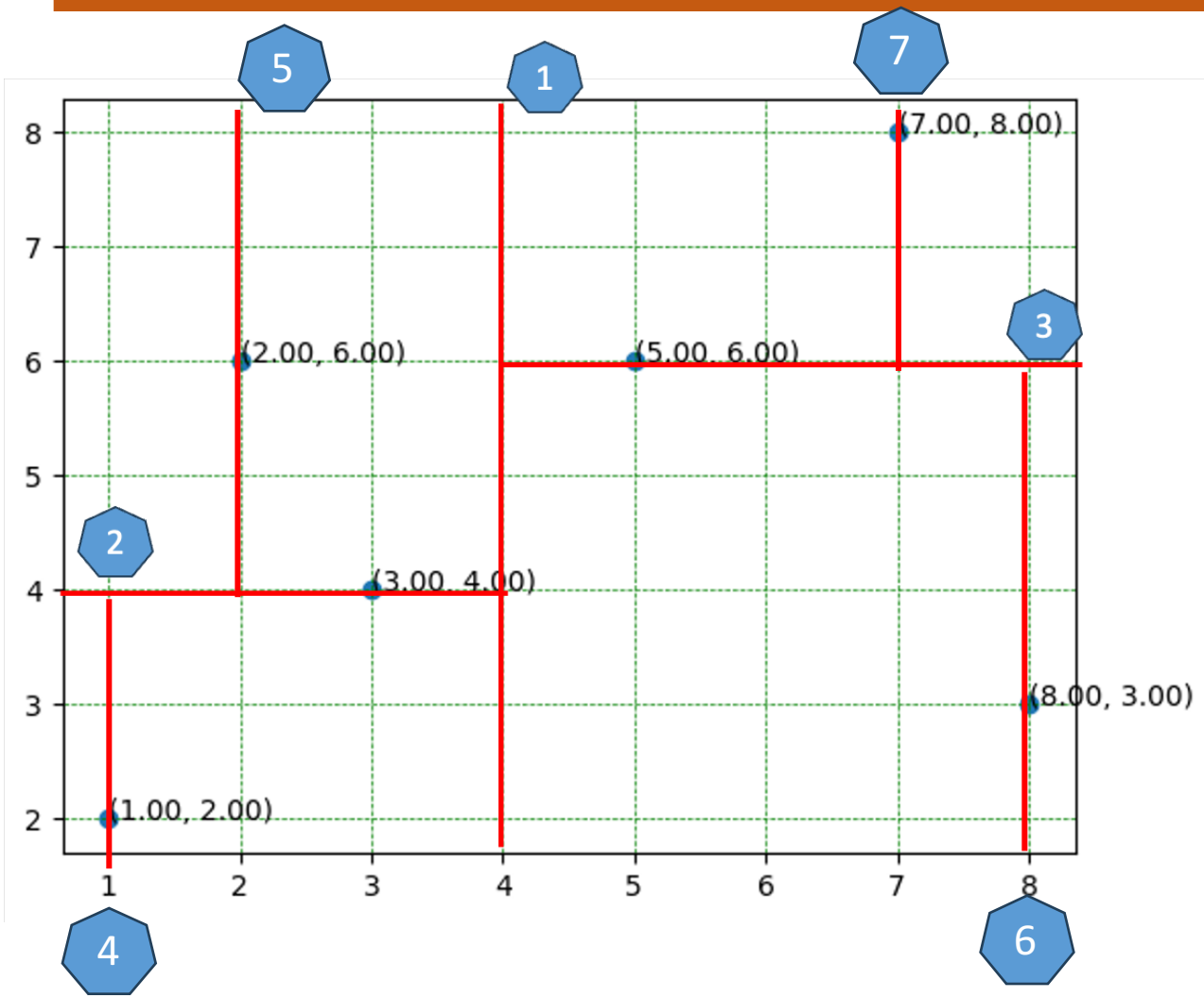


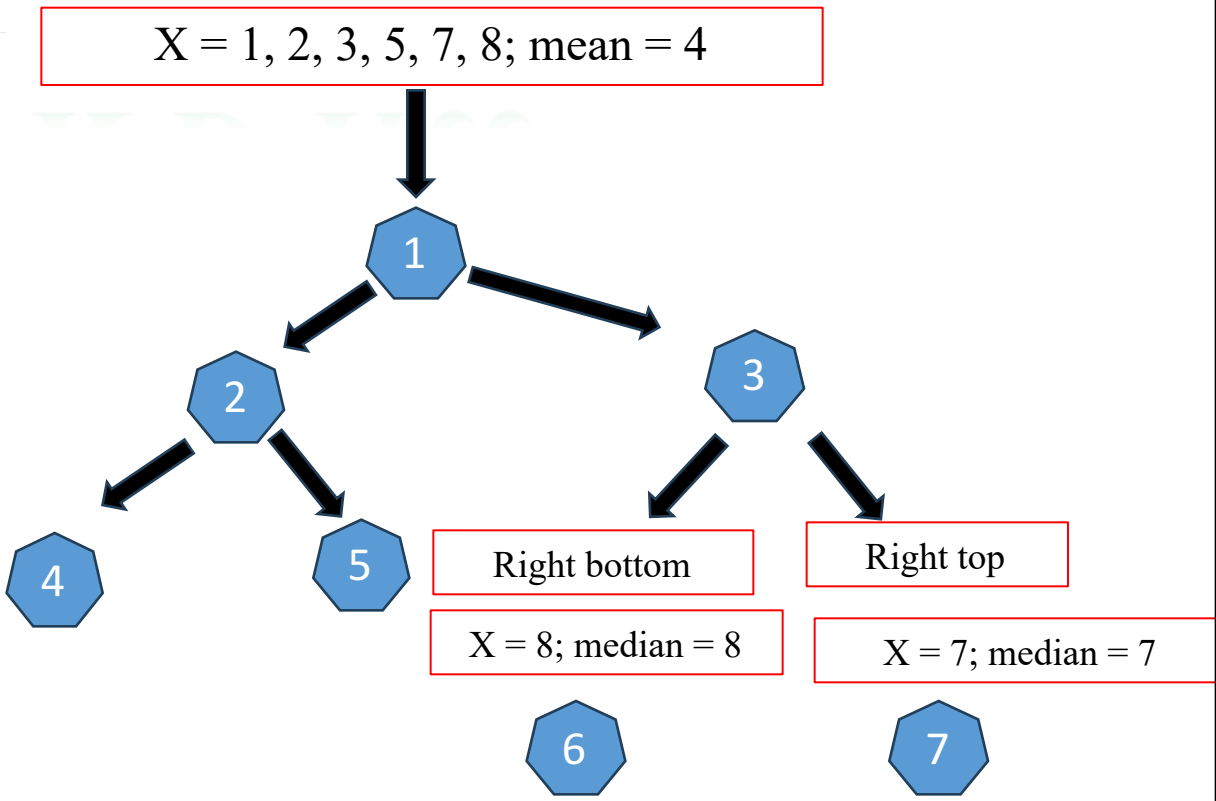
KNN with K-D Tree



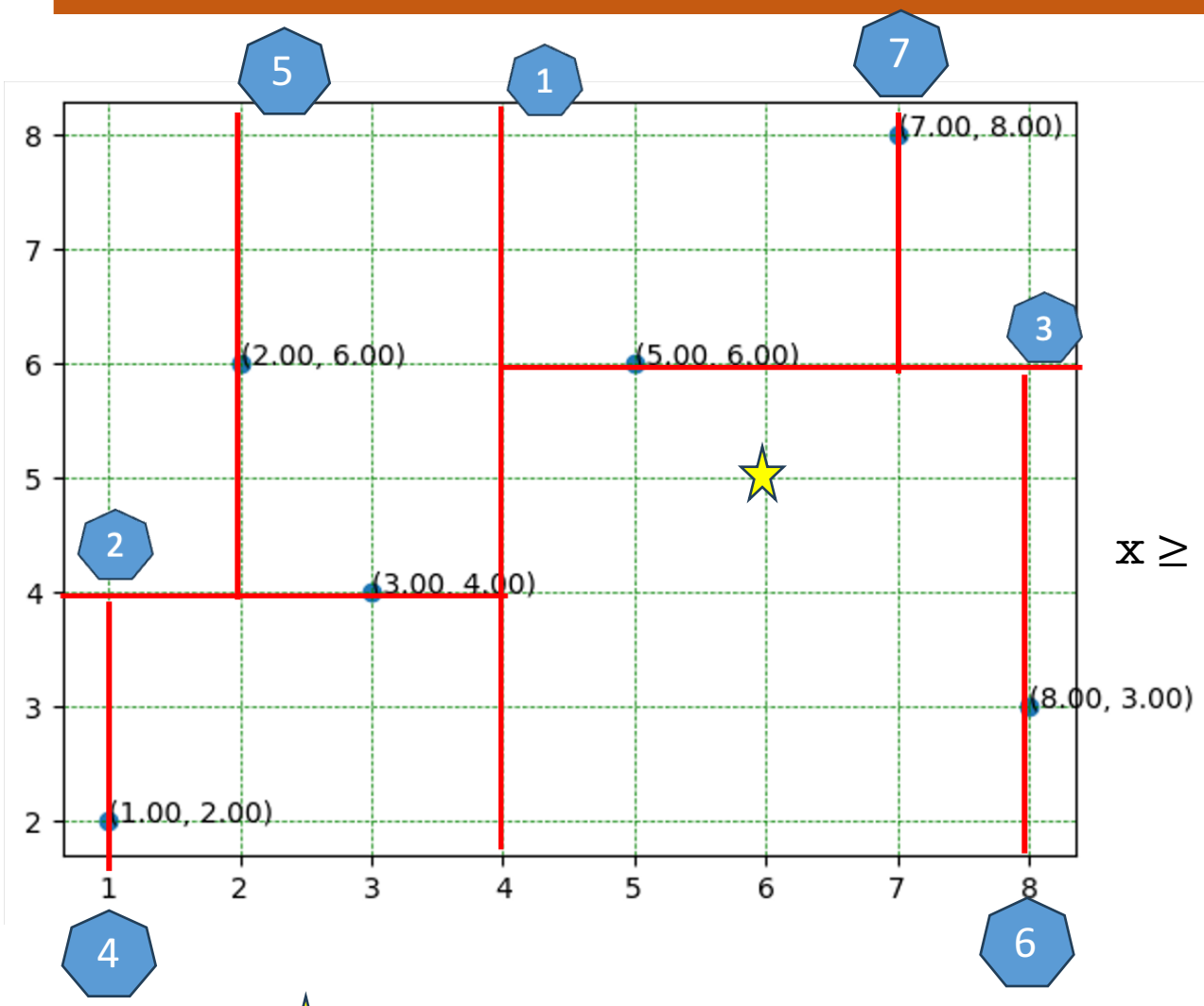


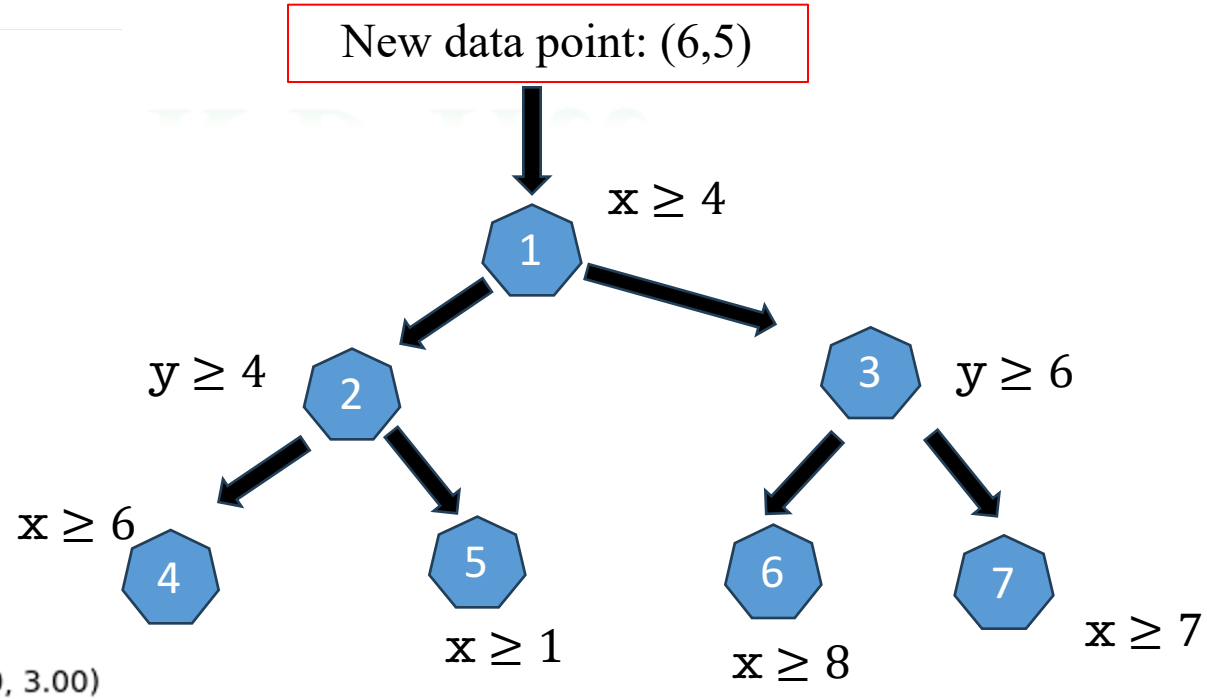
KNN with K-D Tree



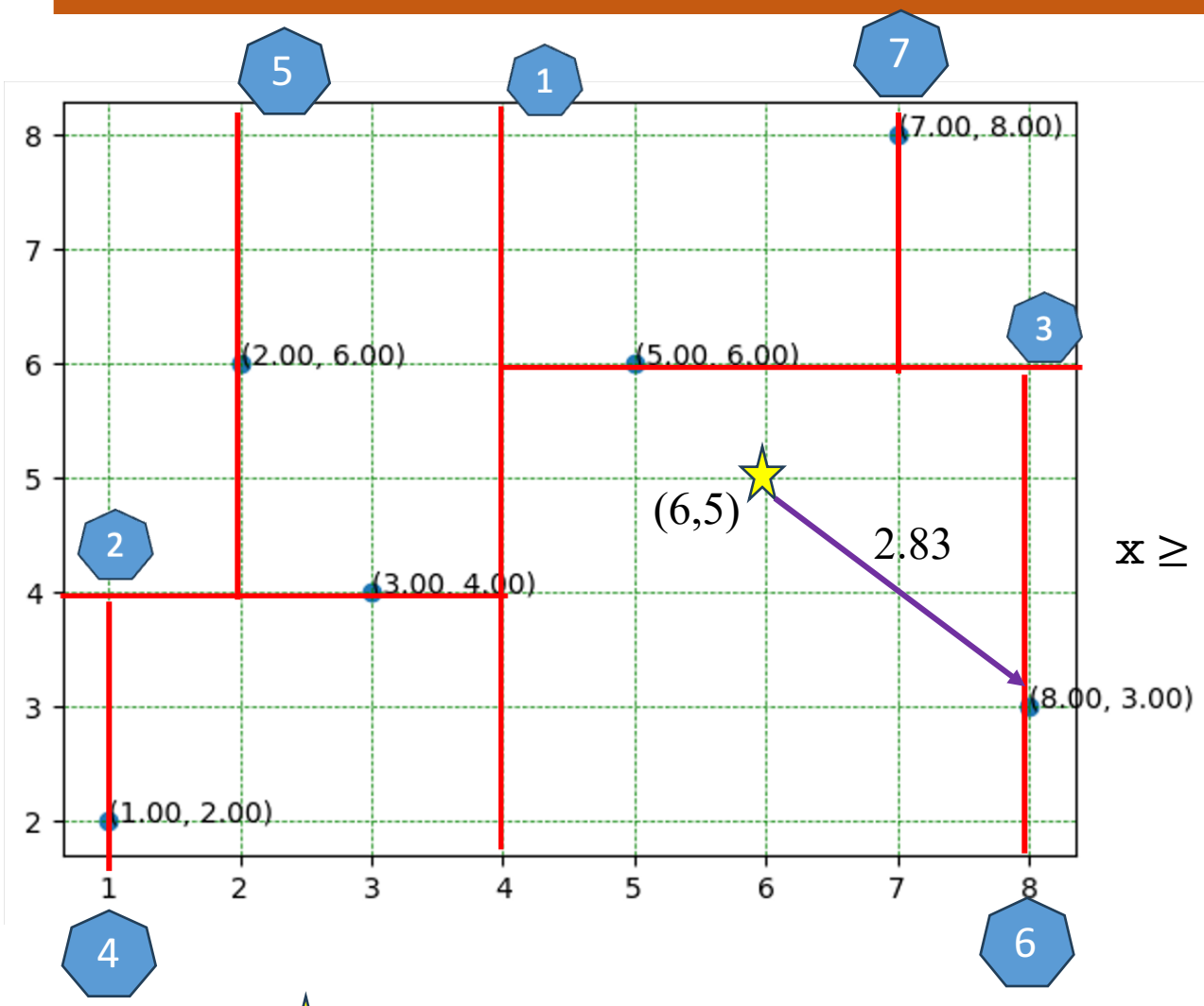


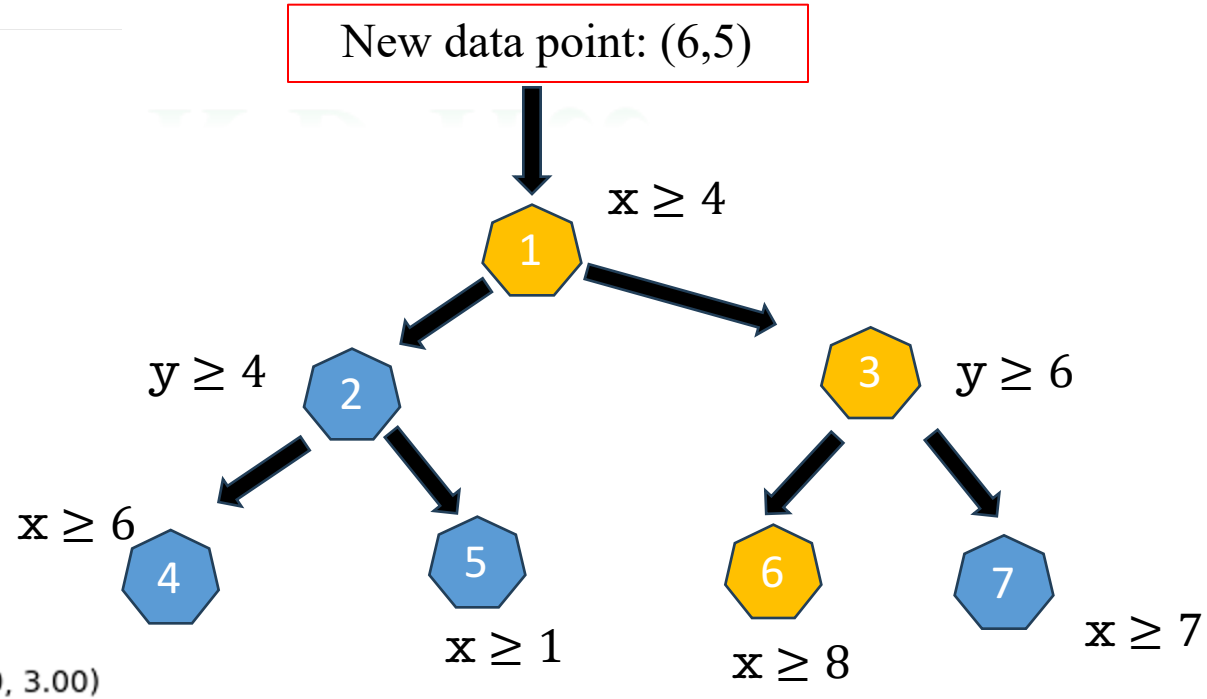
KNN with K-D Tree





KNN with K-D Tree

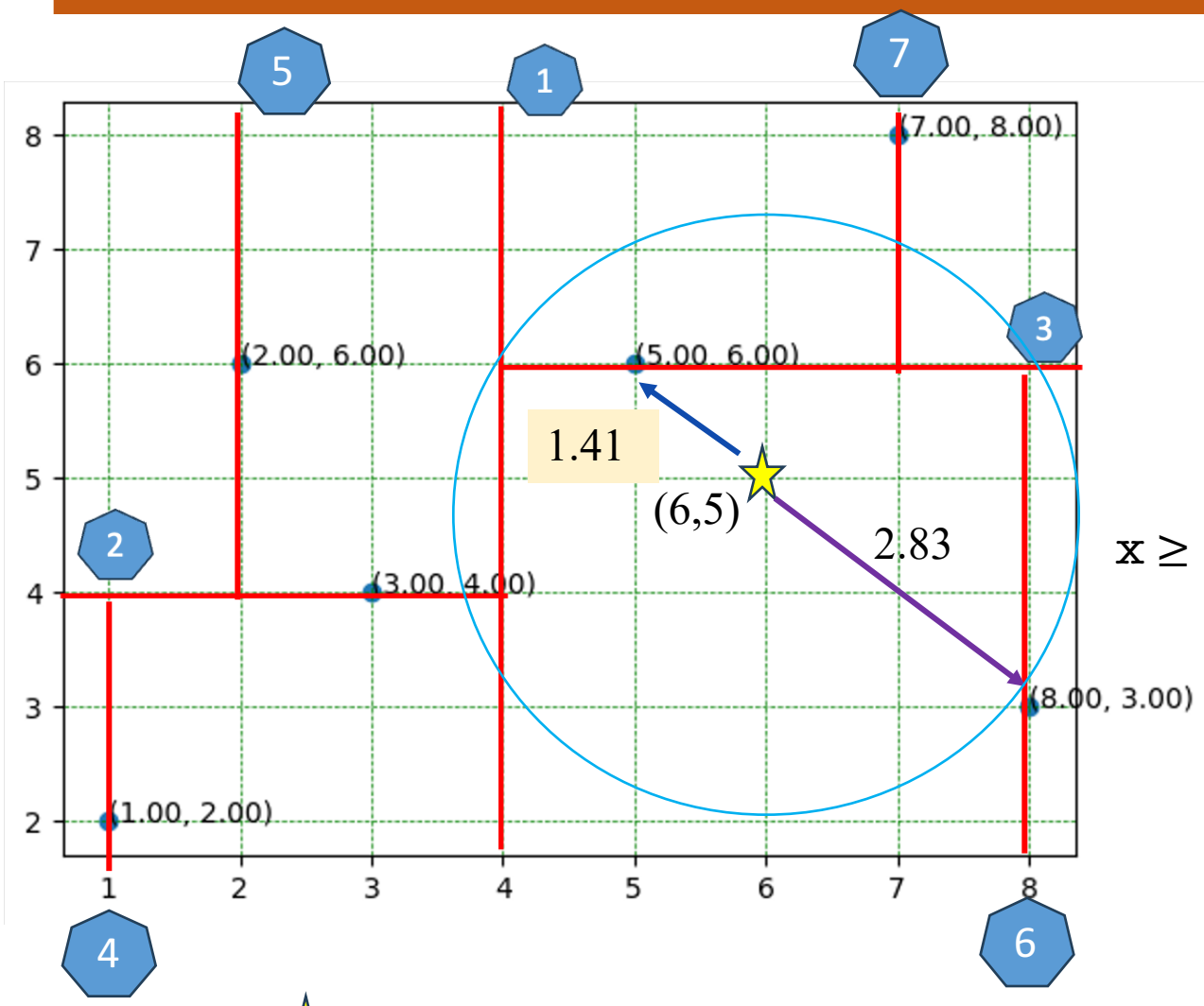




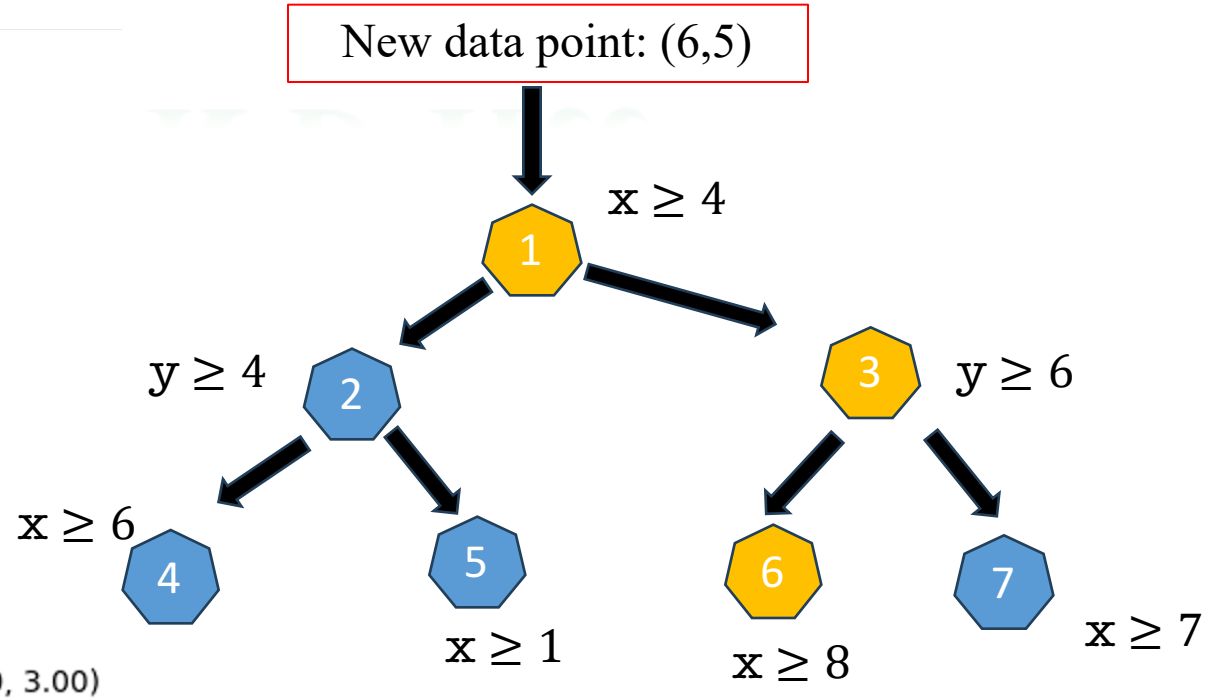
Minimum distance = 2.83

New datapoint

KNN with K-D Tree



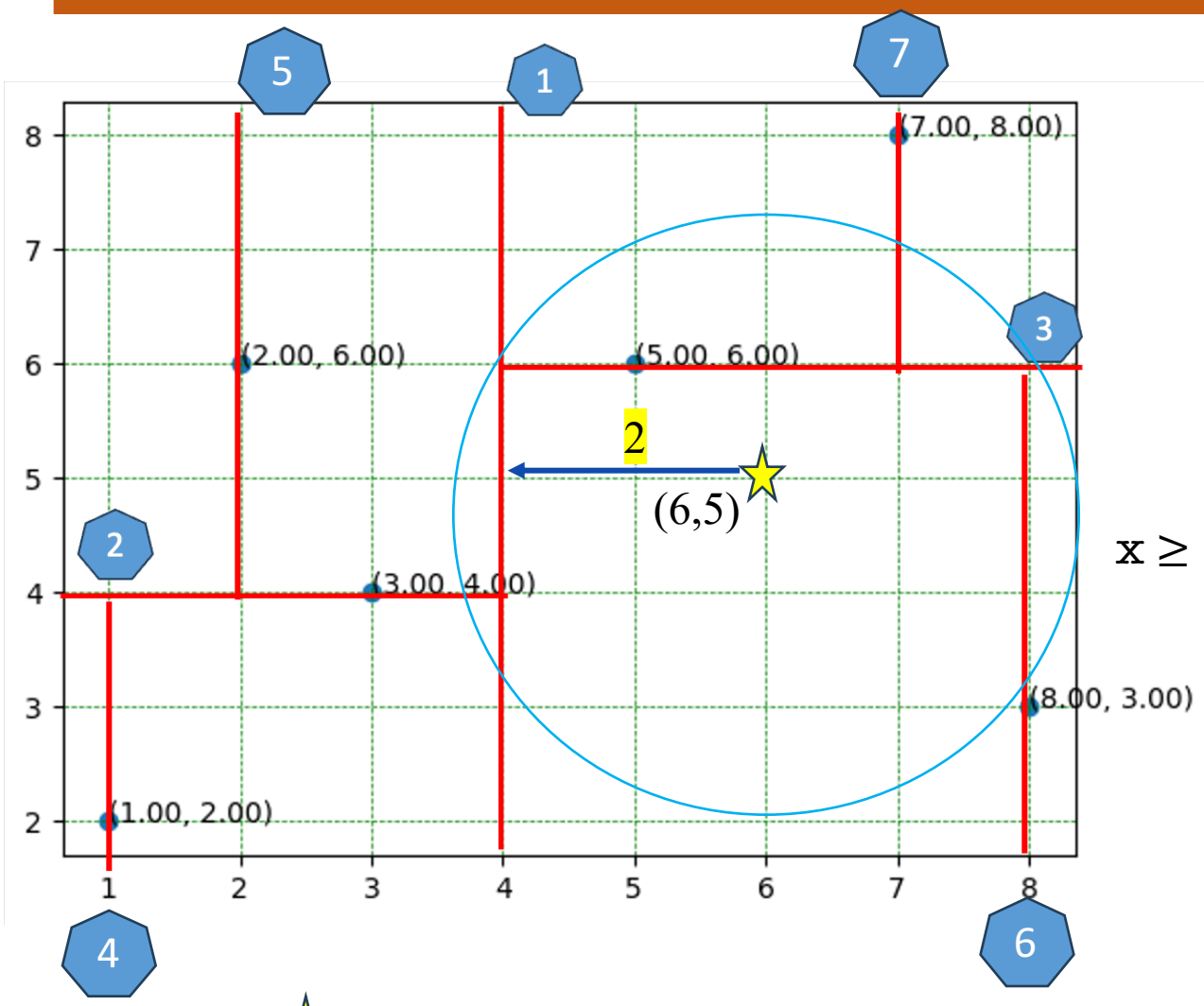
New datapoint



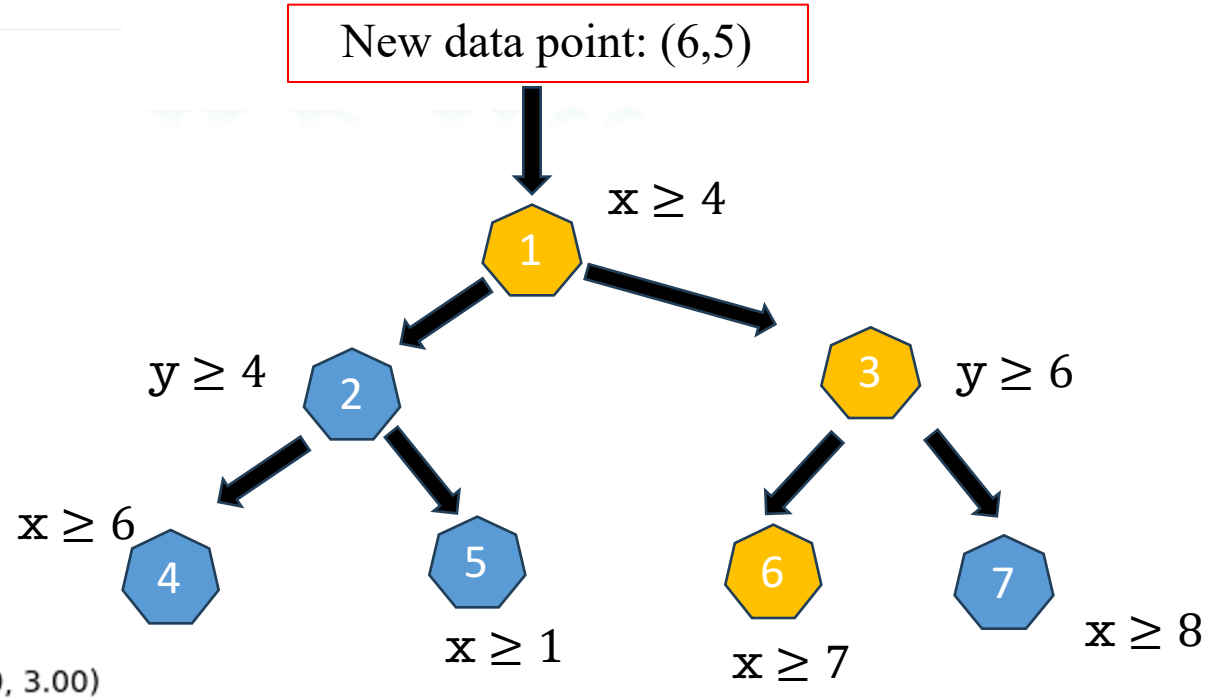
Minimum distance = 2.83

Minimum distance = 1.41

KNN with K-D Tree



New datapoint

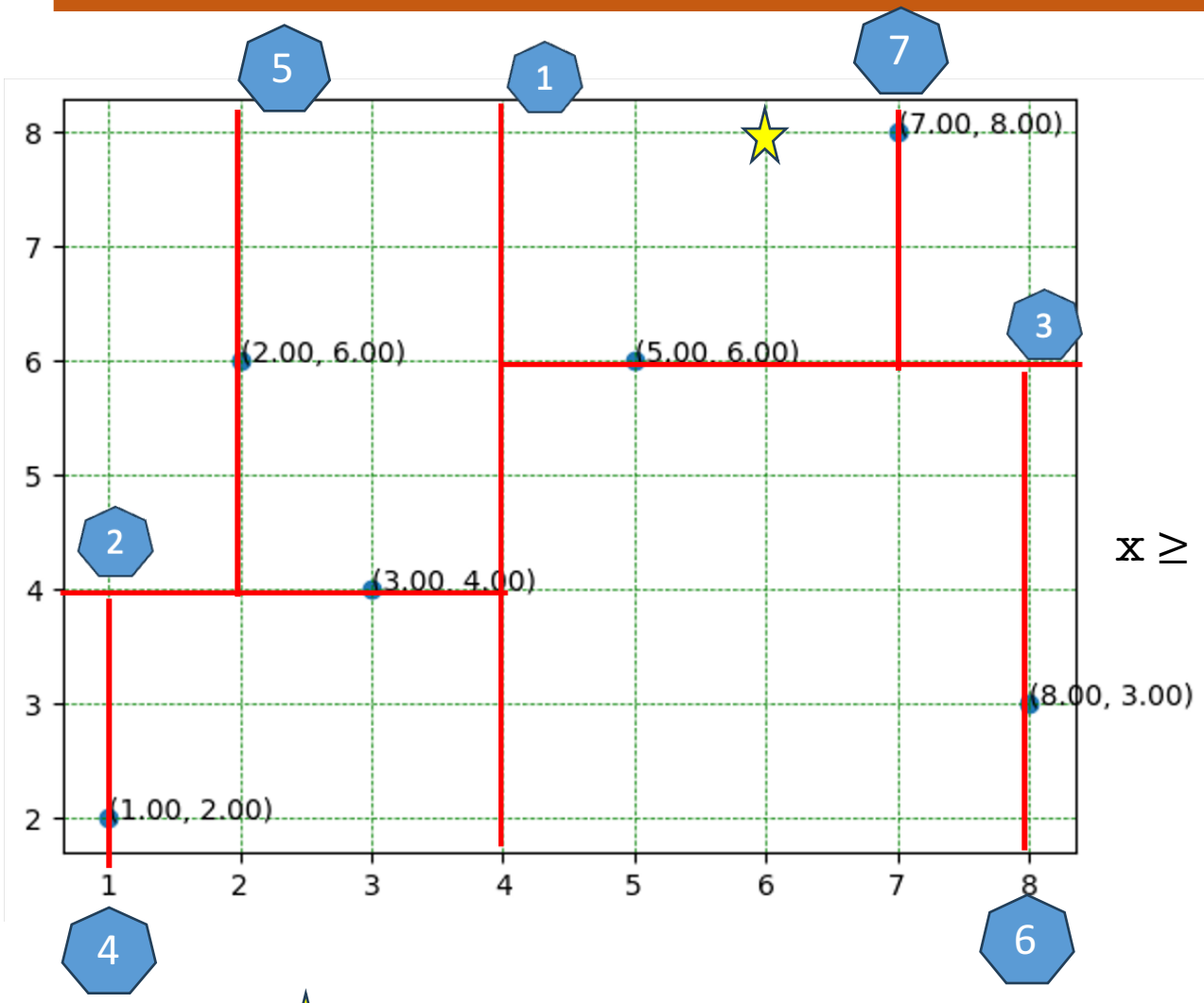


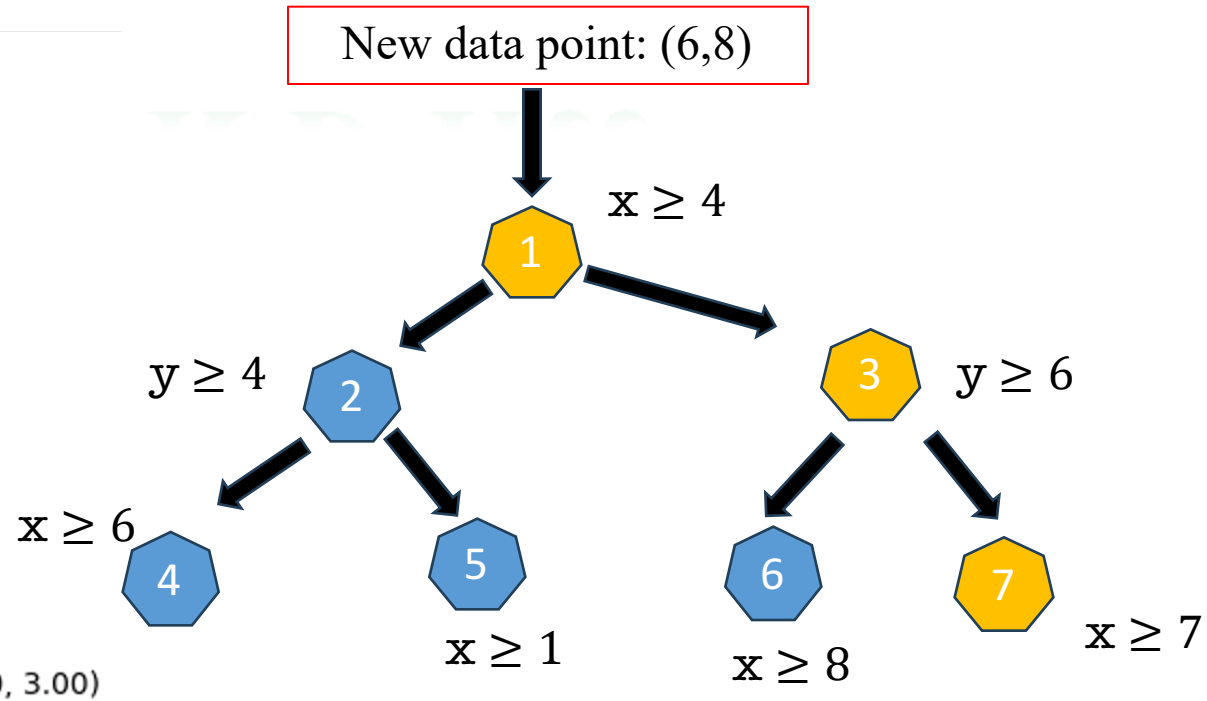
Minimum distance = 2.83

Minimum distance = 1.41

(5,6) is the nearest point

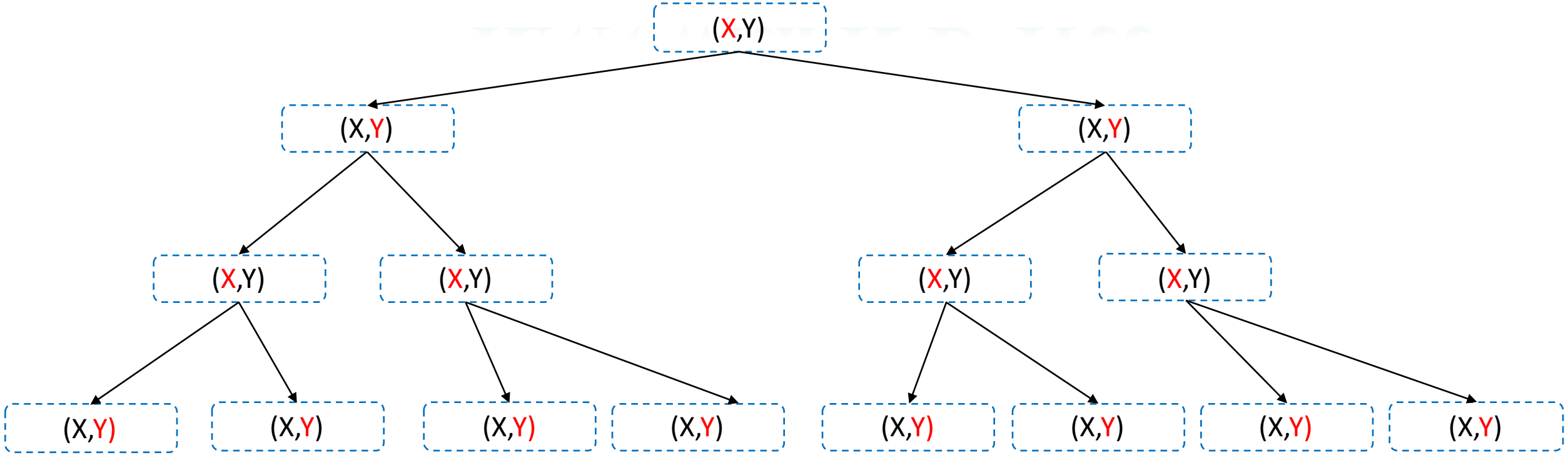
KNN with K-D Tree





What is the nearest neighbor?

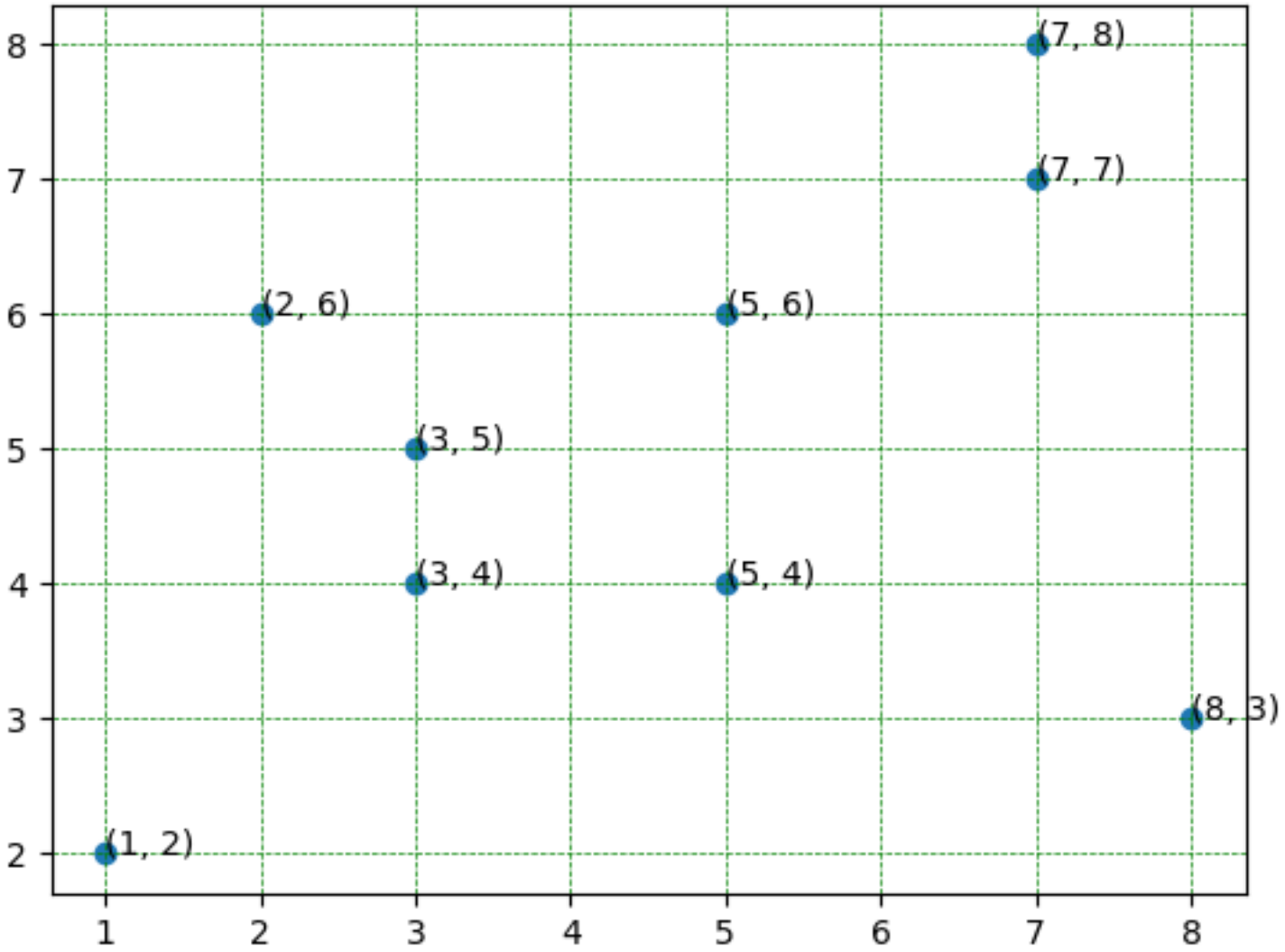
KNN with K-D Tree



Outline

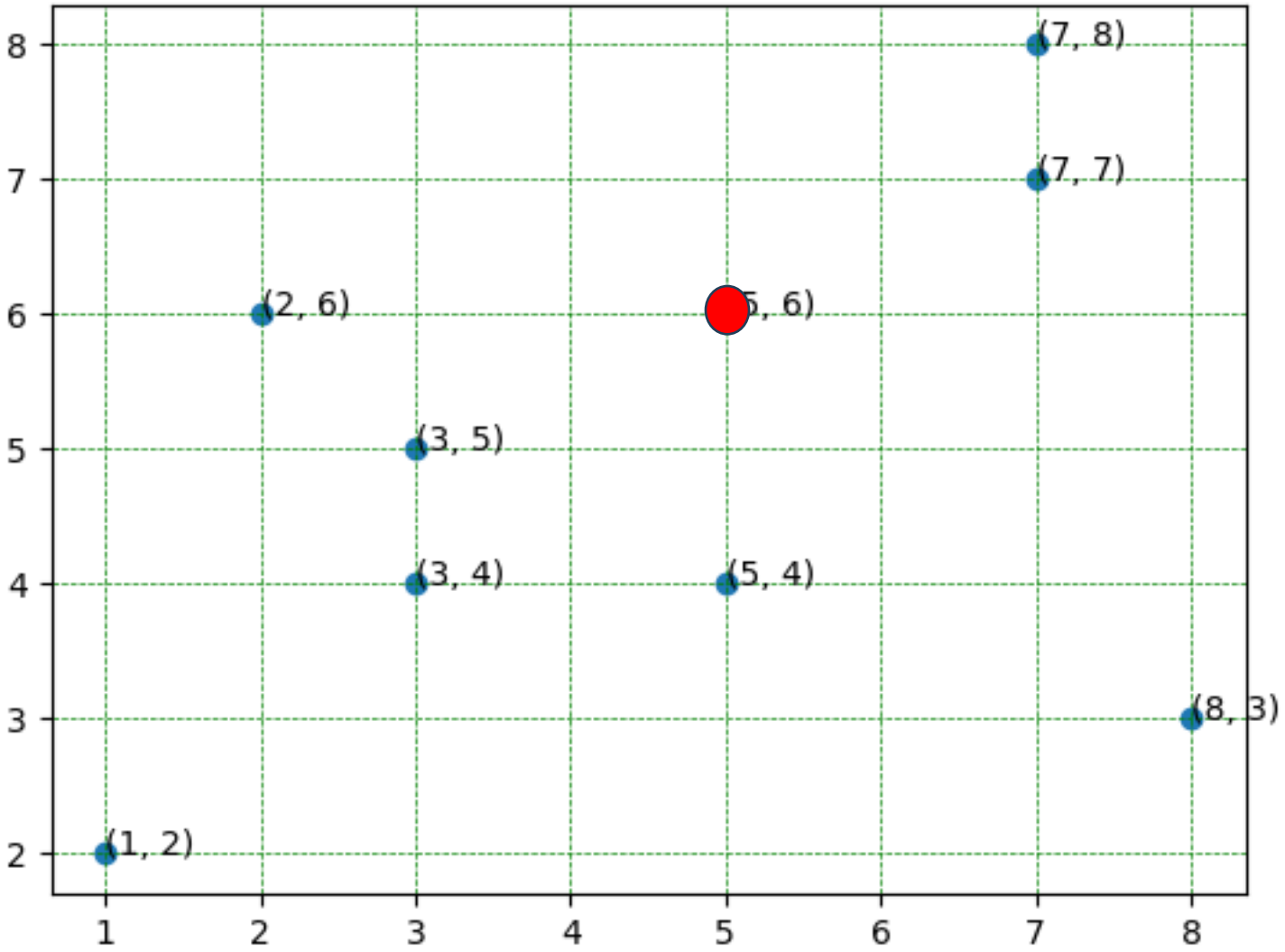
- Overview Machine Learning
- KNN Motivation
- KNN for Classification
- How to Select k in KNN
- KNN for Regression
- KNN with Brute force
- KNN with K-D Tree
- KNN with Ball Tree

KNN with Ball Tree



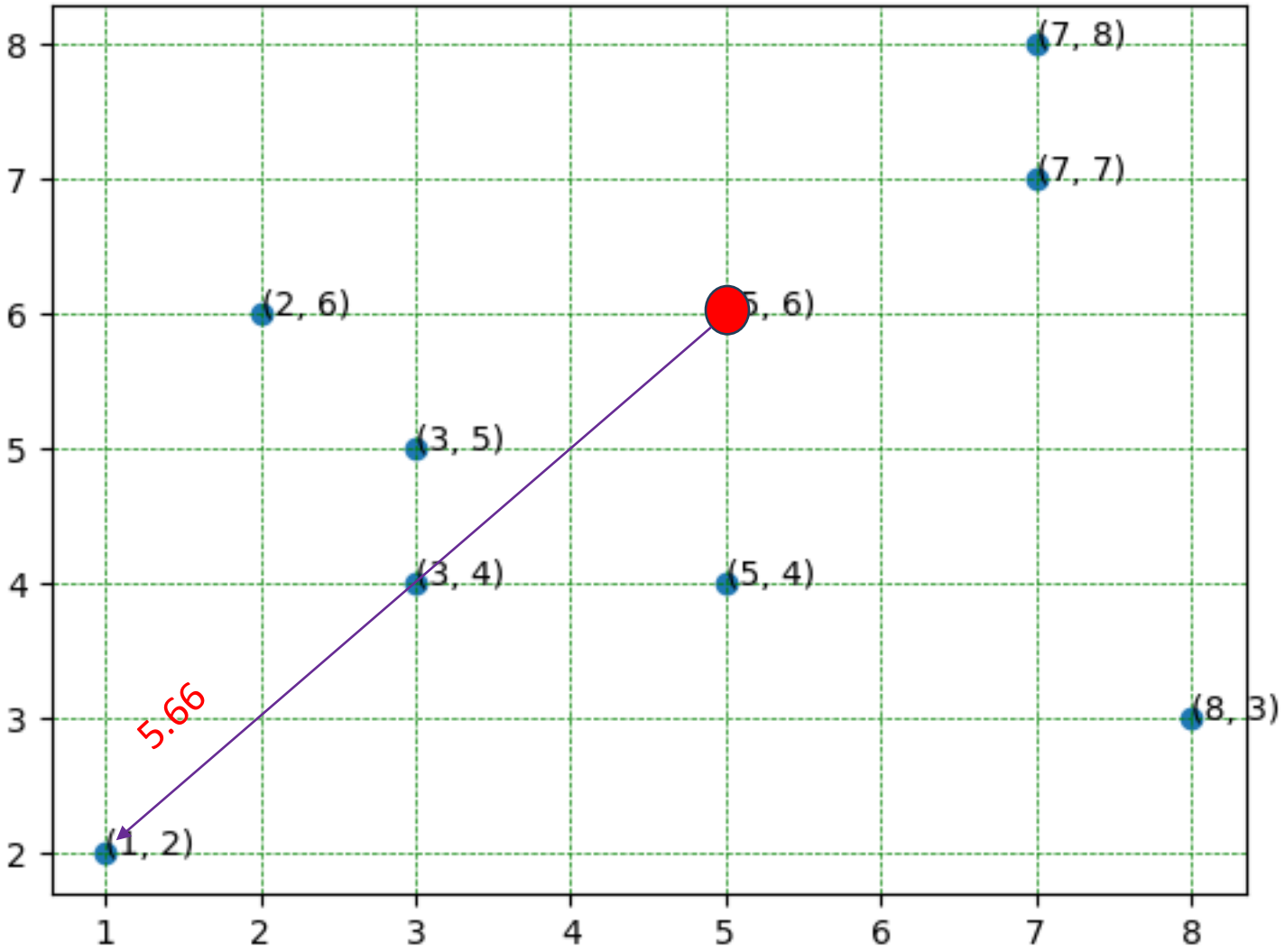
1. Take any point
2. Find a point farthest to that point
3. Find a point farthest to this fatherst point
4. Project all points on the line joining farthest points
5. Find the median to divide the space into two halves
6. Find the centroid in each half
7. Draw ball (cirlce of radius equal to the distance to the farthest point in that half)

KNN with Ball Tree



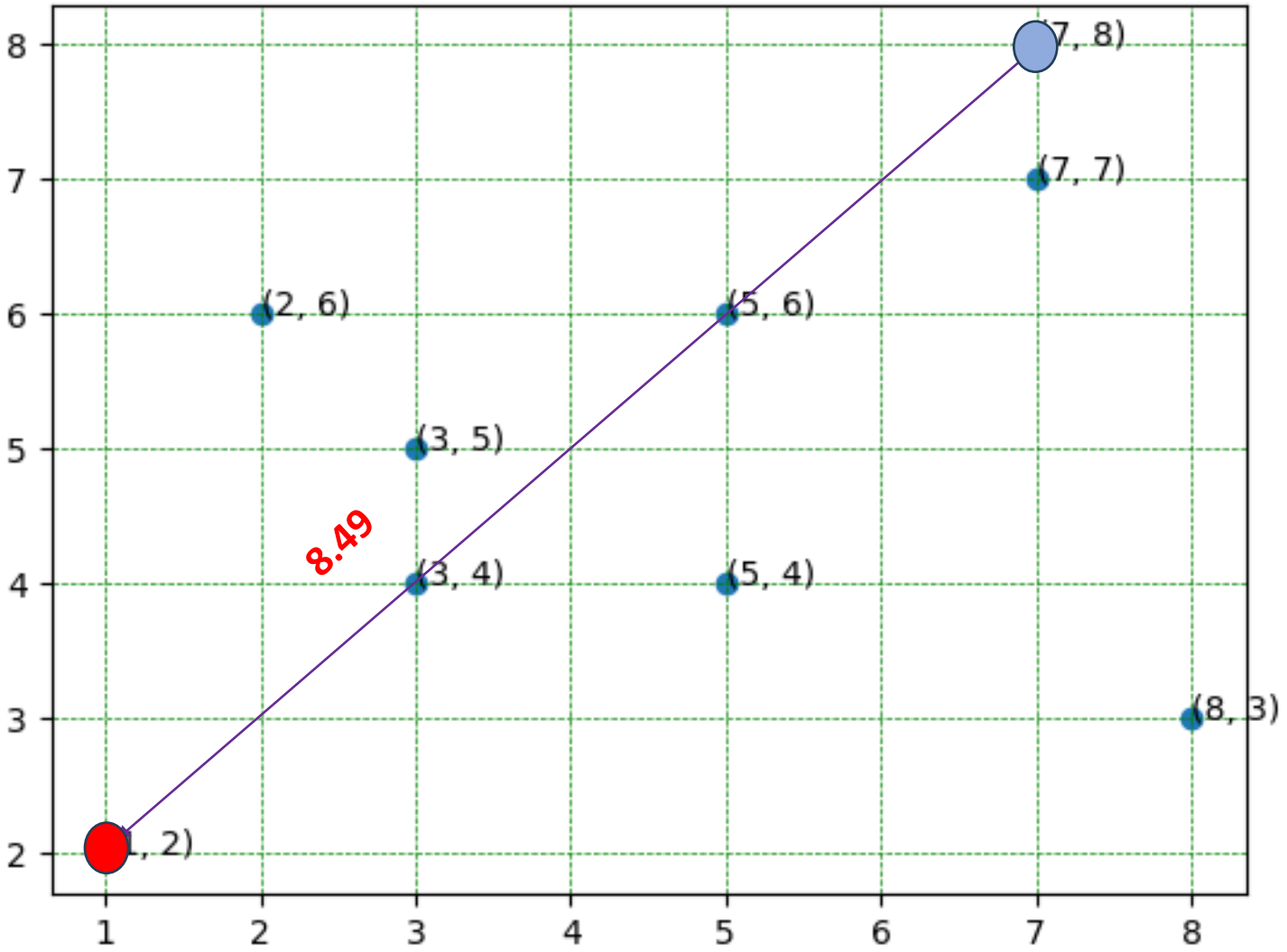
1. Take any point
2. Find a point farthest to that point
3. Find a point farthest to this fatherst point
4. Project all points on the line joining farthest points
5. Find the median to divide the space into two halves
6. Find the centroid in each half
7. Draw ball (cirlce of radius equal to the distance to the farthest point in that half)

KNN with Ball Tree



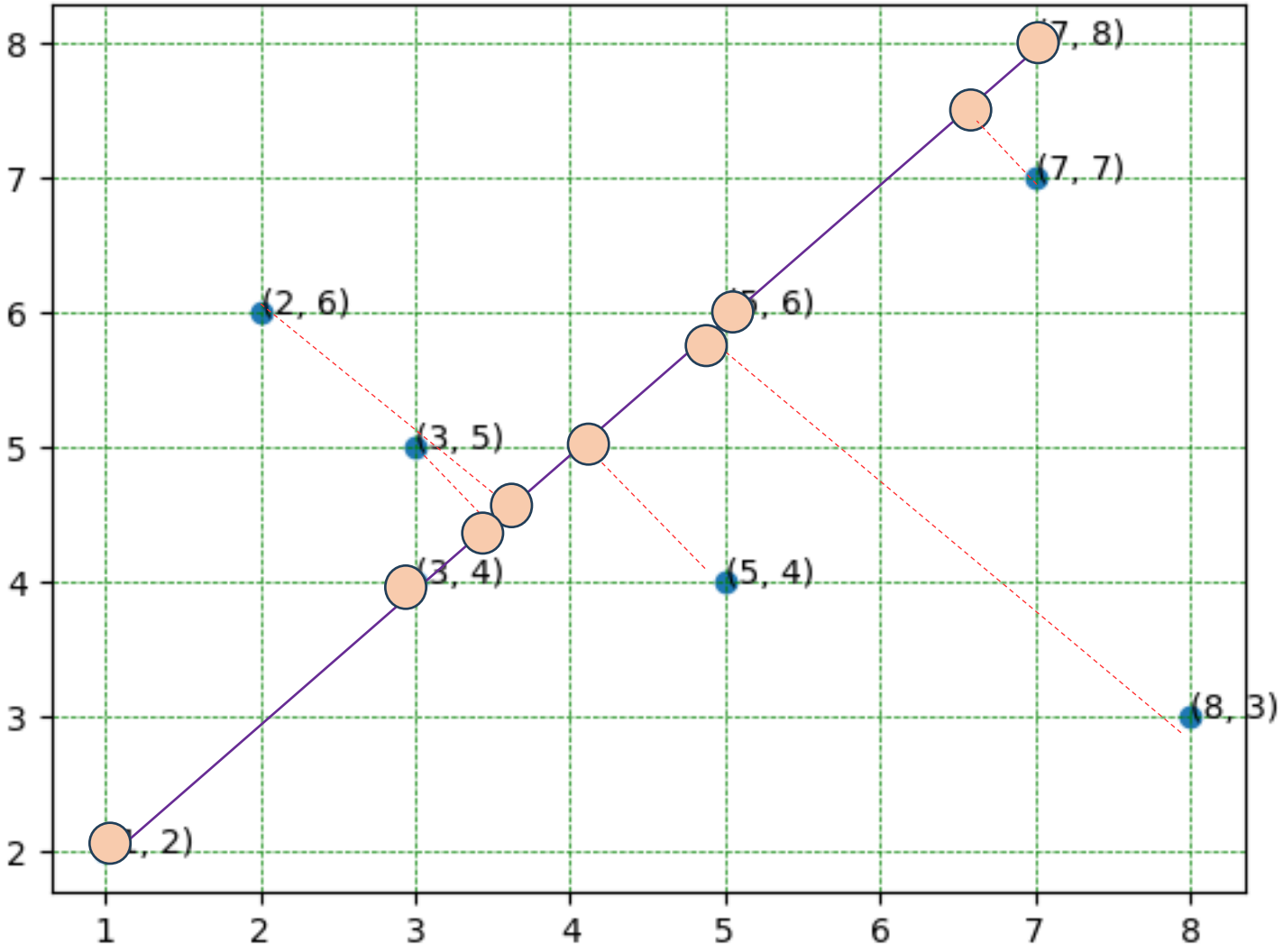
1. Take any point
2. Find a point farthest to that point
3. Find a point farthest to this fatherst point
4. Project all points on the line joining farthest points
5. Find the median to divide the space into two halves
6. Find the centroid in each half
7. Draw ball (cirlce of radius equal to the distance to the farthest point in that half)

KNN with Ball Tree



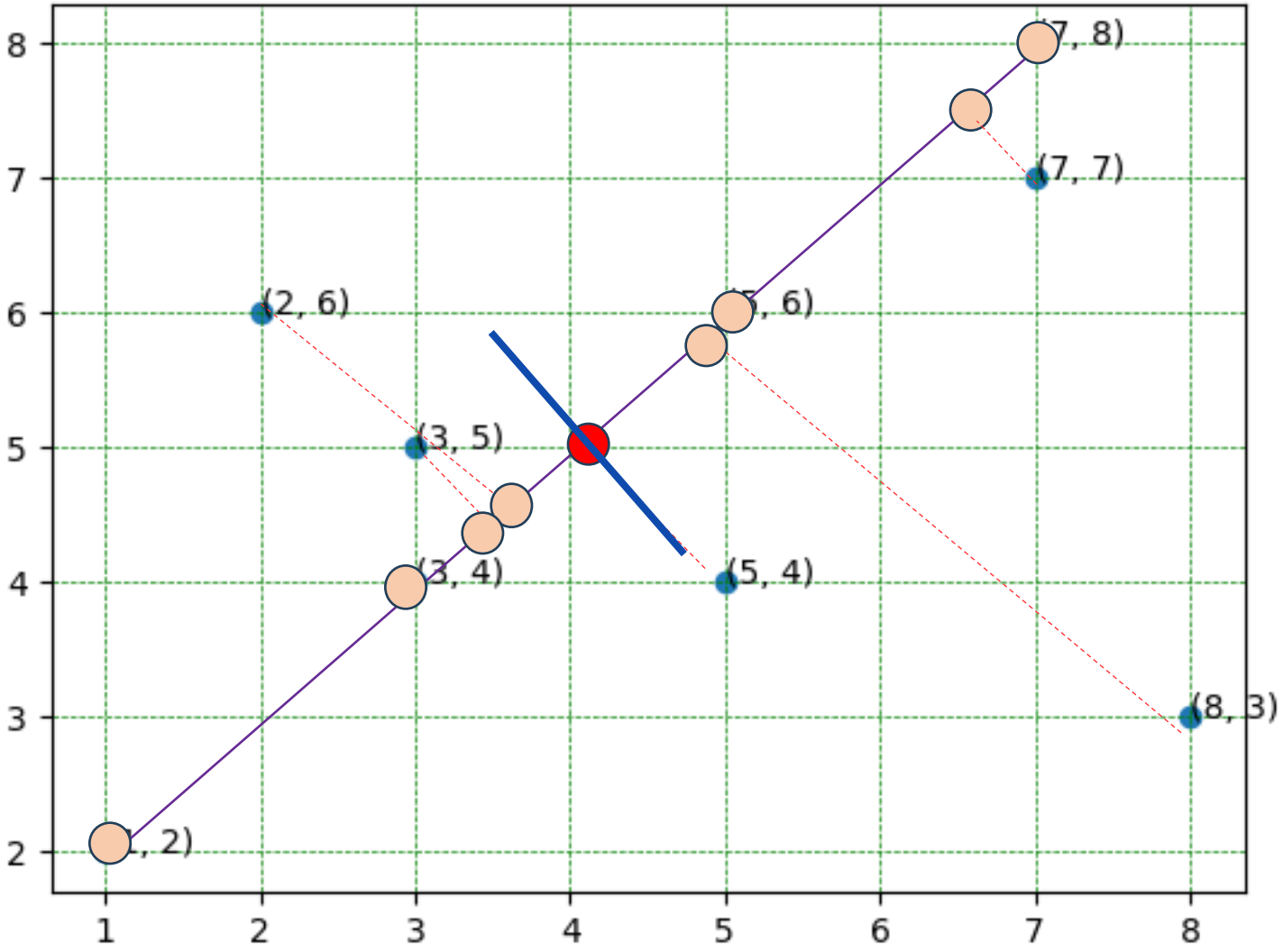
1. Take any point
2. Find a point farthest to that point
3. Find a point farthest to this farthest point
4. Project all points on the line joining farthest points
5. Find the median to divide the space into two halves
6. Find the centroid in each half
7. Draw ball (cirlce of radius equal to the distance to the farthest point in that half)

KNN with Ball Tree



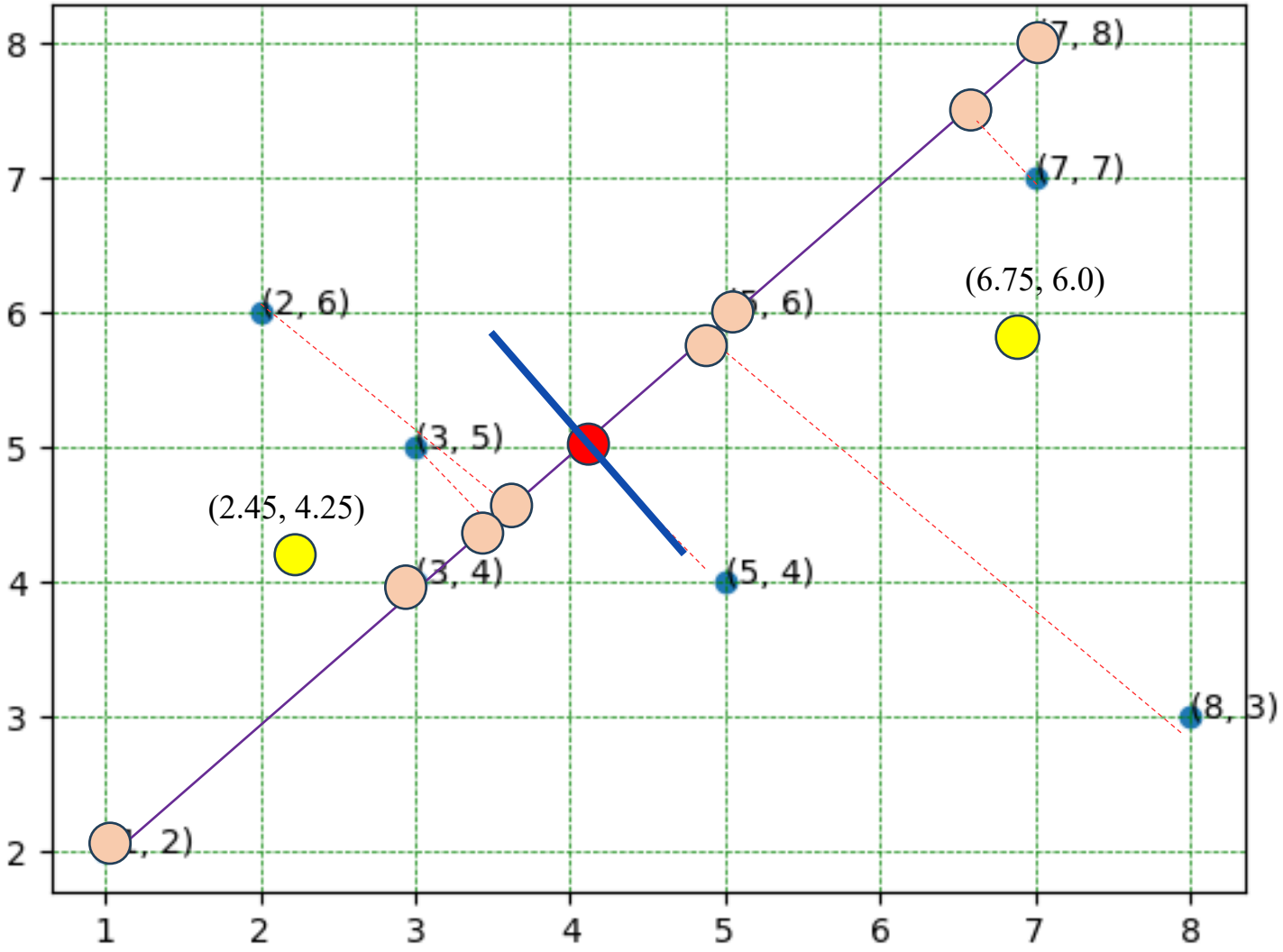
1. Take any point
2. Find a point farthest to that point
3. Find a point farthest to this farthest point
4. Project all points on the line joining farthest points
5. Find the median to divide the space into two halves
6. Find the centroid in each half
7. Draw ball (cirlce of radius equal to the distance to the farthest point in that half)

KNN with Ball Tree



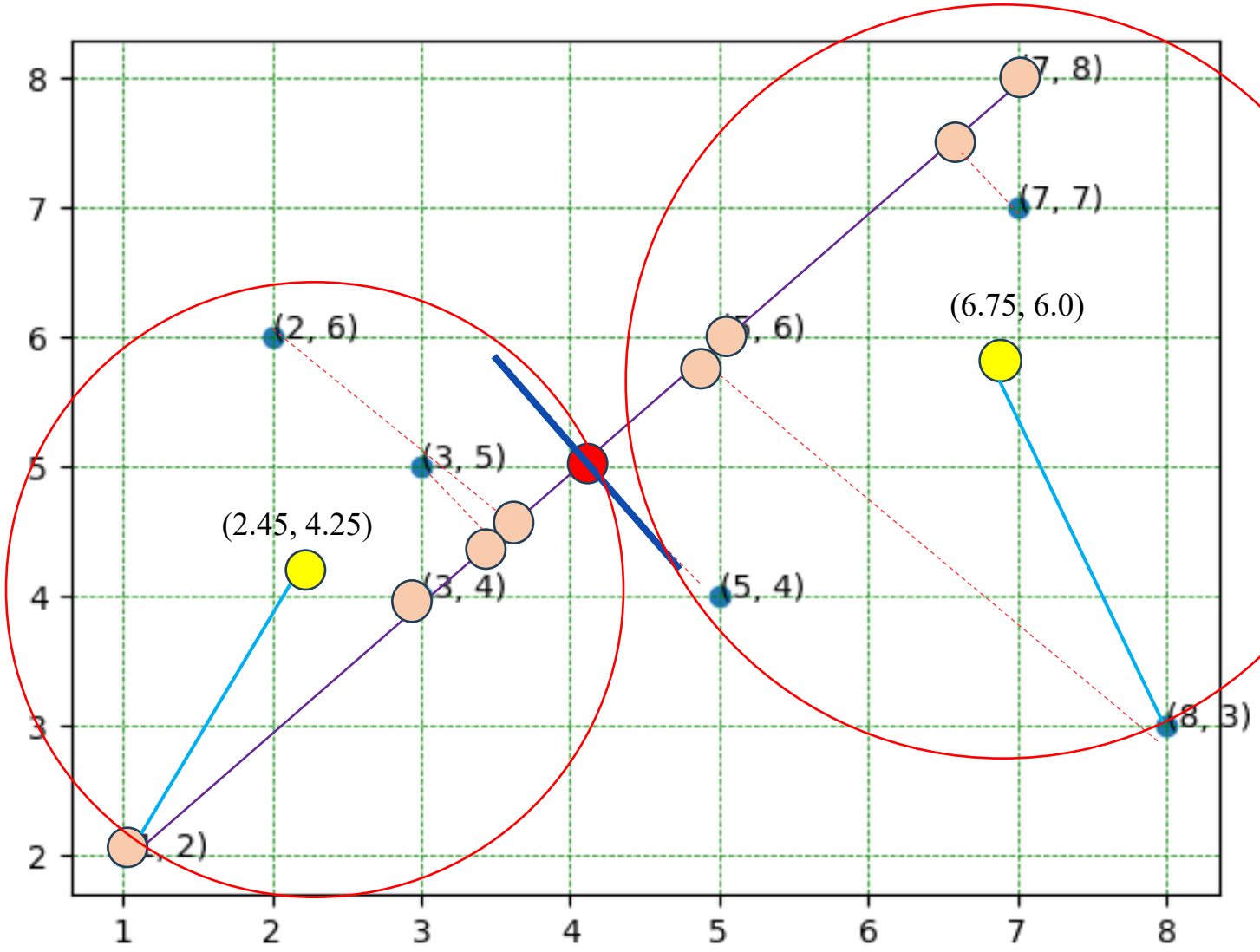
1. Take any point
2. Find a point farthest to that point
3. Find a point farthest to this farthest point
4. Project all points on the line joining farthest points
5. Find the median to divide the space into two halves
6. Find the centroid in each half
7. Draw ball (cirlce of radius equal to the distance to the farthest point in that half)

KNN with Ball Tree



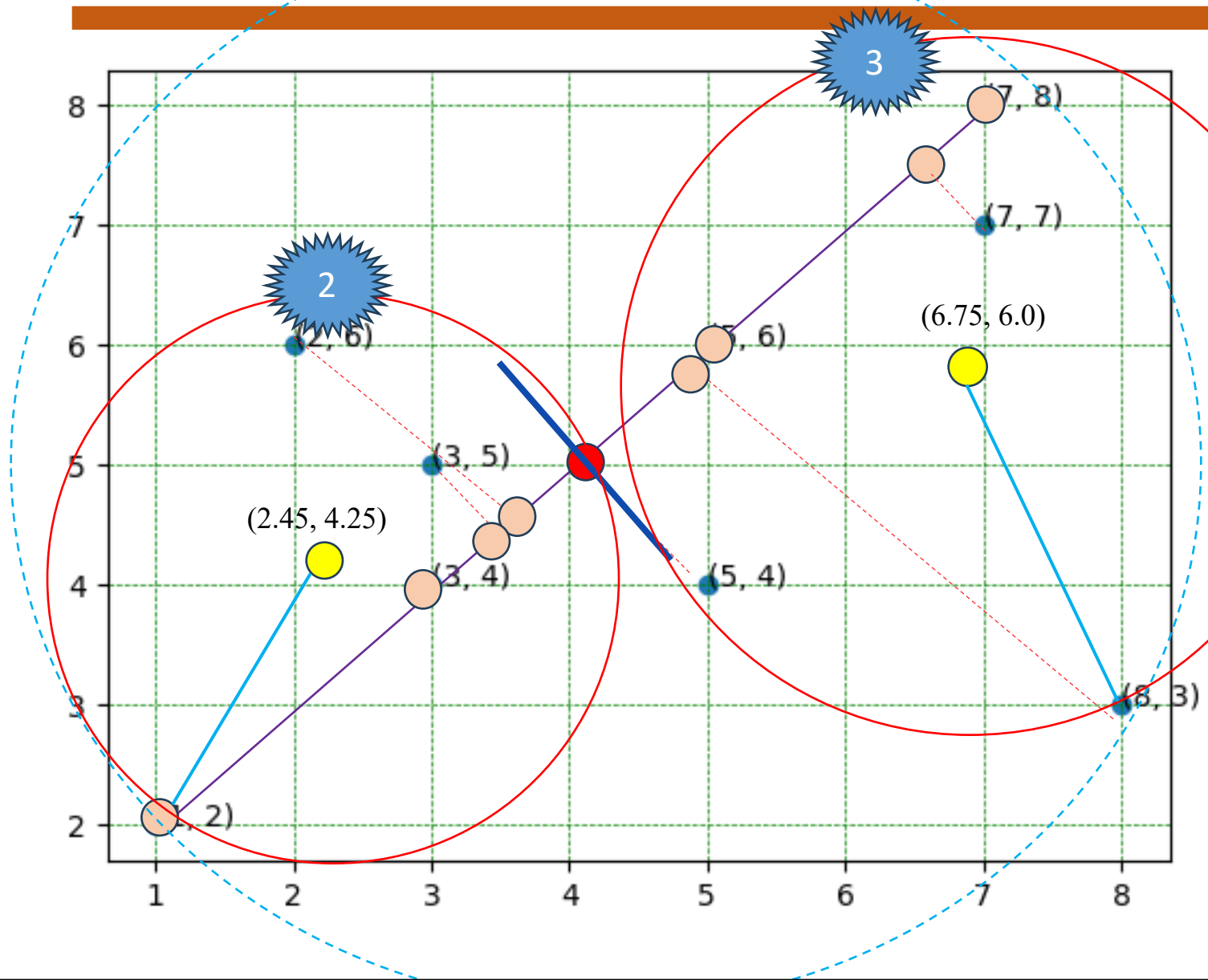
1. Take any point
2. Find a point farthest to that point
3. Find a point farthest to this farthest point
4. Project all points on the line joining farthest points
5. Find the median to divide the space into two halves
6. Find the centroid in each half
7. Draw ball (cirlce of radius equal to the distance to the farthest point in that half)

KNN with Ball Tree

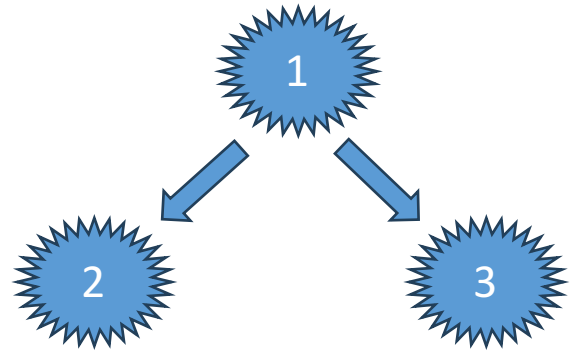


1. Take any point
2. Find a point farthest to that point
3. Find a point farthest to this farthest point
4. Project all points on the line joining farthest points
5. Find the median to divide the space into two halves
6. Find the centroid in each half
7. Draw ball (cirlce of radius equal to the distance to the farthest point in that half)

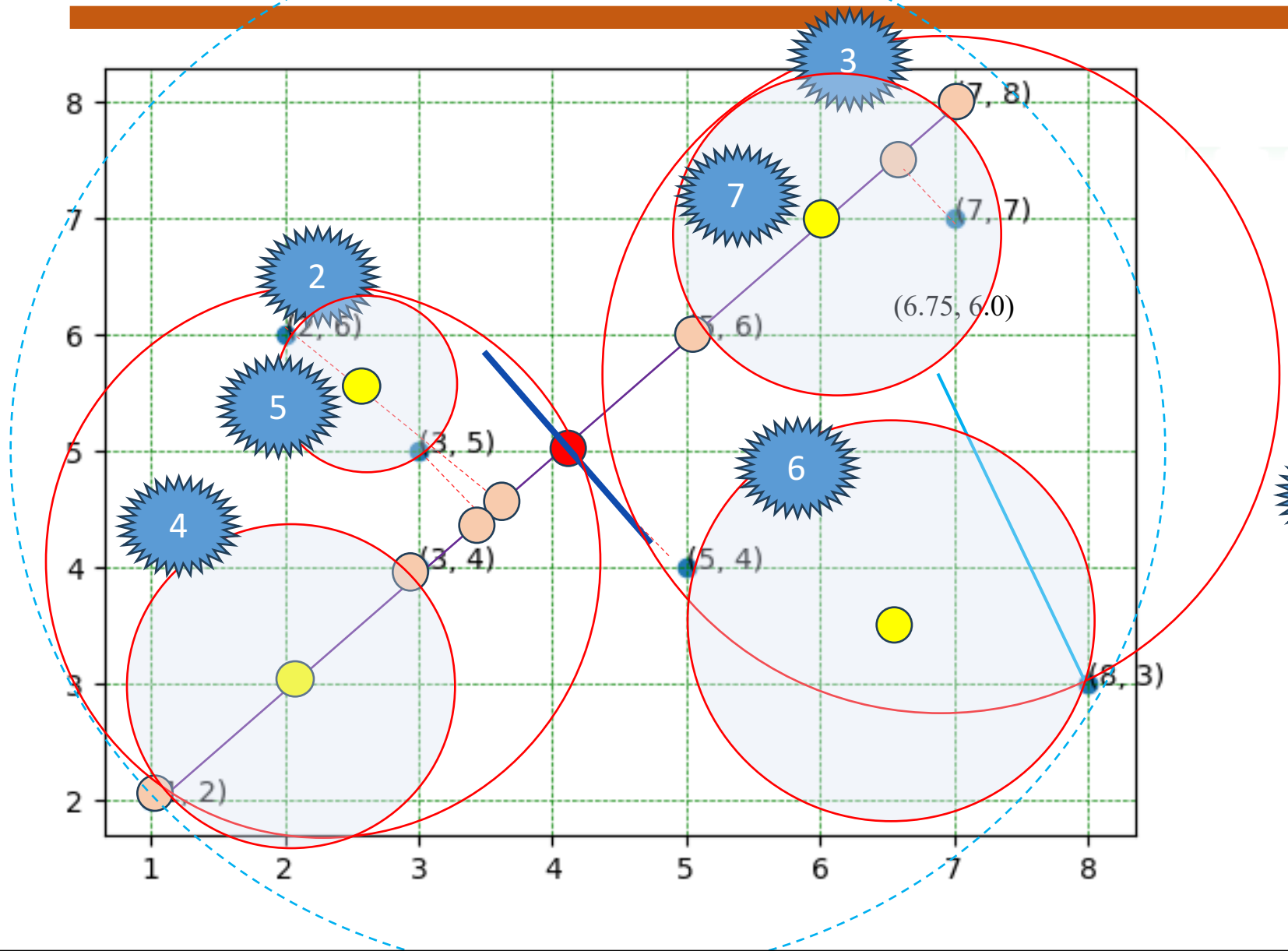
KNN with Ball Tree

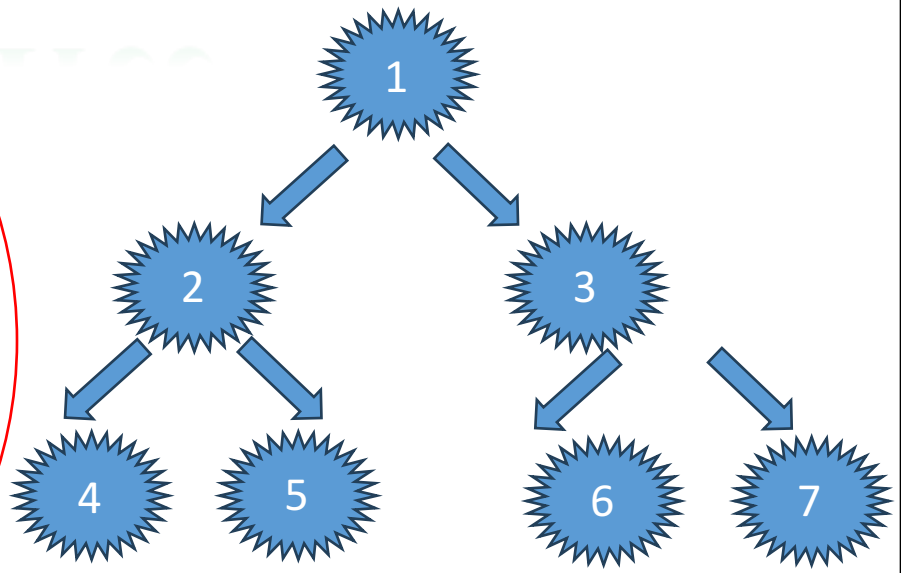


1. Take any point
2. Find a point farthest to that point
3. Find a point farthest to this farthest point
4. Project all points on the line joining farthest points
5. Find the median to divide the space into two halves
6. Find the centroid in each half
7. Draw ball (cirlce of radius equal to the distance to the farthest point in that half)

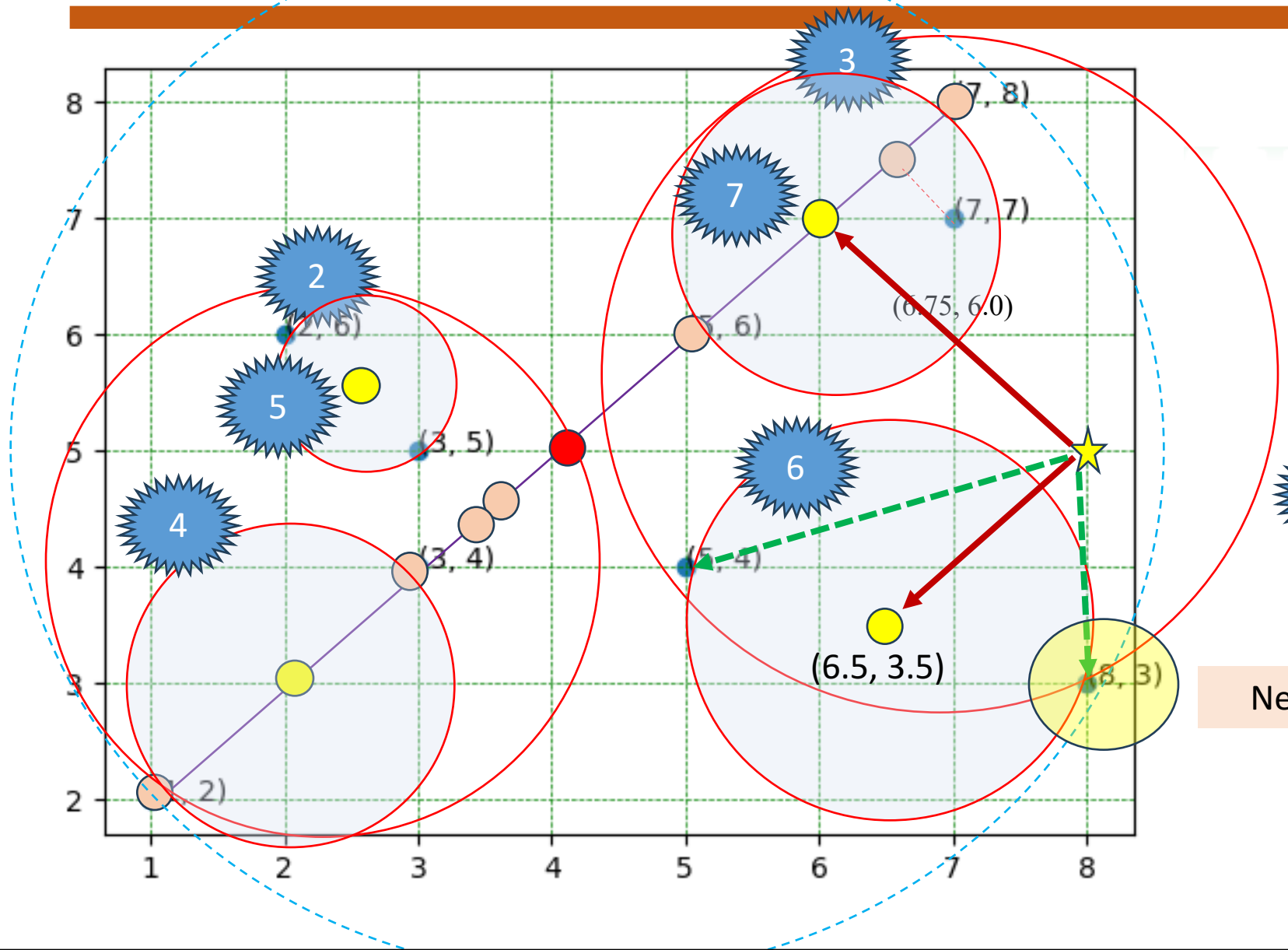


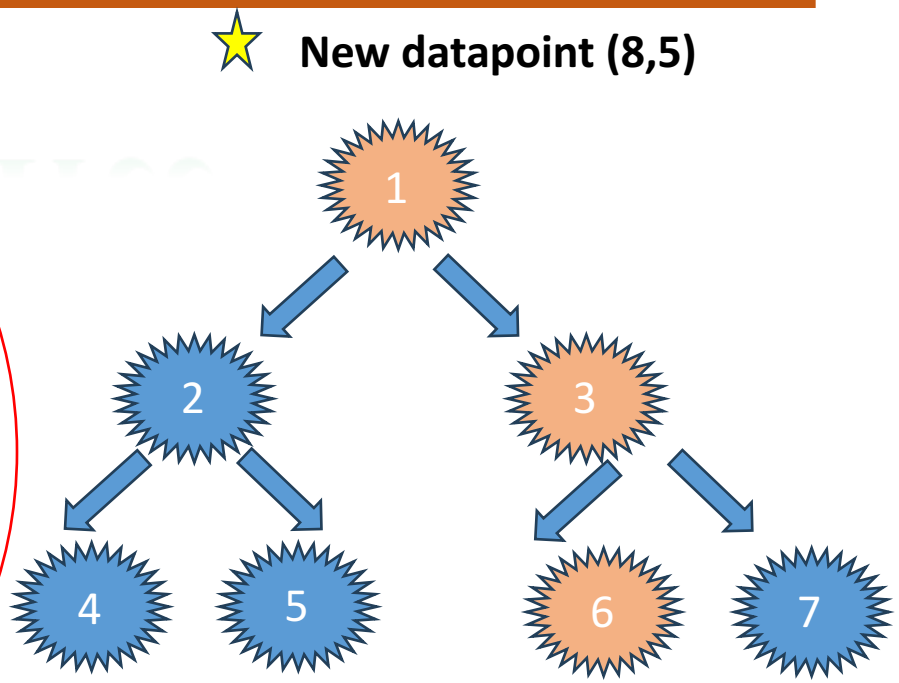
KNN with Ball Tree





KNN with Ball Tree





Nearest neighbor is (8,3)


Thank
you