KNN Algorithm

1. To classify document d into class c

2. Define k-neighborhood N as k nearest neighbors (according to a given distance or similarity measure) of d

3. Count number of documents kc in N that belong to c

4. Estimate P(c|d) as kc/k

5. Choose as class argmaxc P(c|d) [= majority class]

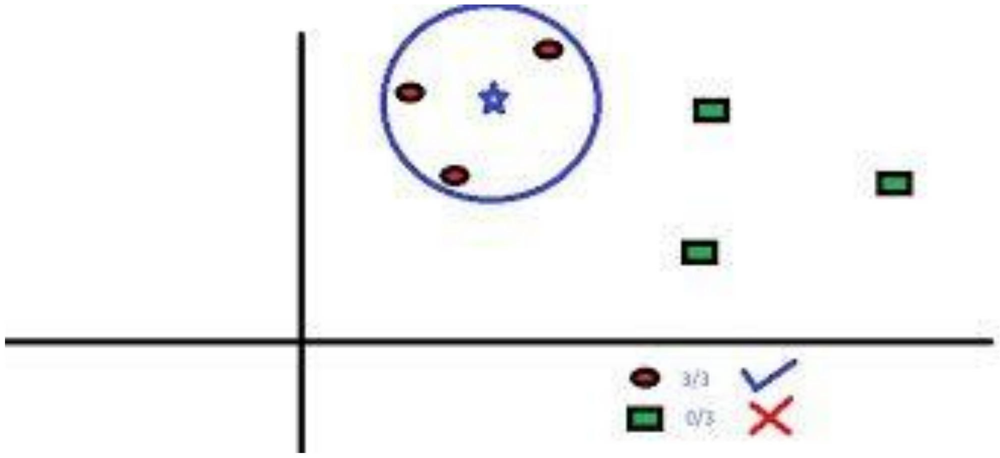
Finding similar rows

Distance functions

Euclidean $$\sqrt{\sum_{i=1}^{k}(x_i - y_i)^2}$$

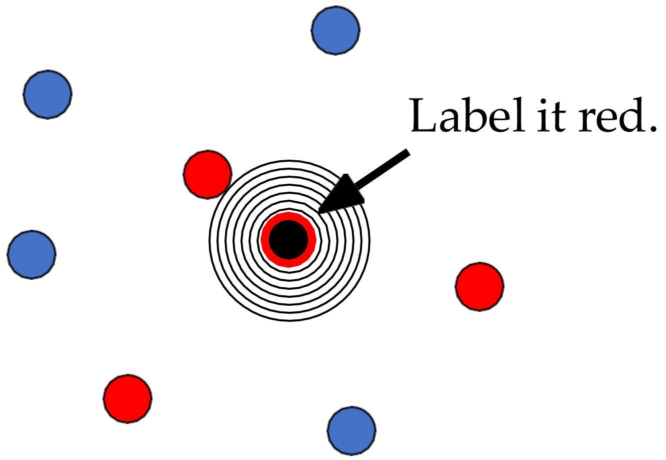
Manhattan $$\sum_{i=1}^{k}|x_i - y_i|$$

Minkowski $$\left(\sum_{i=1}^{k}(|x_i - y_i|)^q\right)^{1/q}$$



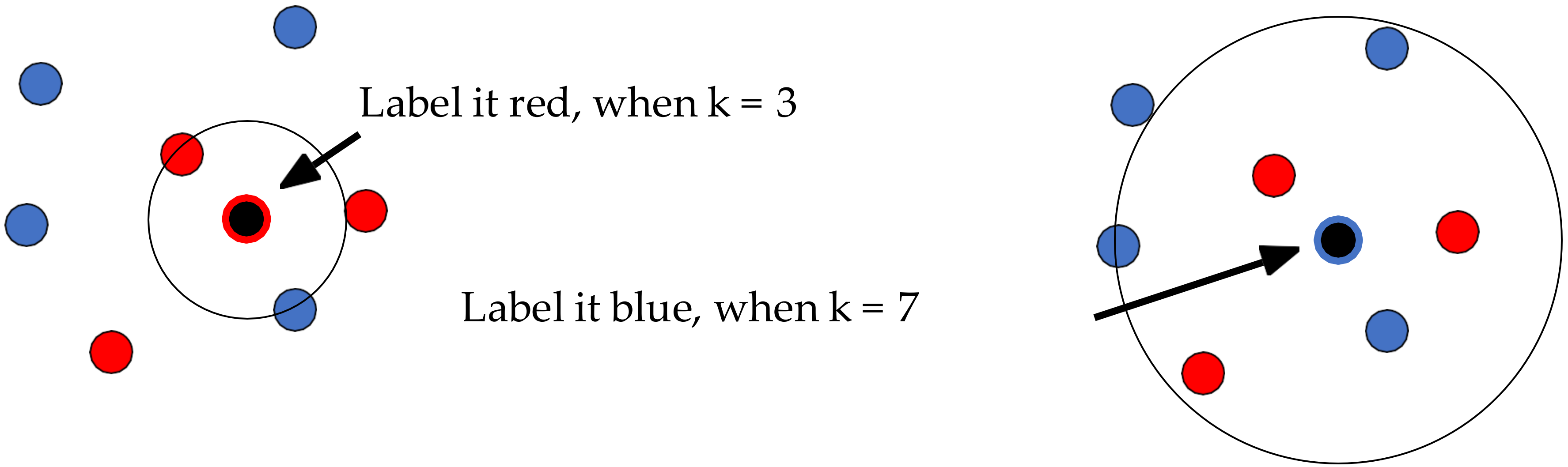
1-Nearest Neighbor

- One of the simplest of all machine learning classifiers
- Simple idea: label a new point the same as the closest known point



k – Nearest Neighbor

- Generalizes 1-NN to smooth away noise in the labels
- A new point is now assigned the most frequent label of its *k* nearest neighbors



KNN Example

	Food (3)	Chat (2)	Fast (2)	Price (3)	Bar (2)	BigTip
1	great	yes	yes	normal	no	yes
2	great	no	yes	normal	no	yes
3	mediocre	yes	no	high	no	no
4	great	yes	yes	normal	yes	yes

Similarity metric: Number of matching attributes (k=2)

- New examples:
  - Example 1 (great, no, no, normal, no) Yes
    - □ most similar: number 2 (1 mismatch, 4 match) □ yes
    - □Second most similar example: number 1 (2 mismatch, 3 match) □ yes
  - Example 2 (mediocre, yes, no, normal, no) Yes/No
    - □ Most similar: number 3 (1 mismatch, 4 match) □ no
    - □Second most similar example: number 1 (2 mismatch, 3 match) □ yes

We have data from survey (to ask people opinion) and objective testing with two attributes(acid durability and strength) to classify whether a special paper tissue is good or not. Here is four training samples

X1(Acid) in seconds	X2(Strength) in kg/square meter	Y = Classification
7	7	Bad
7	4	Bad
3	4	Good
1	4	Good

We have data from survey (to ask people opinion) and objective testing with two attributes(acid durability and strength) to classify whether a special paper tissue is good or not. Here is four training samples

X1(Acid) in seconds	X2(Strength) in kg/square meter	Y = Classification
7	7	Bad
7	4	Bad
3	4	Good
1	4	Good

Now the factory produces a new paper tissue that pass laboratory test with X1 = 3 and X2 = 7.

Without another expensive survey, can we guess what the classification of this new tissue is?

Step 1: Determine Parameter K = number of nearest neighbours. Suppose use k = 3

Step 1: Determine Parameter K = number of nearest neighbours. Suppose use k = 3

Step 2: Calculate the distance between the query-instance and all the training samples Coordinate of query instance is (3,7), instead of calculating the distance we compute square distance which is faster to calculate(without square root)

Step 1: Determine Parameter K = number of nearest neighbours. Suppose use k = 3

Step 2: Calculate the distance between the query-instance and all the training samples
Coordinate of query instance is (3,7), instead of calculating the distance we compute square distance which is faster to calculate(without square root)

X1(Acid) in seconds	X2(Strength) in kg/square meter	Square Distance to query instance(3,7)
7	7	$(7-3)^2 + (7-7)^2 = 16$
7	4	$(7-3)^2 + (4-7)^2 = 25$
3	4	$(3-3)^2 + (4-7)^2 = 9$
1	4	$(1-3)^2 + (4-7)^2 = 13$

Step 1: Determine Parameter K = number of nearest neighbours. Suppose use k = 3

Step 2: Calculate the distance between the query-instance and all the training samples Coordinate of query instance is (3,7), instead of calculating the distance we compute square distance which is faster to calculate(without square root)

Step 3 : Sort the distance and determine nearest neighbours based on the K-th minimum distance

Step 1: Determine Parameter K = number of nearest neighbours. Suppose use k = 3

Step 2: Calculate the distance between the query-instance and all the training samples Coordinate of query instance is (3,7), instead of calculating the distance we compute square distance which is faster to calculate(without square root)

Step 3 : Sort the distance and determine nearest neighbours based on the K-th minimum distance

X1(Acid) in seconds	X2(Strength) in kg/square meter	Square Distance to query instance(3,7)	Rank minimum distance	Is it included in 3-Nearest Neighbors?
7	7	(7-3)^2 + (7-7)^2 = 16	3	Yes
7	4	(7-3)^2 + (4-7)^2= 25	4	No
3	4	(3-3)^2 + (4-7)^2 = 9	1	Yes
1	4	(1-3)^2 + (4-7)^2 = 13	2	Yes

Step 1: Determine Parameter K = number of nearest neighbours. Suppose use k = 3

Step 2: Calculate the distance between the query-instance and all the training samples
Coordinate of query instance is (3,7), instead of calculating the distance we compute square distance which is faster to calculate(without square root)

Step 3 : Sort the distance and determine nearest neighbours based on the K-th minimum distance

Step 4 : Gather the category(Y) of the nearest neighbours. Notice in the second row last column that the category of nearest neighbor(Y) is not included because the rank of this data is more that 3

Step 4 : Gather the category(Y) of the nearest neighbours. Notice in the second row last column that the category of nearest neighbor(Y) is not included because the rank of this data is more that 3

X1(Acid) in seconds	X2(Strength) in kg/square meter	Square Distance to query instance(3,7)	Rank minimum distance	Is it included in 3- Nearest Neighbors?	Y = Category of nearest Neighbor
7	7	(7-3)^2 + (7-7)^2 = 16	3	Yes	Bad
7	4	(7-3)^2 + (4-7)^2= 25	4	No	-
3	4	(3-3)^2 + (4-7)^2 = 9	1	Yes	Good
1	4	(1-3)^2 + (4-7)^2 = 13	2	Yes	Good

Step 1: Determine Parameter K = number of nearest neighbours. Suppose use k = 3

Step 2: Calculate the distance between the query-instance and all the training samples Coordinate of query instance is (3,7), instead of calculating the distance we compute square distance which is faster to calculate(without square root)

Step 3 : Sort the distance and determine nearest neighbours based on the K-th minimum distance

Step 4 : Gather the category(Y) of the nearest neighbours. Notice in the second row last column that the category of nearest neighbor(Y) is not included because the rank of this data is more that 3

Step 5 : Use simple majority to the category of nearest neighbours as the prediction value of the query instance

Step 5 : Use simple majority to the category of nearest neighbours as the prediction value of the query instance

X1(Acid) in seconds	X2(Strength) in kg/square meter	Square Distance to query instance(3,7)	Rank minimum distance	Is it included in 3- Nearest Neighbors?	Y = Category of nearest Neighbor
7	7	$(7-3)^2 + (7-7)^2 = 16$	3	Yes	Bad
7	4	$(7-3)^2 + (4-7)^2 = 25$	4	No	-
3	4	$(3-3)^2 + (4-7)^2 = 9$	1	Yes	Good
1	4	$(1-3)^2 + (4-7)^2 = 13$	2	Yes	Good

We have 2 good and 1 bad, since 2>1 then we conclude that a new paper tissue that pass laboratory test with X1 = 3 and X2 =7 is included in Good category

KNN – Distance

Age	Loan	Default	Distance
25	$40,000	N	102000
35	$60,000	N	82000
45	$80,000	N	62000
20	$20,000	N	122000
35	$120,000	N	22000
52	$18,000	N	124000
23	$95,000	Y	47000
40	$62,000	Y	80000
60	$100,000	Y	42000
48	$220,000	Y	78000
33	$150,000	Y	8000
48	**$142,000**	**?**	

Euclidean Distance

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

KNN – Standardized Distance

Age	Loan	Default	Distance
0.125	0.11	N	0.7652
0.375	0.21	N	0.5200
0.625	0.31	N	0.3160
0	0.01	N	0.9245
0.375	0.50	N	0.3428
0.8	0.00	N	0.6220
0.075	0.38	Y	0.6669
0.5	0.22	Y	0.4437
1	0.41	Y	0.3650
0.7	1.00	Y	0.3861
0.325	0.65	Y	0.3771
0.7	**0.61**	**?**	

Standardized Variable

$$X_s = \frac{X - Min}{Max - Min}$$

Behaviour

Large k : Smoother boundaries (class separating)

Large N : Large storage req. (space complexity)

Large p : lower accuracy (curse of dimensionality)

Step 1

```
import numpy as np
from sklearn.preprocessing import Imputer
from sklearn.cross_validation import train_test_split
from sklearn.neighbors import KNeighborsClassifier
from sklearn.metrics import accuracy_score
```

Step 2 - Import Data

Step 3

```
X_train, X_test, y_train, y_test = train_test_split(
 X, Y, test_size = 0.3, random_state = 100)
y_train = y_train.ravel()
y_test = y_test.ravel()
```

Step 4

```
for K in range(25):
 K_value = K+1
 neigh = KNeighborsClassifier(n_neighbors = K_value, weights='uniform', algorithm='auto')
 neigh.fit(X_train, y_train)
 y_pred = neigh.predict(X_test)
 print "Accuracy is ", accuracy_score(y_test,y_pred)*100,"% for K-Value:",K_value
```

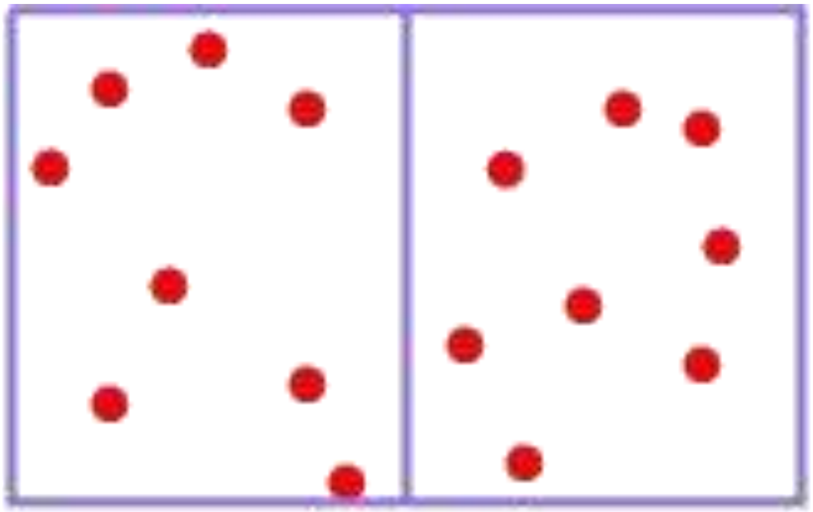
KNN Advantage

- Makes no assumptions about distributions of classes in feature space
- Can work for multi classes simultaneously
- Easy to implement and understand
- Not impacted by outliers

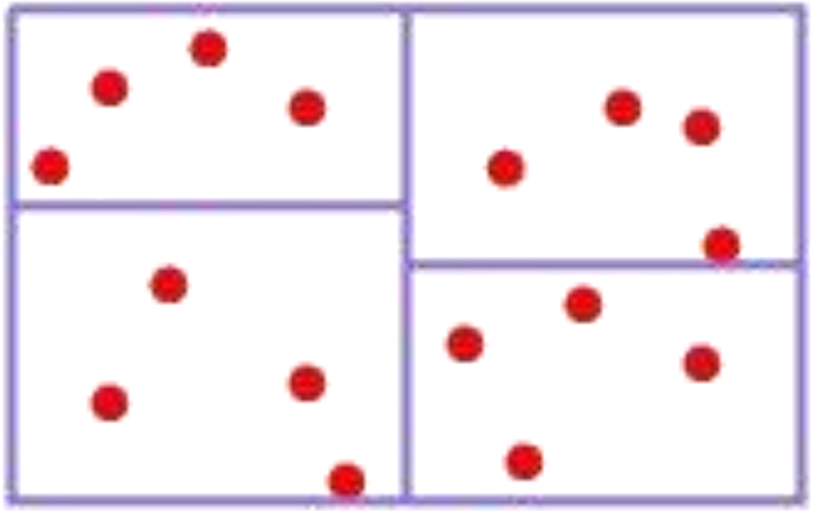
KNN Disadvantage

- Fixing the optimal value of K is a challenge
- Will not be effective when the class distributions overlap
- Does not output any models. Calculates distances for every new point (lazy learner)
- Computationally intensive (O(D(N^2))), can be addressed using KD algorithms which take time to prepare

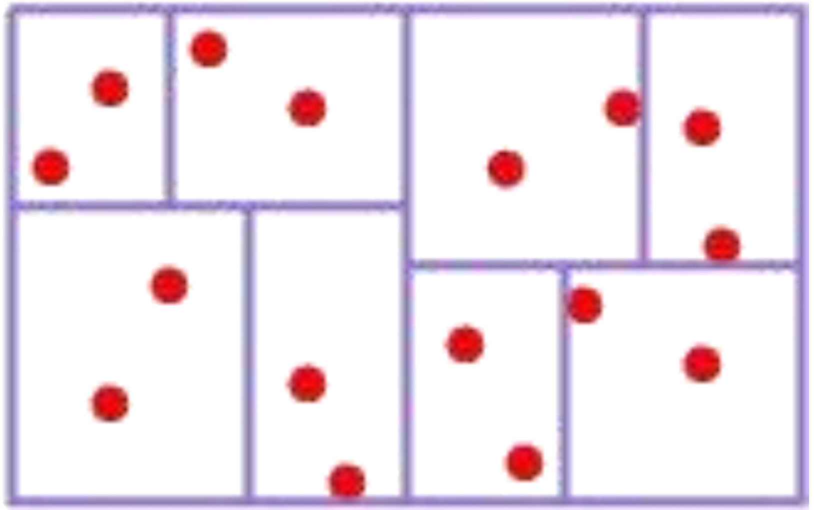
1.



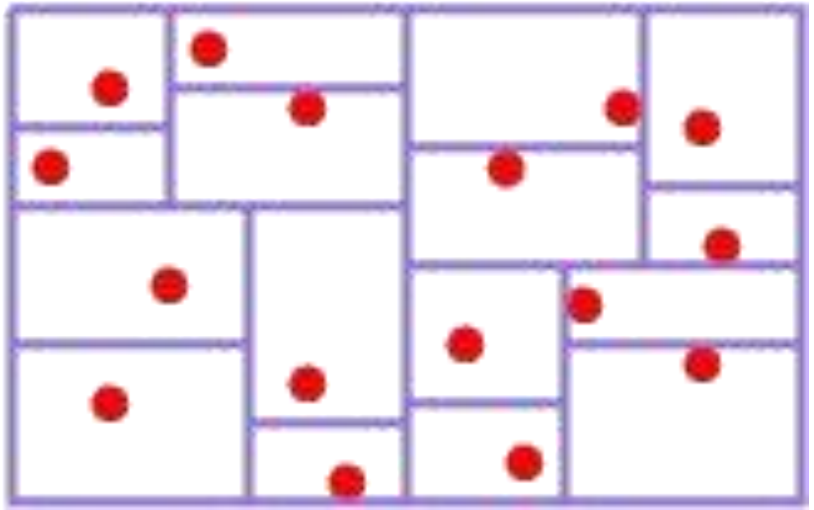
2.

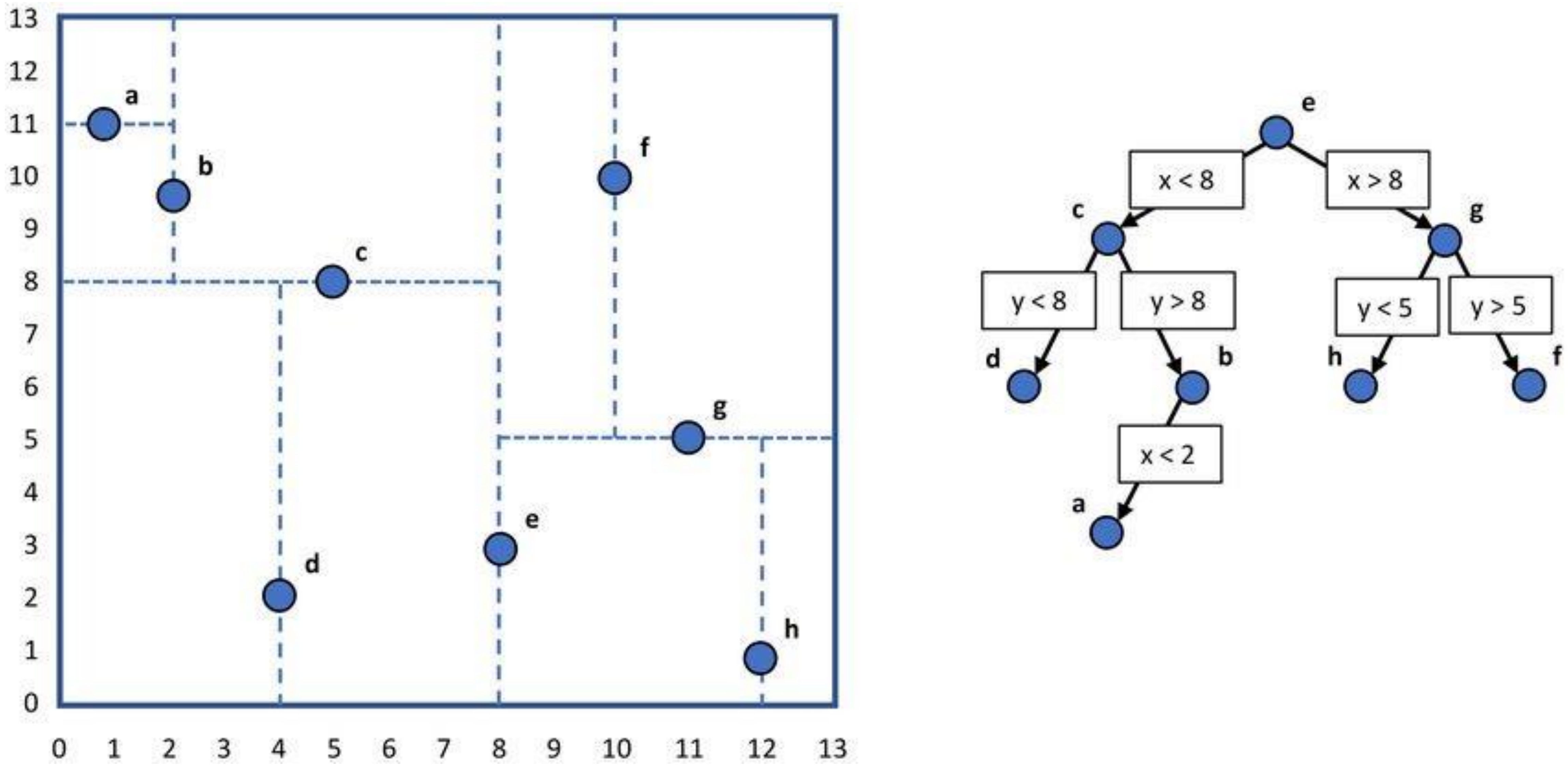


3.

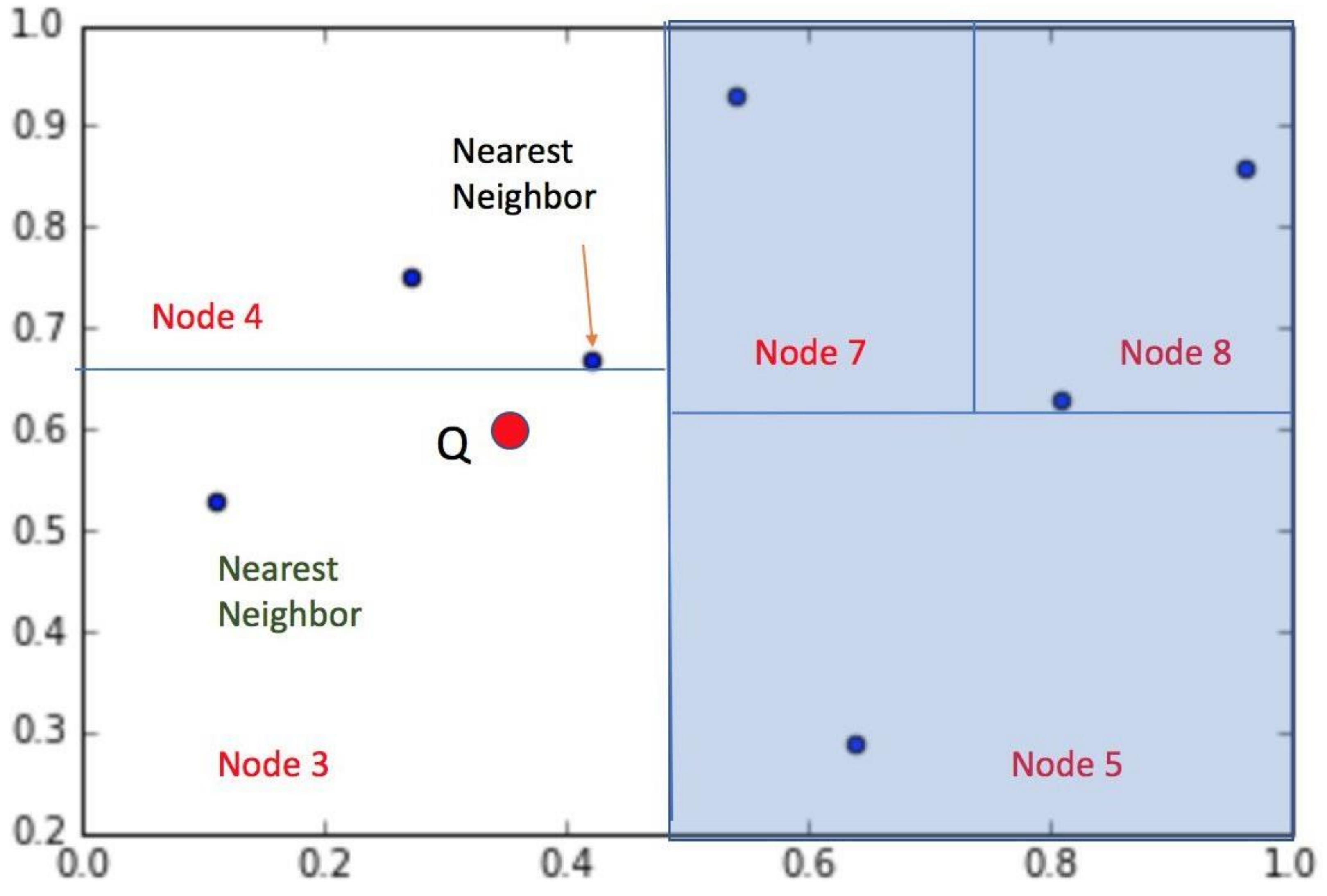


4.





13
12
11
10
9
8
7
6
5
4
3
2
1
0
0 1 2 3 4 5 6 7 8 9 10 11 12 13
a
b
c
d
e
f
g
h
e
x < 8
x > 8
c
g
y < 8
y > 8
y < 5
y > 5
d
b
h
f
x < 2
a



Nearest Neighbor
Node 4
Q
Node 7
Node 8
Nearest Neighbor
Node 3
Node 5
1.0
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.0
0.2
0.4
0.6
0.8
1.0

THANK YOU