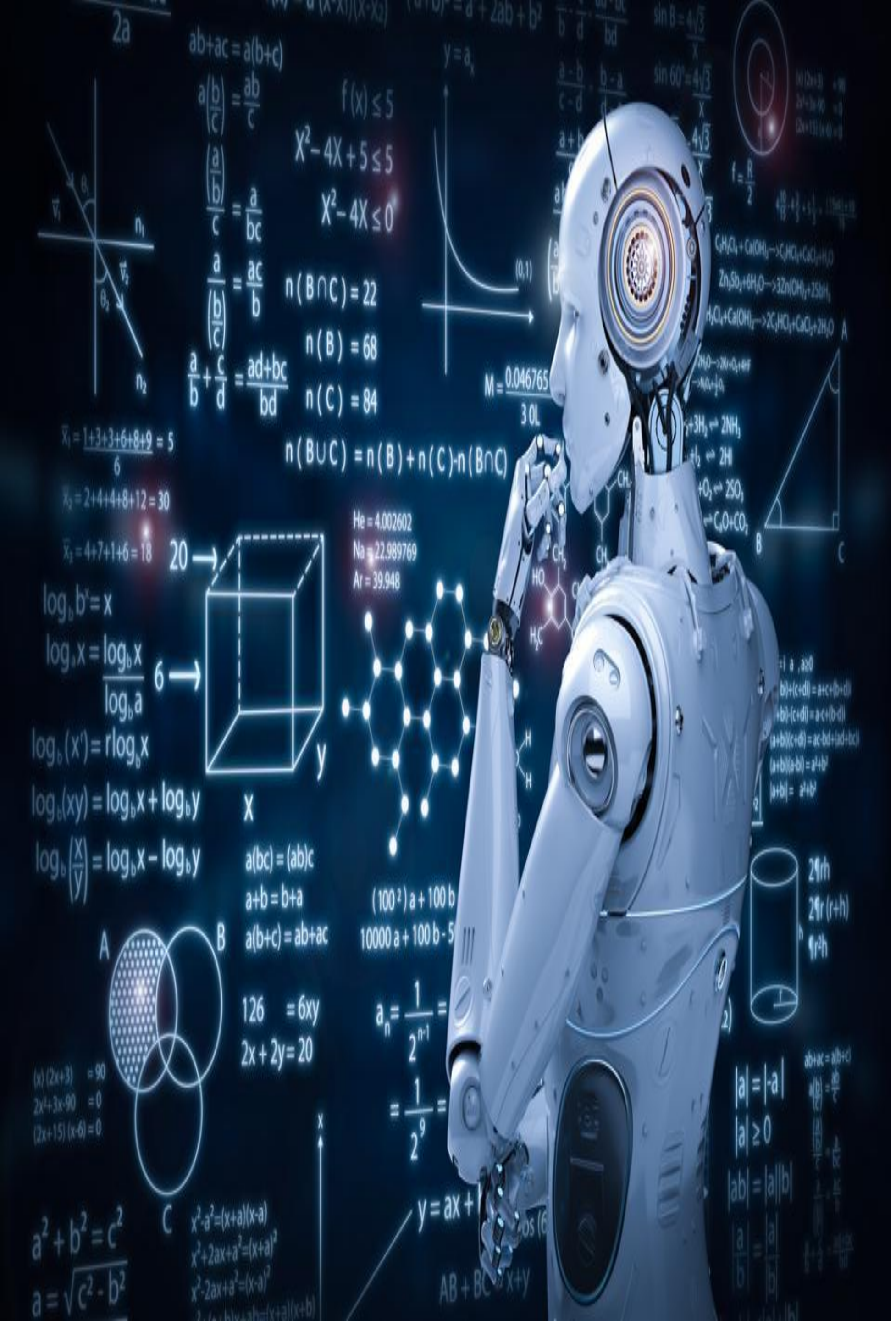


Practical Machine Learning

Day 10: Mar22 DBDA

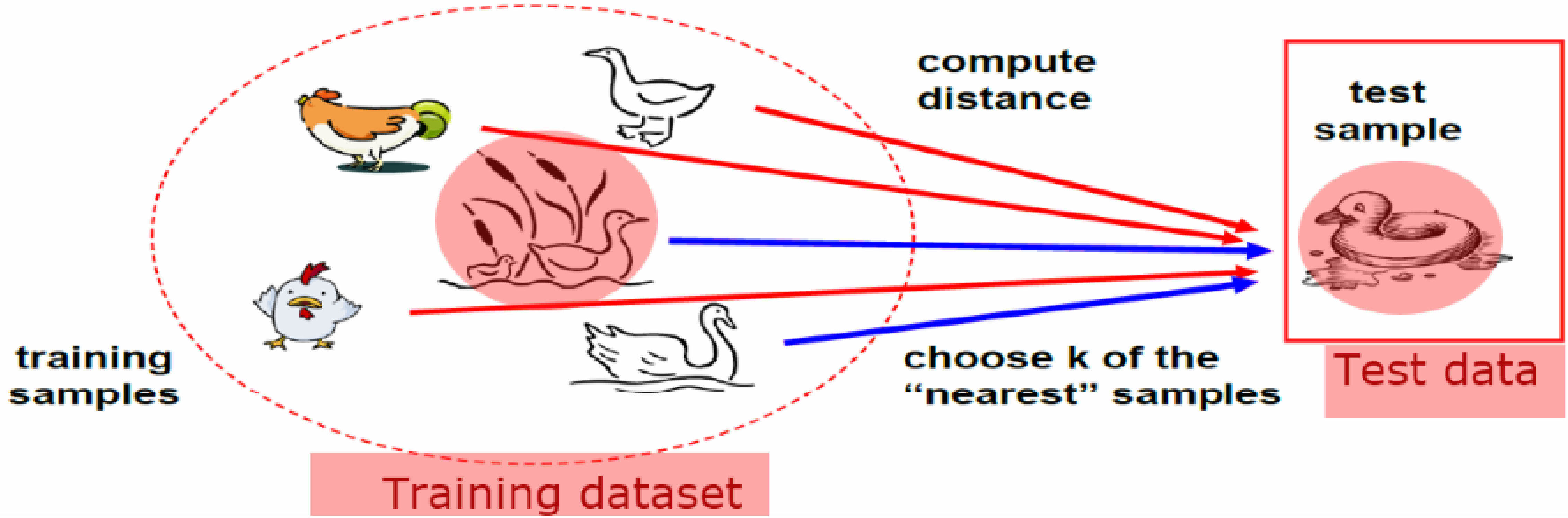
Kiran Waghmare

Agenda

- Classification Algorithm
- kNN
- Naïve Bayes

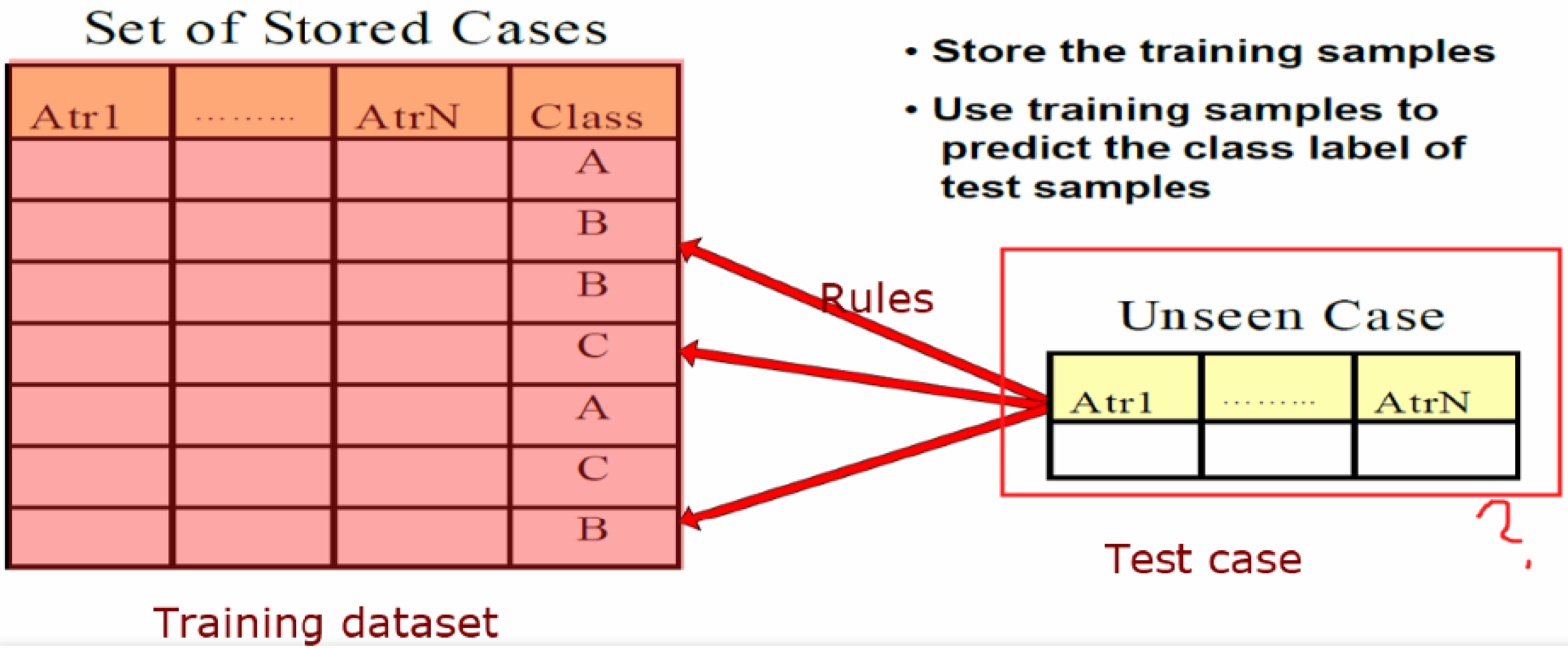
Nearest Neighbor Classifiers → Pattern recognition

- Basic idea: Similarity: distance
  - If it walks like a duck, quacks like a duck, then it's probably a duck



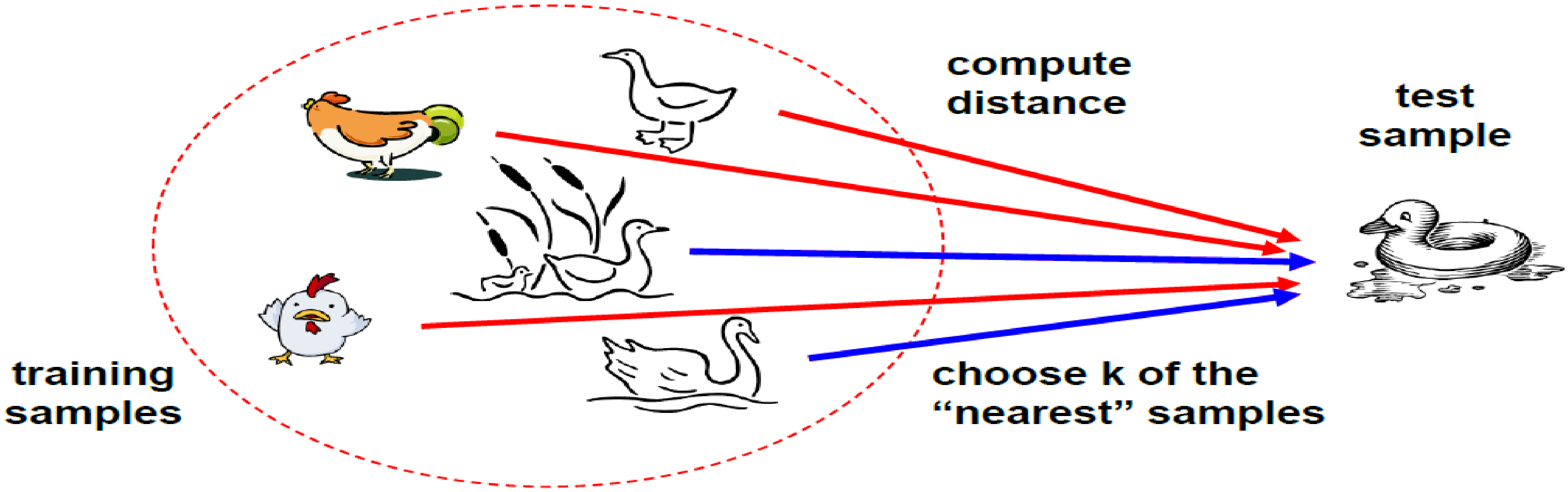
Instance based classifiers

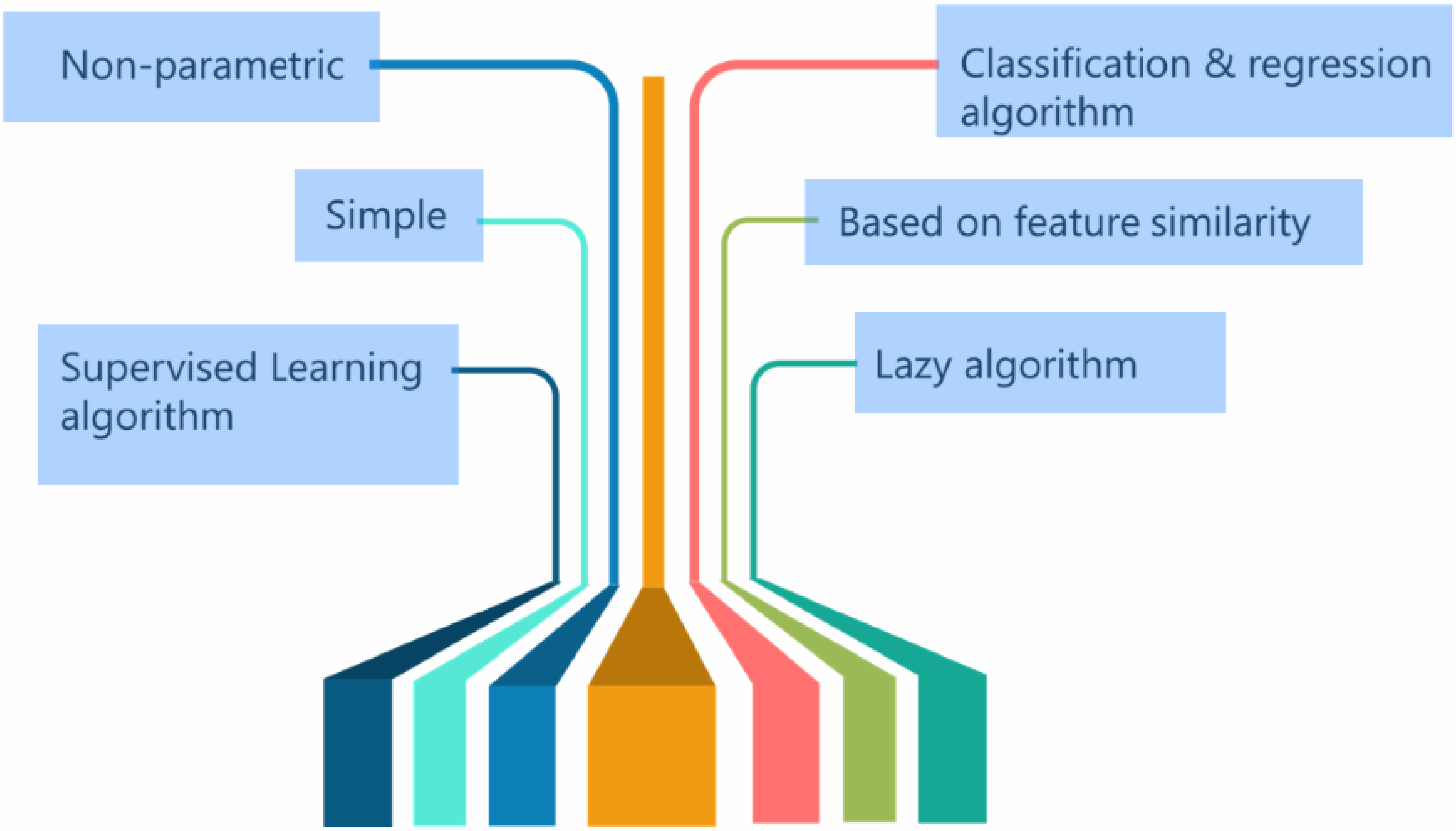
- Store the training samples
- Use training samples to predict the class label of test samples



Nearest Neighbor Classifiers

- Basic idea:
  - If it walks like a duck, quacks like a duck, then it's probably a duck




Non-parametric
Simple
Supervised Learning algorithm
Classification & regression algorithm
Based on feature similarity
Lazy algorithm

Lazy learners

•'**Lazy**': Do not create a model of the training instances in advance

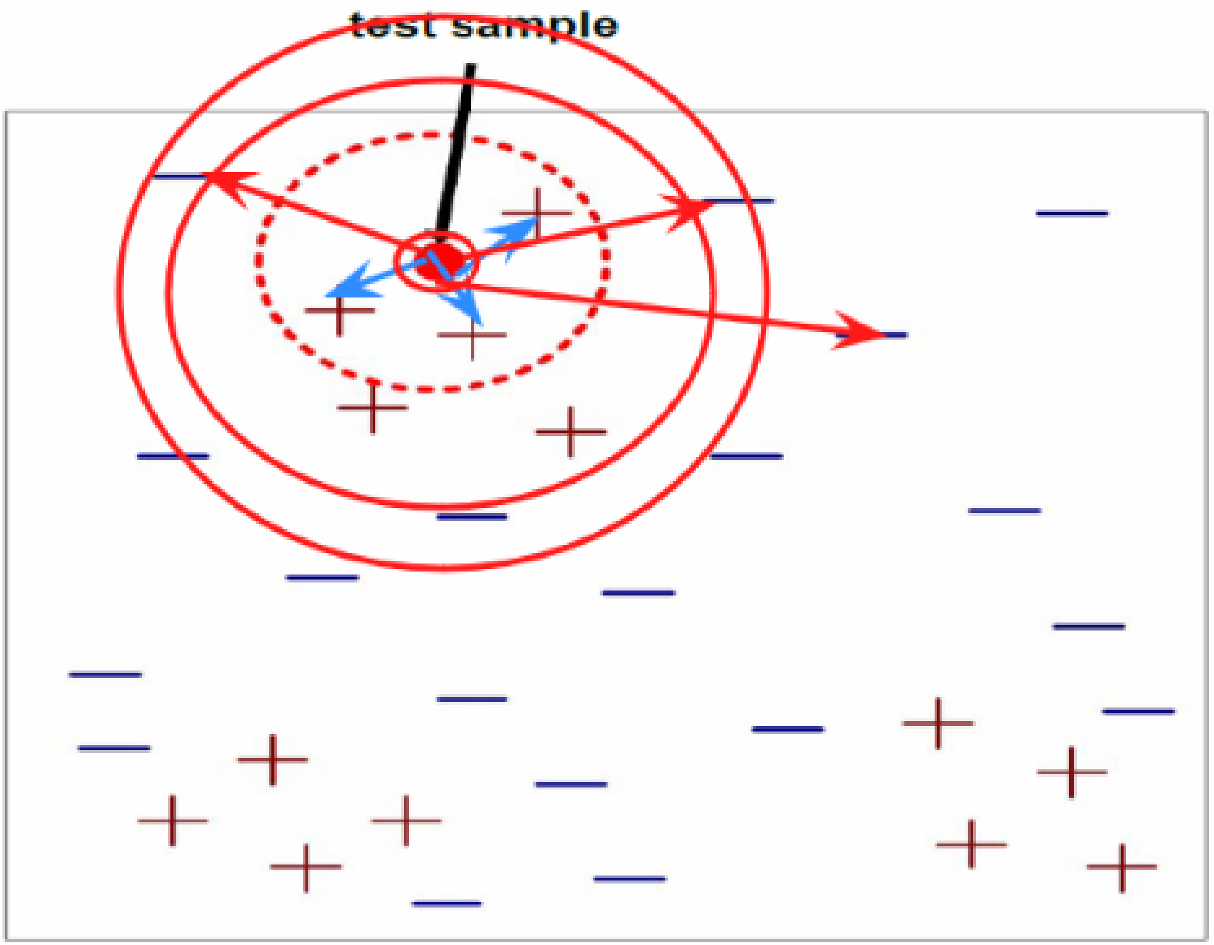
•When an instance arrives for testing, runs the algorithm to get the class prediction

•**Example, K** – nearest neighbor classifier

(K – NN classifier)

"One is known by the company one keeps"

Nearest Neighbor Classifiers



Requires three inputs:

1. The set of stored samples
2. Distance metric to compute distance between samples
3. The value of k, the number of nearest neighbors to retrieve

Distances for nearest neighbors

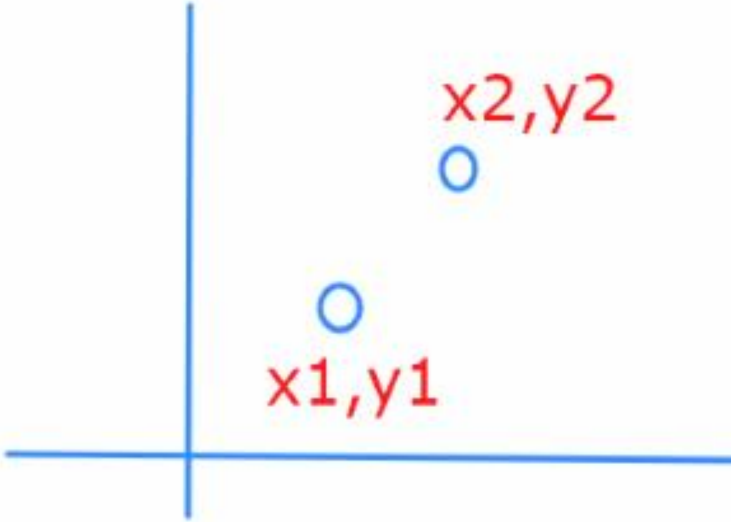
- Options for computing distance between two samples:
  - Euclidean distance

$$d(\mathbf{x}, \mathbf{y}) = \sqrt{\sum_i (x_i - y_i)^2}$$

  - Cosine similarity

$$d(\mathbf{x}, \mathbf{y}) = \mathbf{x} \cdot \mathbf{y}$$

  - Hamming distance
  - String edit distance
  - Kernel distance
  - Many others



Distance measure for Continuous Variables

Distance functions

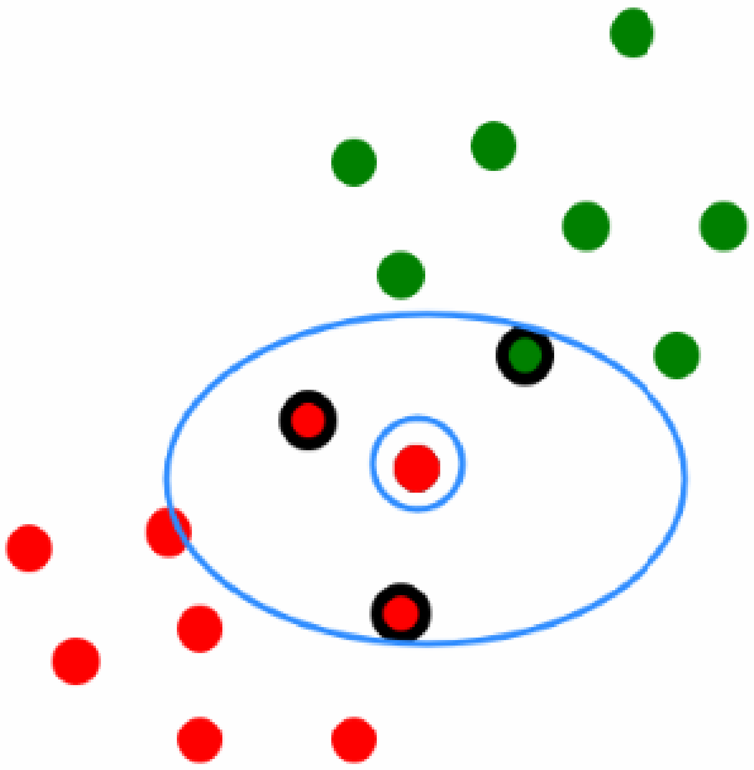
- Euclidean: $\sqrt{\sum_{i=1}^{k}(x_i - y_i)^2}$
- Manhattan: $\sum_{i=1}^{k}|x_i - y_i|$
- Minkowski: $\left(\sum_{i=1}^{k}(|x_i - y_i|)^q\right)^{1/q}$

Numeric value

K-NN classifier schematic

For a test instance,

1) Calculate distances from training pts.
2) Find K-nearest neighbours (say, K = 3)
3) Assign class label based on majority



Distance Between Neighbors

- Calculate the distance between new example (E) and all examples in the training set.

- *Euclidean* distance between two examples.
  - $X = [x_1, x_2, x_3, .., x_n]$
  - $Y = [y_1, y_2, y_3, ..., y_n]$

  - The Euclidean distance between *X* and *Y* is defined as:

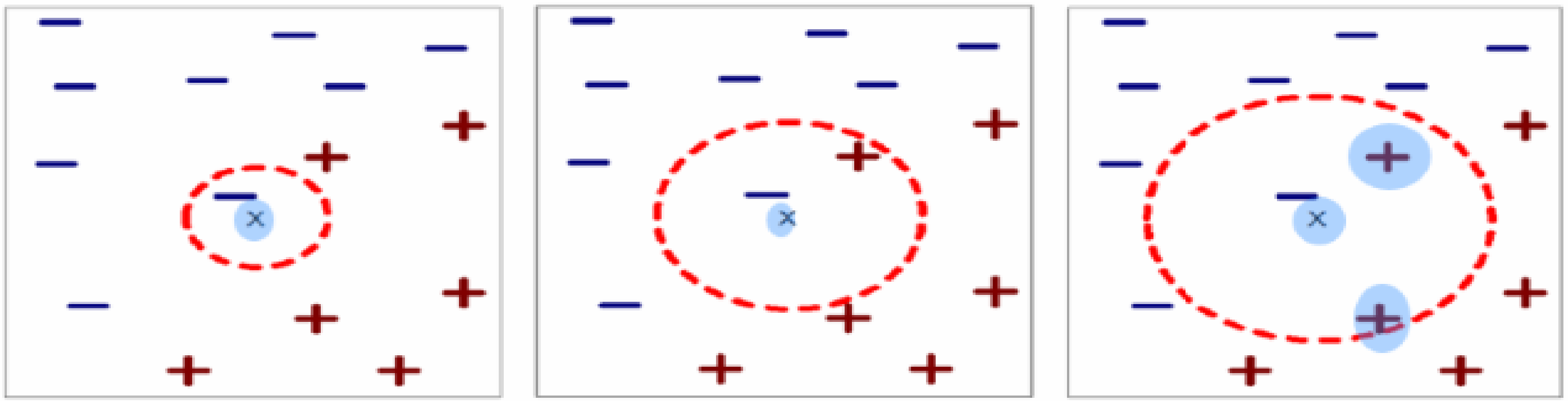
$$D(X,Y) = \sqrt{\sum_{i=1}^{n} (x_i - y_i)^2}$$

1

Predicting class from nearest neighbors

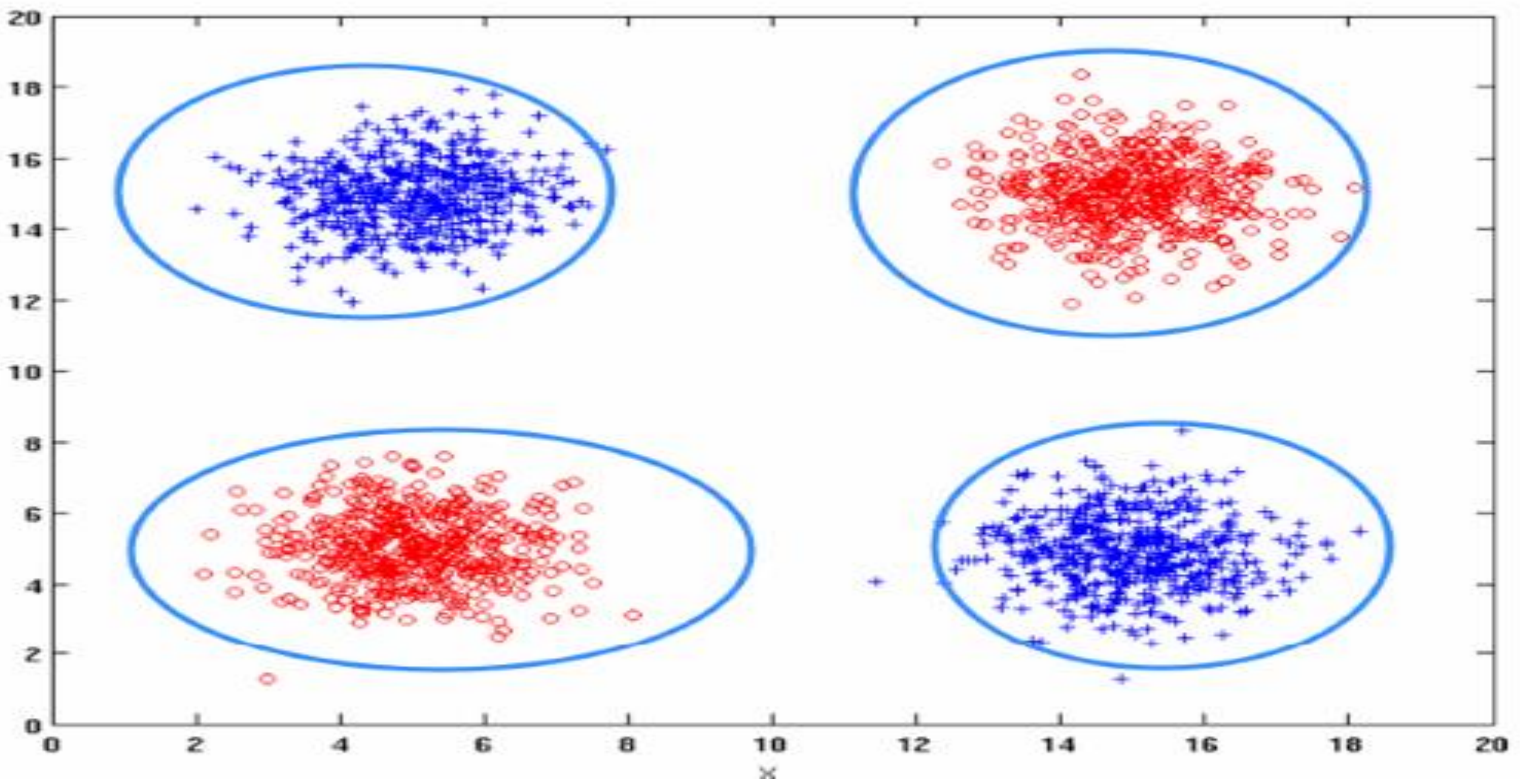
- Options for predicting test class from nearest neighbor list
  - Take majority vote of class labels among the *k*-nearest neighbors
  - Weight the votes according to distance
    - example: weight factor $w = 1 / d^2$

Predicting class from nearest neighbors



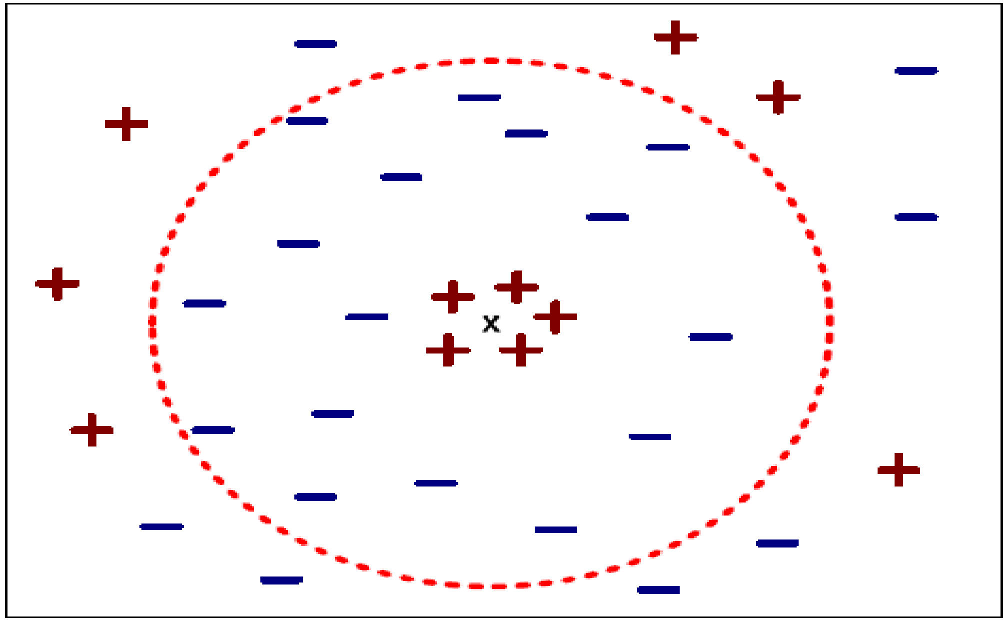
nearest neighbors	1	2	3
majority vote	–	?	+
distance-weighted vote	–	–	– or +

K-NN Classifiers: Handling attributes that are interacting



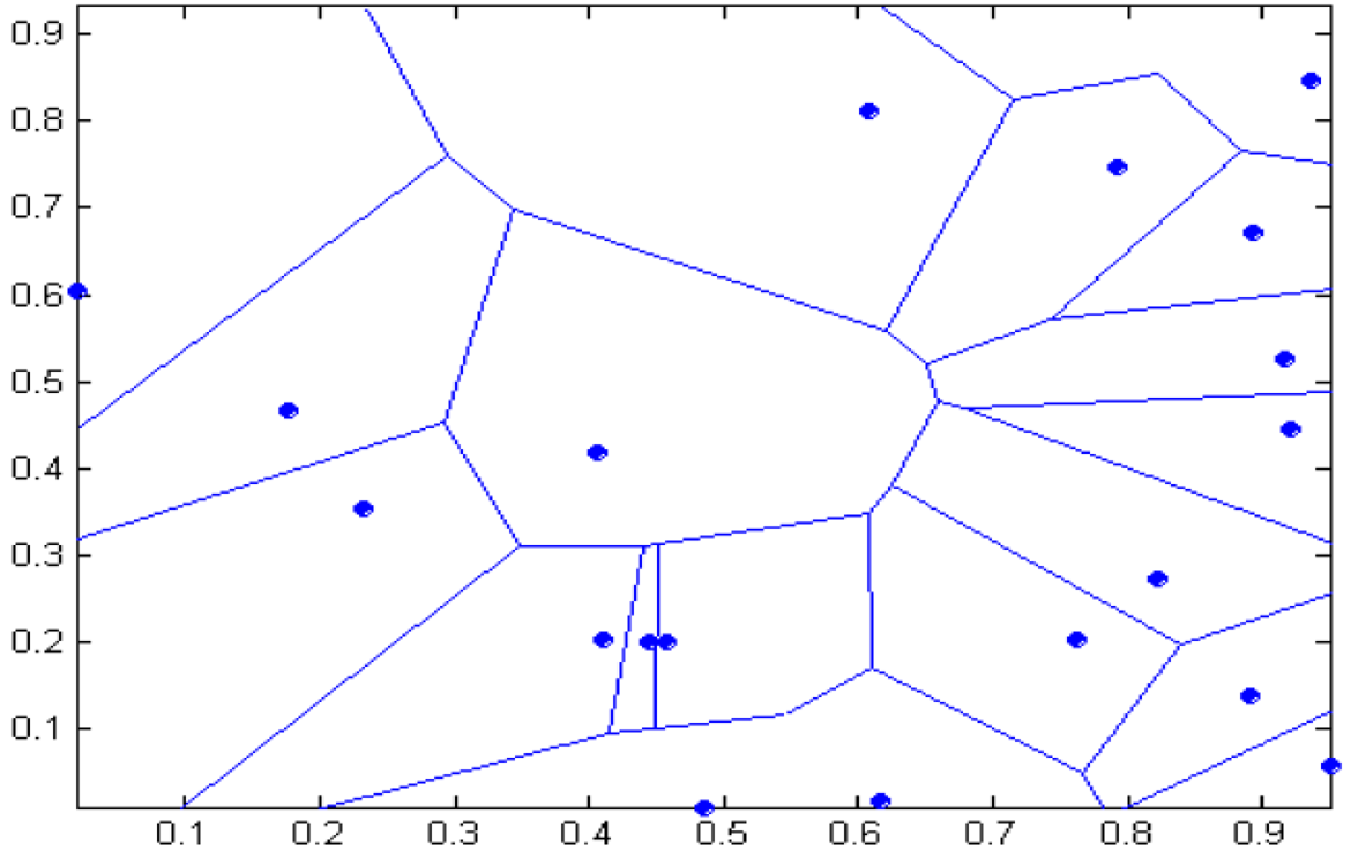
Predicting class from nearest neighbors

- Choosing the value of *k*:
  - If *k* is too small, sensitive to noise points
  - If *k* is too large, neighborhood may include points from other classes



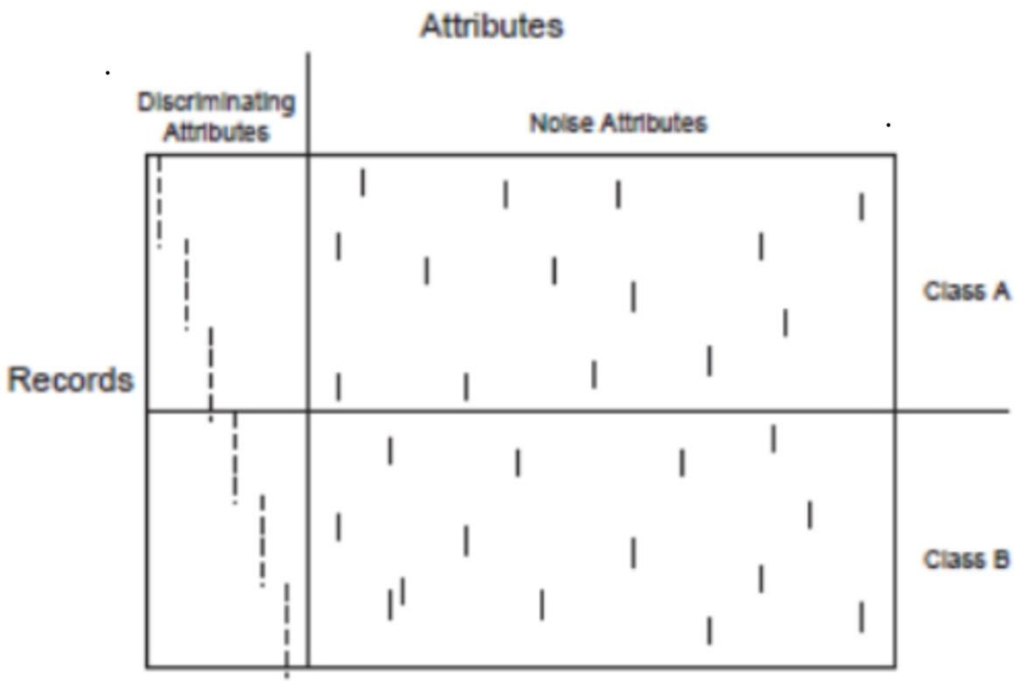
1-nearest neighbor

Voronoi diagram



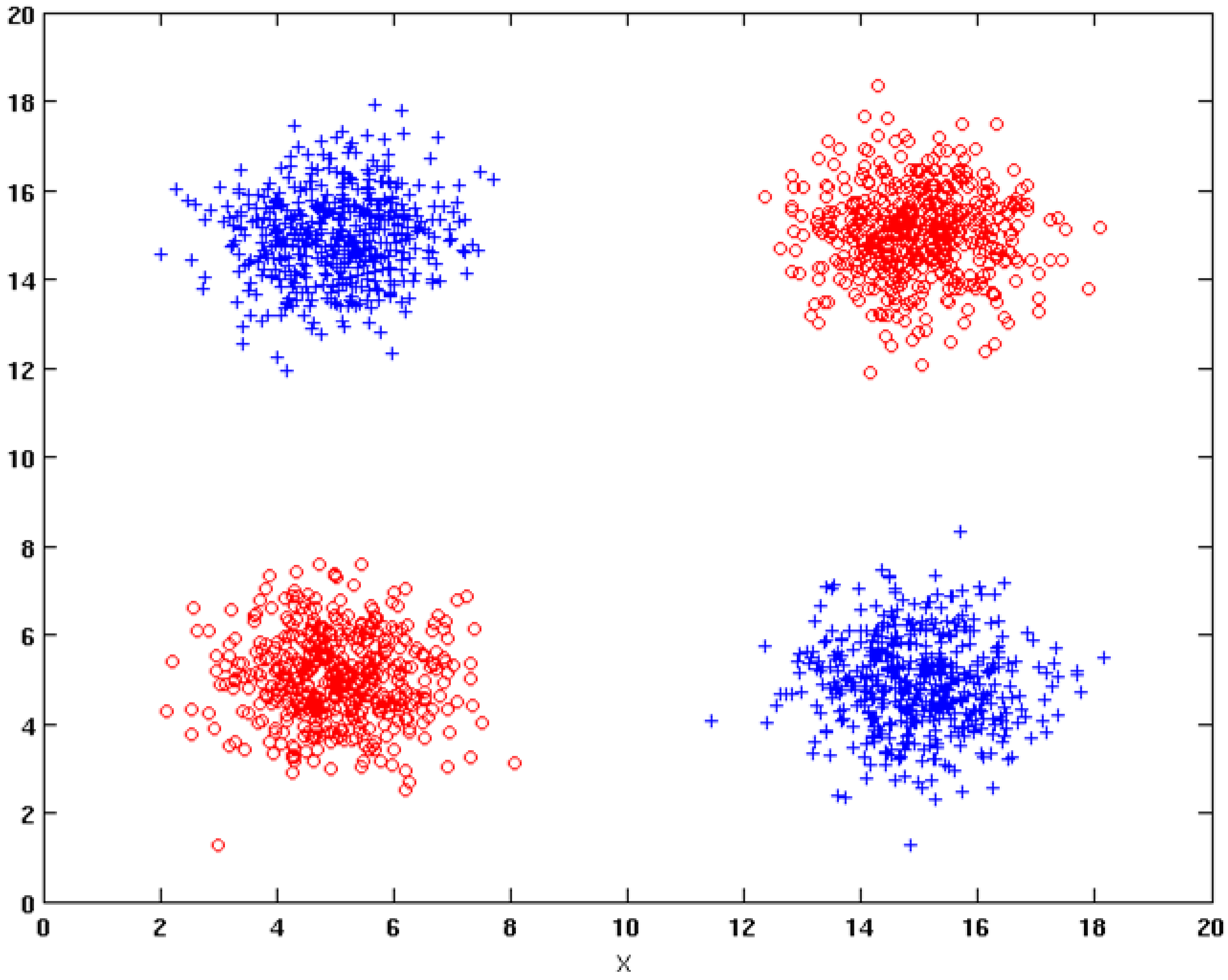
K-NN Classificiers…
Handling Irrelevant and Redundant Attributes

- Irrelevant attributes add noise to the proximity measure
- Redundant attributes bias the proximity measure towards certain attributes

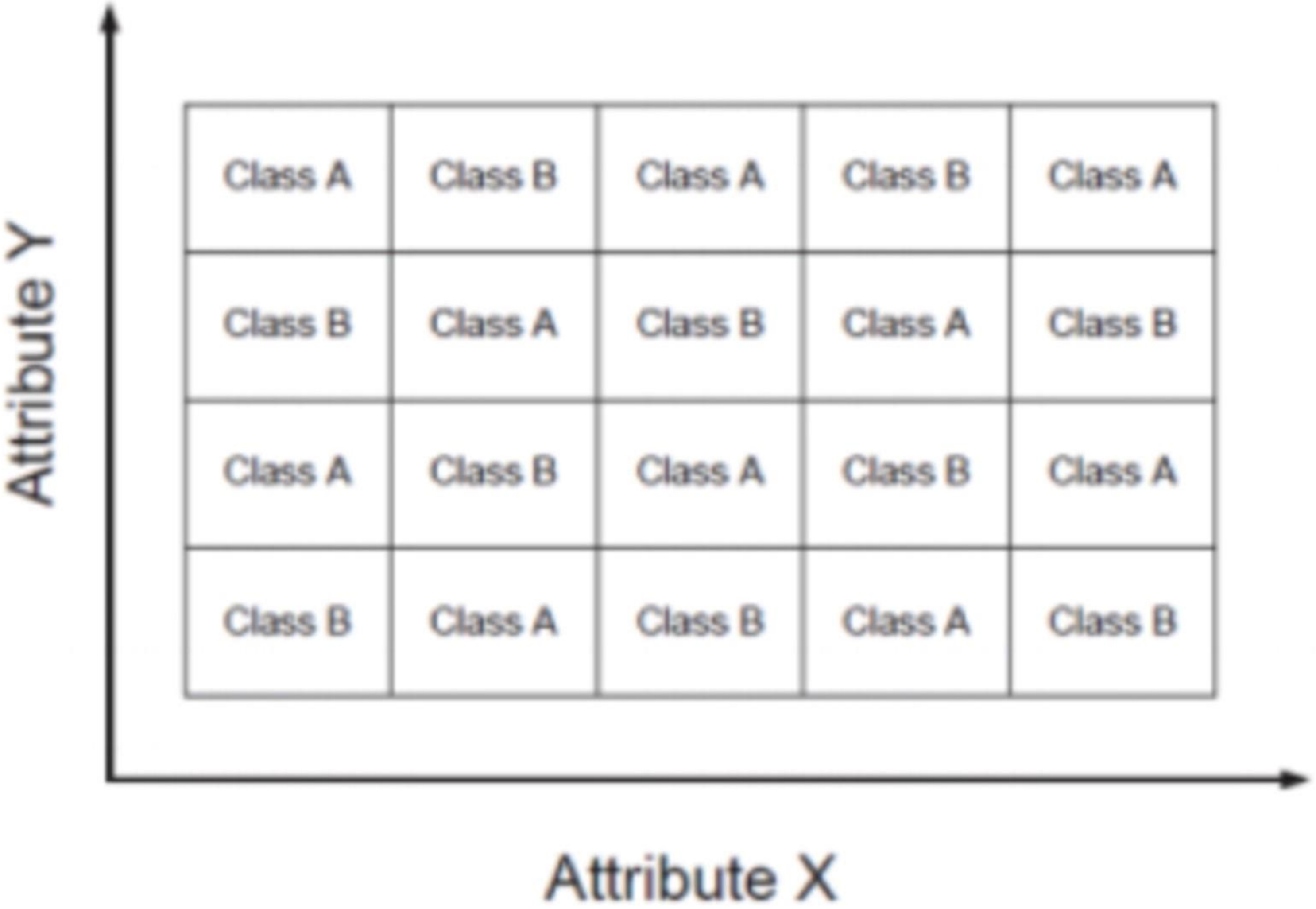


(a) Synthetic data set 1.

K-NN Classifiers: Handling attributes that are interacting



Handling attributes that are interacting



Decision boundaries in global vs. local models

logistic regression

- global
- stable
- can be inaccurate

15-nearest neighbor **1-nearest neighbor**

- local
- unstable
- accurate

stable: model decision boundary not sensitive to addition or removal of samples from training set

What ultimately matters: ***GENERALIZATION***

KNN Classification – 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

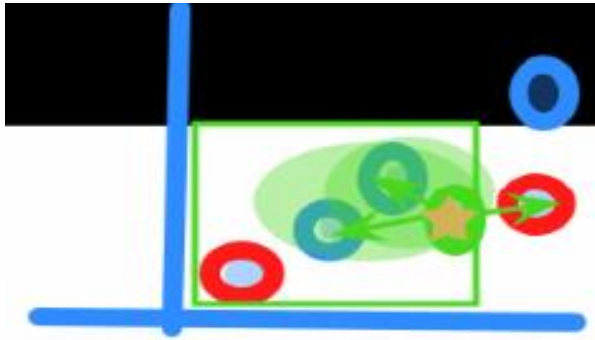
$$D = \sqrt{(x_1 - x_2)^2 + (y_1 - y_2)^2}$$

KNN Classification – 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

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



3-KNN: Example(1)

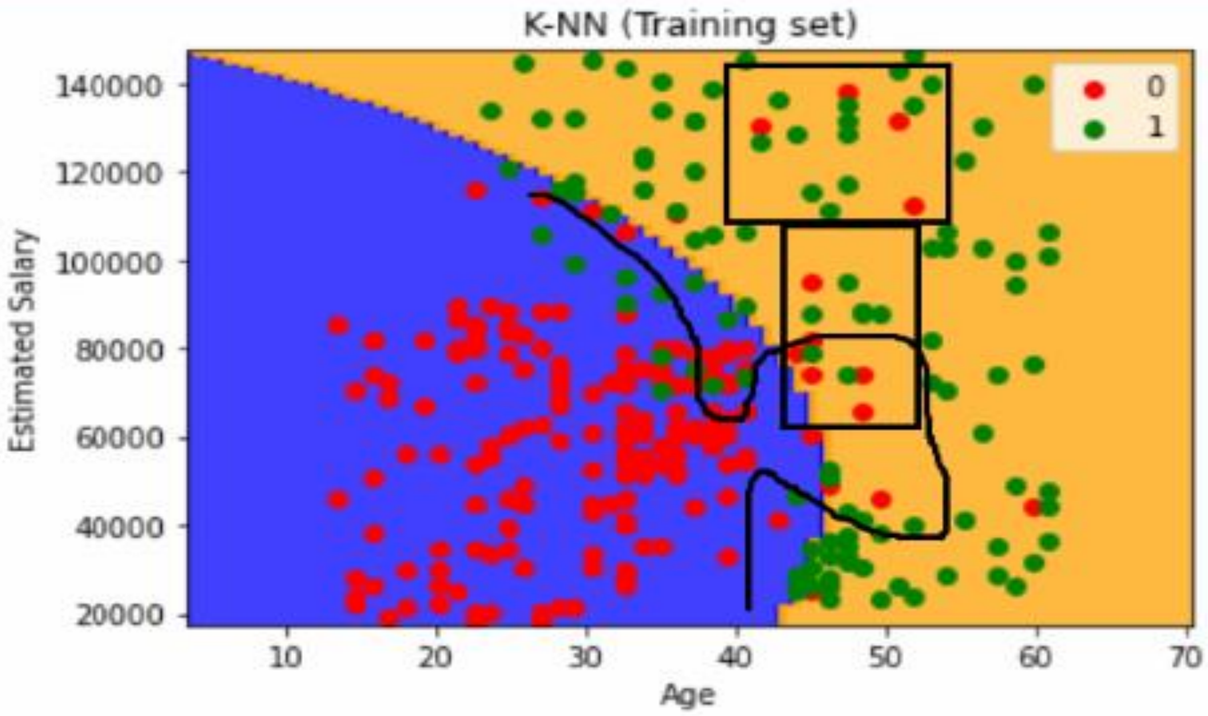
Customer	Age	Income	No. credit cards	Class	Distance from John
George	35	35K	3	No	sqrt $[(35-37)^2+(35-50)^2+(3-2)^2]$=15.16
Rachel	22	50K	2	Yes	sqrt $[(22-37)^2+(50-50)^2+(2-2)^2]$=15
Steve	63	200K	1	No	sqrt $[(63-37)^2+(200-50)^2+(1-2)^2]$=152.23
Tom	59	170K	1	No	sqrt $[(59-37)^2+(170-50)^2+(1-2)^2]$=122
Anne	25	40K	4	Yes	sqrt $[(25-37)^2+(40-50)^2+(4-2)^2]$=15.74
John	37	50K	2	YES	

Improving KNN Efficiency

- Avoid having to compute distance to all objects in the training set
  - Multi-dimensional access methods (k-d trees)
  - Fast approximate similarity search
  - Locality Sensitive Hashing (LSH)
- Condensing
  - Determine a smaller set of objects that give the same performance
- Editing
  - Remove objects to improve efficiency

```
Out[17]: array([[64,  4],
                [ 3, 29]], dtype=int64)

In [18]: accuracy_score(y_test,y_pred)

Out[18]: 0.93

In [19]: #Visualising the Training data

         from matplotlib.colors import ListedColormap
         X_set, y_set = sc.inverse_transform(X_train), y_train
         X1, X2 = np.meshgrid(np.arange(start = X_set[:, 0].min() - 10, stop = X_set[:, 0]
                              np.arange(start = X_set[:, 1].min() - 1000, stop = X_set[:,

         plt.contourf(X1, X2, classifier.predict(sc.transform(np.array([X1.ravel(), X2.rav
                      alpha = 0.75, cmap = ListedColormap(('blue', 'orange')))
         plt.xlim(X1.min(), X1.max())
         plt.ylim(X2.min(), X2.max())
         for i, j in enumerate(np.unique(y_set)):
             plt.scatter(X_set[y_set == j, 0], X_set[y_set == j, 1], c = ListedColormap((
         plt.title('K-NN (Training set)')
         plt.xlabel('Age')
         plt.ylabel('Estimated Salary')
         plt.legend()
         plt.show()

*c* argument looks like a single numeric RGB or RGBA sequence, which should be
avoided as value-mapping will have precedence in case its length matches with *
x* & *y*.  Please use the *color* keyword-argument or provide a 2D array with a
single row if you intend to specify the same RGB or RGBA value for all points.
```

x* & *y*. Please use the *color* keyword-argument or provide a 2D array with a single row if you intend to specify the same RGB or RGBA value for all points.



In [68]:
```
from matplotlib.colors import ListedColormap
X_set, y_set = sc.inverse_transform(X_test), y_test
X1, X2 = np.meshgrid(np.arange(start = X_set[:, 0].min() - 10, stop = X_set[:, 0
                     np.arange(start = X_set[:, 1].min() - 1000, stop = X_set[:,
plt.contourf(X1, X2, classifier.predict(sc.transform(np.array([X1.ravel(), X2.ra
             alpha = 0.75, cmap = ListedColormap(('blue', 'orange')))
plt.xlim(X1.min(), X1.max())
plt.ylim(X2.min(), X2.max())
```