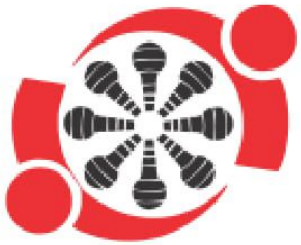


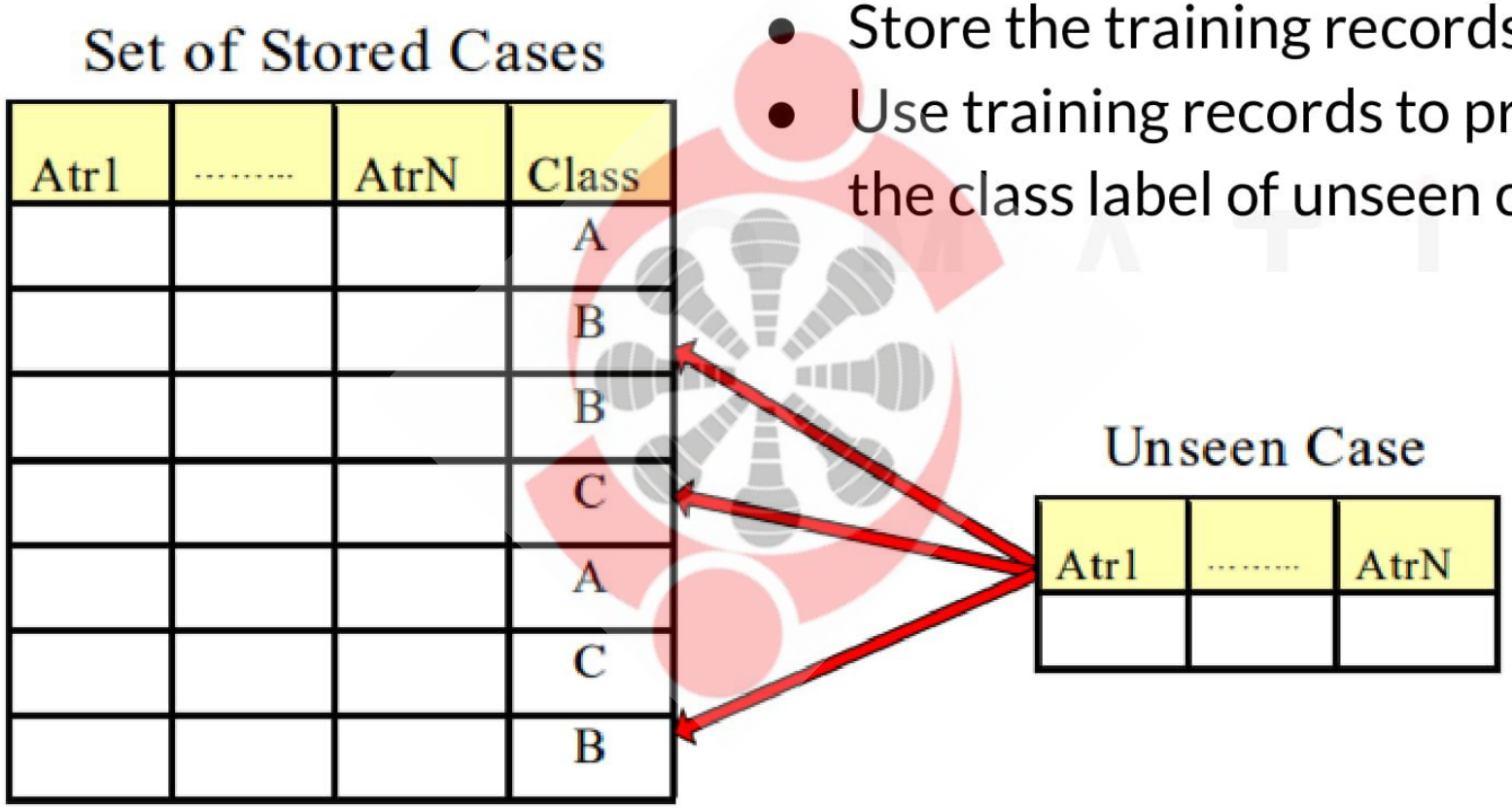
INNOMATICS
RESEARCH LABS

# K-Nearest Neighbour

# Content

- k-Means vs k-NN
- When to Consider Nearest Neighbors
- How to determine the good value for k?
- K-Nearest Neighbor algorithm
- A few Applications and Examples of KNN
- Some pros and cons of KNN
- Quick summary of KNN
- For n denotes the number of data points, d the number of (original) features Problems with training and testing on the same data
- Downsides of train/test split

# Instance-Based Classifiers




- Store the training records
- Use training records to predict the class label of unseen cases

INNOMATICS
RESEARCH LABS

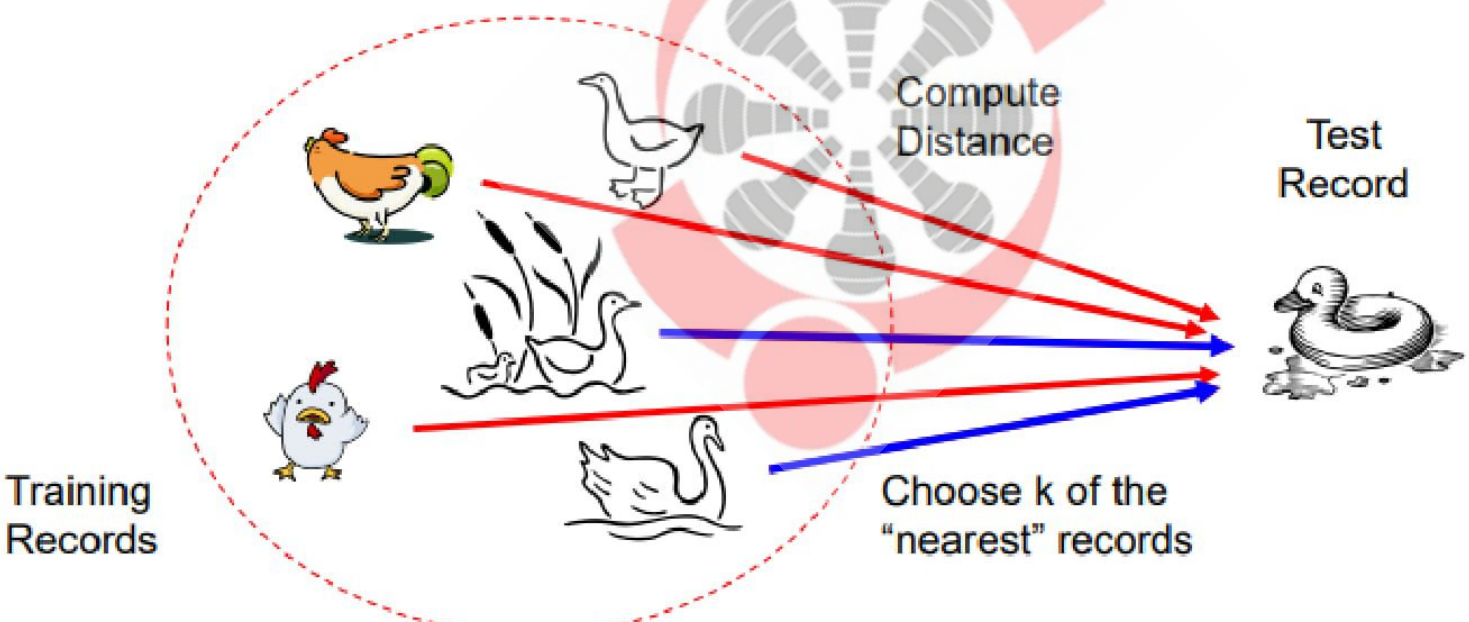
# Instance Based Classifiers

- Examples:
  - Rote-learner
    - Memorizes entire training data and performs classification only if attributes of record match one of the training examples exactly
  - Nearest neighbor
    - Uses k “closest” points (nearest neighbors) for performing classification

# Nearest Neighbor Classifiers

- Basic idea:

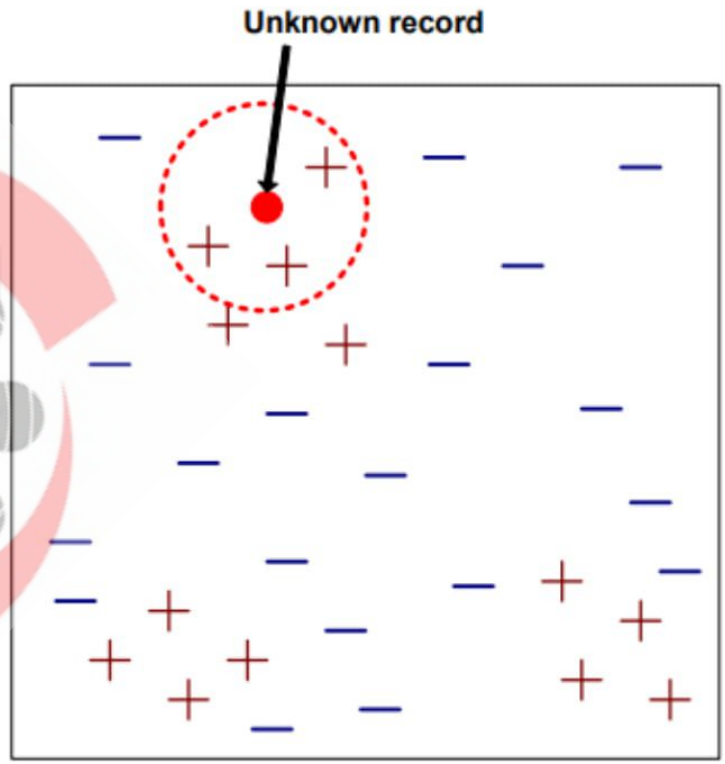
– If it walks like a duck, quacks like a duck, then it's probably a duck



# Nearest-Neighbor Classifiers

Requires three things :

– The set of stored records
– Distance Metric to compute distance between records
– The value of k, the number of nearest neighbors to retrieve



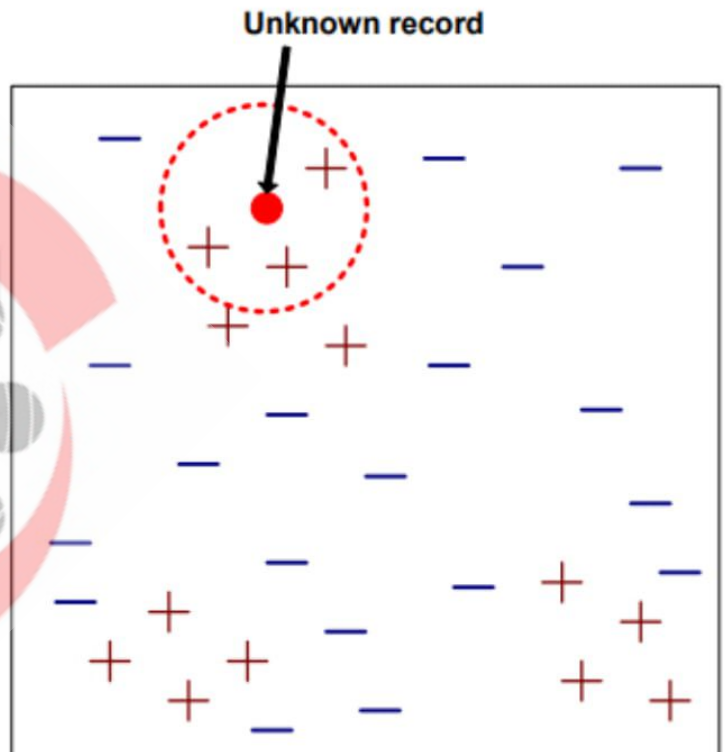
# Nearest-Neighbor Classifiers

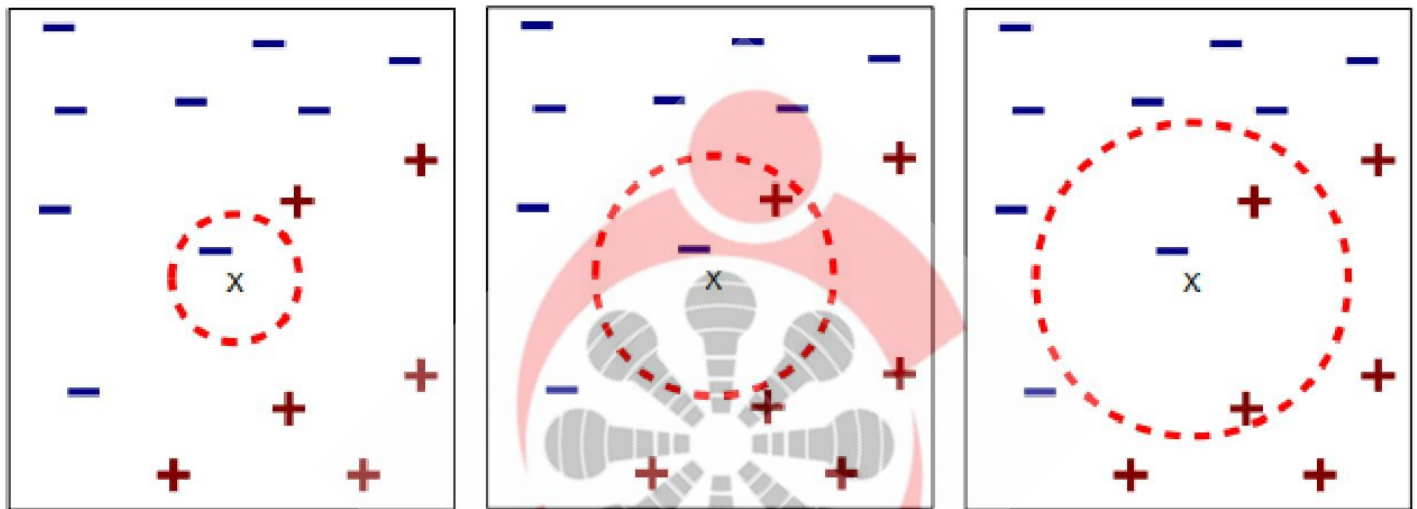
To classify an unknown record:

– Compute distance to other training records

– Identify k nearest neighbors

– Use class labels of nearest neighbors to determine the class label of unknown record (e.g., by taking majority vote)



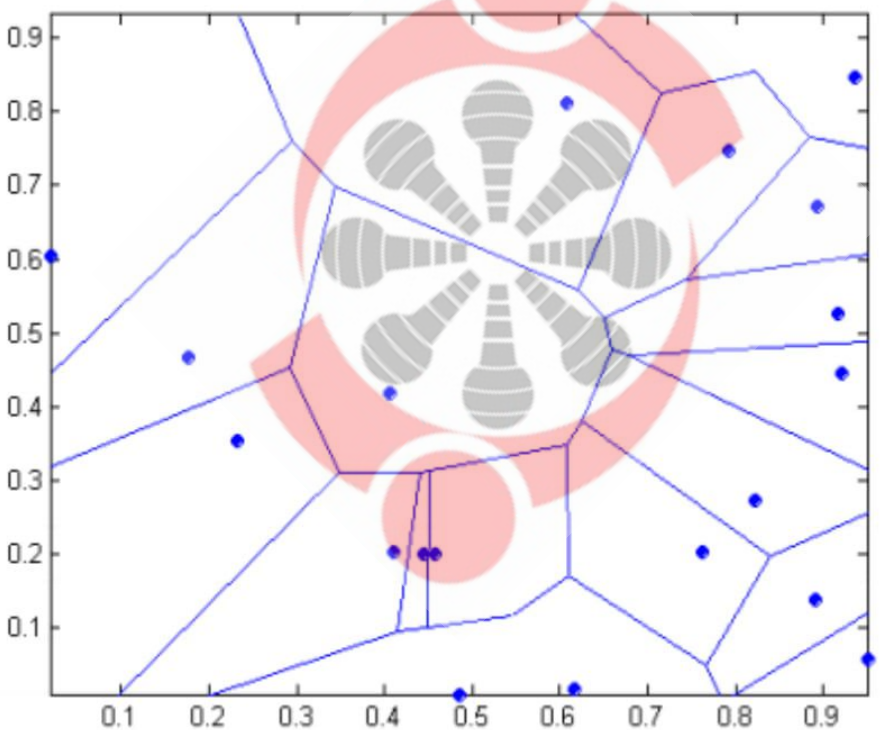



(a) 1-nearest neighbor (b) 2-nearest neighbor (c) 3-nearest neighbor

K-nearest neighbors of a record x are data points that have the k smallest distance to x

# 1 nearest-neighbor

## Voronoi Diagram



# Nearest Neighbor Classification

Compute distance between two points:

– Euclidean distance

$$d(p,q) = \sqrt{\sum_i (p_i - q_i)^2}$$

– Manhattan distance

$$d(p,q) = \sum_i |p_i - q_i|$$

– q norm distance

$$d(p,q) = (\sum_i |p_i - q_i|^q)^{1/q}$$

- Determine the class from nearest neighbor list

– Take the majority vote of class labels among the k-nearest neighbors

$$y' = \underset{v}{\operatorname{argmax}} \sum_{(x_i, y_i) \in D_z} I(v = y_i)$$

➔ where $D_z$ is the set of k closest training examples to z.

– Weigh the vote according to distance

$$y' = \underset{v}{\operatorname{argmax}} \sum_{(x_i, y_i) \in D_z} w_i \times I(v = y_i)$$

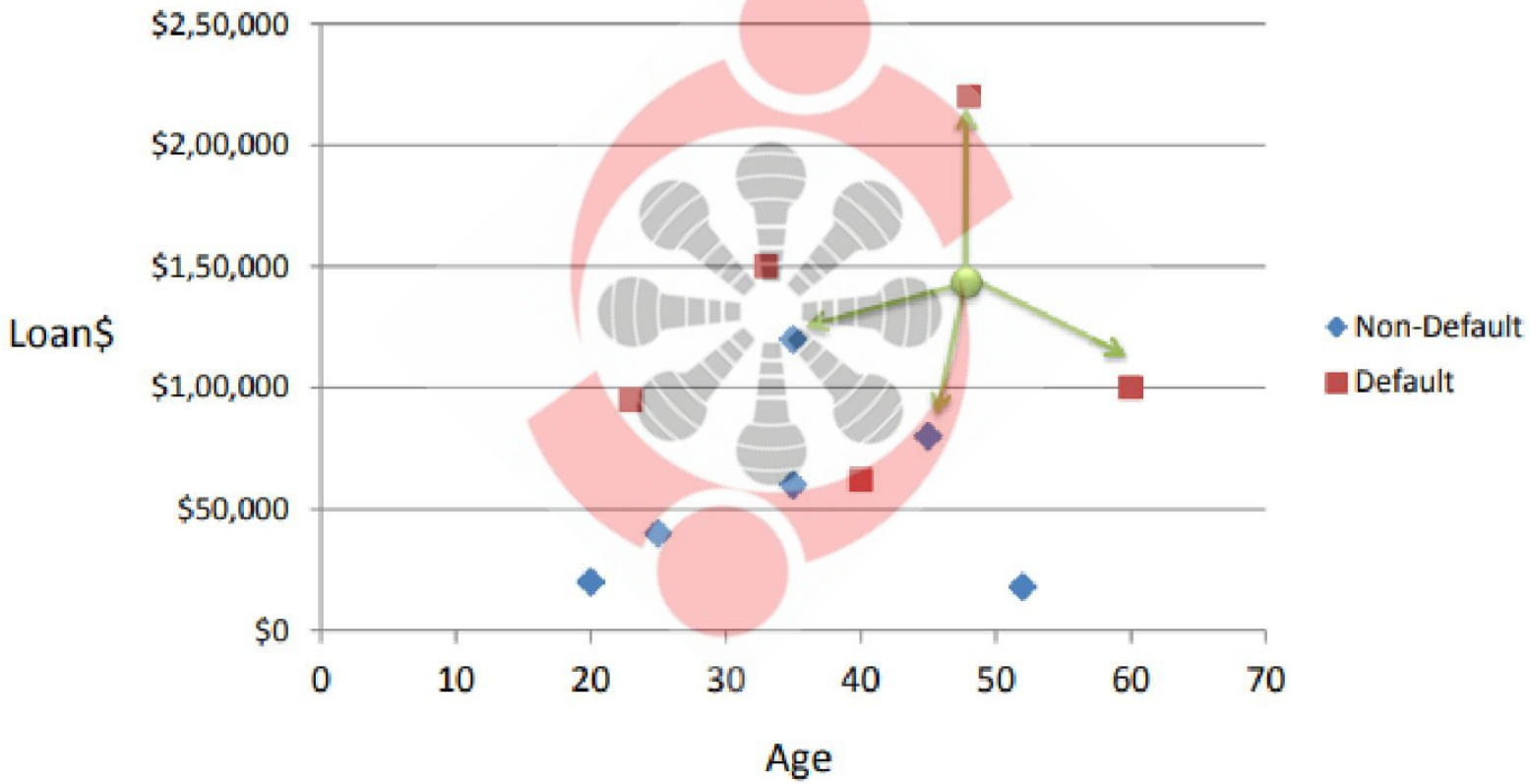
➔ weight factor, $w = 1/d^2$

# The KNN classification algorithm

Let k be the number of nearest neighbors and D be the set of training examples.

1. **for** each test example z = (**x'**,y') **do**
2. Compute d(**x'**,**x**), the distance between z and every example, (**x**,y) ϵ D
3. Select $D_2 \subseteq D$, the set of k closest training examples to z.
4. $y' = \underset{v}{\operatorname{argmax}} \sum_{(x_i,y_i) \in D_z} I(v = y_i)$
5. **end for**

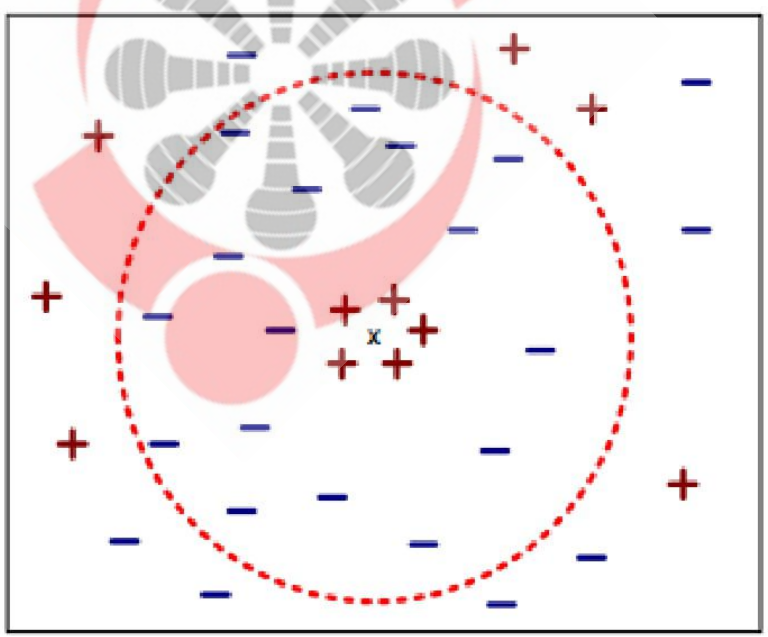
# KNN Classification




INNOMATICS
RESEARCH LABS

# Nearest Neighbor Classification

- 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



# Nearest Neighbor Classification

- Scaling issues
  - Attributes may have to be scaled to prevent distance measures from being dominated by one of the attributes
  - Example:
    - Height of a person may vary from 1.5m to 1.8m
    - Weight of a person may vary from 60 KG to 100KG
    - Income of a person may vary from Rs10K to Rs 2 Lakh

# Nearest Neighbor Classification

- Problem with Euclidean measure:
  - High dimensional data
    - curse of dimensionality: all vectors are almost equidistant to the query vector
  - Can produce undesirable results

| | | | |
|---|---|---|---|
| 1 1 1 1 1 1 1 1 1 1 1 0 | | 1 0 0 0 0 0 0 0 0 0 0 0 | |
| | vs | | |
| 0 1 1 1 1 1 1 1 1 1 1 1 | | 0 0 0 0 0 0 0 0 0 0 0 1 | |
| d = 1.4142 | | d = 1.4142 | |

    - Solution: Normalize the vectors to unit length

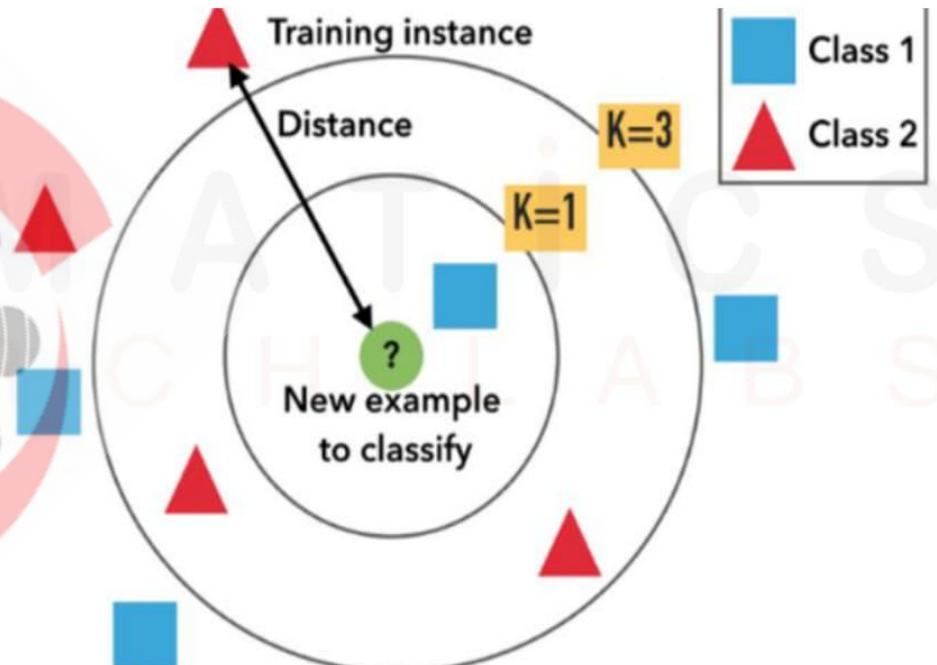
# Nearest Neighbor Classification

- k-NN classifiers are lazy learners
  - It does not build models explicitly
  - Unlike eager learners such as decision tree induction and rule-based systems
  - Classifying unknown records are relatively expensive

# Neighborhoods As Predictors

INNOMATICS RESEARCH LABS

- The test sample (green circle) should be classified either to the first class of blue squares or to the second class of red triangles.
- If k = 3 (solid line circle) it is assigned to the second class because there are 2 triangles and only 1 square inside the inner circle.
- If k = 5 (dashed line circle) it is assigned to the first class (3 squares vs. 2 triangles inside the outer circle).



# K-Nearest Neighbor algorithm

KNN is also a **lazy** algorithm (as opposed to an *eager* algorithm).

This means is that it does not use the training data points to do any *generalization*. In other words, there is *no explicit training phase* or it is very minimal. This also means that the training phase is pretty fast . Lack of generalization means that KNN keeps all the training data.

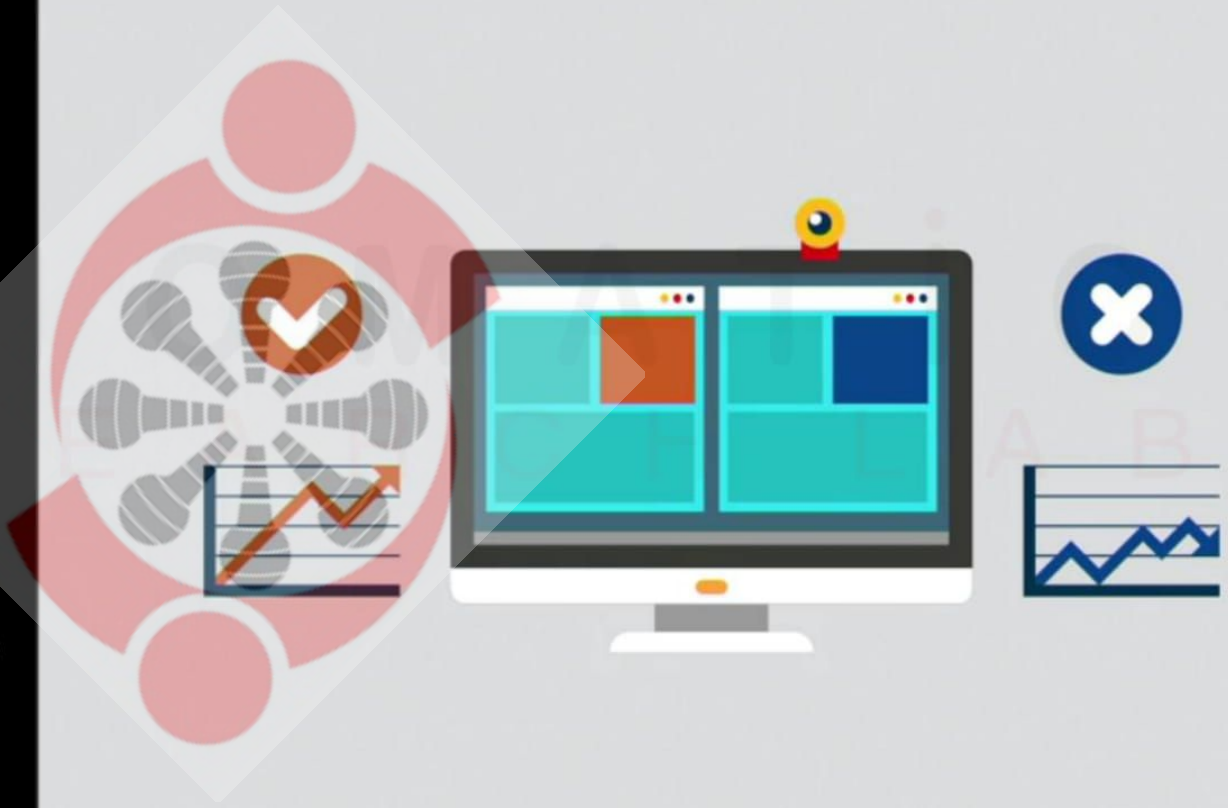
- Most basic instance-based method
- Data are represented in a vector space
- Supervised learning

## k-NN

- k = number of neighbors
- Instance-based learning
- “Lazy learner”
- Very simple algorithm

## Distance

- j predictor variables
- Euclidean distance in j-dimensional space
- Overlap metric
- Reduce dimensions before k-NN



# How to determine the good value for k?

- Determined experimentally
- Start with k=1 and use a test set to validate the error rate of the classifier
- Repeat with k=k+2
- Choose the value of k for which the error rate is minimum
- Note: k should be odd number to avoid ties

## Choosing k

- How many neighbors?
- More = smoother
- Risk of random noise and misclassification
- Can weight neighbors
- Many variations

# When to Consider Nearest Neighbors

Advantages:

- Training is very fast
- Learn complex target functions
- Do not lose information

Disadvantages:

- Slow at query time
- Presorting and indexing training samples into search trees reduces time
- Easily fooled by irrelevant features (attributes)

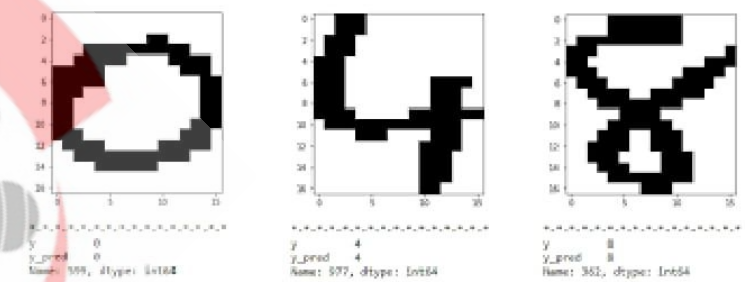
Need:

- Instances map to points in $R_d$
- Less than 20 features (attributes) per instance, typically normalized
- Lots of training data

# Applications and of KNN

- Credit ratings — collecting financial characteristics vs. comparing people with similar financial features to a database.
- By the very nature of a credit rating, people who have similar financial details would be given similar credit ratings.
- Therefore, they would like to be able to use this existing database to predict a new customer's credit rating, without having to perform all the calculations.

- Classing a potential voter to a “will vote” or “will not vote”, or to “vote Democrat” or “vote Republican”.



- Handwriting detection (like OCR), image recognition and even video recognition.

# Some pros and cons of KNN

**Pros:**

- No assumptions about data — useful, for example, for nonlinear data
- Simple algorithm — to explain and understand/interpret
- High accuracy (relatively) — it is pretty high but not competitive in comparison to better supervised learning models
- Versatile — useful for classification or regression

**Cons:**

- Computationally expensive — because the algorithm stores all of the training data
- High memory requirement
- Stores all (or almost all) of the training data
- Prediction stage might be slow (with big N)
- Sensitive to irrelevant features and the scale of the data

# Quick summary of KNN

The algorithm can be summarized as:

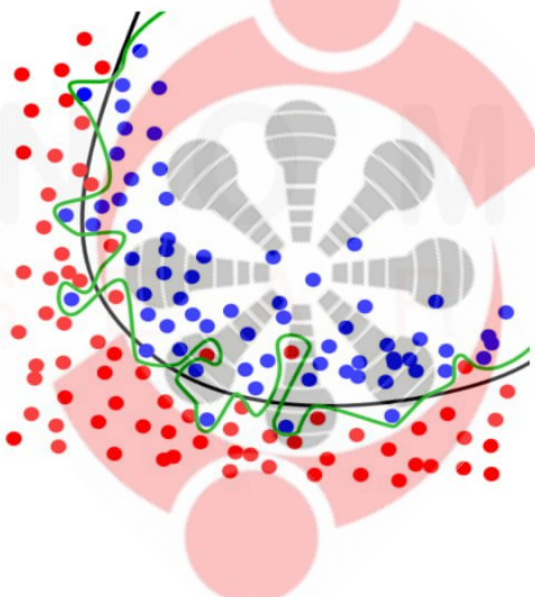
1. A positive integer k is specified, along with a new sample
2. We select the k entries in our database which are closest to the new sample
3. We find the most common classification of these entries
4. This is the classification we give to the new sample

A few other features of KNN:

- KNN stores the entire training dataset which it uses as its representation.
- KNN does not learn any model.
- KNN makes predictions just-in-time by calculating the similarity between an input sample and each training instance.

# Problems with training and testing on the same data

- Goal is to estimate likely performance of a model on out-of-sample data
- But, maximizing training accuracy rewards overly complex models that won't necessarily generalize
- Unnecessarily complex mo



- Green line (decision boundary): overfit
    - Your accuracy would be high but may not generalize well for future observations
    - Your accuracy is high because it is perfect in classifying your training data but not out-of-sample data
- Black line (decision boundary): just right
    - Good for generalizing for future observations