# Instance-based Classifiers

Anna Monreale
Computer Science Department

Introduction to Data Mining, 2nd Edition
Chapter 5.2

# Instance-based Classifiers

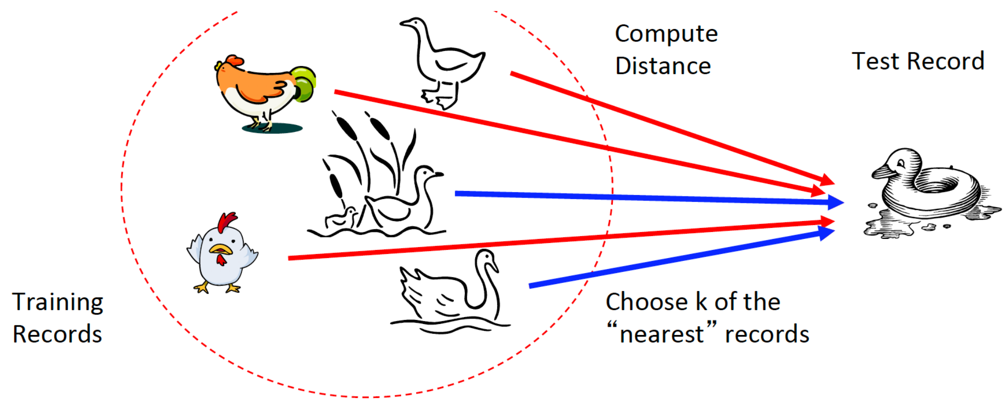
- Instead of performing explicit generalization, compare new instances with instances seen in training, which have been stored in memory.
- Sometimes called **memory-based** learning.
- **Advantages**
  - Adapt its model to previously unseen data by storing a new instance or throwing an old instance away.
- **Disadvantages**
  - Lazy learner: it does not build a model explicitly.
  - Classifying unknown records is relatively expensive: in the worst case, given *n* training items, the complexity of classifying a single instance is *O(n)*.

# Nearest-Neighbor Classifier (K-NN)

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

Requires three things

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



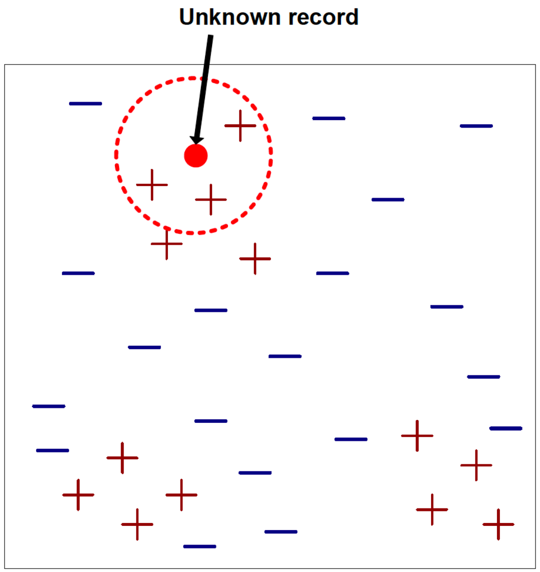
# Nearest-Neighbor Classifier (K-NN)

**Algorithm 5.2** The $k$-nearest neighbor classification algorithm.

Let $k$ be the number of nearest neighbors and $D$ be the set of training examples.<br>
**for** each test example $z = (\mathbf{x}', y')$ **do**<br>
    Compute $d(\mathbf{x}', \mathbf{x})$, the distance between $z$ and every example, $(\mathbf{x}, y) \in D$.<br>
    Select $D_z \subseteq D$, the set of $k$ closest training examples to $z$.<br>
    $y' = \underset{v}{\operatorname{argmax}} \sum_{(\mathbf{x}_i, y_i) \in D_z} I(v = y_i)$<br>
**end for**

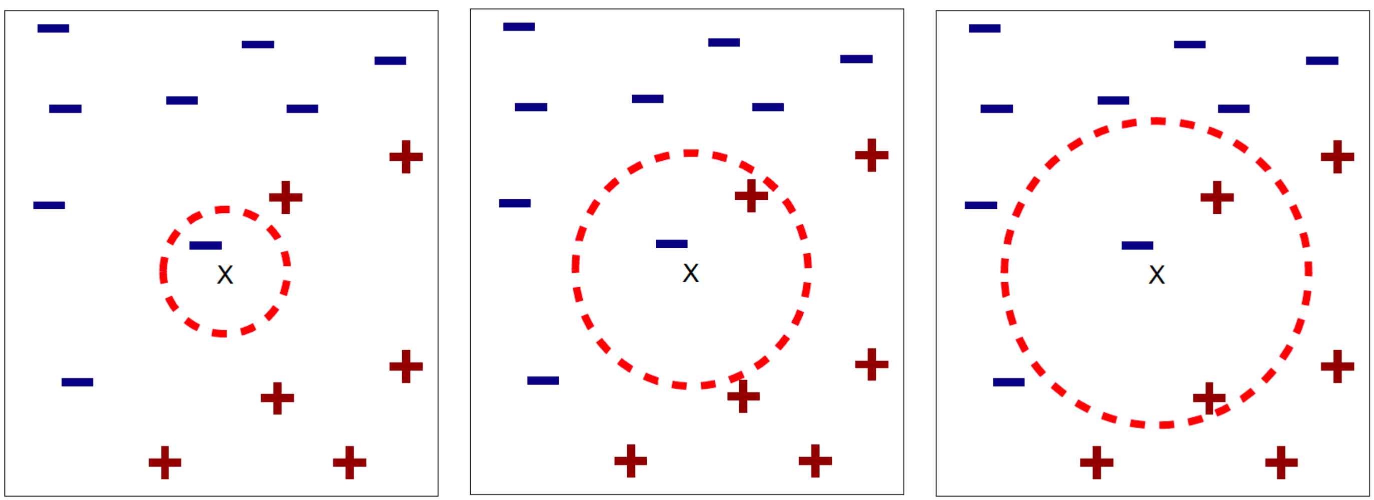
Given a set of training records (memory), and a test record:

1. **Compute the distances** from the records in the training to the test.
2. **Identify the k "nearest" records**.
3. Use class labels of nearest neighbors to **determine the class label** of unknown record (e.g., by taking majority vote).



# Definition of Nearest Neighbor

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



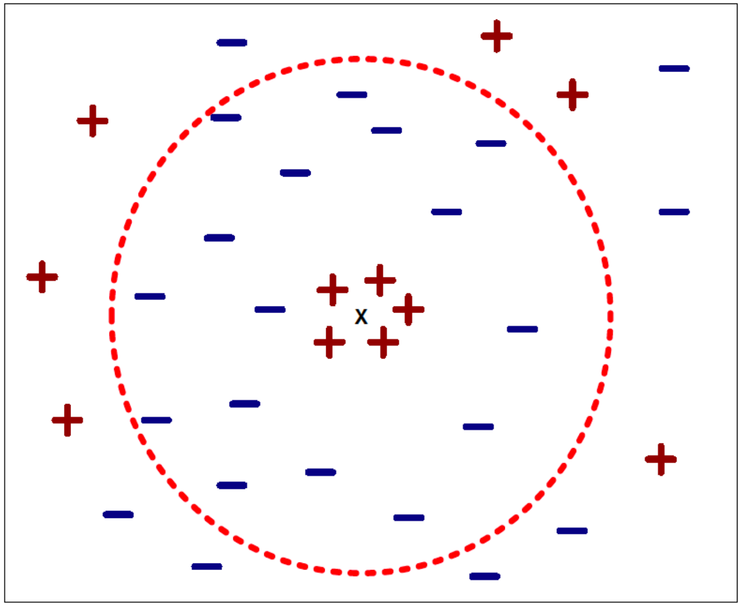

(a) 1-nearest neighbor (b) 2-nearest neighbor (c) 3-nearest neighbor

# Choosing the Value of K

- If k is too **small**, it is sensitive to **noise** points and it can lead to overfitting to the noise in the training set.

- If k is too **large**, the neighborhood may **include** points from **other classes**.

- General practice **k = sqrt(N)** where N is the number of samples in the training dataset.



# Nearest Neighbor Classification

Compute distance between two points:

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

Determine the class from nearest neighbors

- take the majority vote of class labels among the k nearest neighbors
- weigh the vote according to distance (e.g. weight factor, w = $1/d^2$)

$$\text{Distance-Weighted Voting: } y' = \underset{v}{\operatorname{argmax}} \sum_{(\mathbf{x}_i, y_i) \in D_z} w_i \times I(v = y_i).$$

# Dimensionality and Scaling Issues

- Problem with Euclidean measure: high dimensional data can cause curse of dimensionality.
  - Solution: normalize the vectors to unit length
    - Vector (3,4) becomes (0.6, 0.98) because length of the vector is SQRT($3^2 + 4^2$)=5
- 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 10kg to 200kg
  - income of a person may vary from $10K to $1M

# Parallel Exemplar-Based Learning System (PEBLS)

- PEBLS is a nearest-neighbor learning system (k=1) designed for applications where the instances have symbolic feature values
- Works with both continuous and nominal features.
- For nominal features, the distance between two nominal values is computed using Modified Value Difference Metric (MVDM)

$$d(V_1, V_2) = \sum_i \left| \frac{n_{1i}}{n_1} - \frac{n_{2i}}{n_2} \right|$$

- Where $n_1$ is the number of records that consists of nominal attribute value $V_1$ and $n_{1i}$ is the number of records whose target label is class $i$.

# Distance Between Nominal Attribute Values

- d(Status=Single, Status=Married) = | 2/4 – 0/4 | + | 2/4 – 4/4 | = 1
- d(Status=Single, Status=Divorced) = | 2/4 – 1/2 | + | 2/4 – 1/2 | = 0
- d(Status=Married, Status=Divorced) = | 0/4 – 1/2 | + | 4/4 – 1/2 | = 1
- d(Refund=Yes, Refund=No) = | 0/3 – 3/7 | + | 3/3 – 4/7 | = 6/7

| Tid | Refund | Marital Status | Taxable Income | Cheat |
|---|---|---|---|---|
| 1 | Yes | Single | 125K | No |
| 2 | No | Married | 100K | No |
| 3 | No | Single | 70K | No |
| 4 | Yes | Married | 120K | No |
| 5 | No | Divorced | 95K | Yes |
| 6 | No | Married | 60K | No |
| 7 | Yes | Divorced | 220K | No |
| 8 | No | Single | 85K | Yes |
| 9 | No | Married | 75K | No |
| 10 | No | Single | 90K | Yes |

| Class | Marital Status | | |
|---|---|---|---|
| | Single | Married | Divorced |
| Yes | 2 | 0 | 1 |
| No | 2 | 4 | 1 |

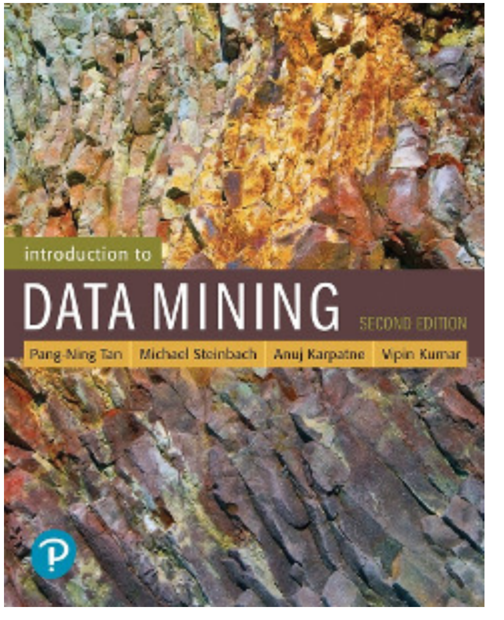
| Class | Refund | |
|---|---|---|
| | Yes | No |
| Yes | 0 | 3 |
| No | 3 | 4 |

# Characteristics of Nearest Neighbor Classifiers

- Instance-based learner: makes predictions without maintaining abstraction, i.e., building a model like decision trees.
- It is a lazy learner: classifying a test example can be expensive because need to compute the proximity values between test and training examples.
- In contrast **eager** learners **spend time** in **building** the model but then the classification is fast.
- Make their prediction on local information and for low *k* they are susceptible to noise.
- Can produce wrong predictions if inappropriate distance functions and/or preprocessing steps are performed.

# References

- Nearest Neighbor classifiers. Chapter 5.2. Introduction to Data Mining.

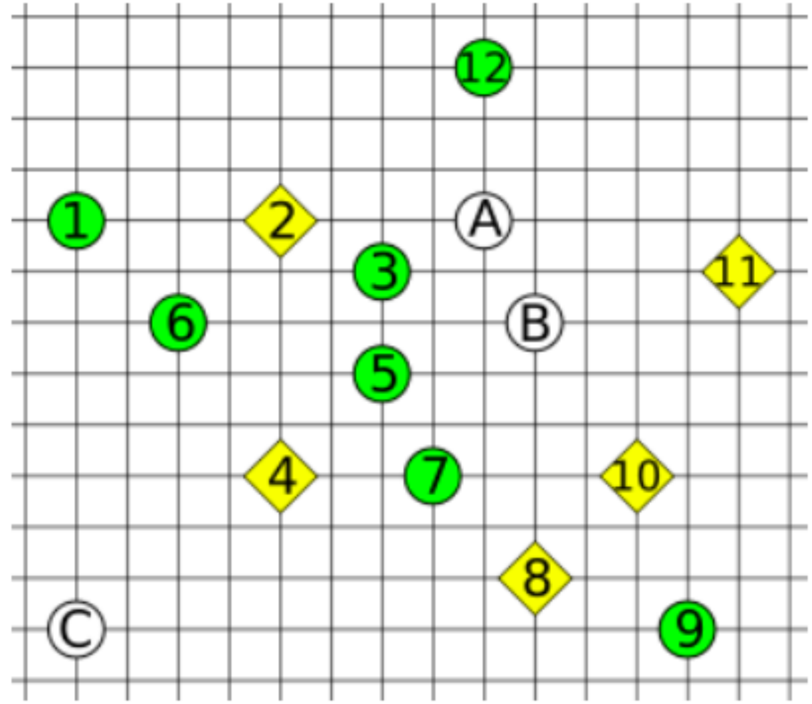


# EXERCISES - KNN

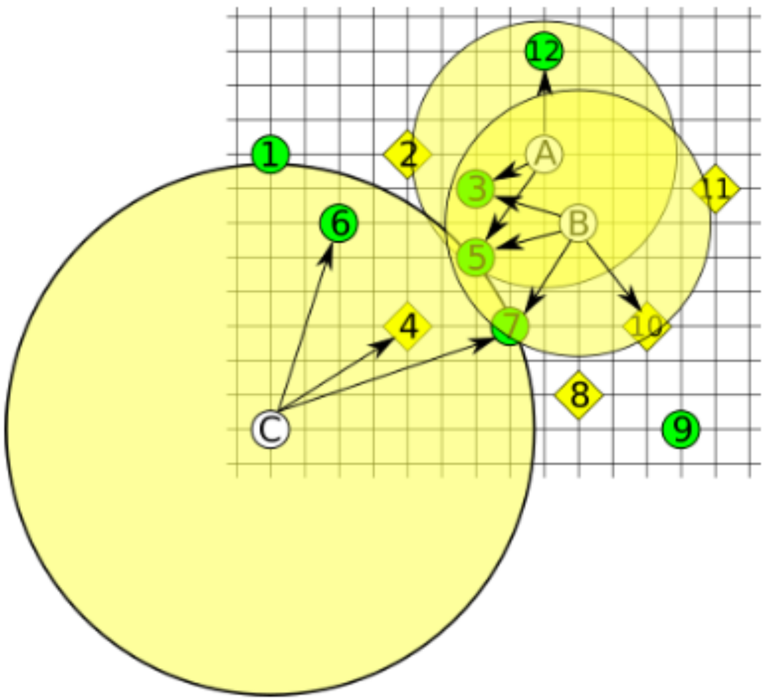
b) k-NN (**3 points**)

Given the training set on the right, composed of elements numbered from 1 to 12, and labelled as circles and diamonds, use it to classify the remaining 3 elements (letters A, B and C) using a k-NN classifier with k=3. For each point to classify, list the points of the dataset that belong to its k-NN set.

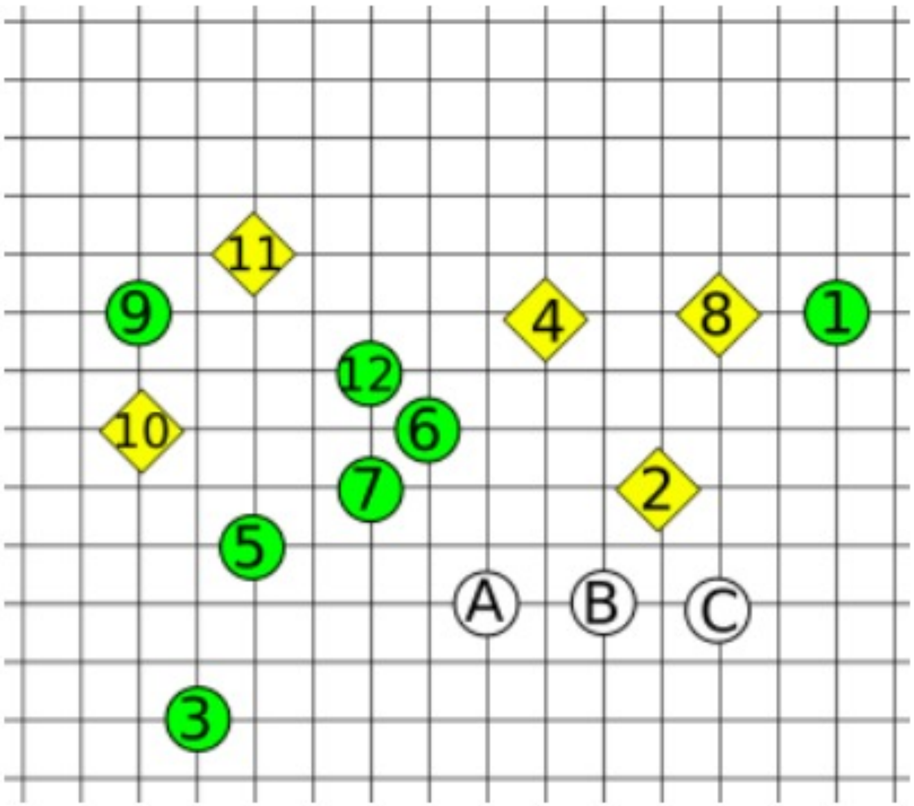
Notice: A, B and C belong to the test set, not to the training set. Also, the Euclidean distance should be used.




**Answer:**
**kNN(A) = {3, 5, 12} → CIRCLE**
**kNN(B) = { 3, 5, 7, 10 } → CIRCLE**
**kNN(C) = { 4, 6, 7 } → CIRCLE**



Given the training set on the right, composed of elements numbered from 1 to 12, and labelled as circles and diamonds, use it to classify the remaining 3 elements (letters A, B and C) using a k-NN classifier with k=3.
For each point to classify, list the points of the dataset that belong to its k-NN set.
Notice: A, B and C belong to the test set, not to the training set. Also, the Euclidean distance should be used.



# k-Nearest Neighbor Classifier

A medical expert is going to build up a case-based reasoning system for diagnosis tasks. Cases correspond to individual persons where the case problem parts are made up of a number of features describing possible symptoms and the solution parts represent the diagnosis (classification of disease). The case base contains the seven cases provided in the table below.

| Training | Fever | Vomiting | Diarrhea | Shivering | Classification |
|---|---|---|---|---|---|
| $c_1$ | no | no | no | no | healty (H) |
| $c_2$ | average | no | no | no | influenza (I) |
| $c_3$ | high | no | no | yes | influenza (I) |
| $c_4$ | high | yes | yes | no | salmonella poisoning (S) |
| $c_5$ | average | no | yes | no | salmonella poisoning (S) |
| $c_6$ | no | yes | yes | no | bowel inflammation (B) |
| $c_7$ | average | yes | yes | no | bowel inflammation (B) |

**Similarity provided by an expert**

$sim_F$

| q \ c | no | avg | high |
|---|---|---|---|
| no | 1.0 | 0.7 | 0.2 |
| avg | 0.5 | 1.0 | 0.8 |
| high | 0.0 | 0.3 | 1.0 |

$sim_V = sim_D = sim_{Sh}$

| q | yes | no |
|---|---|---|
| yes | 1.0 | 0.0 |
| no | 0.2 | 1.0 |

Weights

$w_F = 0.3$

$w_V = 0.2$

$W_D = 0.2$

$w_{Sh} = 0.3$

**Classify the new instance q = (high; no; no; no) by applying the KNN algorithm with K=1,2,3**

Calculate the similarity between all cases from the case base and the new instance q = **(**high; no; no; no**)**

**c1 = (no; no; no; no):**

Sim(q; c1) = 0.3*0.0 + 0.2 *1.0 + 0.2*1.0 + 0.3* 1.0 = 0.70

**c2 = (average; no; no; no):**

Sim(q; c2) = 0.3* 0.3 + 0.2 *1.0 + 0.2*1.0 + 0.3*1.0 = 0.79

**c3 = (high; no; no; yes)**

Sim(q; c3) = 0.3*1.0 + 0.2*1.0 + 0.2*1.0 + 0.3*0.2 = 0.76

**c4 = (high; yes; yes; no):**

Sim(q; c4) = 0.3*1.0 + 0.2*0.2 + 0.2*0.2 + 0.3*1.0 = 0.68

**c5 = (average; no; yes; no):**

Sim(q; c5) = 0.3*0.3 + 0.2*1.0 + 0.2*0.2 + 0.3*1.0 = 0.63

**c6 = (no; yes; yes; no):**

Sim(q; c6) = 0.3*0.0 + 0.2*0.2 + 0.2*0.2 + 0.3*1.0 = 0.28

**c7 = (average; yes; yes; no):**

Sim(q; c7) = 0.3*0.3 + 0.2*0.2 + 0.2*0.2 + 0.3*1.0 = 0.47

$sim_F$

| q \ c | no | avg | high |
|---|---|---|---|
| no | 1.0 | 0.7 | 0.2 |
| avg | 0.5 | 1.0 | 0.8 |
| high | 0.0 | 0.3 | 1.0 |

$sim_V$=$sim_D$=$sim_{Sh}$

| q \ | yes | no |
|---|---|---|
| yes | 1.0 | 0.0 |
| no | 0.2 | 1.0 |

Weights

$w_F$=0.3

$w_V$=0.2

$W_D$=0.2

$w_{Sh}$=0.3

# KNN Classification for K=1

**c1 = (no; no; no; no):**

Sim(q; c1) = 0.3*0.0 + 0.2 *1.0 + 0.2*1.0 + 0.3* 1.0 = 0.70

**c2 = (average; no; no; no):**

Sim(q; c2) = 0.3* 0.3 + 0.2 *1.0 + 0.2*1.0 + 0.3*1.0 = 0.79

**c3 = (high; no; no; yes)**

Sim(q; c3) = 0.3*1.0 + 0.2*1.0 + 0.2*1.0 + 0.3*0.2 = 0.76

**c4 = (high; yes; yes; no):**

Sim(q; c4) = 0.3*1.0 + 0.2*0.2 + 0.2*0.2 + 0.3*1.0 = 0.68

**c5 = (average; no; yes; no):**

Sim(q; c5) = 0.3*0.3 + 0.2*1.0 + 0.2*0.2 + 0.3*1.0 = 0.63

**c6 = (no; yes; yes; no):**

Sim(q; c6) = 0.3*0.0 + 0.2*0.2 + 0.2*0.2 + 0.3*1.0 = 0.28

**c7 = (average; yes; yes; no):**

Sim(q; c7) = 0.3*0.3 + 0.2*0.2 + 0.2*0.2 + 0.3*1.0 = 0.47

$sim_F$

| q \ c | no | avg | high |
|---|---|---|---|
| no | 1.0 | 0.7 | 0.2 |
| avg | 0.5 | 1.0 | 0.8 |
| high | 0.0 | 0.3 | 1.0 |

Weights

$w_F$=0.3

$w_V$=0.2

$W_D$=0.2

$w_{Sh}$=0.3

**Class: Influenza**

# KNN Classification for K=2

**c1 = (no; no; no; no):**

Sim(q; c1) = 0.3*0.0 + 0.2 *1.0 + 0.2*1.0 + 0.3* 1.0 = 0.70

**c2 = (average; no; no; no):**

Sim(q; c2) = 0.3* 0.3 + 0.2 *1.0 + 0.2*1.0 + 0.3*1.0 = 0.79

**c3 = (high; no; no; yes):**

Sim(q; c3) = 0.3*1.0 + 0.2*1.0 + 0.2*1.0 + 0.3*0.2 = 0.76

**c4 = (high; yes; yes; no):**

Sim(q; c4) = 0.3*1.0 + 0.2*0.2 + 0.2*0.2 + 0.3*1.0 = 0.68

**c5 = (average; no; yes; no):**

Sim(q; c5) = 0.3*0.3 + 0.2*1.0 + 0.2*0.2 + 0.3*1.0 = 0.63

**c6 = (no; yes; yes; no):**

Sim(q; c6) = 0.3*0.0 + 0.2*0.2 + 0.2*0.2 + 0.3*1.0 = 0.28

**c7 = (average; yes; yes; no):**

Sim(q; c7) = 0.3*0.3 + 0.2*0.2 + 0.2*0.2 + 0.3*1.0 = 0.47

$sim_F$

| q \ c | no | avg | high |
|---|---|---|---|
| no | 1.0 | 0.7 | 0.2 |
| avg | 0.5 | 1.0 | 0.8 |
| high | 0.0 | 0.3 | 1.0 |

Weights

$w_F$=0.3

$w_V$=0.2

$W_D$=0.2

$w_{Sh}$=0.3

**C2: Influenza**

**C3: Influenza**

**Class: Influenza**

# KNN Classification for K=3

**c1 = (no; no; no; no):**

Sim(q; c1) = 0.3*0.0 + 0.2 *1.0 + 0.2*1.0 + 0.3* 1.0 = 0.70

**c2 = (average; no; no; no):**

Sim(q; c2) = 0.3* 0.3 + 0.2 *1.0 + 0.2*1.0 + 0.3*1.0 = 0.79

**c3 = (high; no; no; yes):**

Sim(q; c3) = 0.3*1.0 + 0.2*1.0 + 0.2*1.0 + 0.3*0.2 = 0.76

**c4 = (high; yes; yes; no):**

Sim(q; c4) = 0.3*1.0 + 0.2*0.2 + 0.2*0.2 + 0.3*1.0 = 0.68

**c5 = (average; no; yes; no):**

Sim(q; c5) = 0.3*0.3 + 0.2*1.0 + 0.2*0.2 + 0.3*1.0 = 0.63

**c6 = (no; yes; yes; no):**

Sim(q; c6) = 0.3*0.0 + 0.2*0.2 + 0.2*0.2 + 0.3*1.0 = 0.28

**c7 = (average; yes; yes; no):**

Sim(q; c7) = 0.3*0.3 + 0.2*0.2 + 0.2*0.2 + 0.3*1.0 = 0.47

$sim_F$

| q \ c | no | avg | high |
|---|---|---|---|
| no | 1.0 | 0.7 | 0.2 |
| avg | 0.5 | 1.0 | 0.8 |
| high | 0.0 | 0.3 | 1.0 |

Weights

$w_F=0.3$

$w_V=0.2$

$W_D=0.2$

$w_{Sh}=0.3$

**C1: healty**

**C2: Influenza**

**C3: Influenza**

**Class: Influenza**