Instance-based Learning

CS576 Machine Learning

Dr. Jin S. Yoo, Professor
Department of Computer Science
Purdue University Fort Wayne

Reference

- Kelleher et al., Fundamentals of ML, Ch 5
- Mitchell, Machine Learning, Ch 8.1-8.2

Outline

- Introduction to Instance-Based Learning
- Similarity (Distance) Metrics
- Standard Approach: Nearest Neighbor Learner
- *k*-Nearest Neighbor Learner
- Data Normalization
- Predicting Continuous Targets
- Other Measures of Similarity
- Feature Selection
- Summary

Instance-Based Learning

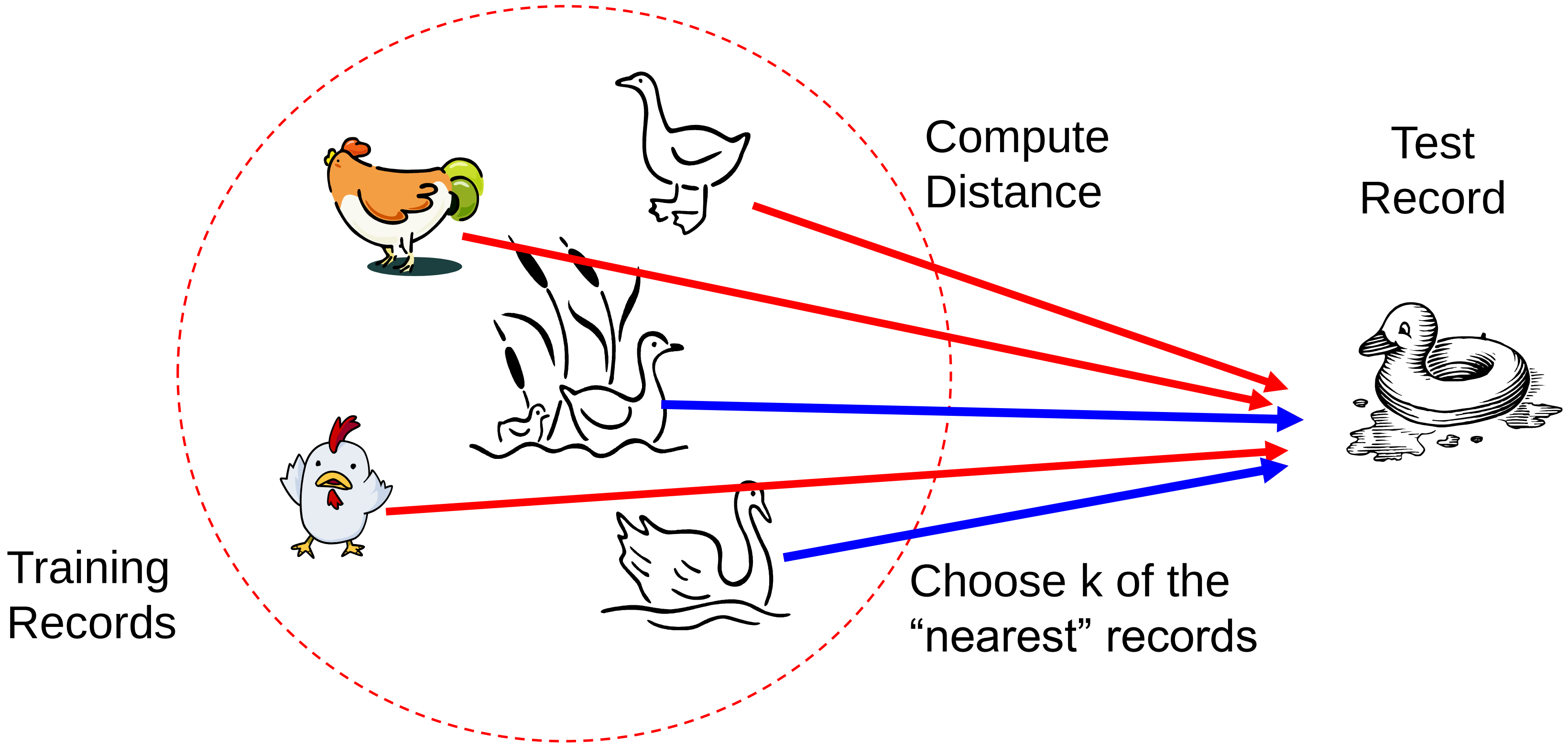
- **Instance-based learning** (also known as similarity-based learning) is a machine learning approach where, rather than explicitly building a general model during the training phase, the method stores the training instances and directly references them for making predictions.
- Instance-based learning is frequently labeled as "**lazy**" learning because computations are deferred until a prediction is needed.
- This approach is also commonly referred to as "**memory-based**" learning, as predictions are derived directly from the training data.

Instance-Based Learning: Principle

- Instance-based prediction models attempt to mimic a very human way of reasoning by basing predictions for a target feature value on the most similar instances in memory
- **Instance-based learning operates on the premise that input instances with similar attributes yield similar output values.**

Example

- Imagine classifying an unfamiliar animal by comparing its features with those of animals you've previously encountered.
- E.g., If an animal walks like a duck and quacks like a duck, then it’s probably a duck



Instance-Based Learning Methods

- ***k*-Nearest Neighbors (*k*-NN)**
  - uses *k* “closest” points (nearest neighbors) for performing classification
  - One of the simplest and best known machine learning algorithms for instance-based reasoning.
- Locally Weighted Learning (LWL)
  - A variation of k-NN used primarily for regression.
- Case-Based Reasoning (CBR)
  - Used mostly in expert systems and decision support, CBR involves storing past cases and their solutions.
- Radial Basis Function (RBF) Networks
- Learning Vector Quantization (LVQ)
- Self-Organizing Maps (SOM)

Eager vs. Lazy Learners

- **Eager Learners:**
  - **Approach**: Eager learners build a prediction model based on the training data before receiving a test instance to make predictions. Once the model is built, no further reference is made to the training data.
  - **Global Approximation:** The learner creates a single comprehensive model that applies to the entire input space.
  - **Examples**: Decision trees (like ID3), Neural Networks, etc.
- **Lazy Learners**:
  - **Approach**: Lazy learners don't build an explicit model until they're required to make a prediction. When given a test instance, they search through the training data to find and use the most relevant data to make a prediction.
  - **Local Approximations**: Instead of building a general model for the entire dataset, they create "local approximations" based on specific portions of the data that are most relevant to the current test instance.
  - **Examples**: k-Nearest Neighbors (k-NN)

Eager vs. Lazy Learners

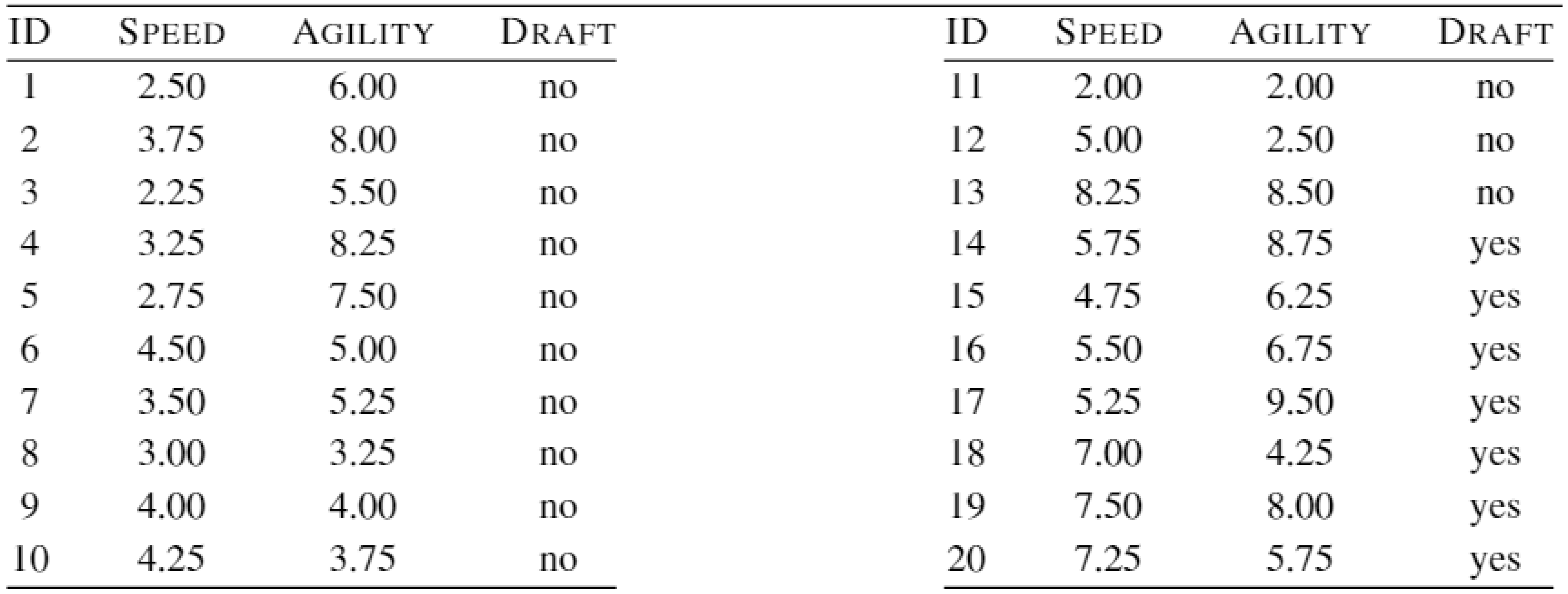
- **Eager Learners:**
  - **Pros**: They tend to be faster at making predictions since the model is already built and ready to use.
  - **Cons**: They might not be as adaptive to new, unseen data or outliers as they've committed to a single global hypothesis.
- **Lazy Learners**:
  - **Pros**: They can be more accurate in certain situations, especially if there's a lot of variability in the data, since they adapt to the data as needed.
  - **Cons**: They can be computationally intensive and slow during prediction time since they require access to the entire training dataset.

Outline

Concept of Similarity

- The concept of **similarity** (or **distance**) is crucial in instance-based learning, since during prediction, an instance-based learning algorithm scans the training dataset for the most similar instances and leverages them to predict the output for the given query instance.
- Depending on the nature of the data and the specific problem, different measures are used.
  - Common measures of distance include the Euclidean distance or Manhattan distance.
- The choice of distance metric can be crucial to the performance of the algorithm.

Example: College Athlete Dataset



ID	SPEED	AGILITY	DRAFT
1	2.50	6.00	no
2	3.75	8.00	no
3	2.25	5.50	no
4	3.25	8.25	no
5	2.75	7.50	no
6	4.50	5.00	no
7	3.50	5.25	no
8	3.00	3.25	no
9	4.00	4.00	no
10	4.25	3.75	no
11	2.00	2.00	no
12	5.00	2.50	no
13	8.25	8.50	no
14	5.75	8.75	yes
15	4.75	6.25	yes
16	5.50	6.75	yes
17	5.25	9.50	yes
18	7.00	4.25	yes
19	7.50	8.00	yes
20	7.25	5.75	yes

Table. The SPEED and AGILITY ratings for 20 college athletes and whether they were drafted by a professional team.

Feature Space

- A **feature space** is an abstract *n*-dimensional space that is created by taking each of the descriptive features to be the axes of the reference space.
- Each instance is mapped to a point in the feature space based on the values of its descriptive features

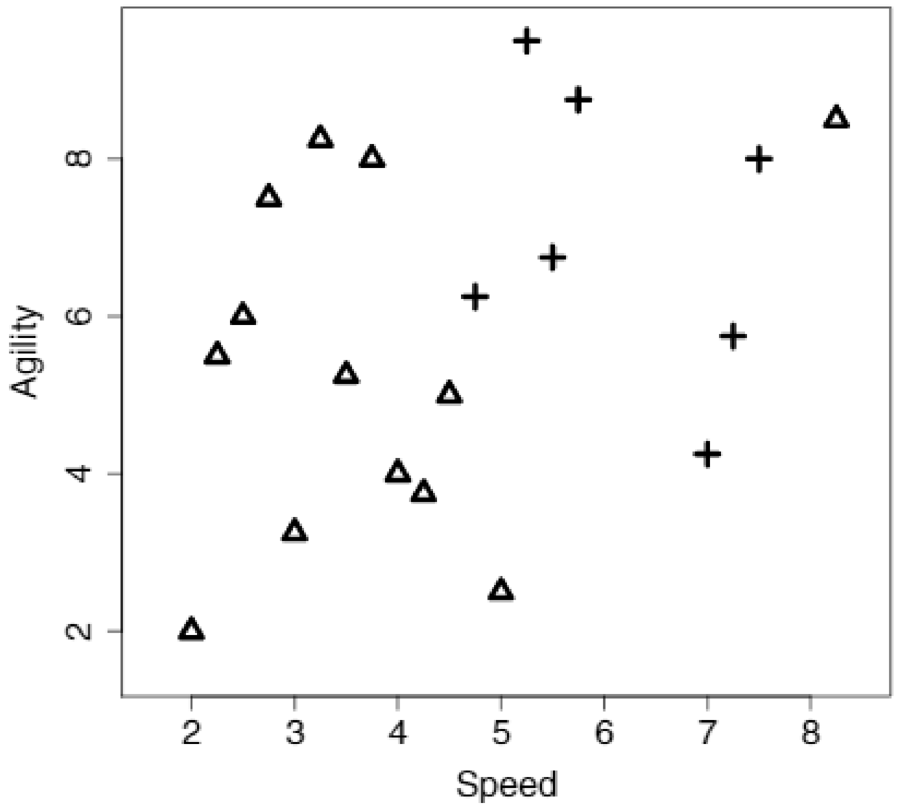


Figure: A feature space plot of the data with .△ for the 'Non-draft' instances and + for the 'Draft' instances.

Measuring Similarity using Distance Metric

- The simplest way to measure the **similarity** (**proximity**) between two instances is to measure the **distance** (**dissimilarity**) between the instances in a feature space.

- **Note**
  - The terms '**similarity**' and '**distance**' are often used interchangeably, as we frequently assess the similarity between two instances based on the distance between them in a feature space.
  - The key distinction to remember is that with distances, smaller values indicate instances are closer in a feature space, while larger similarity values suggest the same.

Distance Metrics

- One of the best-known distance metrics is **Euclidean distance**. The Euclidean distance between two instances a and b in a *n*-dimensional feature space is defined by:
  $$Euclidean(\mathrm{a},\mathrm{b}) = \sqrt{\sum_{i=1}^{n}(\mathrm{a}[i] - \mathrm{b}[i])^2}$$
- Another, less well known, distance measure is the **Manhattan** distance or **taxi-cab distance:**
  $$Manhattan(\mathrm{a},\mathrm{b}) = \sum_{i=1}^{n} abs(\mathrm{a}[\mathrm{i}] - \mathrm{b}[\mathrm{i}])$$
- The Euclidean and Manhattan distances are special cases of **Minkowski distance:**
  $$Minkowski(\mathrm{a},\mathrm{b}) = \left(\sum_{i=1}^{n} abs(\mathrm{a}[\mathrm{i}] - \mathrm{b}[\mathrm{i}])^p\right)^{\frac{1}{p}}$$
  - when *p* =1, the Manhattan distance
  - when *p*=2, the Euclidean distance
  - The larger the value of p the more emphasis is placed on the features with large differences in values because these differences are raised to the power of p

Manhattan vs Euclidean Distance

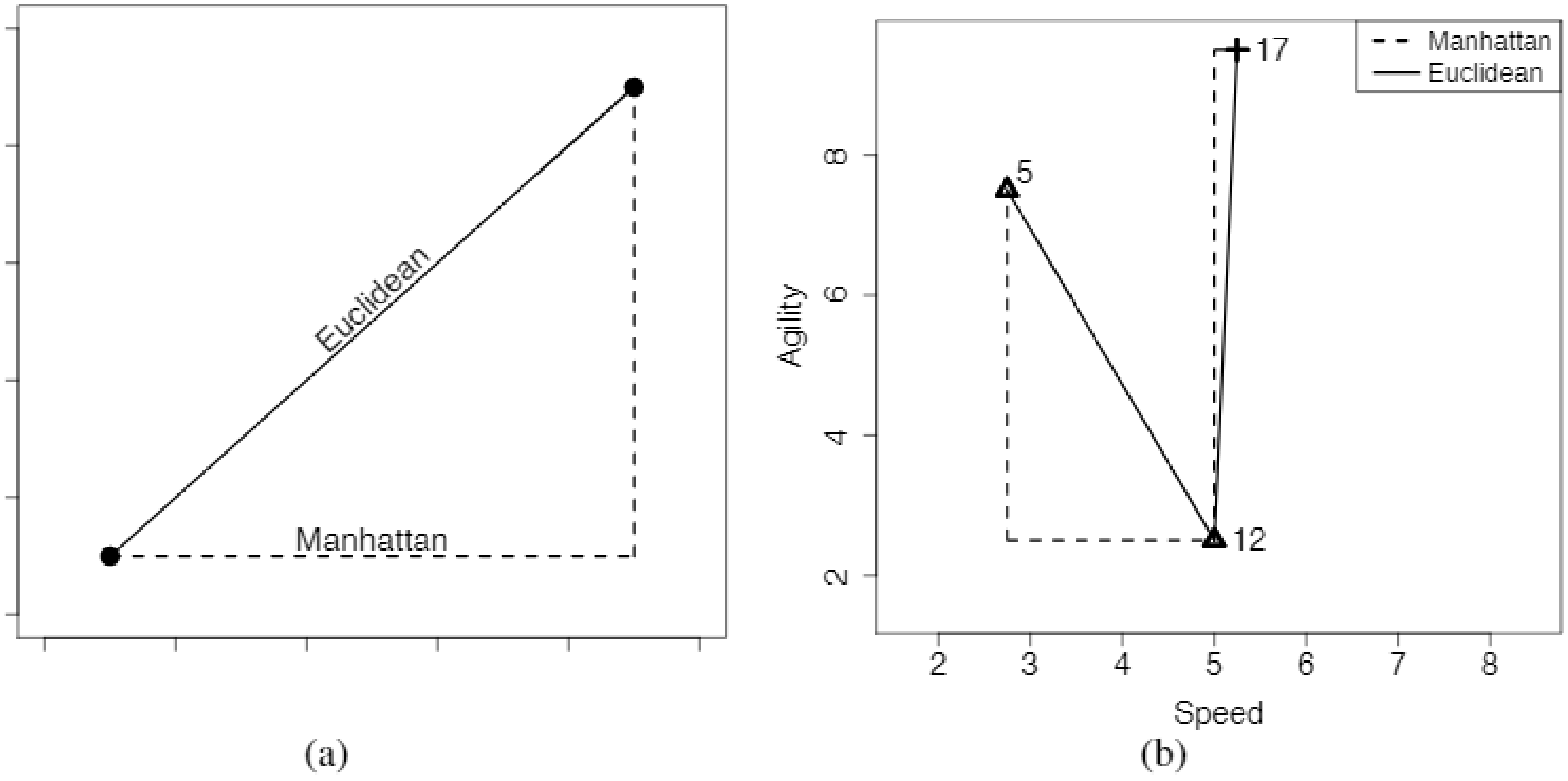


Figure. (a) A generalized illustration of the Manhattan and Euclidean distances between two points; and (b) a plot of the Manhattan and Euclidean distances between instances d_{12} and d_5 , and between d_{12} and d_{17}

Background Concept: Distance Metric

- A **distance metric**, $metric(\mathrm{a}, \mathrm{b})$, is a function that returns the distance between two instances, a and b.
- Mathematically, a **metric** must conform to the following four criteria:
  1. Non-negativity: $metric(\mathrm{a}, \mathrm{b}) \geq 0$
  2. Identity: $metric(\mathrm{a}, \mathrm{b}) = 0 \iff \mathrm{a} = \mathrm{b}$
  3. Symmetry: $metric(\mathrm{a}, \mathrm{b}) = metric(\mathrm{b}, \mathrm{a})$
  4. Triangular Inequality:

     $$metric(\mathrm{a}, \mathrm{b}) \leq metric(\mathrm{a}, c) + metric(\mathrm{b}, c)$$
- If the similarity doesn't meet the four criteria, it is commonly referred to as an **index**.

Outline

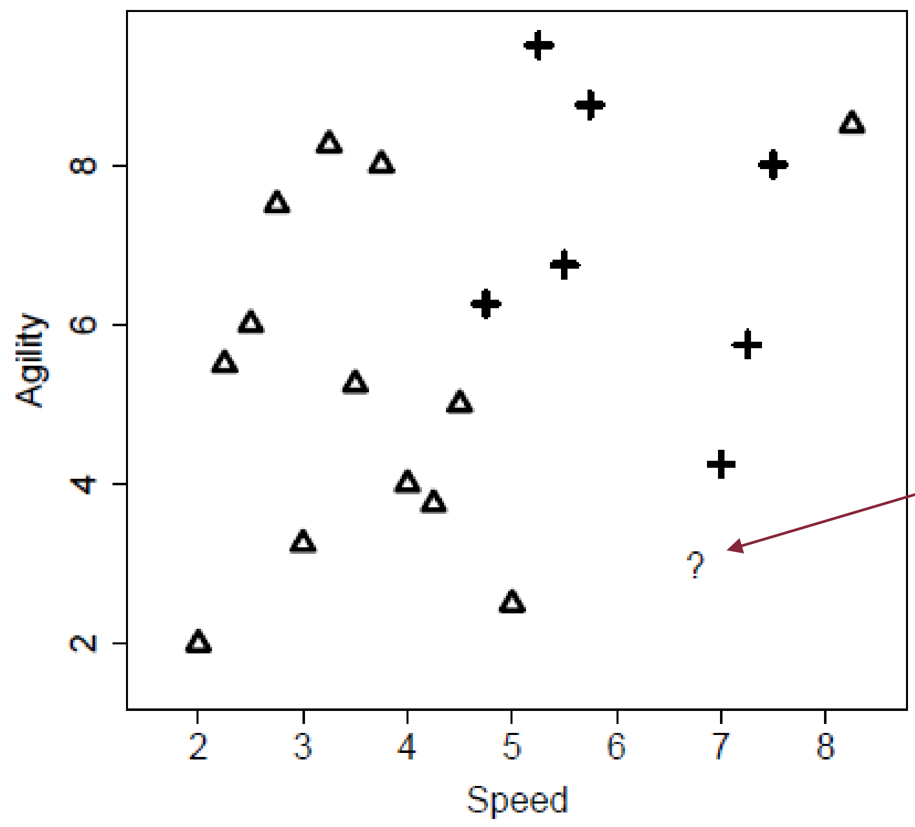
- Introduction to Instance-Based Learning
- Similarity (Distance) Metrics
- ☞ **Standard Approach: Nearest Neighbor Learner**
- *k*-Nearest Neighbor Learner
- Data Normalization
- Predicting Continuous Targets
- Other Measures of Similarity
- Feature Selection
- Summary

Nearest Neighbor Learner

- **Nearest Neighbor learner:** Given a new instance x_q, first locates nearest training instance x_n, then estimates $\hat{f}(x_q) \leftarrow f(x_n)$
- **Algorithm**
  - Given:
    - a set of training instances
    - a query instance to be classified
  - **Training phase**:
    - Just store all the training instances in memory
  - **Prediction phase**:
    1. Iterate across the training instances in memory and find the instance that is shortest distance from the query instance position in the feature space.
    2. Make a prediction for the query instance equal to the value of the target feature of the nearest neighbor.

Example

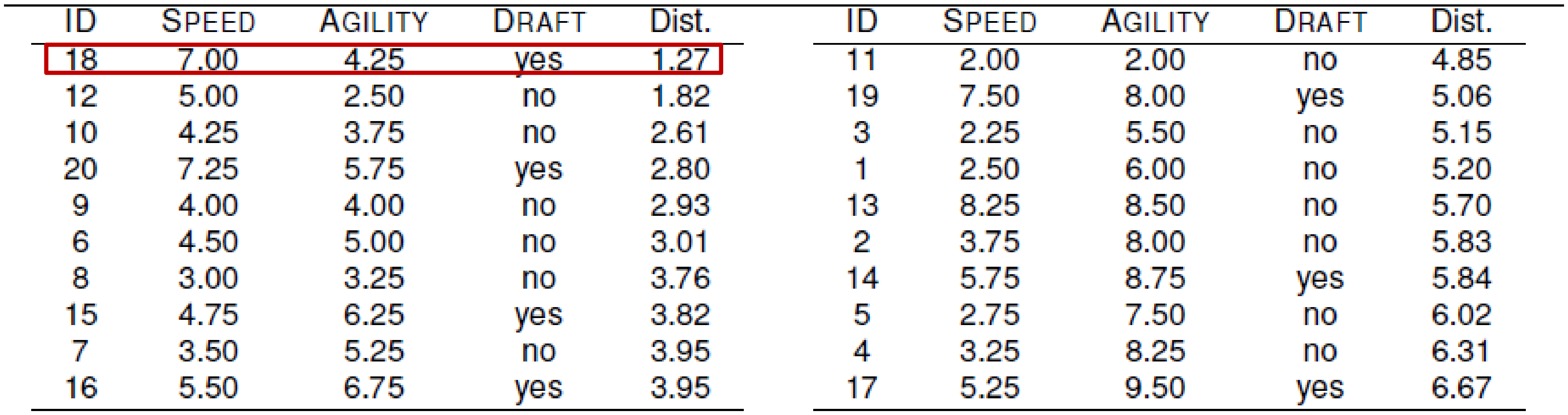
- Should we draft an athlete with the following profile: SPEED = 6.75, AGILITY = 3 ?



The query instance

Example (cont.)

- Compute the distances (Dist.) between the query instance with SPEED = 6.75 and AGILITY = 3.00 and each example in the training set.
- And then sort the training instances by distance.

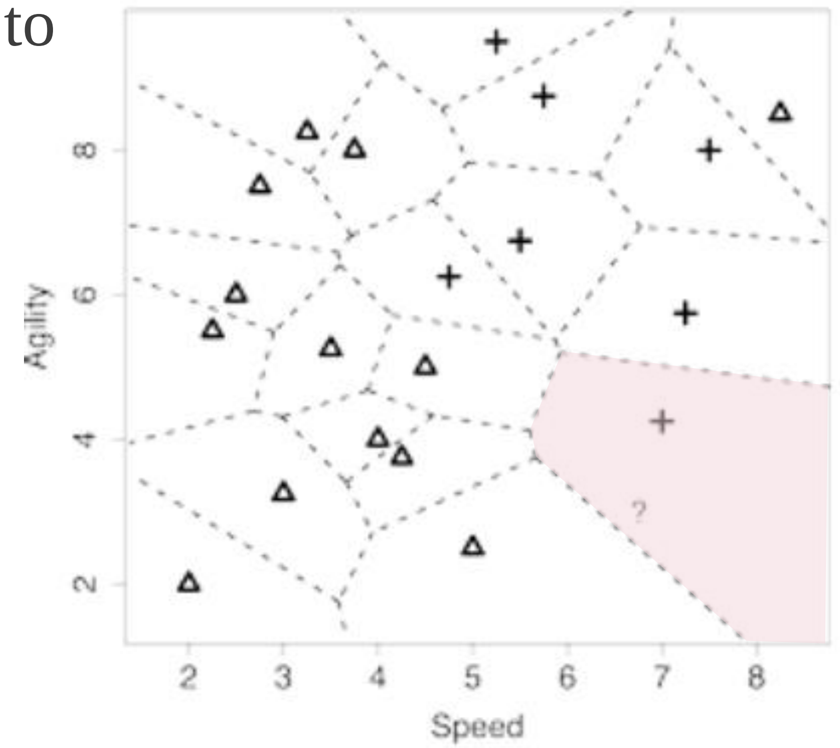


ID	SPEED	AGILITY	DRAFT	Dist.
18	7.00	4.25	yes	1.27
12	5.00	2.50	no	1.82
10	4.25	3.75	no	2.61
20	7.25	5.75	yes	2.80
9	4.00	4.00	no	2.93
6	4.50	5.00	no	3.01
8	3.00	3.25	no	3.76
15	4.75	6.25	yes	3.82
7	3.50	5.25	no	3.95
16	5.50	6.75	yes	3.95
11	2.00	2.00	no	4.85
19	7.50	8.00	yes	5.06
3	2.25	5.50	no	5.15
1	2.50	6.00	no	5.20
13	8.25	8.50	no	5.70
2	3.75	8.00	no	5.83
14	5.75	8.75	yes	5.84
5	2.75	7.50	no	6.02
4	3.25	8.25	no	6.31
17	5.25	9.50	yes	6.67

- The nearest neighbor with the shortest distance is ID18. It's DRAFT='yes'
- So, the query athlete is drafted.

Nearest Neighbor Search using Vornoi Partition

- There are many algorithms for nearest neighbor search
- One of the algorithms is to try to
  - Partition the feature space into what is known as a **Voronoi tessellation**
  - Decide which Voronoi region the query belongs to.



- In this example, the nearest neighbor classifier finds that the query is inside a Voronoi region defined by a training example with a target level of *yes* (+).

Voronoi Region and Decision Boundary

- A **decision boundary** is the boundary between regions of the feature space in which different target levels will be predicted.
- The decision boundary is generated by aggregating the neighboring Vornoi regions (**local models**) that make the same prediction.
- The nearest neighbor search algorithm using Vornoi partition can also create **a global model** based on the full dataset.
- The decision boundary is a global representation of the predictions associated with each example in the training set.

Example: Decision Boundary

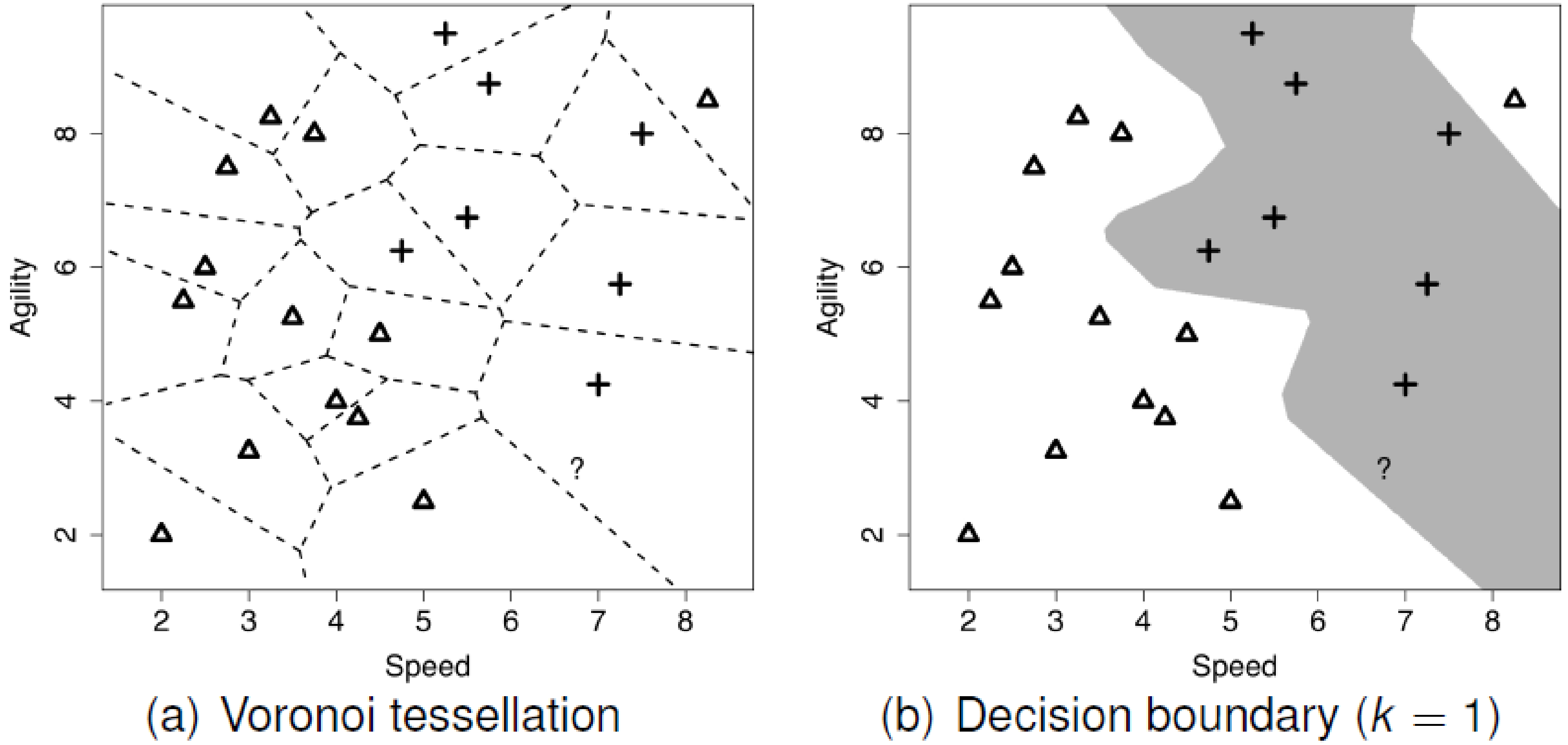
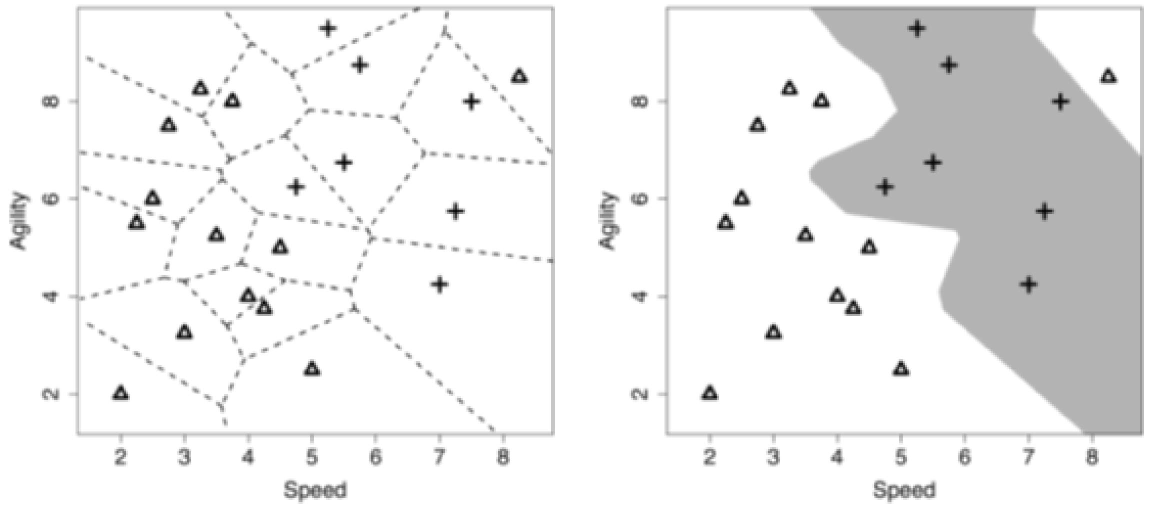


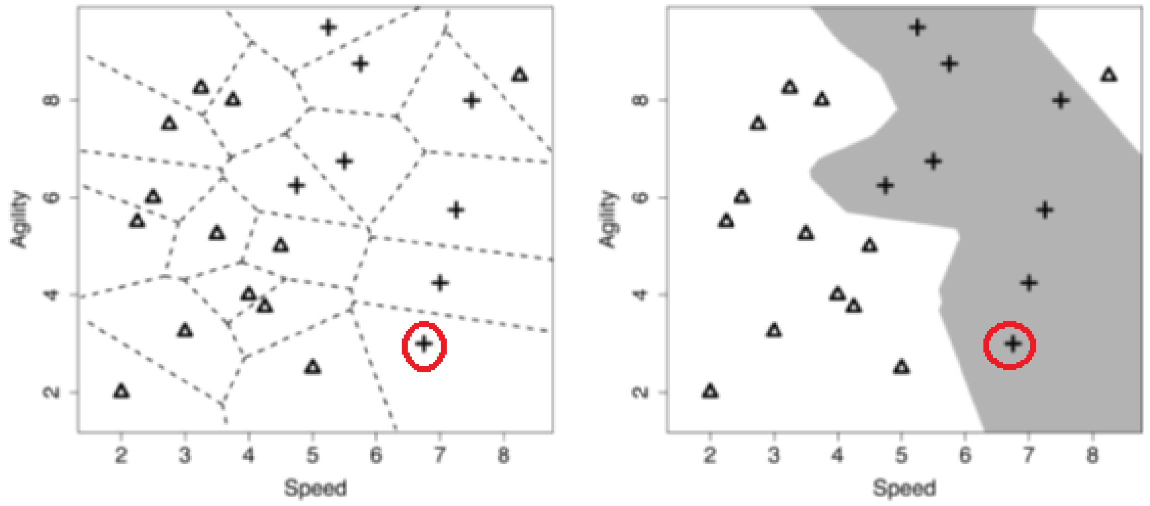
Figure: (a) The **Voronoi tessellation** of the feature space for the dataset with the position of the query represented by the **?** marker; (b) the **decision boundary** created by aggregating the neighboring Voronoi regions that belong to the same target level (label).

Updated Nearest Neighbor Model

- **It is relatively easy to update the nearest neighbor model.**
- E.g., When a new labeled instance with <SPEED= 6.75, AGILITY= 3, DRAFT=yes> is available, its nearby Vornoi regions are just changed



(a) Vornoi tesseliation and Decision boundary



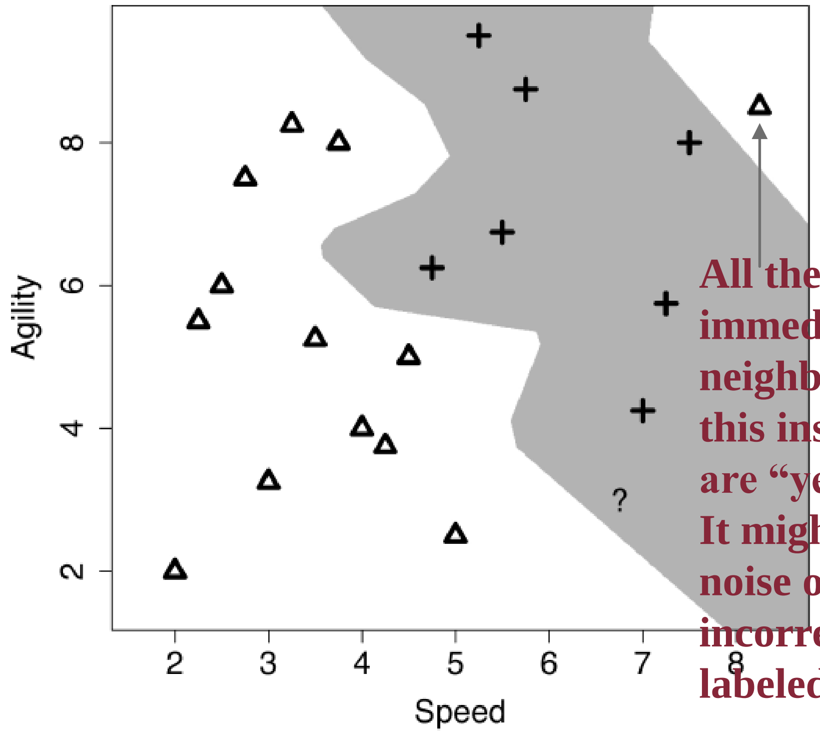
(b) Updated Vornoi tesseliation and Decision boundary after adding a new instance

Characteristics of Nearest Neighbor Learning

- **Strength**
  - Highly effective inference method for many practical and large problems
  - Training is very fast – because no model is required to fit
  - Simple but can learn complex target functions (e.g., decision boundary is very irregular)
  - When new labeled examples become available, it is relatively simple to update the model
- **Weakness**
  - Slow at query time (for the prediction of a new instance)
  - Easily fooled by irrelevant attributes due to distance-based measure.
  - Sensitive to noise

Noisy Data and Nearest Neighbor Leaner

- The standard nearest neighbor approach creates “local approximations” based on the training instance that is most relevant to the current test instance.



All the immediate neighbors of this instance are “yes”. It might be noise or incorrectly labeled.

- Consequently, the learner is **sensitive to noise.**
  - Any errors in the description or labeling of training data results in erroneous local models and hence incorrect predictions.

K-Nearest Neighbor Learner for Classification

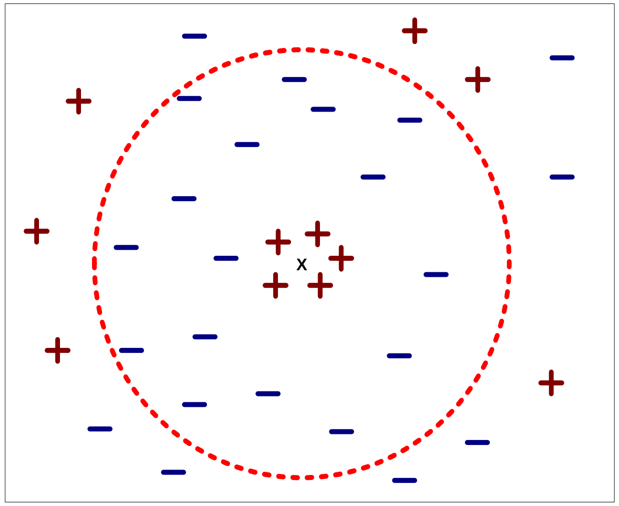
- The ***k*-nearest neighbor learner** predicts the target label with the majority vote from the set of *k* nearest neighbors to the query **q**:

$$\mathbb{M}_k(\mathbf{q}) = \underset{l \in labels(t)}{\arg\max} \sum_{i=1}^{k} \delta(t_i, l)$$

  - $\mathbb{M}_k(\mathbf{q})$ is the prediction of the model M for query **q** given the parameter of the model *k*
  - *labels(t)* is the set of labels in the domain of the target feature (class)
  - t_i is the value of the target feature for instance $\mathbf{d}_i$
  - $\delta(t_i, l)$ is the **Kronecker delta** function, which takes two parameters and return 1 if they are equal and 0 others.

Hyper-Parameter *k*

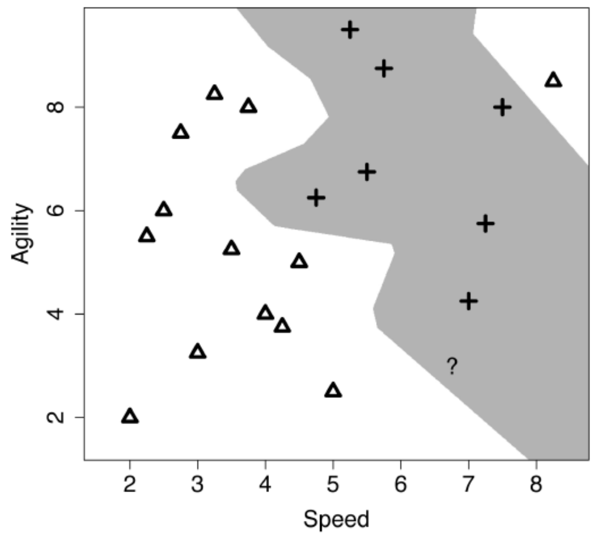
- There is a trade-off in setting the value of *k*
  - **If *k* is too small**, sensitive to noise points
  - **If *k* is too large**, neighborhood may include points from other classes



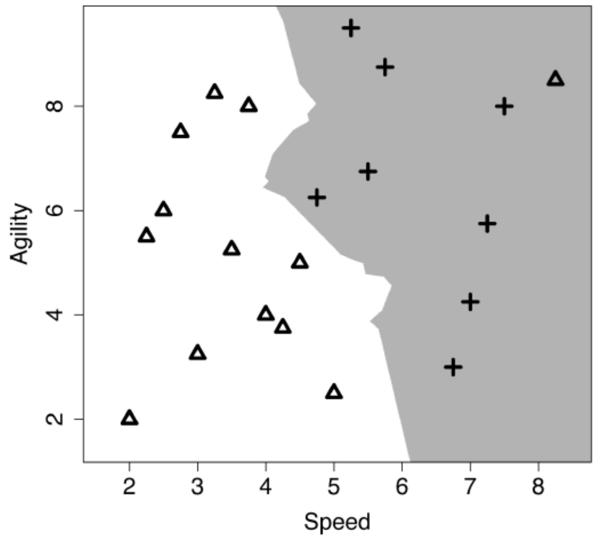
- The risk associated with *k* to a high value are particularly bad with imbalanced data.

Decision Boundary with Different k

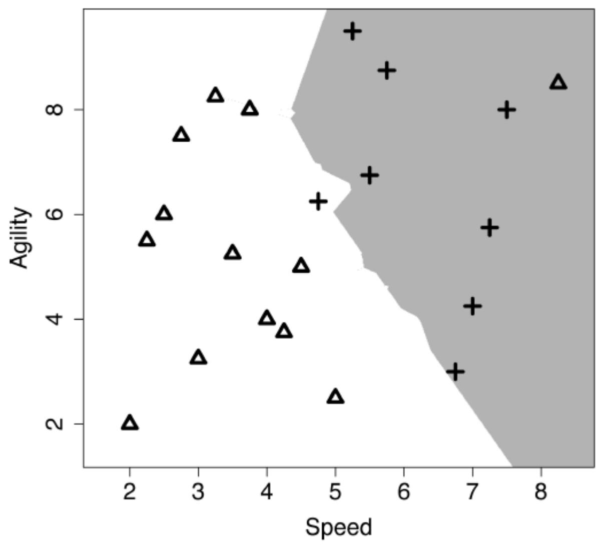
***k*=1**



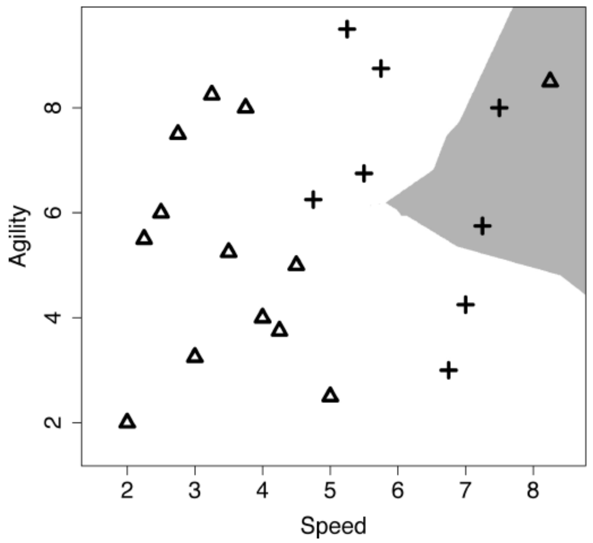
***k*=3**



***k*=5**



***k*=15**



Imbalance data: 13 △ and 7 +

How to Set *k*

- The most common way to set the value of parameter *k* is to perform evaluation experiments to investigate the performance of models with different values for *k* and to select the one that performs best.
- Another way is to use a weighted *k*-nearest neighbor approach.

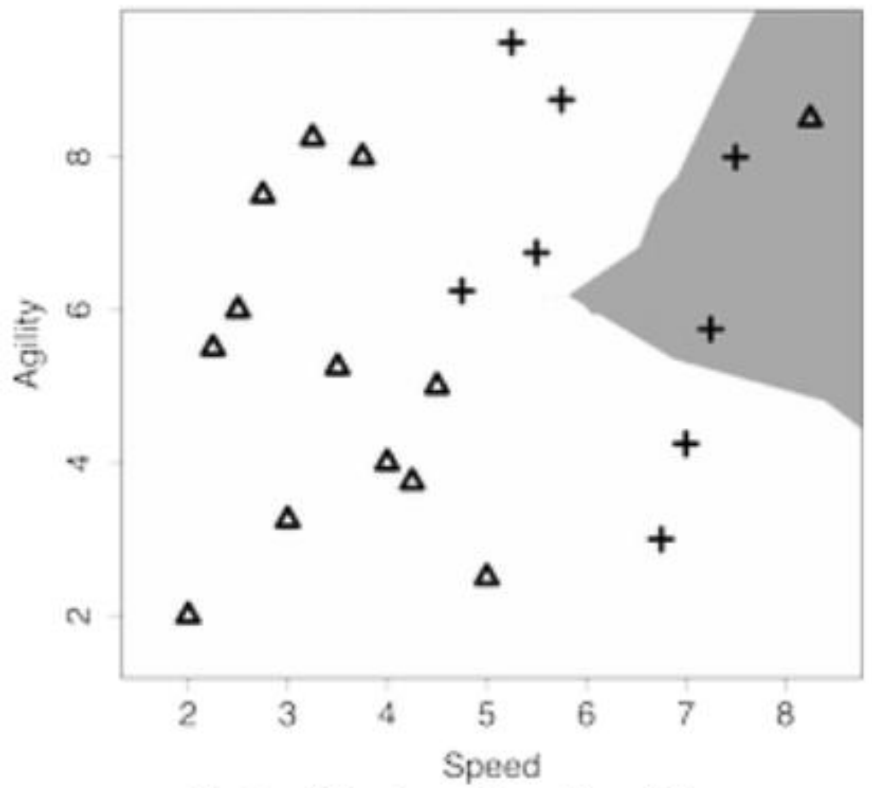
Weighted *k* Nearest Neighbor Learner

- In the distance weighted *k* nearest neighbor model, **neighbors closer to the query have a more significant influence on the outcome.**
- **The distance weighted *k* nearest neighbor model is:**

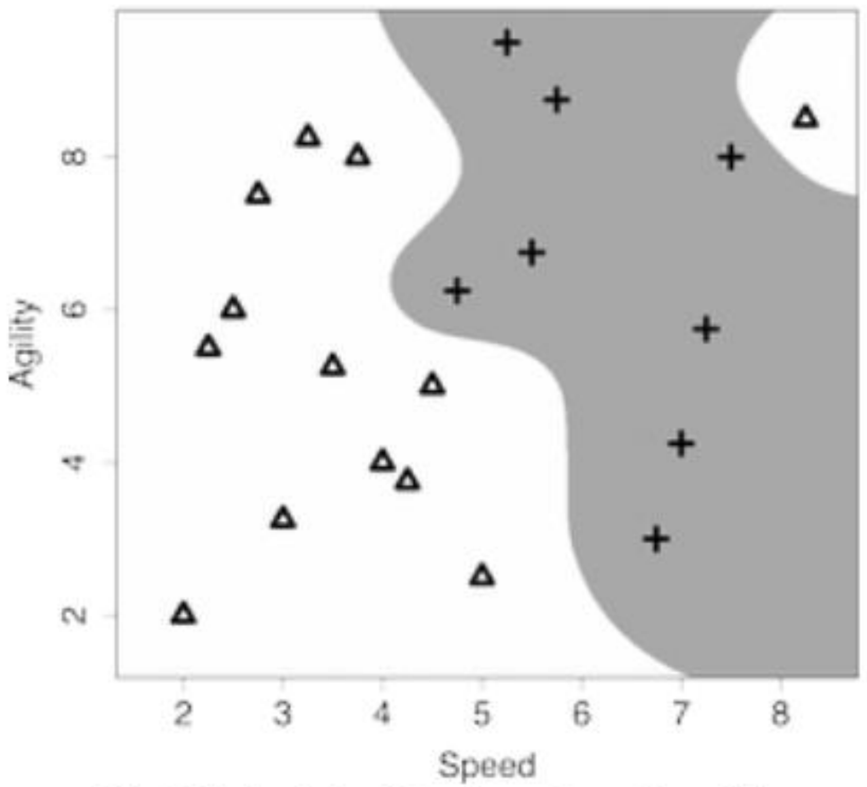
$$\mathbb{M}_k(\mathbf{q}) = \underset{l \in levels(t)}{\arg\max} \sum_{i=1}^{k} \frac{\mathbf{1}}{\boldsymbol{dist}(\mathbf{q}, \mathbf{d}_i)^2} \times \delta(t_i, l)$$

  - Where $levels(t)$ is the set of levels in the domain of the target feature.
  - t_i is the value of the target feature for instance d_i
  - $\delta(t_i, l)$ is the Krnocker delta function, which takes two parameters and return 1 if they are equal and 0 otherwise.
- It weights each neighbor by the reciprocal of the squared distance between the neighbor d and the query q**.** Each neighbor's contribution to the prediction is inversely proportional to its distance from the query.
- The **predicted result** for a specific query is determined by the target level (or class) that accumulates the highest score when summing the weights of the votes from the instances within the k-nearest neighbor's vicinity.

Example



(a) Decision boundary ($k = 15$)



(b) Weighted decision boundary ($k = 21$)

Figure. (a) The decision boundary using majority vote of the ***k*** (=15) **nearest neighbors**; and (b) the **weighted *k*** (=21) **nearest neighbor** model decision boundary (k = 21 = the dataset size)

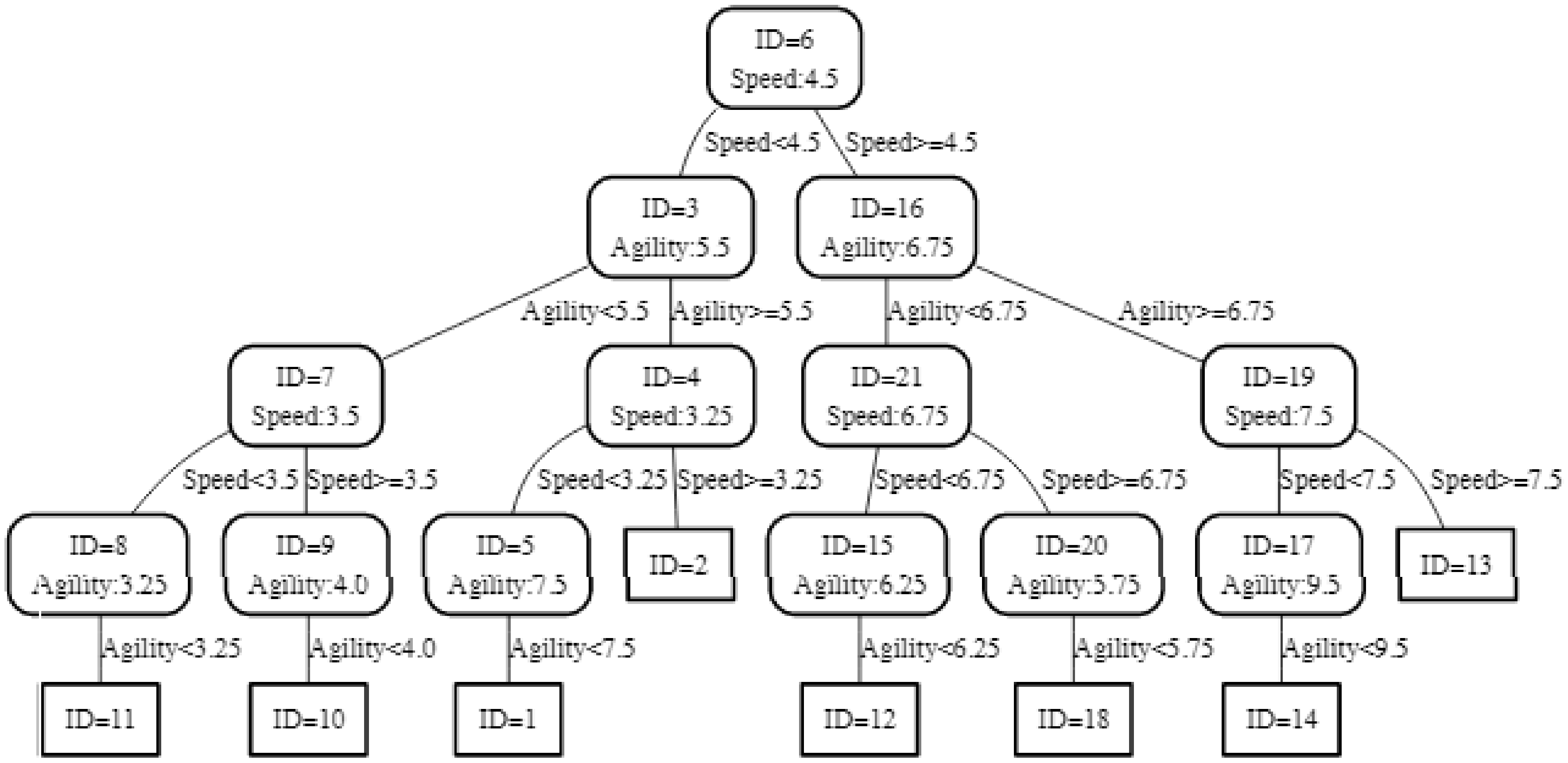
Weighted *k* Nearest Neighbors Model

- **Hyper-parameter *k***
  - We can actually set *k* to the size of the training set to include all the training instances in the prediction process
  - Or It is possible to find a value for *k* using evaluation experiments. - Fitting the parameters of a model is as important as selection which model to use.
- **Strength**:
  - Reduce the impact of noisy instances
  - Smoother decision boundary – A better job of modeling the transition between the different target levels.
- **Weakness**:
  - Still not effective with imbalance data because the majority target level may dominate.
  - Computationally expensive, especially when *k* = dataset size.

Nearest Neighbor Search

- The *k*-NN algorithm stores the full training dataset in memory.
- For large datasets, the computational time required to calculate the distance between a query and all training instances, as well as to retrieve the *k* nearest neighbors may become impractical.
- **An index of the training set** can be utilized to enable efficient retrieval of the nearest neighbors without scanning the entire dataset exhaustively.
- The ***k-d* tree** (*k-dimensional* tree) is a prominent example of such indices.
- A *k-d* tree functions as **a balanced binary tree**, where each node corresponds to an instance from the dataset.

Example: *k-d* Tree Index

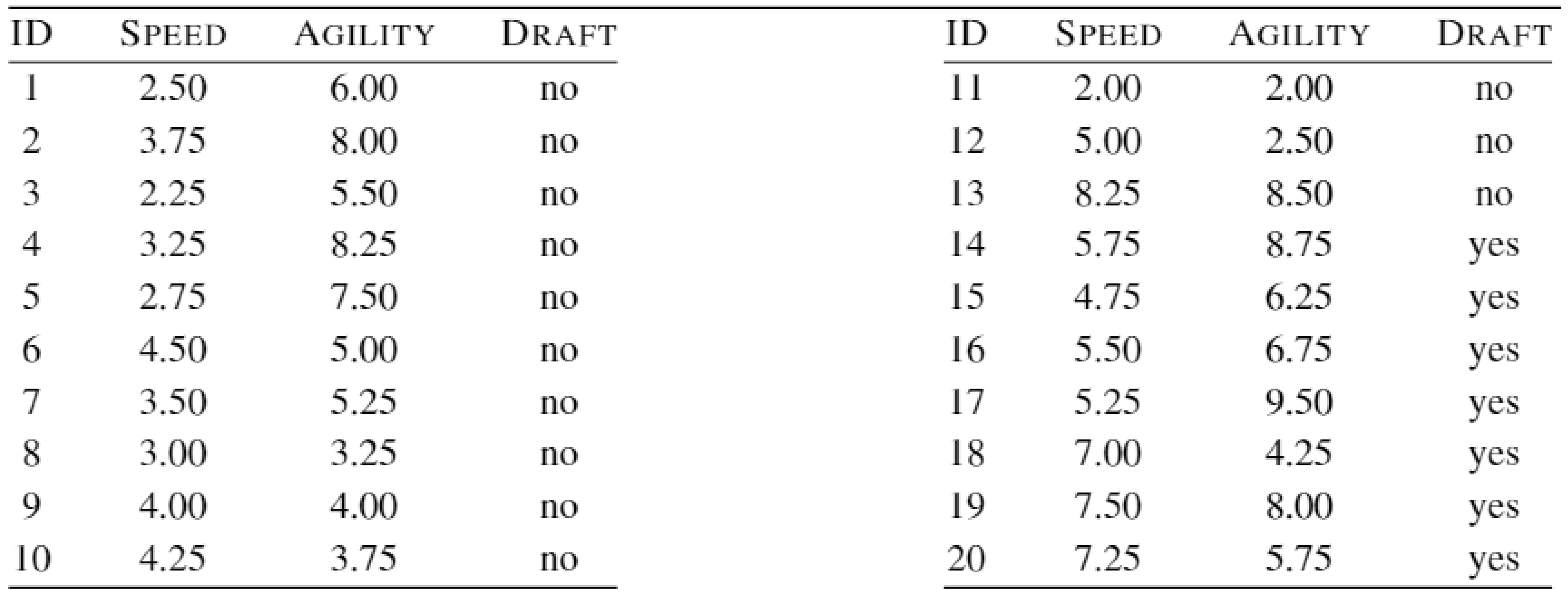


(a)

k-d Tree Construction

- For ***k-d* tree construction**, begin by selecting a feature and partition the data into two subsets using the median value of that feature.
- Subsequently, each of these new subsets is recursively divided, stopping the process once a partition contains fewer than two instances.

Example Dataset

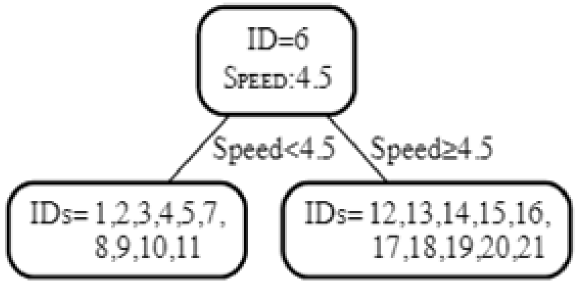


ID	SPEED	AGILITY	DRAFT
1	2.50	6.00	no
2	3.75	8.00	no
3	2.25	5.50	no
4	3.25	8.25	no
5	2.75	7.50	no
6	4.50	5.00	no
7	3.50	5.25	no
8	3.00	3.25	no
9	4.00	4.00	no
10	4.25	3.75	no
11	2.00	2.00	no
12	5.00	2.50	no
13	8.25	8.50	no
14	5.75	8.75	yes
15	4.75	6.25	yes
16	5.50	6.75	yes
17	5.25	9.50	yes
18	7.00	4.25	yes
19	7.50	8.00	yes
20	7.25	5.75	yes

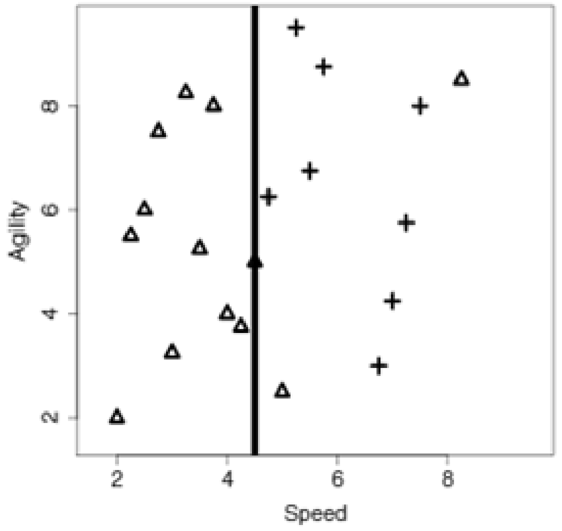
Table. The SPEED and AGILITY ratings for 20 college athletes and whether they were drafted by a professional team.

Example: Construction of *k-d* Tree

Figure (a) The k-d tree generated for the dataset after the initial split using the SPEED feature with a threshold of 4.5; (b) the partitioning of the feature space by the k-d tree in (a);

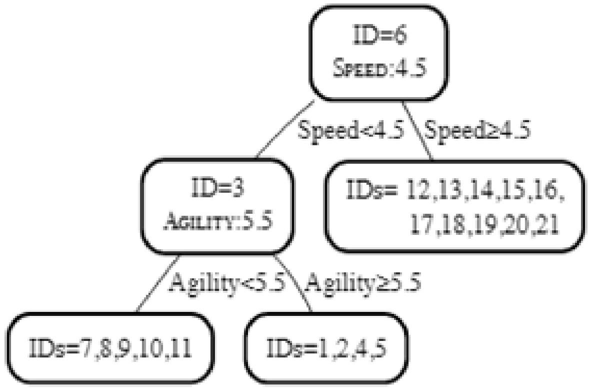

ID=6
SPEED:4.5
Speed<4.5
Speed≥4.5
IDs= 1,2,3,4,5,7, 8,9,10,11
IDs= 12,13,14,15,16, 17,18,19,20,21

(a)

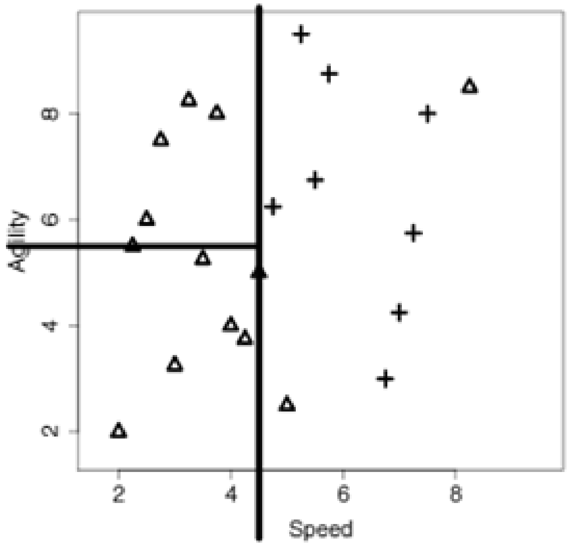


(b)

Figure (c) the k-d tree after the dataset at the left child of the root has been split using the AGILITY feature with a threshold of 5.5; and (d) the partitioning of the feature space by the k-d tree in (c).

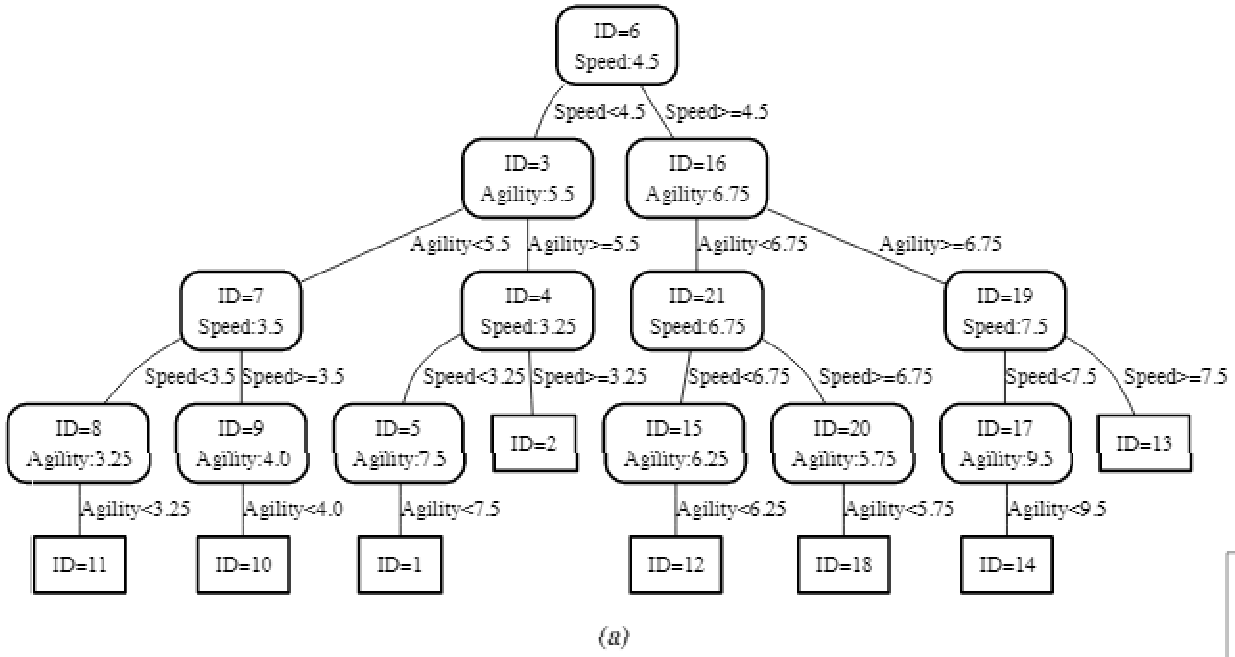


(c)



(d)

Example: Final k-d Tree



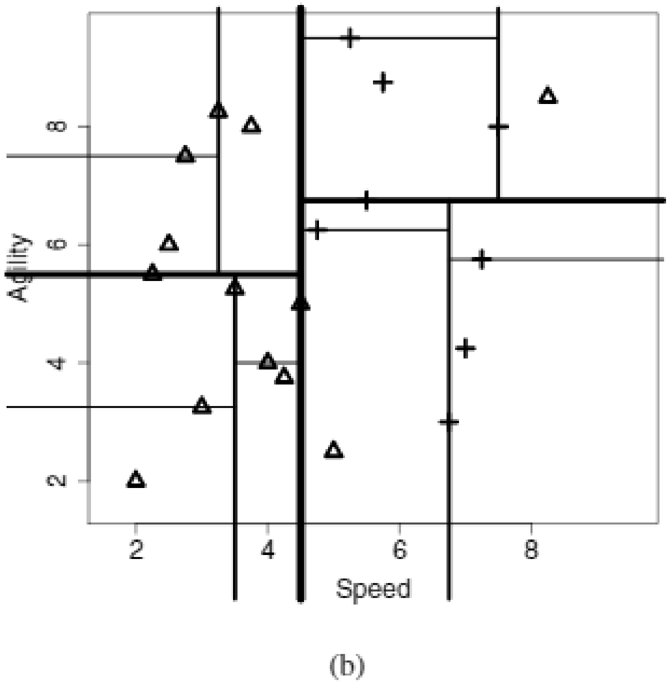
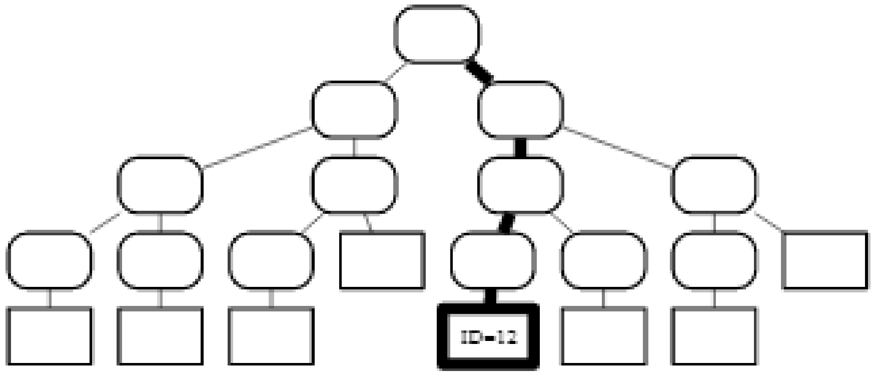
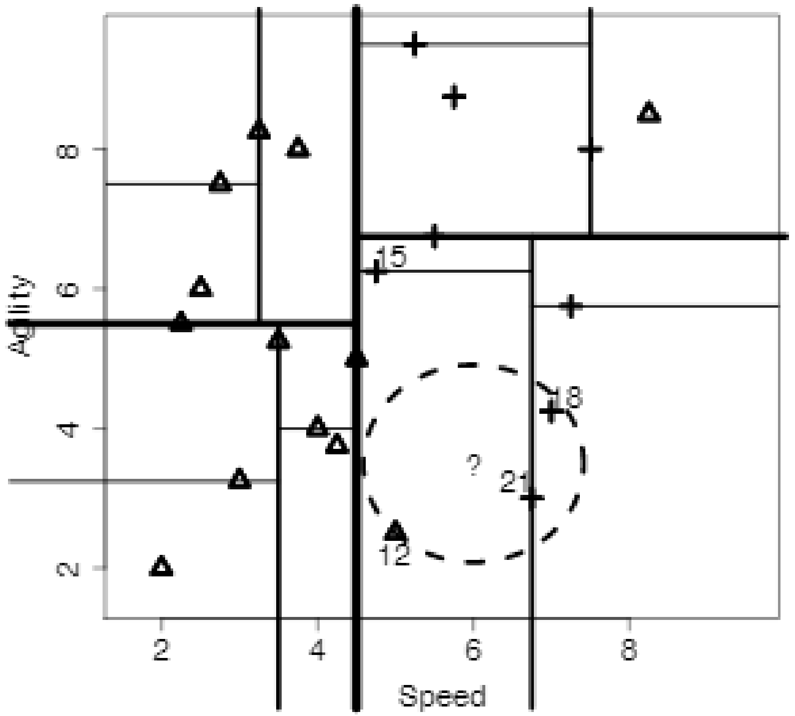


Figure (a) The final k-d tree generated for the dataset; and (b) the partitioning of the feature space defined by this k-d tree.

Nearest Neighbor Search using *k-d* Tree



(a)



(b)

Figure (a) The path taken from the root node to a leaf node when we search the tree with **a query SPEED= 6.00, AGILITY= 3.50**; ID=12 is the nearest neighbor of the query instance.
and (b) the ? marks the location of the query, and the dashed circle plots the extent of the target, and for convenience in the discussion, we have labeled some of the nodes with the IDs of the instances they index (12, 15, 18, and 21).

Outline

- Introduction to Instance-Based Learning
- Similarity (Distance) Metrics
- Standard Approach: Nearest Neighbor Learner
- *k*-Nearest Neighbor Learner

☞ **Data Normalization**

- Predicting Continuous Targets
- Other Measures of Similarity
- Feature Selection
- Summary

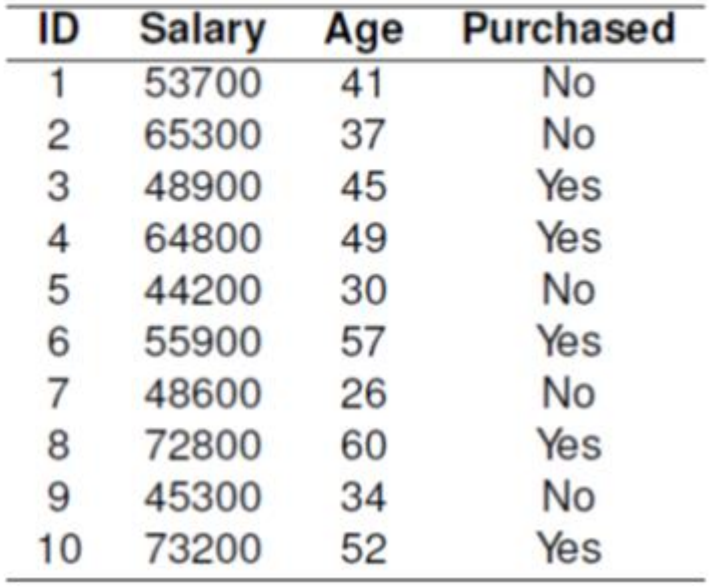
Example: Need of Data Normalization

- **The distance between two instances is dominated by a feature which has a large range of values.**
- Example:

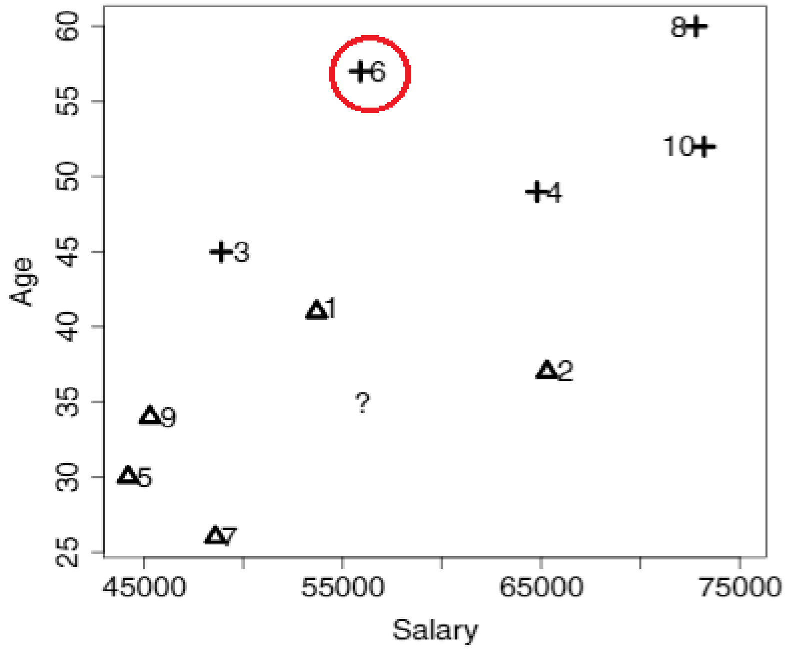
				Salary and Age		Salary Only		Age Only	
ID	Salary	Age	Purch.	Dist.	Neigh.	Dist.	Neigh.	Dist.	Neigh.
1	53700	41	No	2300.0078	2	2300	2	6	4
2	65300	37	No	9300.0002	6	9300	6	2	2
3	48900	45	Yes	7100.0070	3	7100	3	10	6
4	64800	49	Yes	8800.0111	5	8800	5	14	7
5	44200	30	No	11800.0011	8	11800	8	5	5
6	55900	57	Yes	102.3914	1	100	1	22	9
7	48600	26	No	7400.0055	4	7400	4	9	3
8	72800	60	Yes	16800.0186	9	16800	9	25	10
9	45300	34	No	10700.0000	7	10700	7	1	1
10	73200	52	Yes	17200.0084	10	17200	10	17	8

  - Salary feature dominates the computation of Euclidean distance, and this dominance is reflected in the ranking of the instances as neighbors.

Example (cont.)



ID	Salary	Age	Purchased
1	53700	41	No
2	65300	37	No
3	48900	45	Yes
4	64800	49	Yes
5	44200	30	No
6	55900	57	Yes
7	48600	26	No
8	72800	60	Yes
9	45300	34	No
10	73200	52	Yes



- The nearest neighbor based on the distance is ID6 and the prediction is + (*Yes*). ➔ Strange result.
- This odd prediction is caused by features taking different ranges of values, i.e., **features having different variances**.

Data Normalization

- Distance computations are **sensitive** to the value ranges of the features in the dataset.
- We can adjust for the problem of different range values using **normalization**
- When we normalize the features in a dataset, we **control for the variation** across the variances of features and **ensure that each feature can contribute equally to the distance metric**.

Data Normalization Techniques

- The basic **range normalization** (as referred to **min-max normalization**) using the range [0, 1] is:

$$a_i' = \frac{a_i - \min(a)}{\max(a) - \min(a)} \times (high - low) + low$$

  - Example

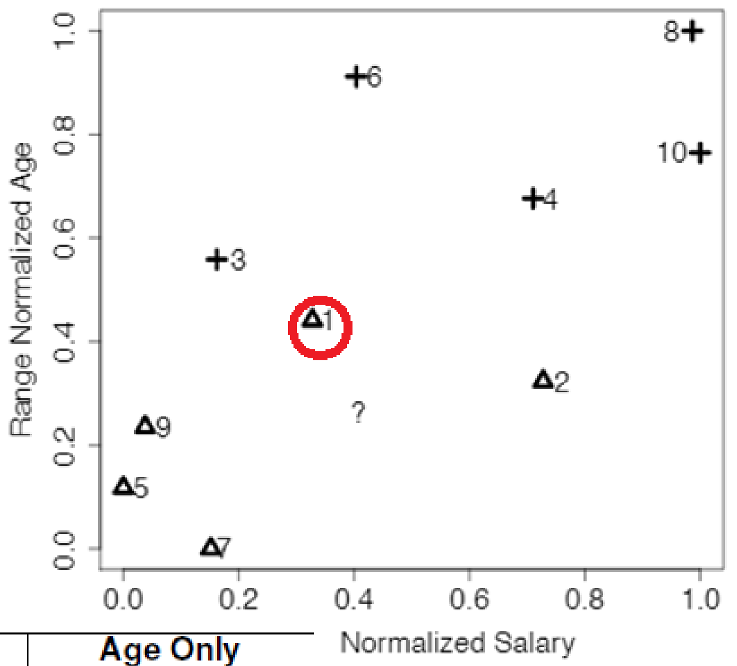
SALARY: $\frac{53700-44200}{73200-44200} \times (1.0 - 0.0) + 0 = 0.3276$

AGE: $\frac{41-26}{60-26} \times (1.0 - 0.0) + 0 = 0.4412$

- Many other normalization methods: **Z-score**, etc.

Example: After Normalization

- **After data normalization** is applied, the nearest neighbor after data normalization is ID1.
- The prediction result of the query is △ (*No*).



	Normalized Dataset			Salary and Age		Salary Only		Age Only	
ID	Salary	Age	Purch.	Dist.	Neigh.	Dist.	Neigh.	Dist.	Neigh.
1	0.3276	0.4412	No	0.1935	1	0.0793	2	0.17647	4
2	0.7276	0.3235	No	0.3260	2	0.3207	6	0.05882	2
3	0.1621	0.5588	Yes	0.3827	5	0.2448	3	0.29412	6
4	0.7103	0.6765	Yes	0.5115	7	0.3034	5	0.41176	7
5	0.0000	0.1176	No	0.4327	6	0.4069	8	0.14706	3
6	0.4034	0.9118	Yes	0.6471	8	0.0034	1	0.64706	9
7	0.1517	0.0000	No	0.3677	3	0.2552	4	0.26471	5
8	0.9862	1.0000	Yes	0.9361	10	0.5793	9	0.73529	10
9	0.0379	0.2353	No	0.3701	4	0.3690	7	0.02941	1
10	1.0000	0.7647	Yes	0.7757	9	0.5931	10	0.50000	8

(1 is closest, 10 is furthest away)

Outline

- Introduction to Instance-Based Learning
- Similarity (Distance) Metrics
- Standard Approach: Nearest Neighbor Learner
- *k*-Nearest Neighbor Learner
- Data Normalization
- ☞ **Predicting Continuous Targets**
- Other Measures of Similarity
- Feature Selection
- Summary

k Nearest Neighbors for Continuous Target

- To handle continuous target features, simply change the *k* nearest neighbor methods to return a prediction of **the average target value of the nearest neighbors**, rather than the majority target level.
- The ***k* nearest neighbor model**:

$$\mathbb{M}_k(\mathbf{q}) = \frac{1}{k}\sum_{i=1}^{k} t_i$$

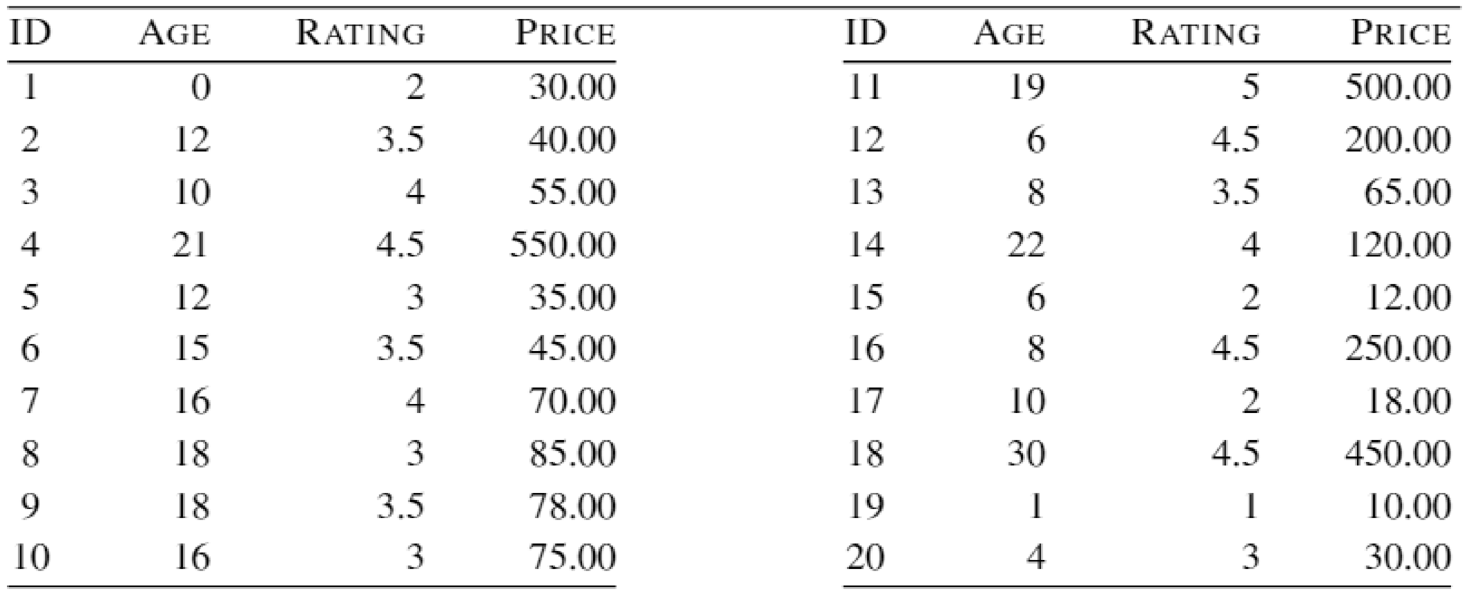
- The **weighted *k* nearest neighbor model**:

$$\mathbb{M}_k(\mathbf{q}) = \frac{\sum_{i=1}^{k} \frac{1}{dist(\mathbf{q},\mathbf{d}_i)^2} \times t_i}{\sum_{i=1}^{k} \frac{1}{dist(\mathbf{q},\mathbf{d}_i)^2}}$$

  - t_i is the value of the target feature for instance i
  - $dist(\mathbf{q},\mathbf{d}_i)$ is the distance between the query instance and its i^{th} nearest neighbor
  - It weights each neighbor by the reciprocal of the squared distance

Example

- A dataset of whiskeys listing the age (in years), the rating (between 1 and 5, with 5 being the best), and the bottle price of each whiskey.



ID	AGE	RATING	PRICE
1	0	2	30.00
2	12	3.5	40.00
3	10	4	55.00
4	21	4.5	550.00
5	12	3	35.00
6	15	3.5	45.00
7	16	4	70.00
8	18	3	85.00
9	18	3.5	78.00
10	16	3	75.00
11	19	5	500.00
12	6	4.5	200.00
13	8	3.5	65.00
14	22	4	120.00
15	6	2	12.00
16	8	4.5	250.00
17	10	2	18.00
18	30	4.5	450.00
19	1	1	10.00
20	4	3	30.00

- Predict the likely sale price of a bottle of whiskey based on the prices achieved by similar bottles at previous auction

Example (cont.)

- A query instance with <AGE=0.0667 and RATING=1.00>.
- Using a ***k* nearest neighbor model (*k*=3),**

$$M_3(<0.0667, 1.00>)$$

$$= \frac{200+250+55}{3} = 168.33$$

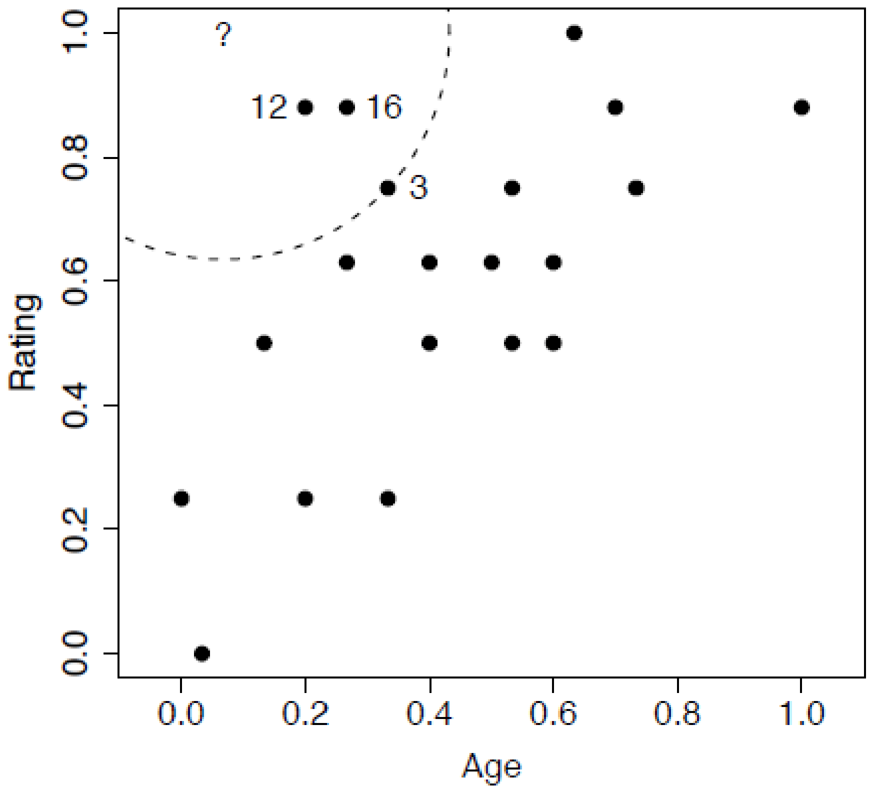
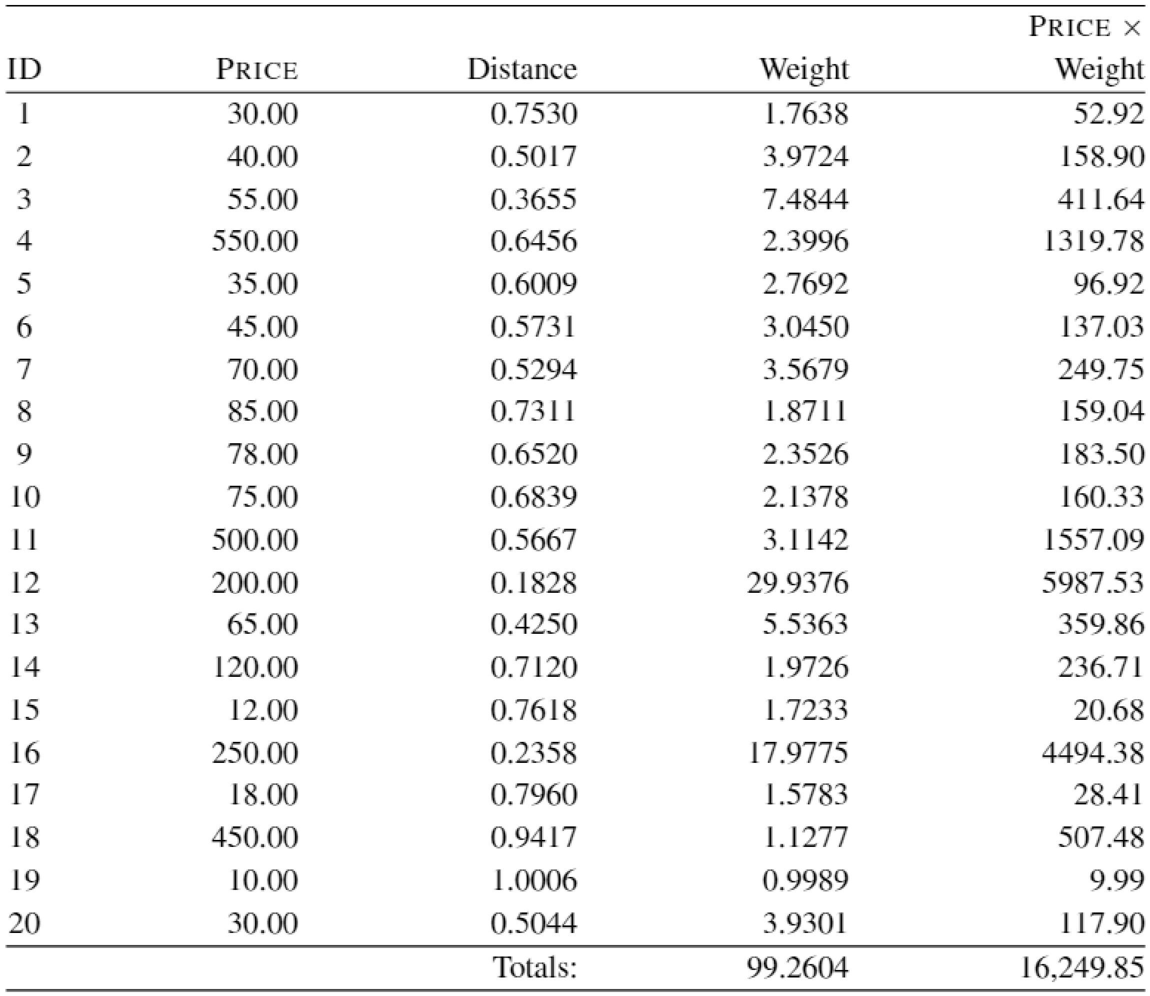


Figure. The AGE and RATING feature space for the whiskey dataset. The data values are normalized. ? is the query point <0.0667, 1.00>. *k*=3

Example (cont.)

- A query instance with AGE=0.0667 and RATING=1.00.
- Using **a weighted *k* nearest neighbor model (*k*=20)**

$$M_3(<0.0667, 1.00>)$$
$$= \frac{16249.85}{99.2604} = 163.71$$



ID	PRICE	Distance	Weight	PRICE × Weight
1	30.00	0.7530	1.7638	52.92
2	40.00	0.5017	3.9724	158.90
3	55.00	0.3655	7.4844	411.64
4	550.00	0.6456	2.3996	1319.78
5	35.00	0.6009	2.7692	96.92
6	45.00	0.5731	3.0450	137.03
7	70.00	0.5294	3.5679	249.75
8	85.00	0.7311	1.8711	159.04
9	78.00	0.6520	2.3526	183.50
10	75.00	0.6839	2.1378	160.33
11	500.00	0.5667	3.1142	1557.09
12	200.00	0.1828	29.9376	5987.53
13	65.00	0.4250	5.5363	359.86
14	120.00	0.7120	1.9726	236.71
15	12.00	0.7618	1.7233	20.68
16	250.00	0.2358	17.9775	4494.38
17	18.00	0.7960	1.5783	28.41
18	450.00	0.9417	1.1277	507.48
19	10.00	1.0006	0.9989	9.99
20	30.00	0.5044	3.9301	117.90
		Totals:	99.2604	16,249.85

Outline

- Introduction to Instance-Based Learning
- Similarity (Distance) Metrics
- Standard Approach: Nearest Neighbor Learner
- *k*-Nearest Neighbor Learner
- Data Normalization
- Predicting Continuous Targets
- **Other Measures of Similarity**
- Feature Selection
- Summary

Other Similarity Metrics and Indexes

- Similarity indexes **for binary descriptive features**
  - Russel-Rao
  - Sokal-Michener
  - Jaccard

- Similarity indexes and metrics **for continuous descriptive features**
  - Cosine Similarity
  - Mahalanobis Distance Metric

Similarity Indexes for Binary Features

- For binary descriptive features, it is often a good idea to use a **similarity index** that defines similarity between instances specifically in terms of **co-presence** or **co-absence** of features, rather than an index based on distance.
- Example

ID	PROFILE	FAQ	HELPFORUM	NEWSLETTER	LIKED	SIGNUP
1	true	true	true	false	true	yes
2	true	false	false	false	false	no

Table A binary dataset listing the behavior of two individuals on a website during a trial period and whether they subsequently signed up for the website.

  - A query instance **q**: PROFILE = true, FAQ = false, HELPFORUM = true, NEWSLETTER = false, LIKED = false

Pairwise Analysis for Binary Feature Similarity

ID	PROFILE	FAQ	HELPFORUM	NEWSLETTER	LIKED	SIGNUP
1	true	true	true	false	true	yes
2	true	false	false	false	false	no

Q =<PROFILE = **true**, FAQ = **false**, HELPFORUM = **true**, NEWSLETTER = **false**, LIKED = **false**>

		q	
		Pres.	Abs.
d_1	Pres.	CP=2	PA=0
	Abs.	AP=2	CA=1

		q	
		Pres.	Abs.
d_2	Pres.	CP=1	PA=1
	Abs.	AP=0	CA=3

Table: The similarity between the current trial user, q, and the two users in the dataset, d_1 and d_2

- **co-presence** (CP), how often a **true** value occurred **for the same feature** in both the query data q and the data for the comparison user (d_1 or d_2)
- **co-absence** (CA), how often a **false** value occurred **for the same feature** in both the query data q and the data for the comparison user (d_1 or d_2)
- **presence-absence** (PA), how often a **true** value occurred in the query data q when a **false** value occurred in the data for the comparison user (d_1 or d_2) for the same feature
- **absence-presence** (AP), how often a **false** value occurred in the query data q when a **true** value occurred in the data for the comparison user (d_1 or d_2) for the same feature

Russel-Rao

- The **Russel-Rao** similarity index is defined as
  - the ratio between the number of **co-presence** (or **co-absence**) and the total number of binary features

$$sim_{RR}(\mathbf{q},\mathbf{d}) = \frac{CP(\mathbf{q},\mathbf{d})}{|\mathbf{q}|}$$

    - $\mathbf{q}, \mathbf{d}$ are two instances
    - $|\mathbf{q}|$ is the total number of features in the dataset
    - $CP(\mathbf{q},\mathbf{d})$ measures the total number of co-presence between $\mathbf{q}$ and $\mathbf{d}$

- **Example**: The current trial user is judged to be more similar to instance $\mathbf{d}_1$ then $\mathbf{d}_2$
  - $sim_{RR}(\mathbf{q},\mathbf{d}_1) = \frac{CP(\mathbf{q},\mathbf{d}_1)}{|\mathbf{q}|} = \frac{2}{5}$, $sim_{RR}(\mathbf{q},\mathbf{d}_2) = \frac{CP(\mathbf{q},\mathbf{d}_2)}{|\mathbf{q}|} = \frac{1}{5}$,

		q	
		Pres.	Abs.
d$_1$	Pres.	CP=2	PA=0
	Abs.	AP=2	CA=1

		q	
		Pres.	Abs.
d$_2$	Pres.	CP=1	PA=1
	Abs.	AP=0	CA=3

Sokal-Michener

- The **Sokal–Michener** similarity index is defined as
  - the ratio between **the total number of co–presence and co–absences** and the total number of binary features

$$sim_{SM}(\mathbf{q},\mathbf{d}) = \frac{CP(\mathbf{q},\mathbf{d}) + CA(\mathbf{q},\mathbf{d})}{|\mathbf{q}|}$$

- Example:
  - $sim_{RM}(\mathbf{q},\mathbf{d}_1) = \frac{3}{5}$, $sim_{SM}(\mathbf{q},\mathbf{d}_2) = \frac{4}{5}$

		q	
		Pres.	Abs.
$\mathbf{d}_1$	Pres.	CP=2	PA=0
	Abs.	AP=2	CA=1

		q	
		Pres.	Abs.
$\mathbf{d}_2$	Pres.	CP=1	PA=1
	Abs.	AP=0	CA=3

Jaccard

- The **Jaccard** similarity index is defined as
  - the ratio between **the total number of co−presence** and the total number of binary features, excluding those that record a co-absence between a pair of instances

$$sim_J(\mathbf{q},\mathbf{d}) = \frac{CP(\mathbf{q},\mathbf{d})}{CP(\mathbf{q},\mathbf{d}) + PA(\mathbf{q},\mathbf{d}) + AP(\mathbf{q},\mathbf{d})}$$

- Example:
  - $sim_J(\mathbf{q},\mathbf{d}_1) = \frac{2}{4}, \quad sim_J(\mathbf{q},\mathbf{d}_2) = \frac{1}{2}$

		q	
		Pres.	Abs.
$\mathbf{d}_1$	Pres.	CP=2	PA=0
	Abs.	AP=2	CA=1

		q	
		Pres.	Abs.
$\mathbf{d}_2$	Pres.	CP=1	PA=1
	Abs.	AP=0	CA=3

Cosine Similarity

- Cosine Similarity is an index that can be used as a measure of the similarity between instances with continuous descriptive features.
- **Cosine similarity** between two instances is the **cosine** of the inner angle between the two vectors that extend from the origin to each instance.

$$sim_{COSINE}(\mathbf{a},\mathbf{b}) = \frac{(\mathbf{a}[1]\times\mathbf{b}[1]) + \cdots + (\mathbf{a}[m]\times\mathbf{b}[m])}{\sqrt{\sum_{i=1}^{m}\mathbf{a}[i]^2}\times\sqrt{\sum_{i=1}^{m}\mathbf{b}[i]^2}}$$

- **Example**: Calculate the cosine similarity between the following two instances: $\mathbf{d}_1$ = <SMS = 97, VOICE = 21> and $\mathbf{d}_2$ = <SMS = 18, VOICE = 184>

$$sim_{COSINE}(\mathbf{d}_1,\mathbf{d}_2) = \frac{(97\times181)+(21\times184)}{\sqrt{97^2+21^2}\times\sqrt{181^2+184^2}} = 0.8362$$

Cosine Similarity

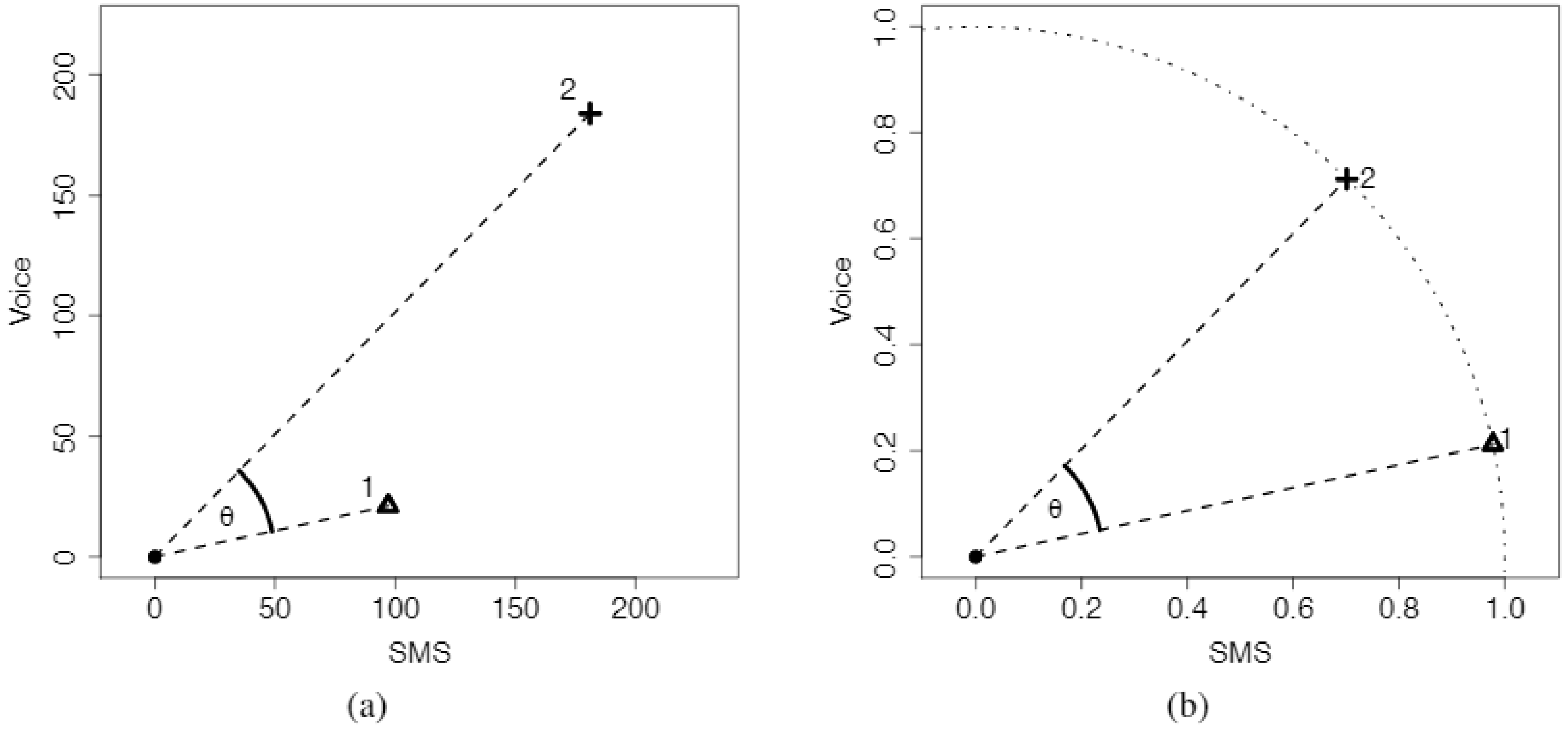


Figure (a) The symbol θ represents the inner angle (i.e., cosine similarity) between the two vectors that extend from the origin of a feature space to each instance d_1 and $d_{2;}$ and (b) shows the illustration with the d_1 and d_2 normalized to the unit circle.

Mahalanobis Distance: Motivation

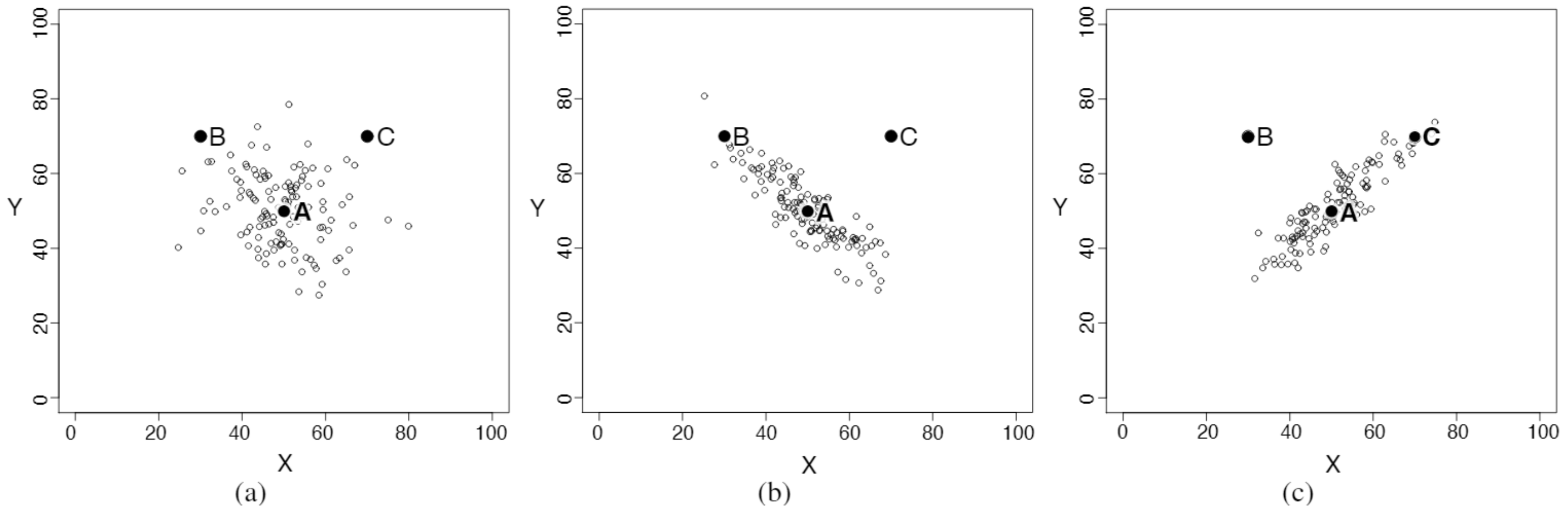


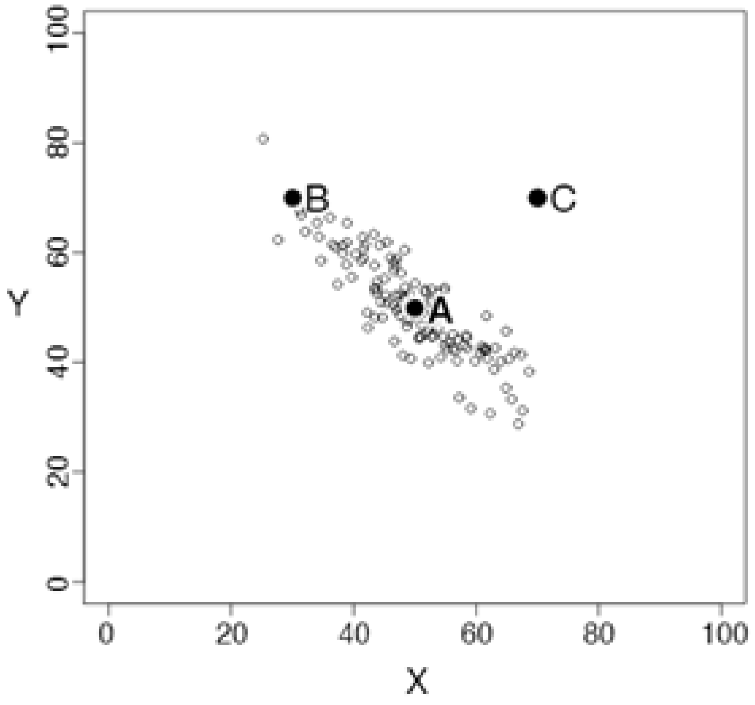
Figure. Scatter plots of three bivariate datasets with the same center point *A* and **two queries *B* and *C*** both **equidistant** from *A*; (a) a dataset **uniformly** spread around the center point; (b) a dataset with **negative covariance**; and (c) a dataset with **positive covariance**.

- ***Q***: Are instance **B** and instance **C** likely to be from the same population form which the dataset has been sampled?

  Ans: In the dataset of Figure (a), **B and C are equally likely to be from the same population as the dataset**. How about in (c) ?

Mahalanobis Distance

- **When judging similarities,** we need to consider not only the central tendency of the group, but also how spread out the members in a group are.
- The **Mahalanobis distance** **reflects how spread out the instances in a dataset**
  - Distances along a direction where the dataset is spread out a lot are scaled down
  - Distances along directions where the dataset is tightly packed are scaled up.
- **Example**: The Mahalanobis distance between B and A is less than the one between C and A.



Mahalanobis Distance (cont.)

- The **mahalanobis distance** uses covariance to scale distances.

$$Mahalanobis(\mathbf{a},\mathbf{b}) = \sqrt{\underbrace{\left[\mathbf{a}[1]-\mathbf{b}[1],\ldots,\mathbf{a}[m]-\mathbf{b}[m]\right]}_{\text{A row vector of differences between each descriptive feature values of the two instances}} \times \underbrace{\sum{}^{-1}}_{\text{Inverse covariance matrix}} \times \underbrace{\begin{bmatrix}\mathbf{a}[1]-\mathbf{b}[1]\\ \vdots \\ \mathbf{a}[m]-\mathbf{b}[m]\end{bmatrix}}_{\text{The column vector of the differences}}}$$

- Mahalanobis distance squares the differences of feature values, but <u>rescales the differences (using the inverse covariance matrix)</u> so that all the features have unit variance and the effects of covariance is removed.

Example

- A: (50,50) marks the central of the dataset, B: (30,70) and C: (70,70)

$$Mahalanobis(A,B) = \sqrt{[50-30, 50-70] \times \begin{bmatrix} 0.059 & -0.521 \\ -0.521 & 0.0578 \end{bmatrix} \times \begin{bmatrix} 50-30 \\ 50-70 \end{bmatrix}} = 9.4049$$

$$Mahalanobis(A,C) = \sqrt{[50-70, 50-70] \times \begin{bmatrix} 0.059 & -0.521 \\ -0.521 & 0.0578 \end{bmatrix} \times \begin{bmatrix} 50-70 \\ 50-70 \end{bmatrix}} = 2.2540$$

, where the inverse covariance matrix is based on the covariance matrix calculated directly from the dataset: $\begin{bmatrix} 82.39 & 74.26 \\ 74.26 & 84.22 \end{bmatrix}$

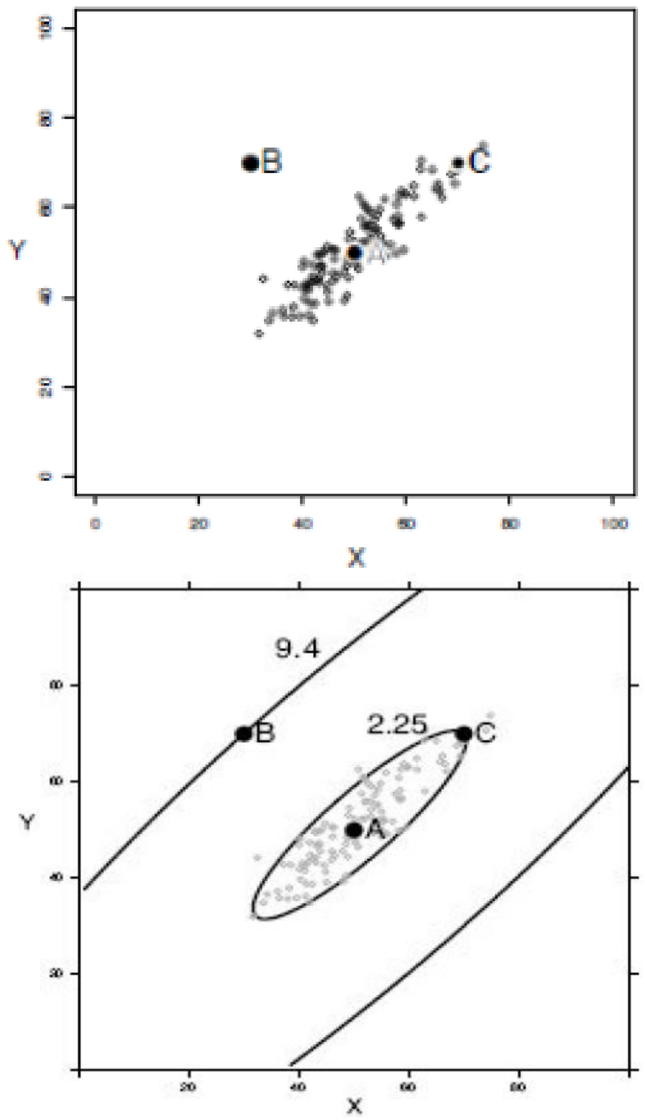


Figure. The ellipses plot the Mahalanobis distance contours from A that B and C lies on

- In Euclidean terms, B and C are equidistant from A; however, **using the Mahalanobis distance, C is much closer to A than B**

Outline

- Introduction to Instance-Based Learning
- Similarity (Distance) Metrics
- Standard Approach: Nearest Neighbor Learner
- *k*-Nearest Neighbor Learner
- Data Normalization
- Predicting Continuous Targets
- Other Measures of Similarity
- ☞ **Feature Selection**
- Summary

Feature Selection

- Nearest Neighbor models are **sensitive to the presence of redundant and irrelevant descriptive features in training data.**
- **Feature selection** is a particularly important process for nearest neighbor learning.
- Feature selection eliminates redundant and irrelevant attributes from the learning process, thereby mitigating the curse of dimensionality.

- Refer to the lecture slide of feature engineering for the feature selection topic!!

Outline

- Introduction to Instance-Based Learning
- Similarity (Distance) Metrics
- Standard Approach: Nearest Neighbor Learner
- *k*-Nearest Neighbor Learner
- Data Normalization
- Predicting Continuous Targets
- Other Measures of Similarity
- Feature Selection
- ☞ **Summary**

Summary

- **Instance-based prediction models** attempt to mimic a very human way of reasoning by basing predictions for a target feature value on the most similar instances in memory—this makes them easy to interpret and understand.
- The **inductive bias** underpinning instance-based machine learning methods is that instances that have similar descriptive features also have the same target feature values.

Summary

- The **nearest neighbor** is one of the simplest and best known machine learning algorithms for instance-based reasoning.
- The definition of neighborhoods is based on **proximity** within the feature space to the labelled training instances.
- **Predictions** are made for a query instance using the target level of the training instance defining the neighborhood in the feature space that contains the query.

Summary (cont.)

- Nearest Neighbor Learning
  - **Just store all training examples $< \boldsymbol{x_i}, \boldsymbol{f}(\boldsymbol{x_i}) >$**
  - **Nearest neighbor approach**:
    - Given query instance x_q, first locate nearest training sample x_n, then estimate $\hat{f}(x_q) \leftarrow f(x_n)$
  - ***k*-Nearest neighbor approach**: Given query instance x_q,
    - take vote among its k nearest neighbors (if discrete-valued target function)
    - Or, take mean of f value of k nearest neighbors (if real-valued), $\hat{f}(x_q) \leftarrow \frac{\sum_{i=1}^{k} f(x_i)}{k}$

Summary (cont.)

- Nearest neighbor models are very **sensitive to noise** in the target feature. The easiest way to solve this problem is to employ a *k* nearest neighbor.
- **Normalization** techniques should almost always be applied when nearest neighbor models are used.
- It is easy to adapt a nearest neighbor model to **continuous targets**.
- There are many different **measures of similarity**.
- As the number of instances becomes large, a nearest neighbor model will become slower—techniques such as the ***k-d* tree** can help with this issue.

Summary (cont.)

- **Advantages**
  - **Adaptable**: Since the "model" is the entire dataset, instance-based learning can quickly adapt to changes.
  - **Simple**: No need for training steps or model-building.
- **Disadvantages**
  - **Computationally Expensive**: Making predictions can be slow, especially with large datasets, since the distance to every instance in the training data might need to be computed.
  - **Storage**: All training data must be stored, which could be space-intensive for large datasets.
  - **Sensitive to Irrelevant Features**: Because predictions are based on distances, the algorithm can be sensitive to irrelevant or noisy features that might affect the distance computations.

Summary (cont.)

- In essence, instance-based learning provides a simple yet effective approach for certain tasks, especially when the dataset isn't exceedingly large, and the dimensionality is manageable.