Chapter 5: Similarity-based Learning

MATH2319

Big Idea

- The big idea underpinning similarity-based learning:

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

- If you are trying to classify something then you should search your memory to find things that are similar and label it with the same class as the most similar thing in your memory.
- **Example:** You meet a new friend from overseas, but you don't know which country she is from.
  - So you go through your memory and try match her characteristics (accent, appearance, etc.) with overseas friends you met in the past.
  - Then you go, like, "hey, are you American?"
  - You have just made a classification!

- One of the simplest and best known machine learning algorithms for this type of reasoning is called the nearest neighbor (NN) algorithm.
- The NN algorithm is one of the most influential ML algorithms in both theory and applications.
- **Theorem:** In the large sample case, the error rate of NN is never worse than twice of the best error ever possible (a.k.a. Bayes error).

Fundamentals

- The fundamentals of similarity-based learning are:
  - Feature space
  - Similarity metrics

Feature Space

Table: The speed and agility ratings for 20 college athletes labelled with the decisions for whether they were drafted or not.

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

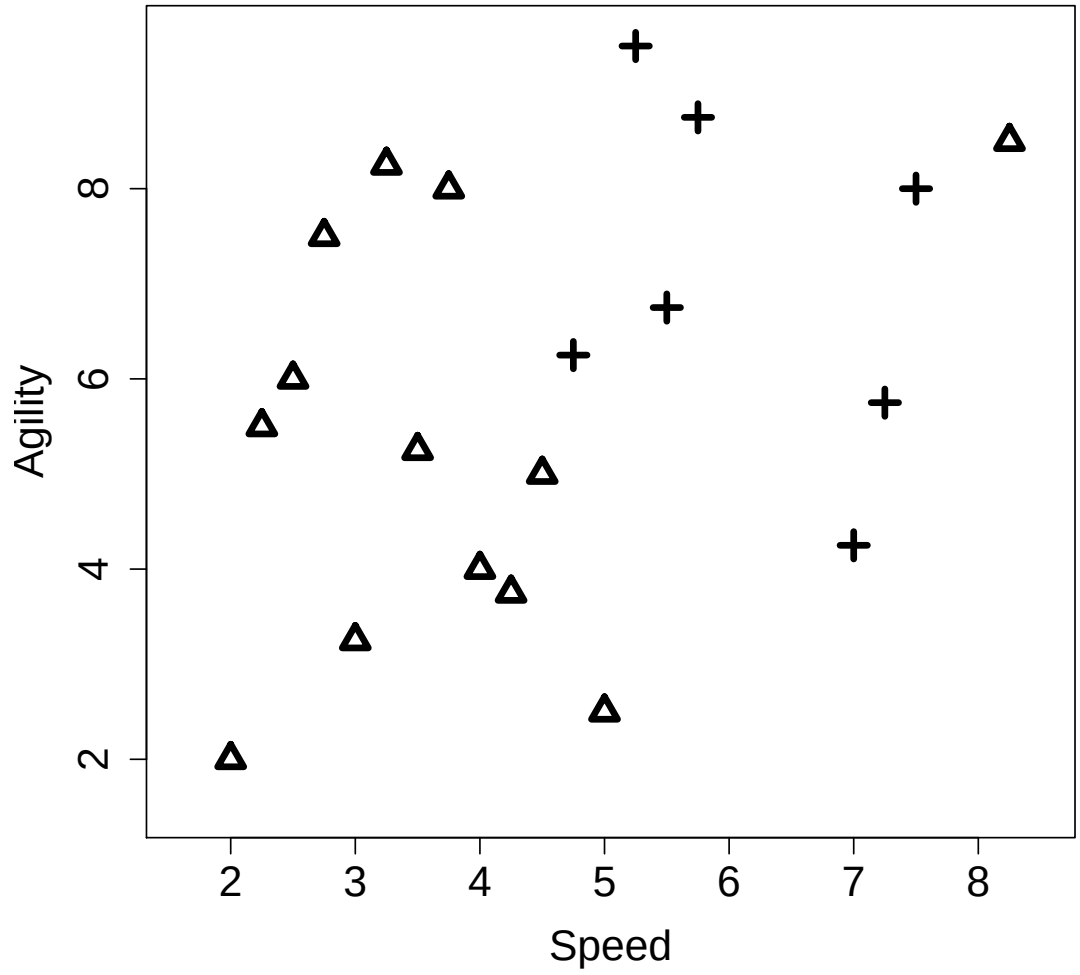


Figure: A feature space plot of the data in the above table. The triangles represent *'Non-draft'* instances and the crosses represent the *'Draft'* instances.

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

Distance Metrics

- A similarity metric measures the similarity between two instances according to a feature space.
- One of the best known metrics is Euclidean distance which computes the length of the straight line between two points. Euclidean distance between two instances **a** and **b** in a m-dimensional feature space is defined as:

$$Euclidean(\mathbf{a}, \mathbf{b}) = \sqrt{\sum_{i=1}^{m} (\mathbf{a}[i] - \mathbf{b}[i])^2} \tag{1}$$

Example

The Euclidean distance between instances d_{12} (SPEED$=$ 5.00, AGILITY$=$ 2.5) and d_5 (SPEED$=$ 2.75,AGILITY$=$ 7.5) in Table 2 [23] is:

Example

The Euclidean distance between instances d_{12} (SPEED= 5.00, AGILITY= 2.5) and d_5 (SPEED= 2.75,AGILITY= 7.5) in Table 2 [23] is:

$$\begin{aligned} \mathit{Euclidean}(\langle 5.00, 2.50\rangle, \langle 2.75, 7.50\rangle) &= \sqrt{(5.00-2.75)^2 + (2.50-7.50)^2} \\ &= \sqrt{30.0625} = 5.4829 \end{aligned}$$

- Another, less well known, distance measure is the Manhattan distance or taxi-cab distance.
- The Manhattan distance between two instances **a** and **b** in a feature space with m dimensions is:[1]

$$Manhattan(\mathbf{a}, \mathbf{b}) = \sum_{i=1}^{m} abs(\mathbf{a}[i] - \mathbf{b}[i]) \tag{2}$$

[1] The *abs*() function surrounding the subtraction term indicates that we use the absolute value, i.e. non-negative value, when we are summing the differences; this makes sense because distances can't be negative.

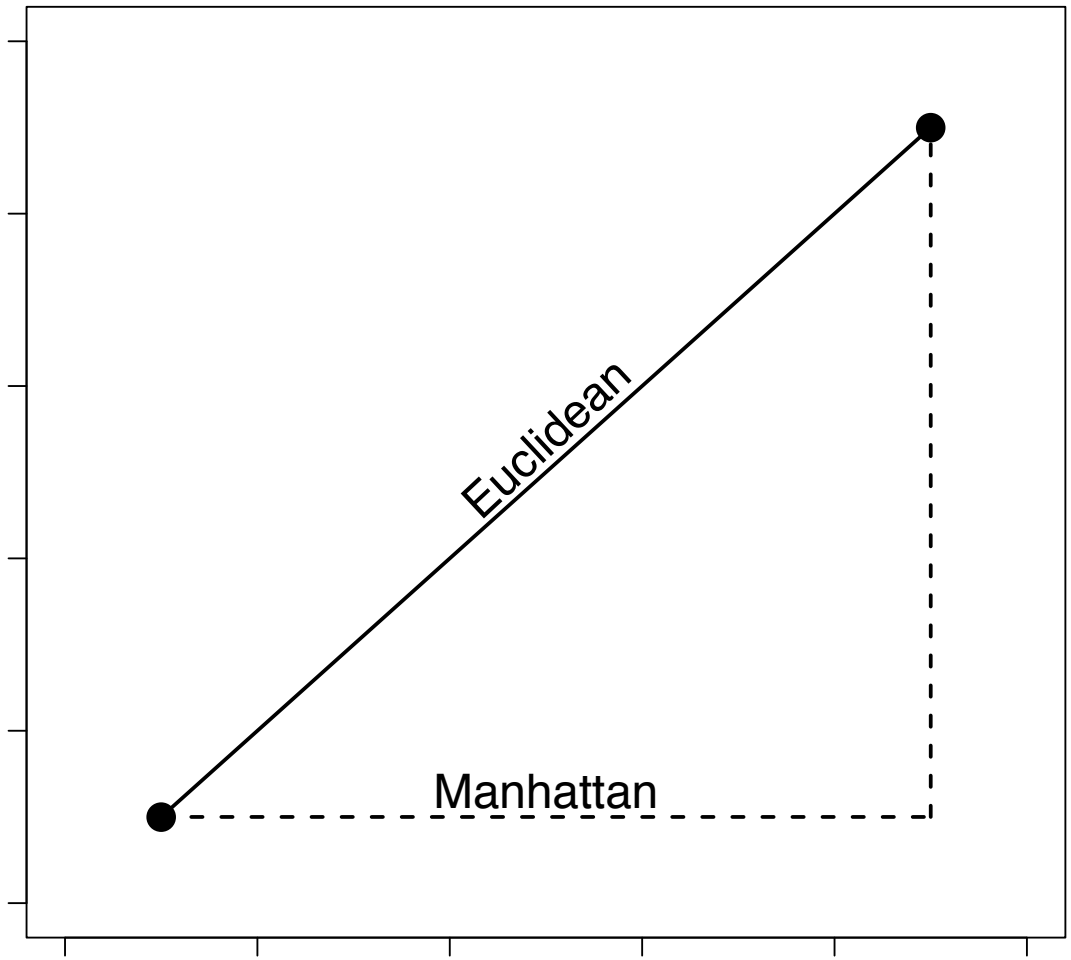


Figure: The Manhattan and Euclidean distances between two points.

Example

The Manhattan distance between instances d_{12} (SPEED= 5.00, AGILITY= 2.5) and d_5 (SPEED= 2.75,AGILITY= 7.5) in Table 2 [23] is:

Example

The Manhattan distance between instances d_{12} (SPEED$=$ 5.00, AGILITY$=$ 2.5) and d_5 (SPEED$=$ 2.75,AGILITY$=$ 7.5) in Table 2 [23] is:

$$\begin{aligned} \mathit{Manhattan}(\langle 5.00, 2.50\rangle, \langle 2.75, 7.50\rangle) &= \mathit{abs}(5.00 - 2.75) + \mathit{abs}(2.5 - 7.5) \\ &= 2.25 + 5 = 7.25 \end{aligned}$$

- The Euclidean and Manhattan distances are special cases of Minkowski distance
- The Minkowski distance between two instances **a** and **b** in a feature space with m descriptive features is:

$$Minkowski(\mathbf{a}, \mathbf{b}) = \left(\sum_{i=1}^{m} abs(\mathbf{a}[i] - \mathbf{b}[i])^p \right)^{\frac{1}{p}} \quad (3)$$

  where different values of the parameter p result in different distance metrics
- The Minkowski distance with $p = 1$ is the Manhattan distance and with $p = 2$ is the Euclidean distance.

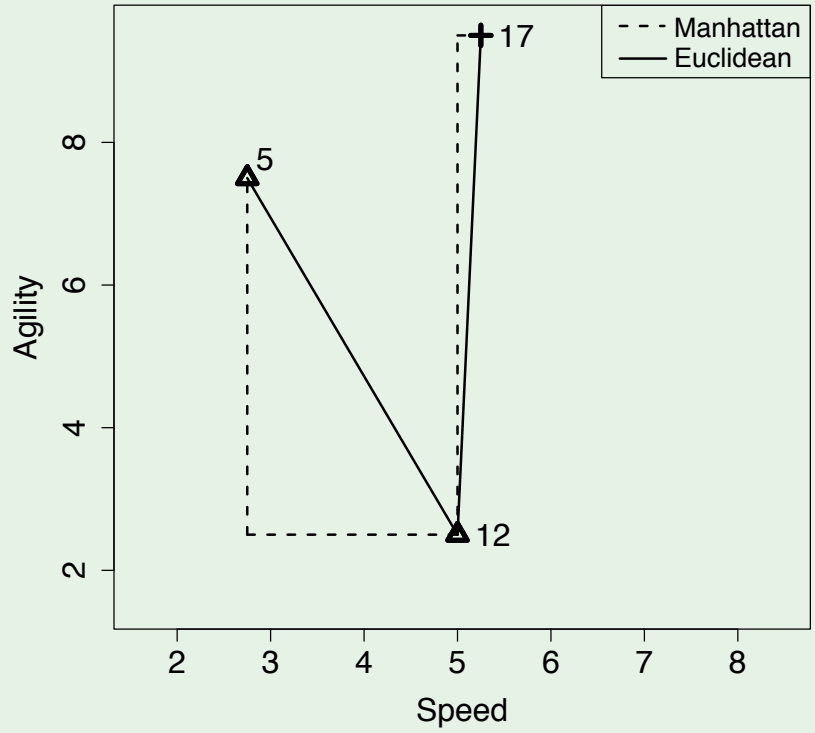
- Minkowski distance in Python using Numpy:
- For two Numpy arrays *a* and *b*, the Minkowski distance between them can be calculated as below:

```
np.power(np.sum(np.power(np.abs(a-b), p)), 1/p)
```

- 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*.

Example

Instance ID	Instance ID	Manhattan (Minkowski p=1)	Euclidean (Minkowski p=2)
12	5	7.25	5.4829
12	17	7.25	8.25



The Manhattan and Euclidean distances between instances $\mathbf{d}_{12}$ (SPEED= 5.00, AGILITY= 2.5) and $\mathbf{d}_5$ (SPEED= 2.75, AGILITY= 7.5) and between instances $\mathbf{d}_{12}$ and $\mathbf{d}_{17}$ (SPEED= 5.25, AGILITY= 9.5).

Standard Approach: The Nearest Neighbor Algorithm

The Nearest Neighbour Algorithm

Require: set of training instances
Require: a query to be classified

1: Iterate across the instances in memory and find the instance that is shortest distance from the query position in the feature space.
2: Make a prediction for the query equal to the value of the target feature of the nearest neighbor.

Table: The speed and agility ratings for 20 college athletes labelled with the decisions for whether they were drafted or not.

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

Example

- Should we draft an athlete with the following profile:

$$\text{SPEED} = 6.75,\ \text{AGILITY} = 3$$

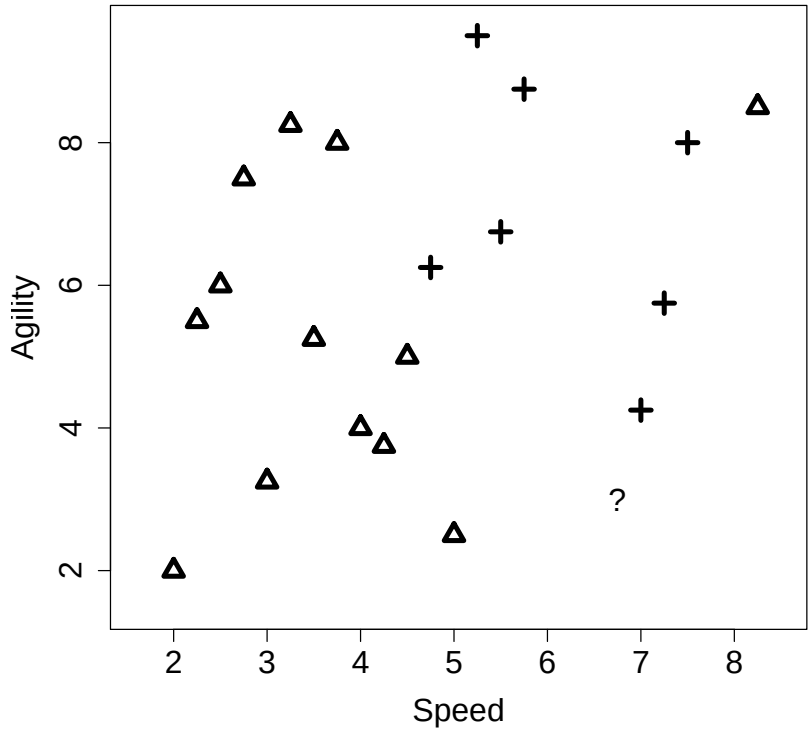
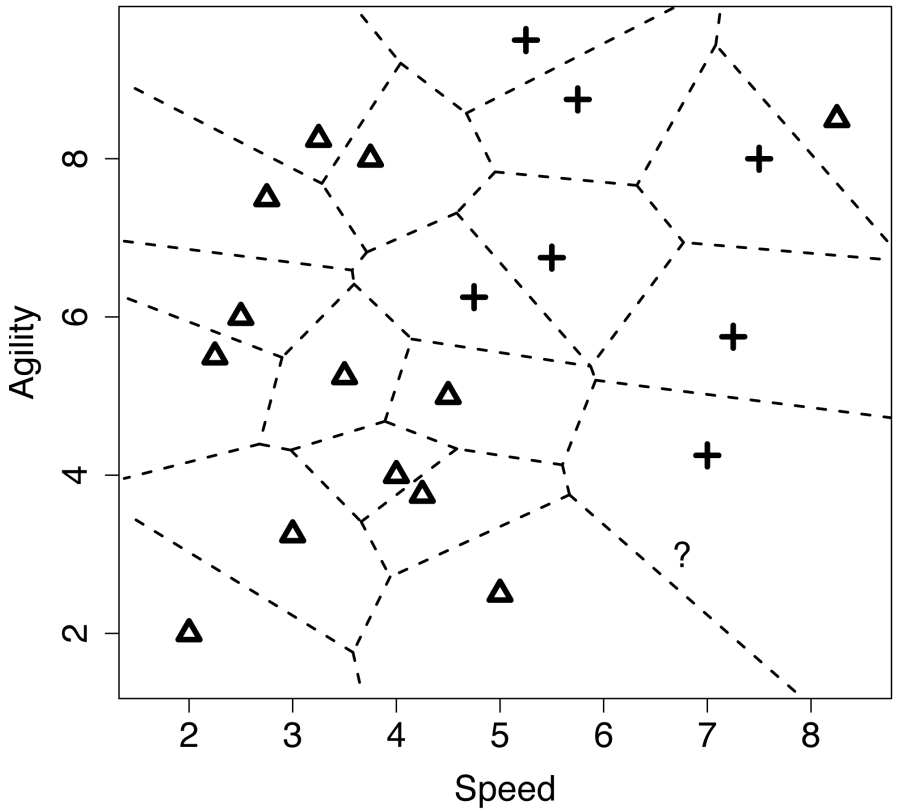


Figure: A feature space plot of the above data with the position in the feature space of the query represented by the ? marker. The triangles represent *'Non-draft'* instances and the crosses represent the *'Draft'* instances.

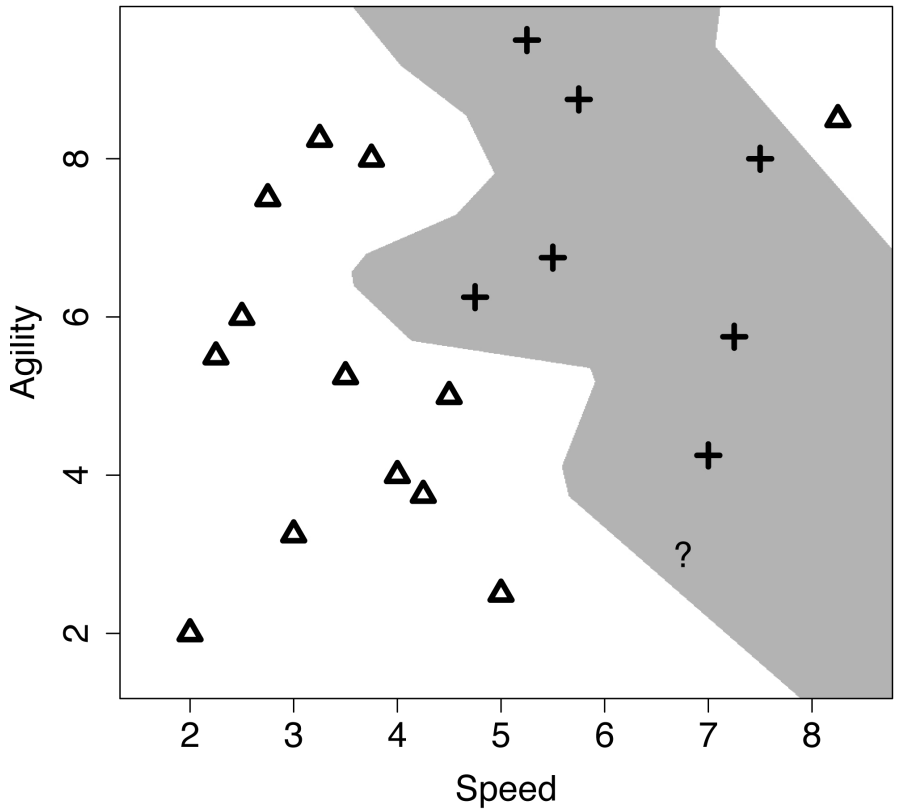
Table: The distances (Dist.) between the query instance with SPEED $= 6.75$ and AGILITY $= 3.00$ and each instance in the above table.

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

ID	SPEED	AGILITY	DRAFT	Dist.
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



(a) Voronoi tessellation



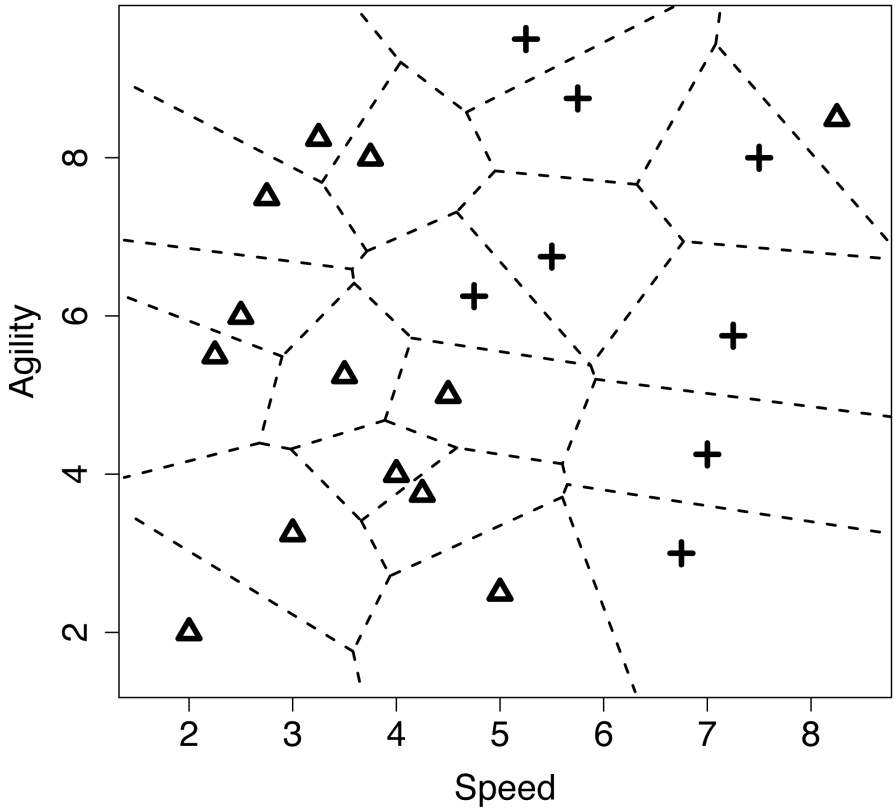
(b) Decision boundary ($k = 1$)

Figure: (a) The Voronoi tessellation of the feature space for the dataset in the above table 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.

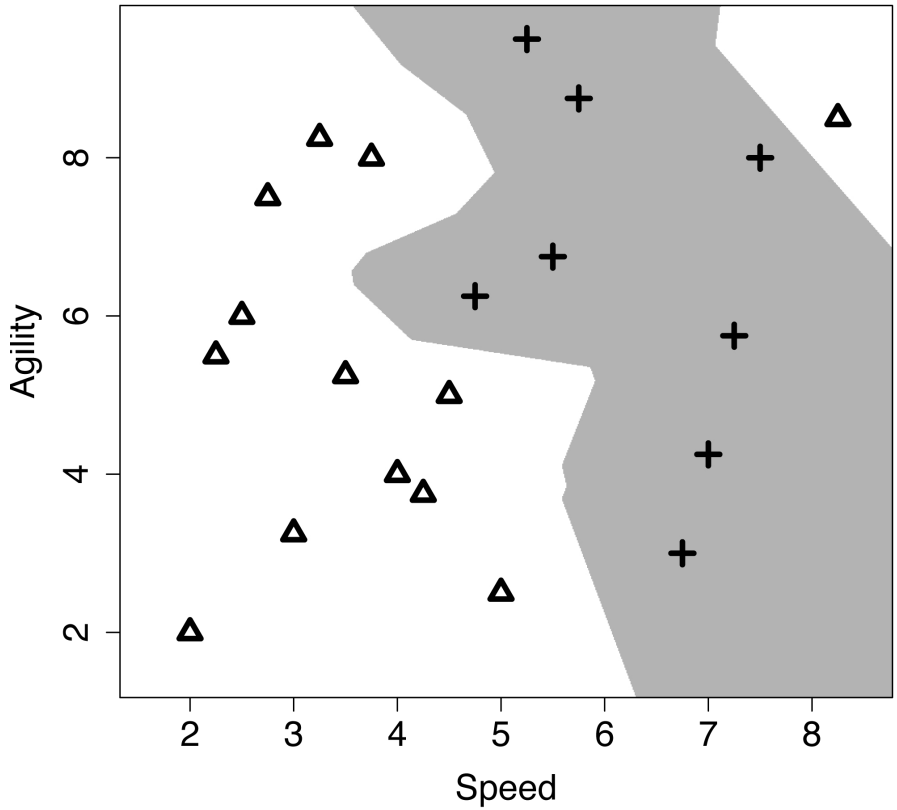
- One of the great things about nearest neighbour algorithms is that we can add in new data to update the model very easily.

Table: The extended version of the college athletes 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
21	6.75	3.00	yes



(a) Voronoi tessellation



(b) Decision boundary ($k = 1$)

Figure: (a) The Voronoi tessellation of the feature space when the dataset has been updated to include the query instance; (b) the updated decision boundary reflecting the addition of the query instance in the training set.

Handling Noisy Data

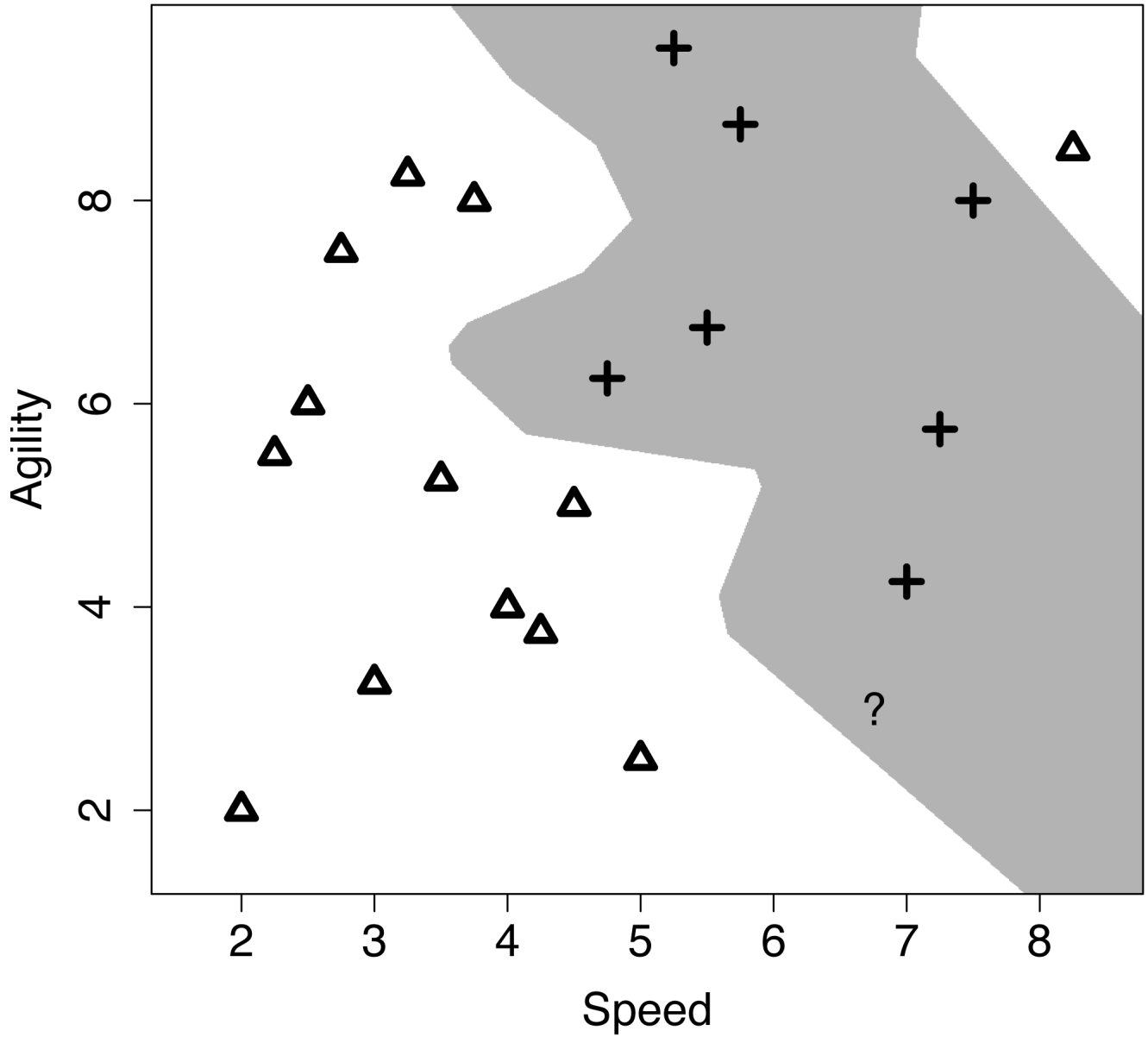


Figure: Is the instance at the top right of the diagram really *noise*?

- The k nearest neighbors model predicts the target level with the majority vote from the set of k nearest neightbors to the query **q**:

$$\mathbb{M}_k(\mathbf{q}) = \underset{l \in levels(t)}{\text{argmax}} \sum_{i=1}^{k} \delta(t_i, l) \tag{4}$$

- where the $\delta()$ is the Kronecker delta function that takes in two variables. It evaluates to 1 if the variables are equal and 0 otherwise.

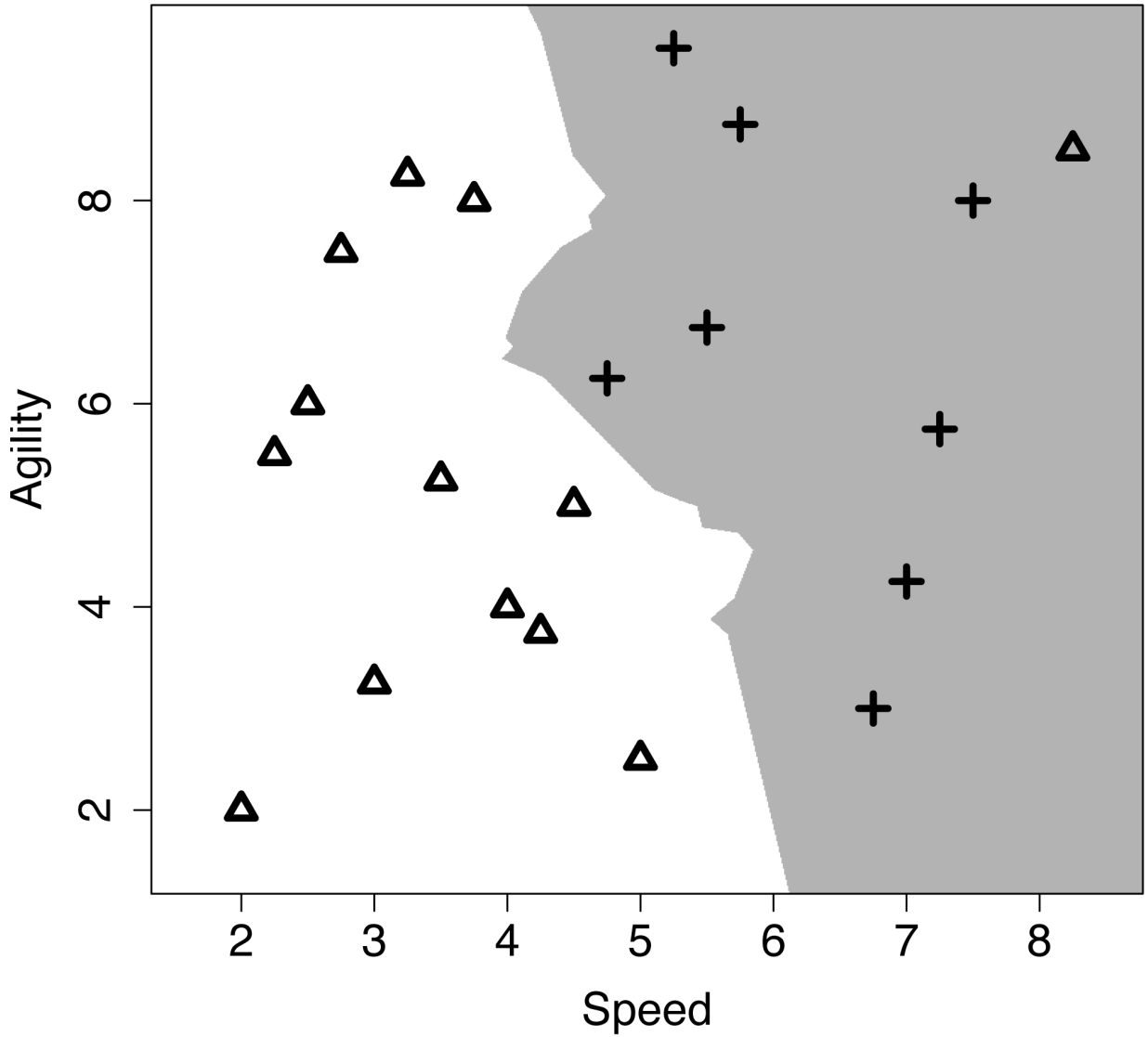


Figure: The decision boundary using majority classification of the nearest 3 neighbors.

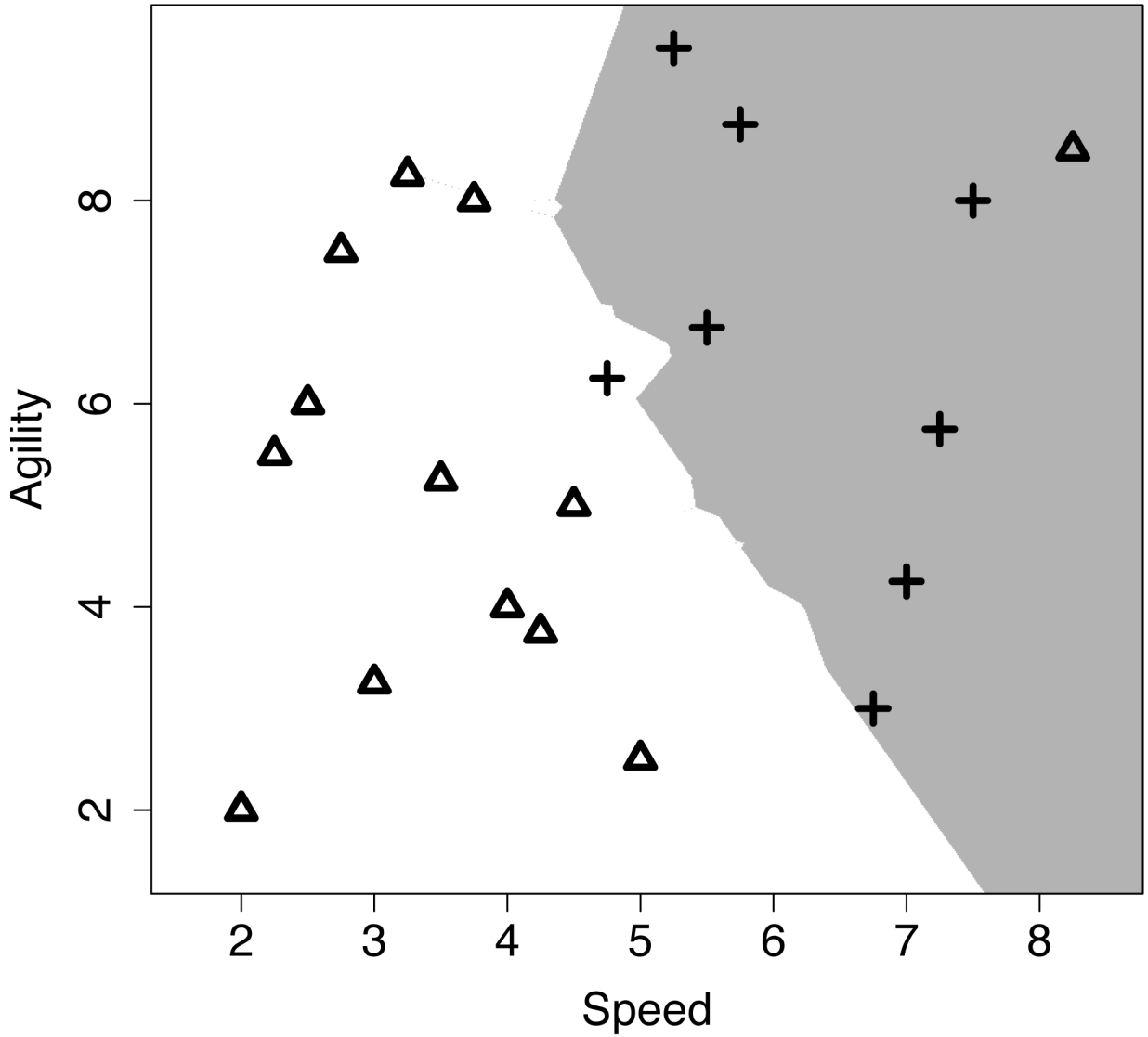


Figure: The decision boundary using majority classification of the nearest 5 neighbors.

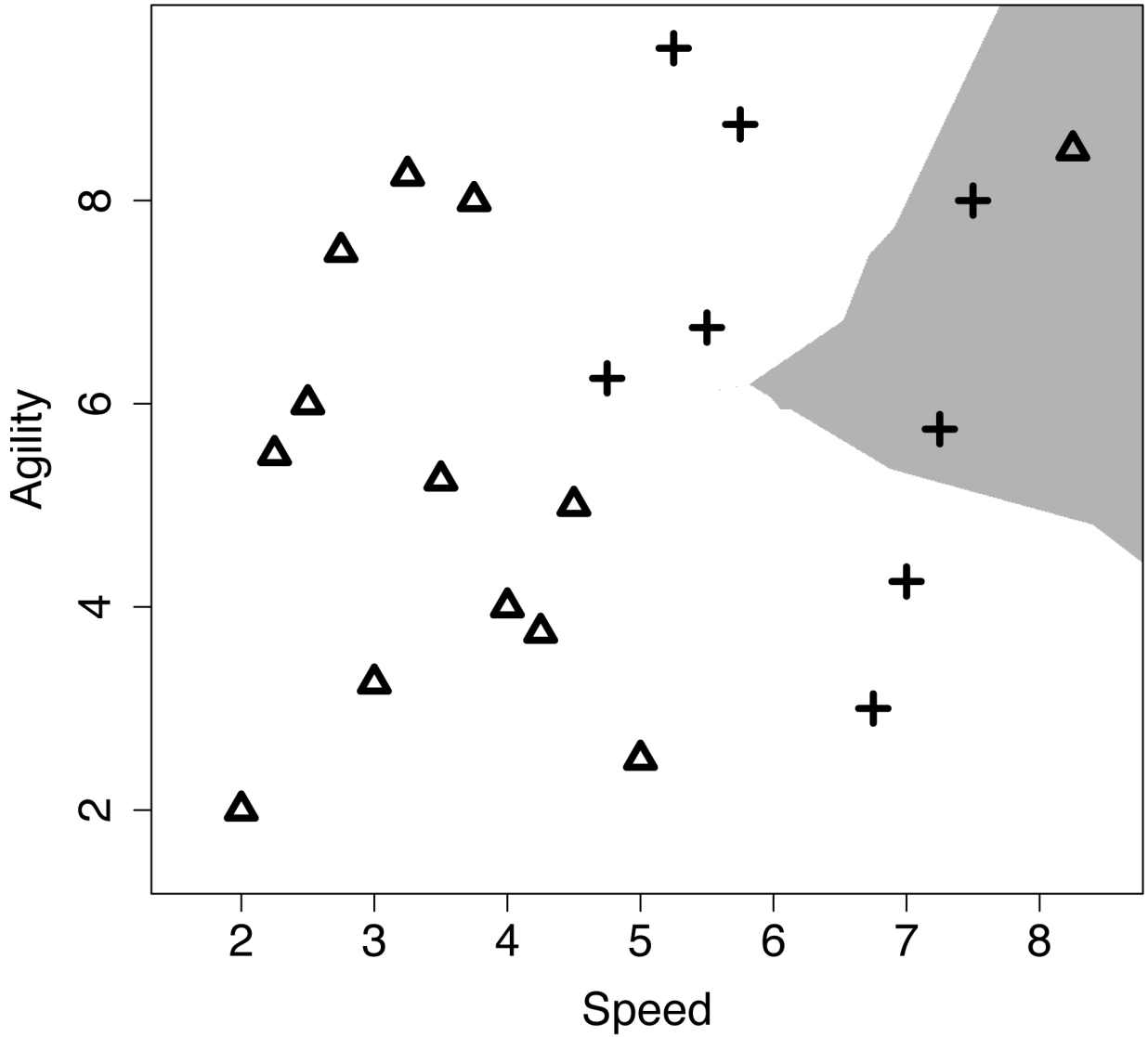


Figure: The decision boundary when k is set to 15.

- In a distance weighted k nearest neighbor algorithm the contribution of each neighbor to the classification decision is weighted by the reciprocal of the squared distance between the neighbor $\mathbf{d}$ and the query $\mathbf{q}$:

$$\frac{1}{dist(\mathbf{q}, \mathbf{d})^2} \tag{5}$$

- The weighted k nearest neighbor model is defined as:

$$\mathbb{M}_k(\mathbf{q}) = \underset{l \in levels(t)}{\text{argmax}} \sum_{i=1}^{k} \frac{1}{dist(\mathbf{q}, \mathbf{d_i})^2} \times \delta(t_i, l) \tag{6}$$

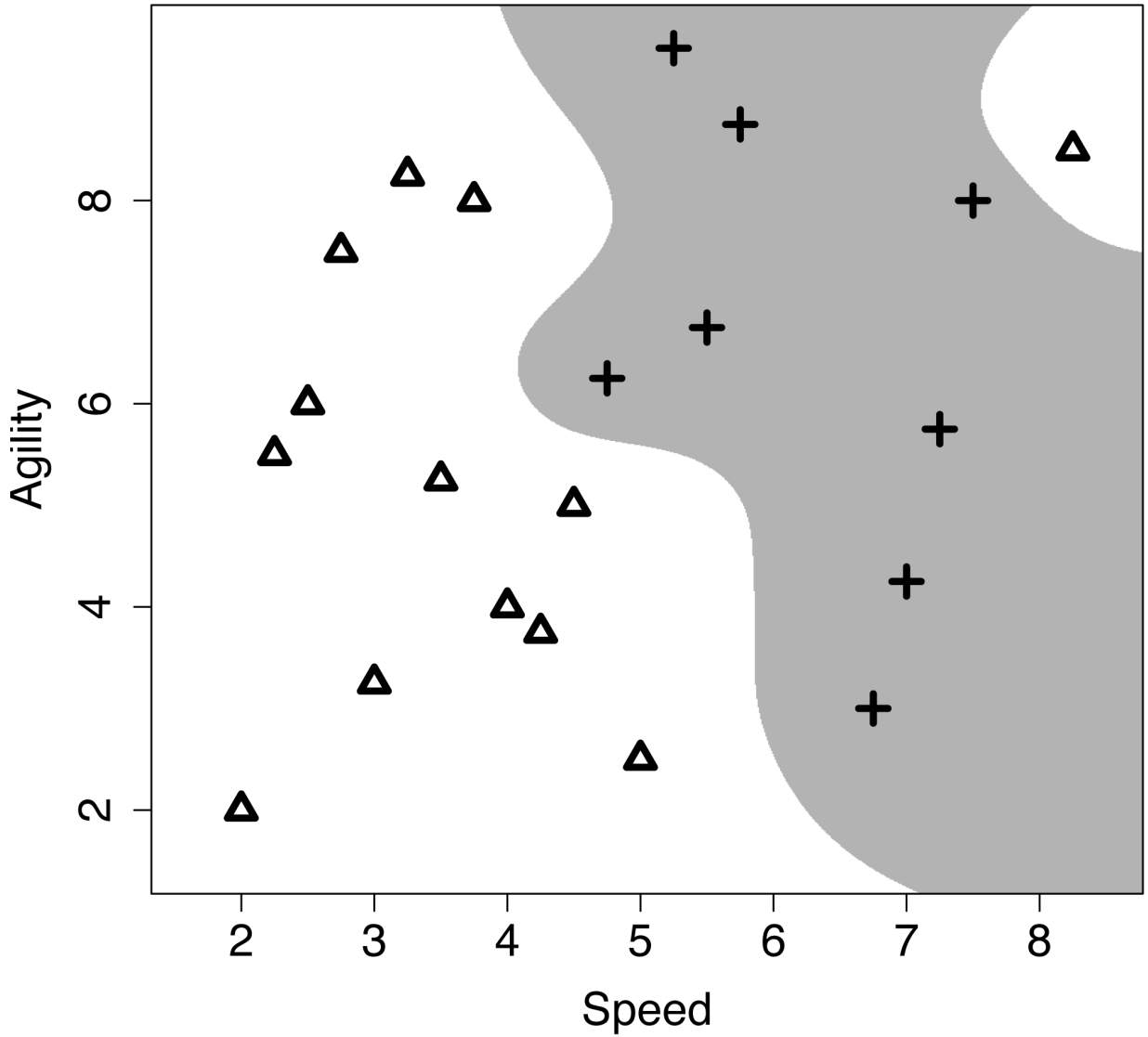


Figure: The weighted k nearest neighbor model decision boundary.

Data Normalization

Table: A dataset listing the salary and age information for customers and whether or not the purchased a pension plan .

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 marketing department wants to decide whether or not they should contact a customer with the following profile:

$$\langle \text{SALARY} = 56,000, \text{AGE} = 35 \rangle$$

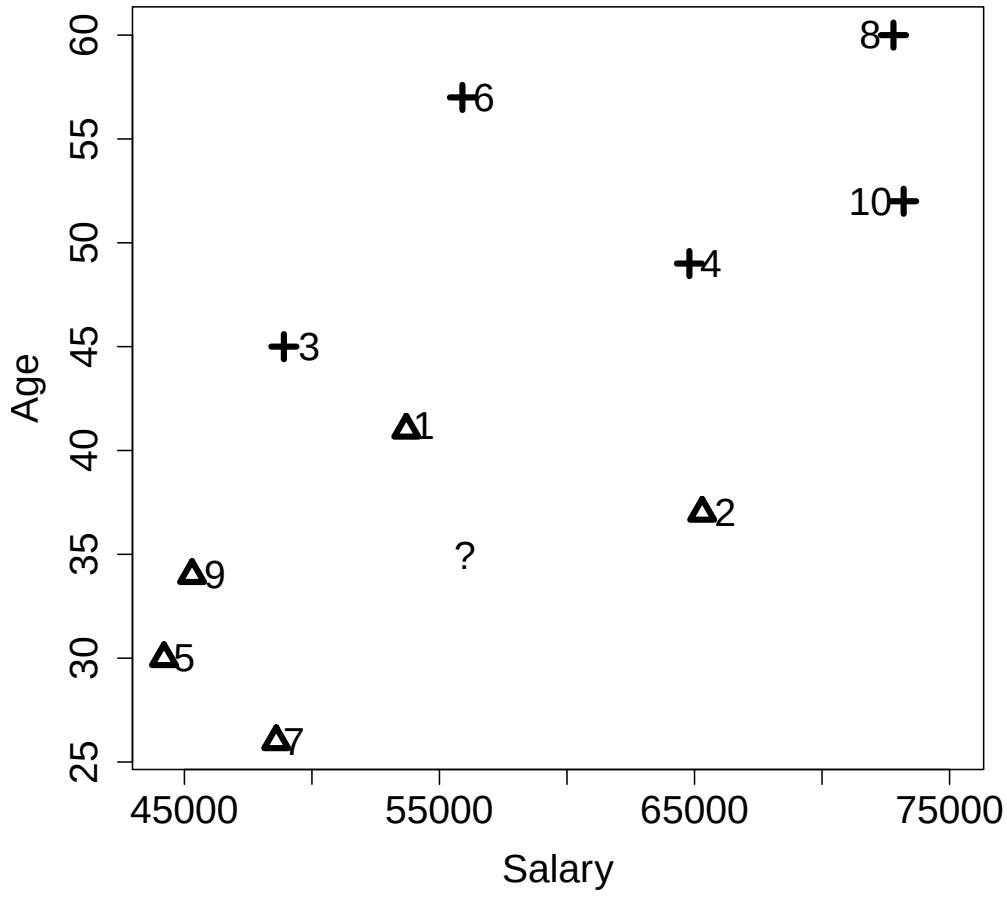


Figure: The salary and age feature space with the data in Table 5 [40] plotted. The instances are labelled their IDs, triangles represent the negative instances and crosses represent the positive instances. The location of the query $\langle \text{SALARY} = 56,000, \text{AGE} = 35 \rangle$ is indicated by the ?.

				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

- This odd prediction is caused by features taking different ranges of values, this is equivalent to features having different variances.
- We can adjust for this using normalization; the equation for range normalization is:

$$a_i' = \frac{a_i - min(a)}{max(a) - min(a)} \times (high - low) + low \tag{7}$$

	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

- Normalizing the data is an important thing to do for almost all machine learning algorithms, not just nearest neighbor!

Predicting Continuous Targets

- Return the average value in the neighborhood:

$$\mathbb{M}_k(\mathbf{q}) = \frac{1}{k} \sum_{i=1}^{k} t_i \tag{8}$$

Table: A dataset of whiskeys listing the age (in years) and 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

ID	Age	Rating	Price
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

Table: The whiskey dataset after the descriptive features have been normalized.

ID	Age	Rating	Price	ID	Age	Rating	Price
1	0.0000	0.25	30.00	11	0.6333	1.00	500.00
2	0.4000	0.63	40.00	12	0.2000	0.88	200.00
3	0.3333	0.75	55.00	13	0.2667	0.63	65.00
4	0.7000	0.88	550.00	14	0.7333	0.75	120.00
5	0.4000	0.50	35.00	15	0.2000	0.25	12.00
6	0.5000	0.63	45.00	16	0.2667	0.88	250.00
7	0.5333	0.75	70.00	17	0.3333	0.25	18.00
8	0.6000	0.50	85.00	18	1.0000	0.88	450.00
9	0.6000	0.63	78.00	19	0.0333	0.00	10.00
10	0.5333	0.50	75.00	20	0.1333	0.50	30.00

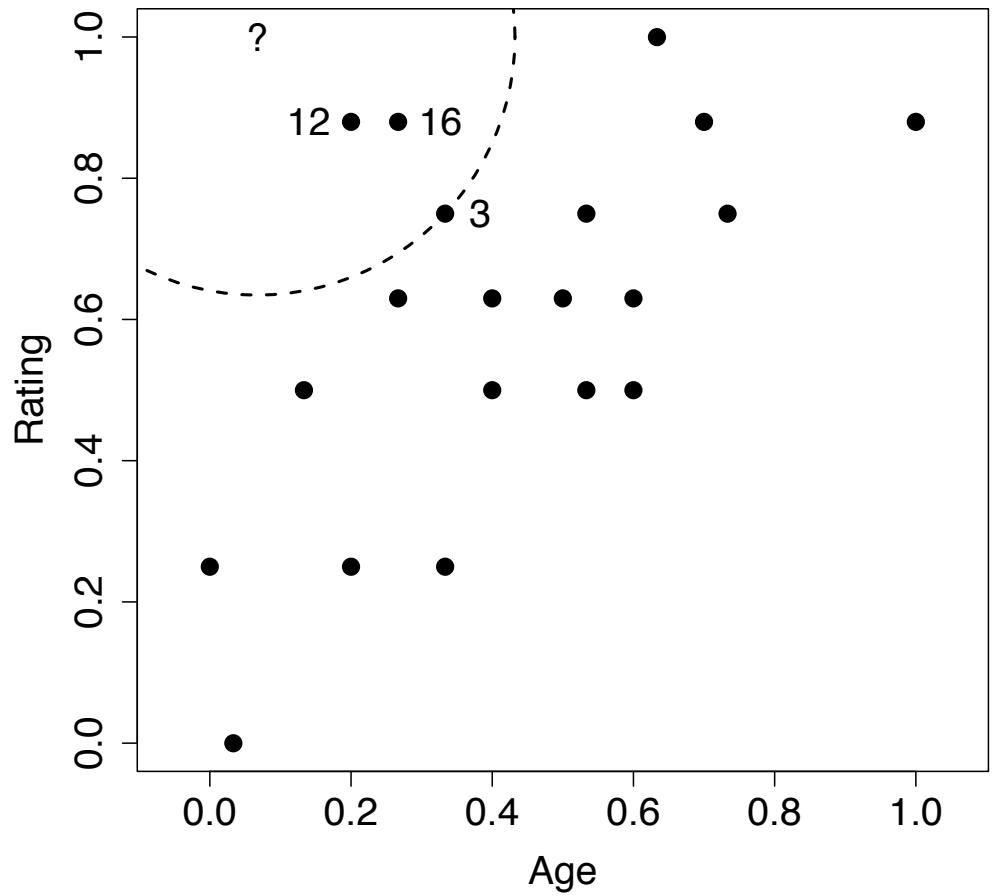


Figure: The AGE and RATING feature space for the whiskey dataset. The location of the query instance is indicated by the ? symbol. The circle plotted with a dashed line demarcates the border of the neighborhood around the query when $k = 3$. The three nearest neighbors to the query are labelled with their ID values.

- The model will return a price prediction that is the average price of the three neighbors:

$$\frac{200.00 + 250.00 + 55.00}{3} = 168.33$$

1. Big Idea
2. Fundamentals
3. Standard Approach: The Nearest Neighbor Algorithm
4. Handling Noisy Data
5. Data Normalization
6. Predicting Continuous Targets