COMP30120

Nearest Neighbour Classifiers

Derek Greene

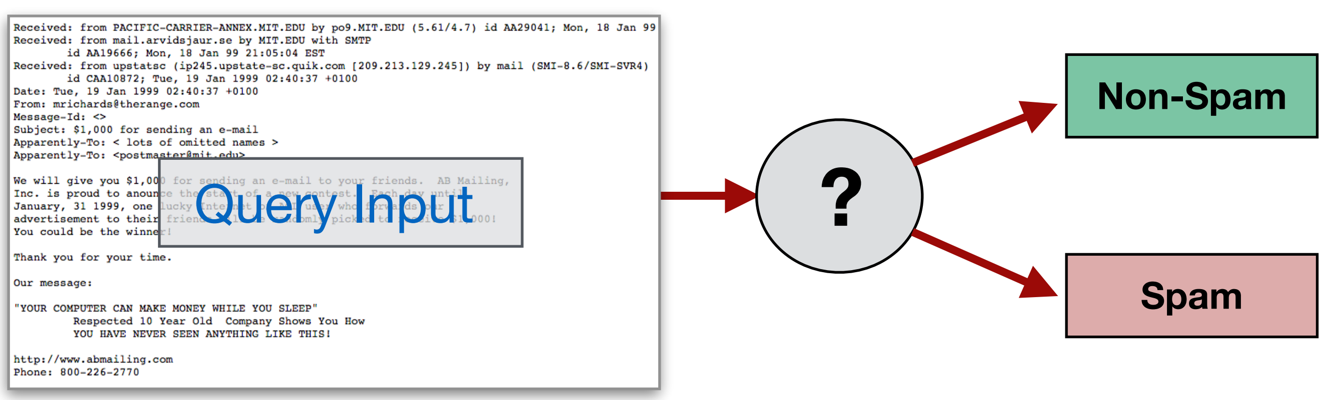
School of Computer Science and Informatics
Autumn 2015

Overview

- Eager v Lazy Classification Strategies
- Similarity-based Learning
- The *k*-Nearest Neighbour Classifier
  - How do we measure distance/similarity?
  - How do we prepare the data?
  - How do we select useful training data?
- Classifying Text Documents
  - Application: Spam Filtering
- *k*-NN in Weka

Reminder: Classification

- Supervised Learning: Algorithm that learns a function from manually-labelled training examples.
- Classification: Training examples, usually represented by a set of features, help decide *class* to which a new unseen query input belongs.
- Binary Classification: Assign one of two possible target class labels to the new query input.



- Multiclass Classification: Assign one of *M>2* possible target class labels to the new query input.

Lazy v Eager Classifiers

- Eager Learning
  - Classifier builds a full model during an initial training phase, to use later when new query examples arrive.
  - More offline setup work, less work at run-time.
  - Generalise before seeing the query example.
  - e.g. Decision Tree classifier
- Lazy Learning
  - Classifier keeps all the training examples for later use.
  - Little work is done offline, wait for new query examples.
  - Focus on the local space around the examples.
  - e.g. *k*-Nearest Neighbour classifier (*k*-NN)

Example: Athlete Selection

- Dataset of performance ratings for 20 college athletes.
- Each athlete described by 2 continuous features: *speed*, *agility.* Binary class label indicates whether or not they were *selected* for the college team.

Athlete	***Speed***	***Agility***	***Selected***
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

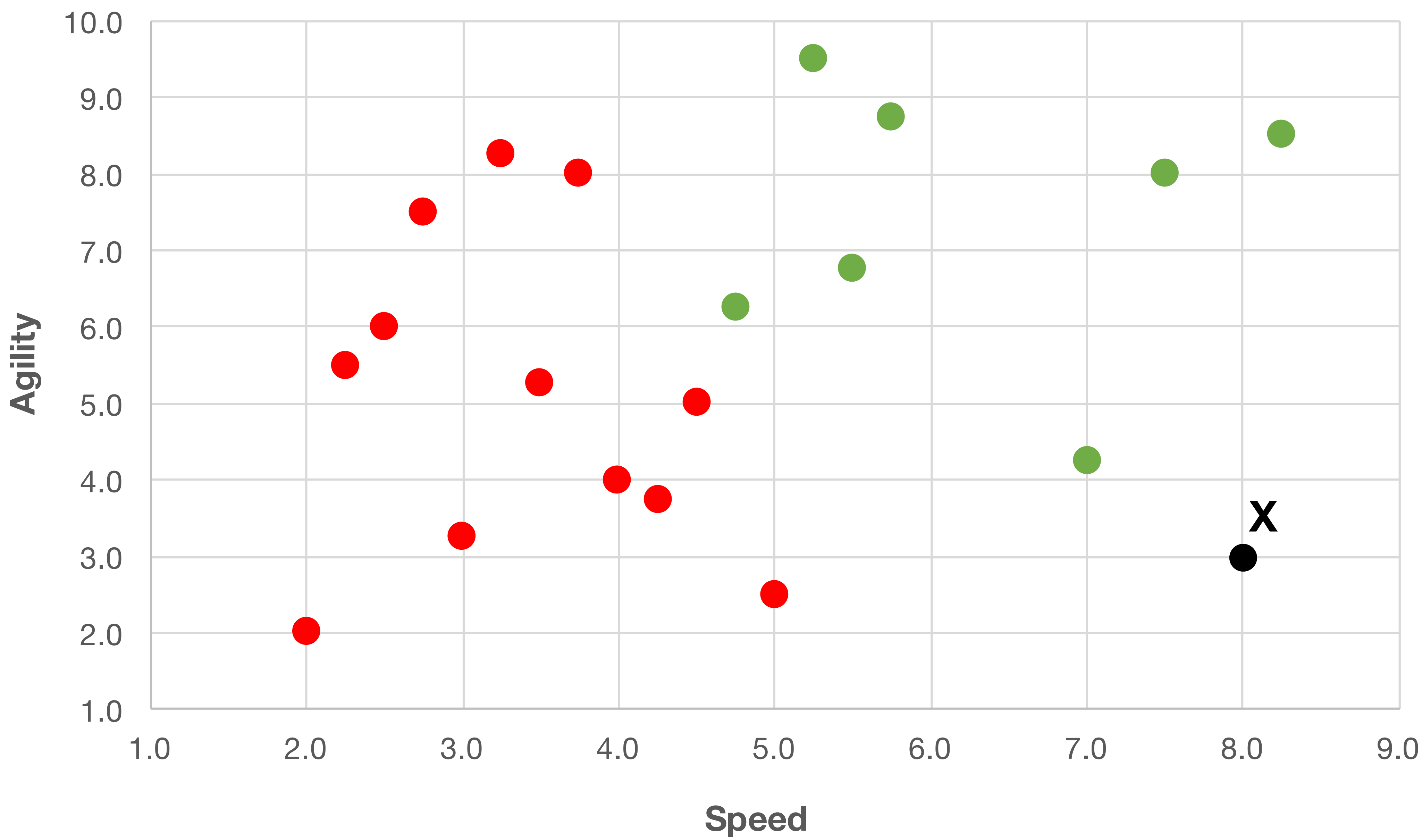
Athlete	***Speed***	***Agility***	***Selected***
11	2.00	2.00	No
12	5.00	2.50	No
13	8.25	8.50	Yes
14	5.75	8.75	Yes
15	4.75	6.25	Yes
16	5.50	6.75	Yes
17	5.25	9.50	Yes
18	7.00	4.25	Yes
19	7.50	8.00	Yes
20	7.25	3.75	Yes

Q. Will athlete X be selected?

Athlete	***Speed***	***Agility***	***Selected***
X	3.00	8.00	???

Example: Athlete Selection

We can use the feature values to visually plot the 20 athletes in a 2-dimensional coordinate space (i.e. *agility* versus *speed*):

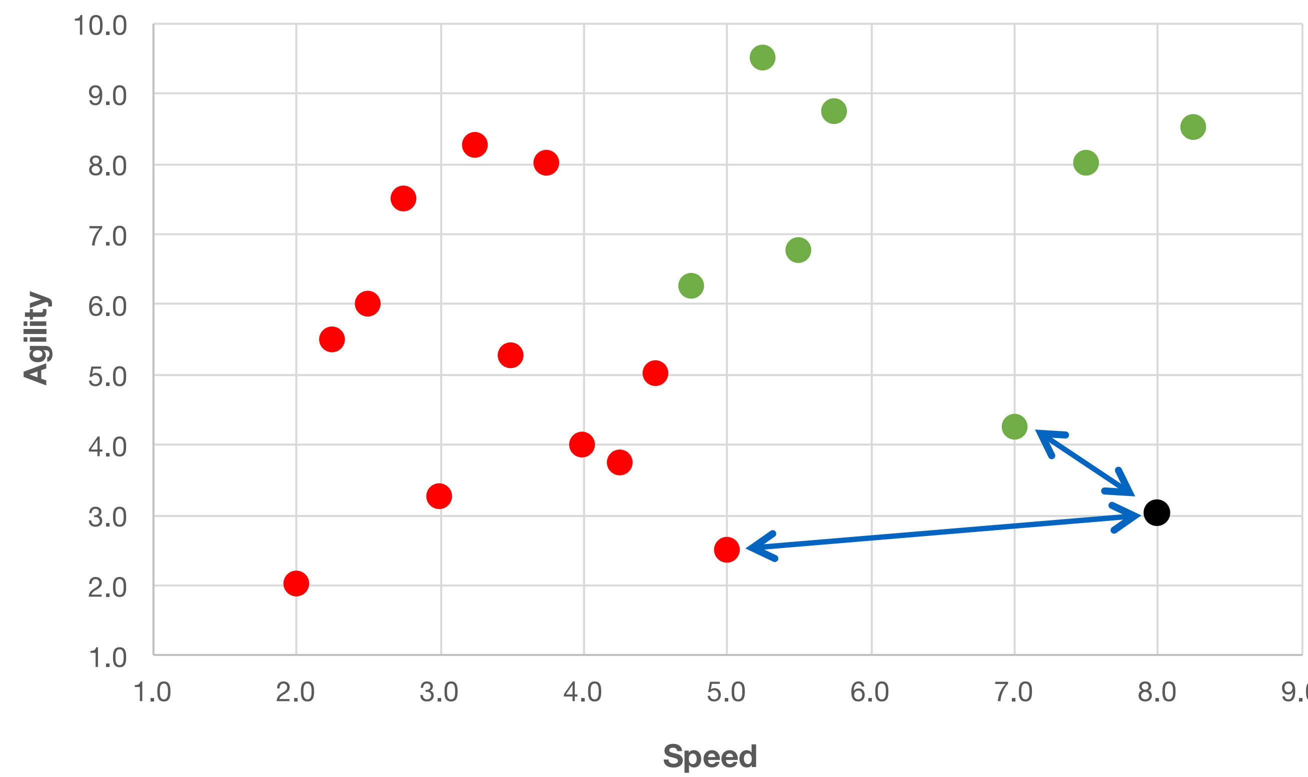


Similarity-based Learning

Fundamental Strategy: "Best way to make predictions is to look at past examples and repeat the same process again".

Features space:
A *D*-dimensional coordinate space used to represent the input examples for a given problem, with one coordinate per descriptive feature.

Similarity measure:
Some function to measure how similar (or distant) two input examples are from one another are in the *D*-dimensional coordinate space.



2 features describing each example (agility & speed)

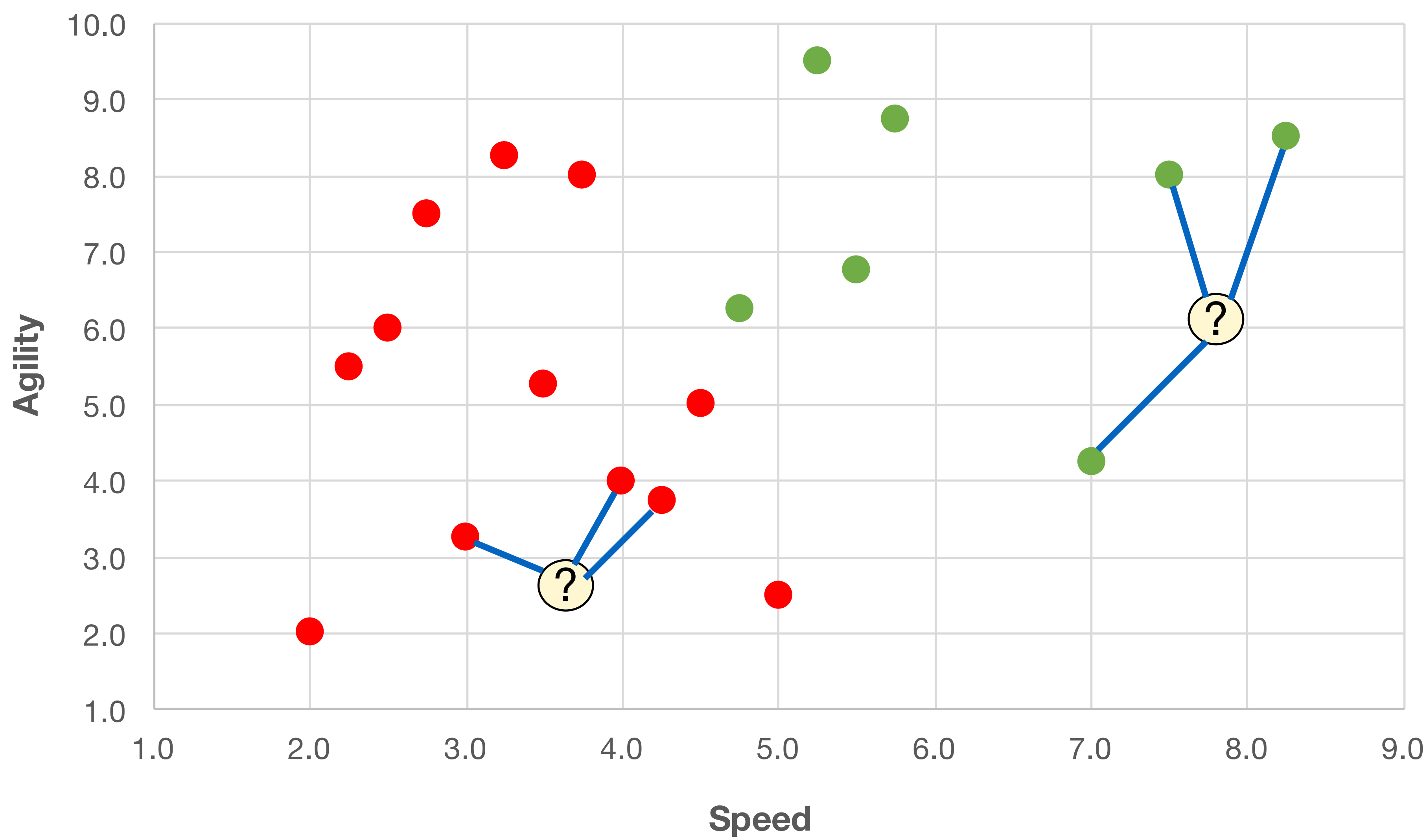
➡ 2 coordinate dimensions for measuring similarity

Typically: $\text{Distance} = 1/\text{Similarity}$ OR $\text{Distance} = 1 - \text{Similarity}$

Nearest Neighbours

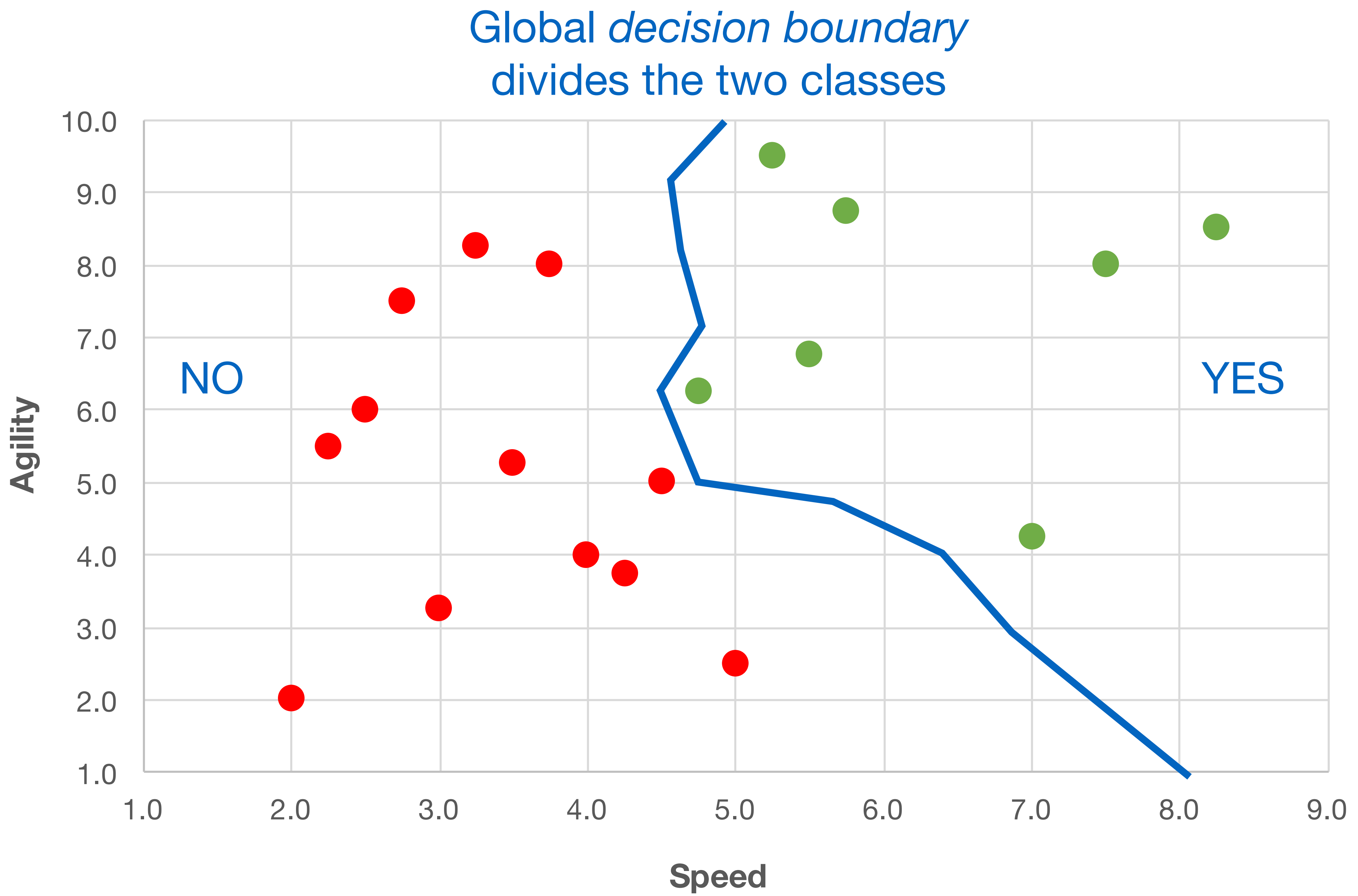
Lazy Learning approach: Identify most similar previous athlete for which a selection decision has already been made (i.e. their nearest neighbours from the training set).

Example: For new query inputs, look at the label of *k* nearest neighbours under both features (e.g. *k=3* neighbours)



Nearest Neighbours

While nearest neighbour methods only consider *local* neighbours of each example, it implicitly allows us to build a *global* model that covers the entire dataset.



k-Nearest Neighbour Classifier

- Inputs:
  - Set of labelled training examples represented by features *F*
  - A query input example ***q*** represented by features *F*
  - User specified parameter value *k* (i.e. number of neighbours)
- Task:
  - Find the *k* nearest neighbours for input ***q*** according to the *distance measure* defined as…

For each $x_i \in D$

$$d(q, x_i) = \sum_{f \in F} w_f \cdot \delta(q_f, x_f)$$

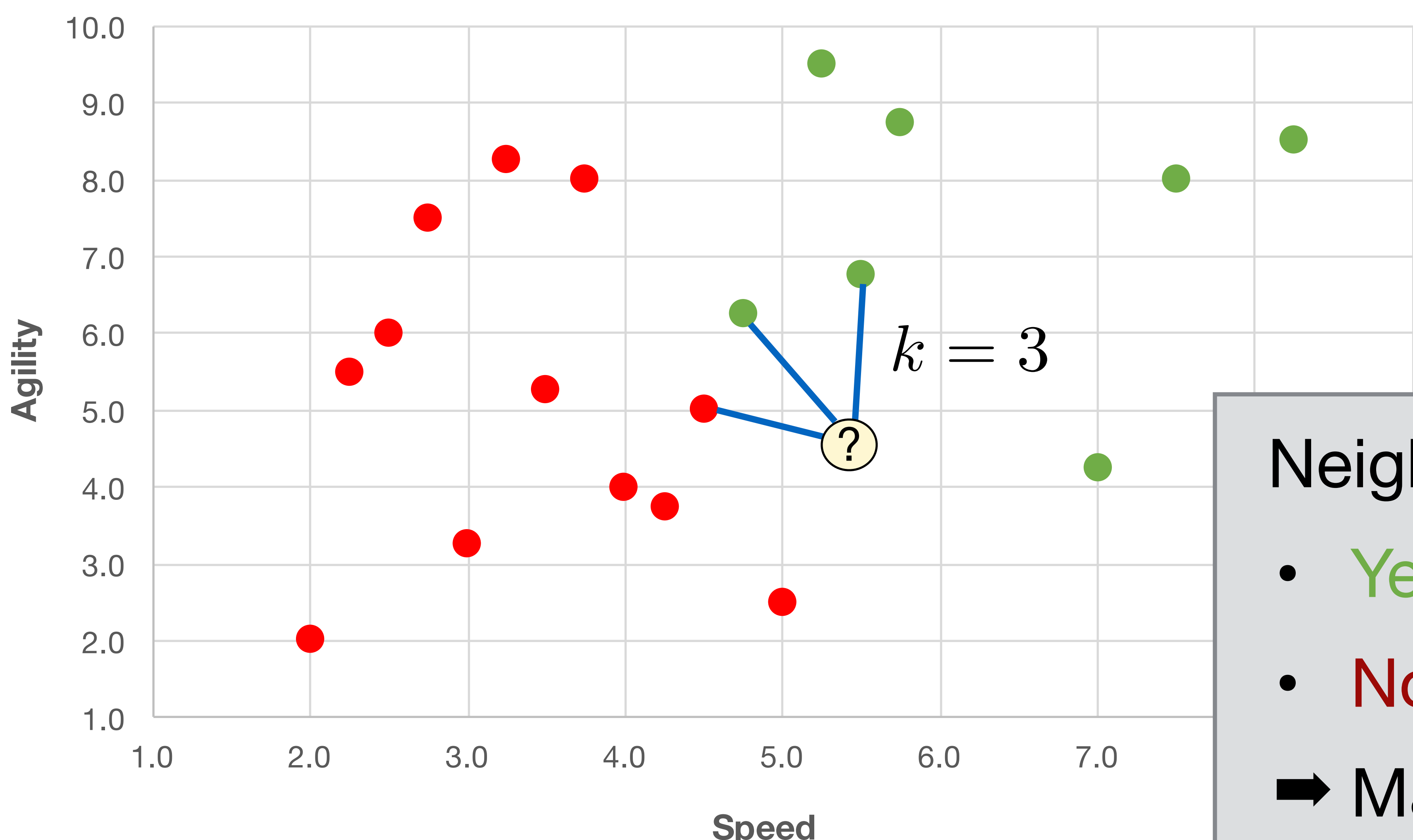
Weighted sum over all features

Difference calculation depends on feature type (e.g. continuous, binary, ordered)

$$d(q_f, x_{if}) = \begin{cases} 0, & \text{if } f \text{ discrete and } q_f = x_{if} \\ 1, & \text{if } f \text{ discrete and } q_f \neq x_{if} \\ \left| q_f - x_{if} \right|, & \text{if } f \text{ continuous} \end{cases}$$

k-Nearest Neighbour Classifier

- Majority voting: The decision on a label for a new query example is decided based on the "votes" of its *k* nearest neighbours, where the neighbours are selected based on minimising the distance $d(q, x_i)$

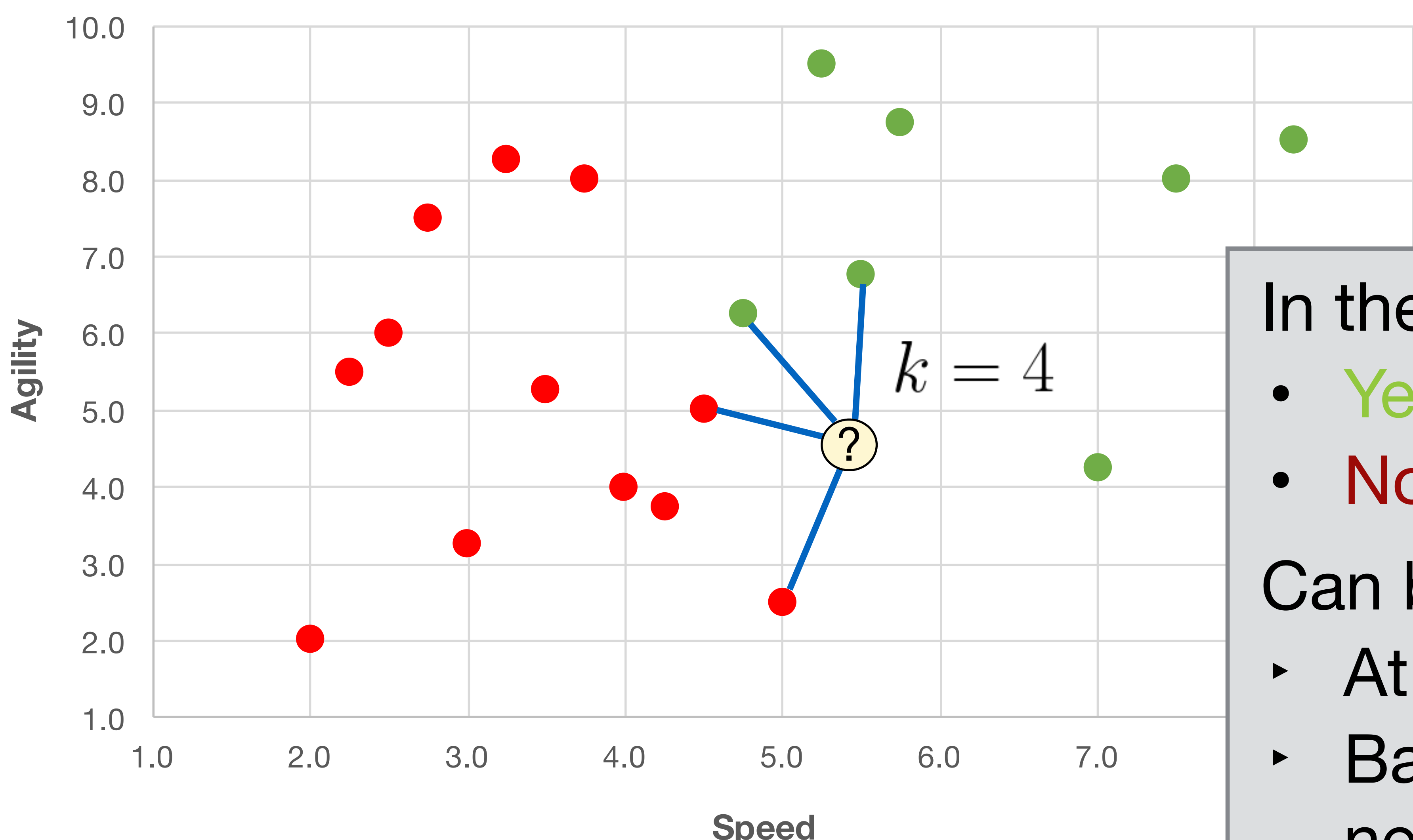


Neighbour counts
- Yes = 2 votes
- No = 1 vote

➡ Majority says Yes!

k-Nearest Neighbour Classifier

- Majority voting: The decision on a label for a new query example is decided based on the “votes” of its *k* nearest neighbours, where the neighbours are selected based on minimising the distance $d(q, x_i)$



In the case that…
- Yes = 2 votes
- No = 2 votes

Can break ties…
- At random
- Based on sum of neighbour distances

Measuring Distance

- Absolute difference: Calculate the absolute value of the difference between the feature values.

Example	Height	Width
p	60	62
q	70	53

```
diff(p,q) = |60-70| + |62-53| = 10+9 = 19
```

- For *ordinal features*, calculate the absolute value of the difference between the two positions in the ordered list of possible values.

Ordinal Feature “Dosage”:
{Low,Medium,High} = {1, 2, 3}

```
diff(Low,High) = |1-3| = 2
diff(Medium,Low) = |2-1| = 1
diff(High,High) = |3-3| = 0
```

- Euclidean distance: Common distance measure between two *continuous features* (inputs represented as numeric vectors).

$$\mathrm{ED}(\mathbf{p},\mathbf{q}) = \sqrt{\sum_{f \in F} (q_f - p_f)^2}$$

$$ED(p,q) = \sqrt{(60-70)^2 + (62-53)^2}$$
$$= 13.45$$

Data Normalisation

- Numeric features often have different ranges, which can skew certain distance measures.
- So that all features have similar range, we apply *feature normalisation*.
- Min-max normalisation:
  Use min and max values for a given feature to rescale to the range [0,1]
- Example: Feature “Age”

Example	Height (Inches)	Weight (Lbs)	Age (Years)
1	65.78	112.99	24
2	71.52	136.49	19
3	69.40	153.03	50
4	68.22	142.34	40
5	67.79	144.30	23
6	68.70	123.30	68
7	69.80	141.49	45
8	70.01	136.46	33
9	67.90	112.37	80
10	66.78	120.67	58

$$z_i = \frac{x_i - \min(x)}{\max(x) - \min(x)}$$

$$\min(x) = 19$$

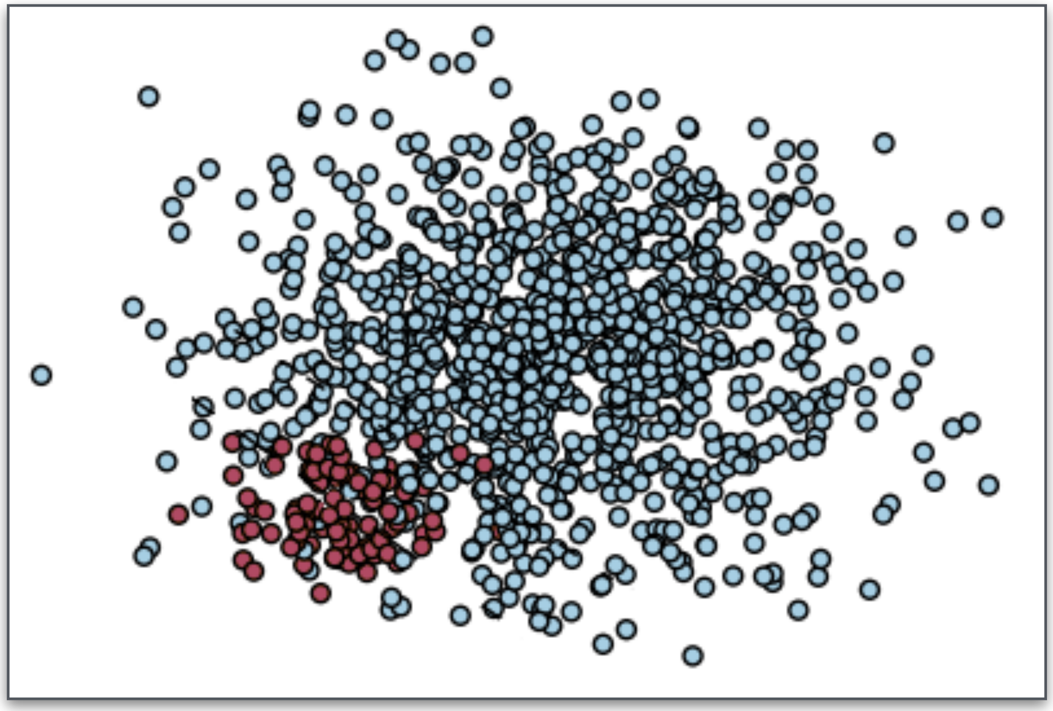
$$\max(x) = 80$$

$$\max(x) - \min(x) = 61$$

Age (Non-normalised)	24	19	50	40	23	68	45	33	80	58
Age (Normalised)	0.08	0.00	0.51	0.34	0.07	0.80	0.43	0.23	1.00	0.64

Noisy Data

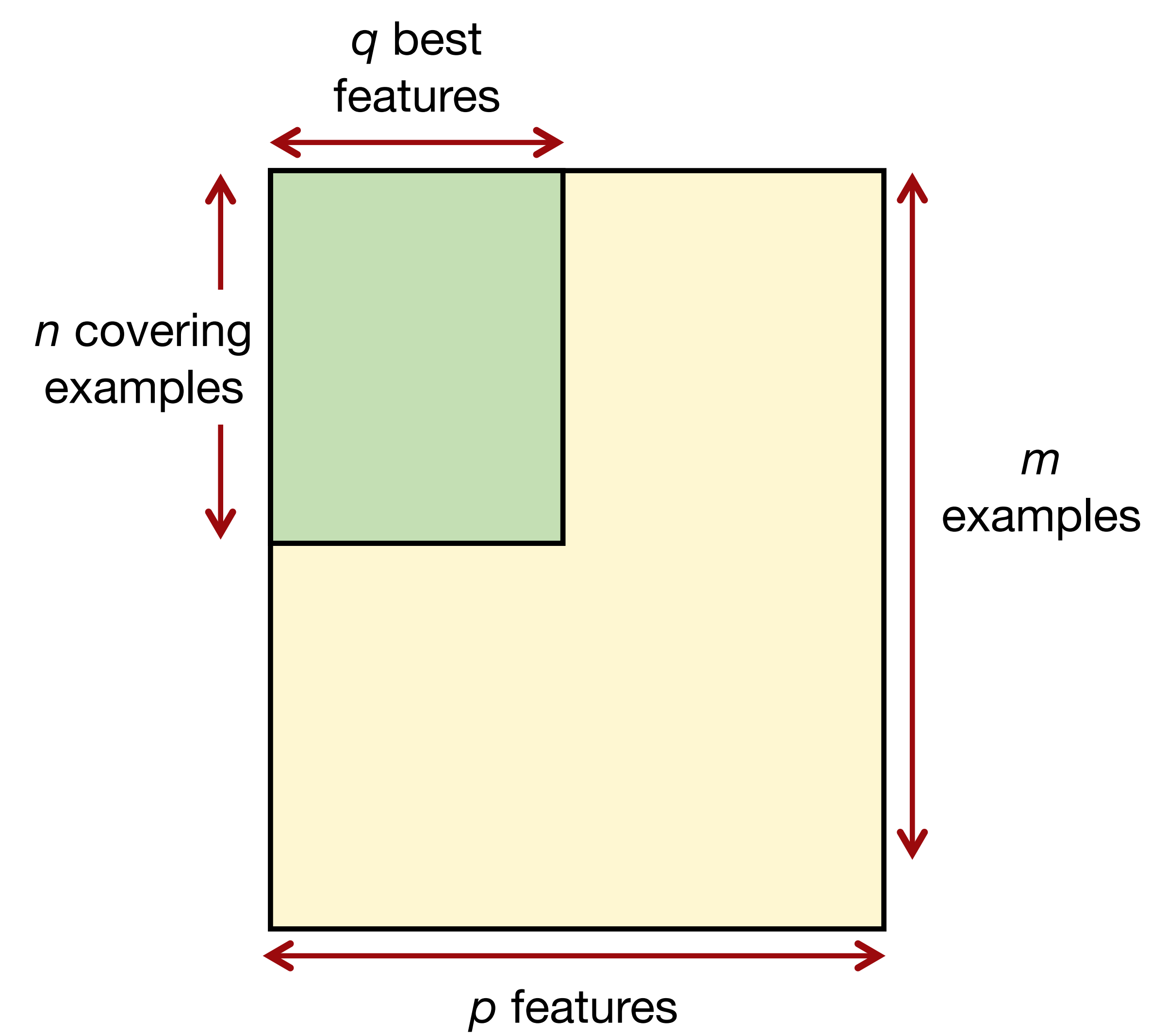
- A simple 1-NN classifier is easy to implement.
- But it will be susceptible to “noise” in the data.
    - ➡ A misclassification will occur every time a single noisy example is retrieved.
- Using a larger neighbourhood size (e.g. $k > 2$) can sometimes make the classifier more robust and overcome this problem.
- But when k is large ($k \rightarrow N$) and classes are *unbalanced*, we always predict the majority class.



Dimension Reduction for *k*-NN

- Feature Selection:
  For a given dataset, not all features may be required. Noisy or redundant features can hinder the algorithm.

- Case Selection:
  For a given dataset, not all training examples may be required. Some are redundant, increasing algorithm training time.



Data matrix

Q. How do we find the best feature and case subsets?

Condensed NN

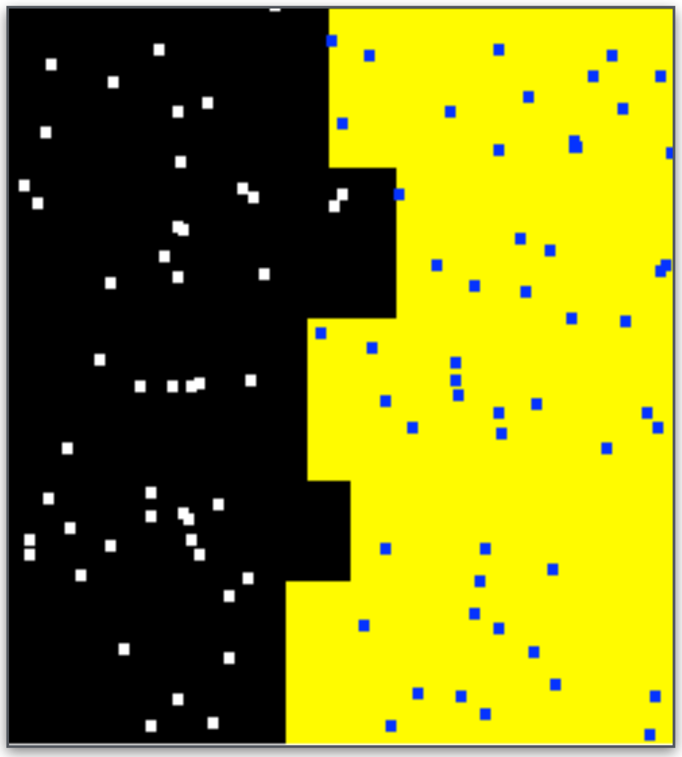
- Input: A set of D training examples.
- Task: Find subset $E \subset D$, where the Nearest Neighbour rule used with E should be as good as the full set D.

Condensed NN (CNN) algorithm

- Choose an example x $\in D$ randomly
- $D \leftarrow D \setminus \{x\}$
- $E \leftarrow \{x\}$
- REPEAT
  - learning $\leftarrow$ False
  - FOR EACH y $\in D$
    - Classify y by nearest neighbours using E
    - IF classification incorrect THEN
      - $D \leftarrow D \setminus \{y\}$
      - $E \leftarrow E \cup \{y\}$
      - learning $\leftarrow$ True
- WHILE learning $\neq$ False

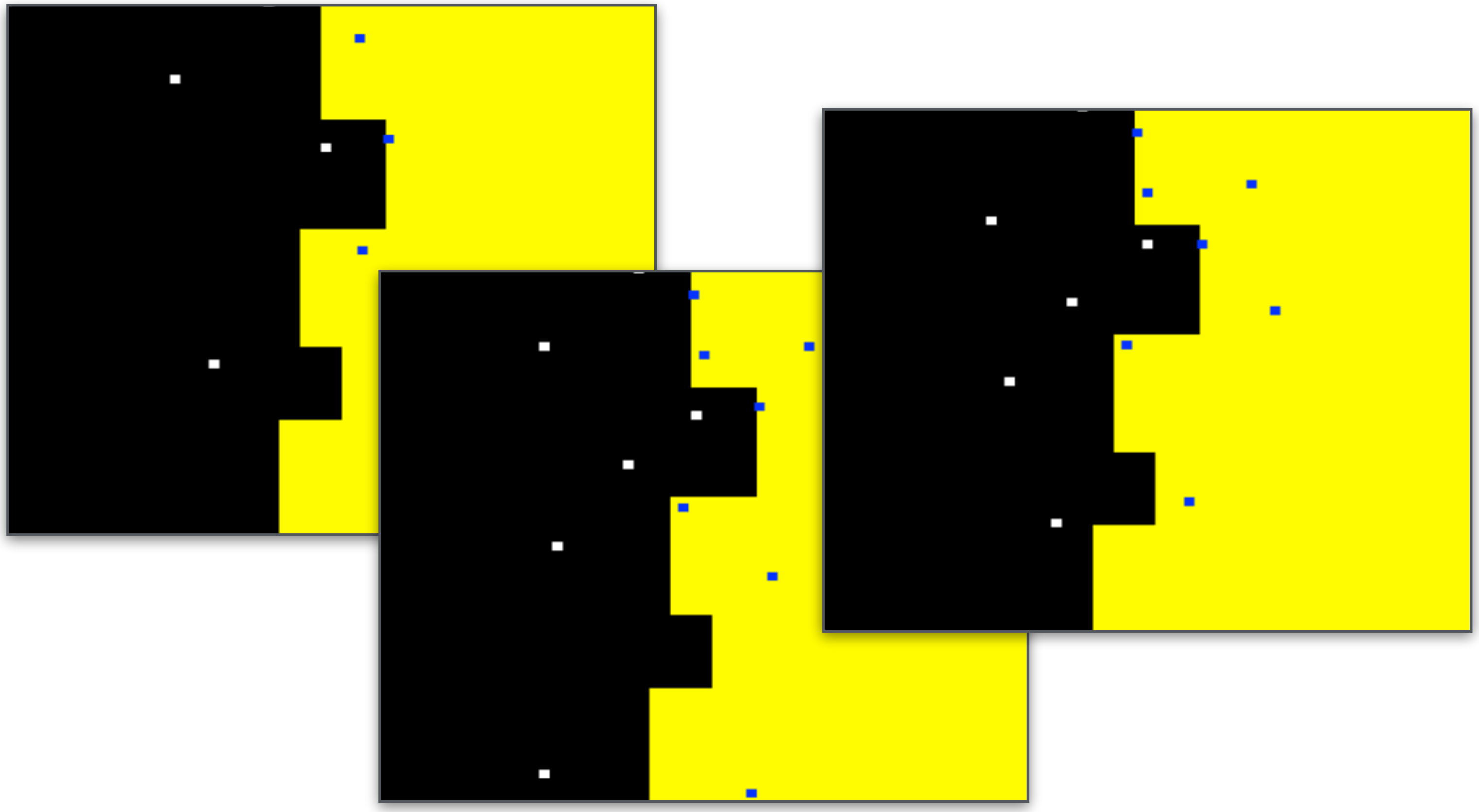
Condensed NN

- Example: 100 examples with 2 target class labels.



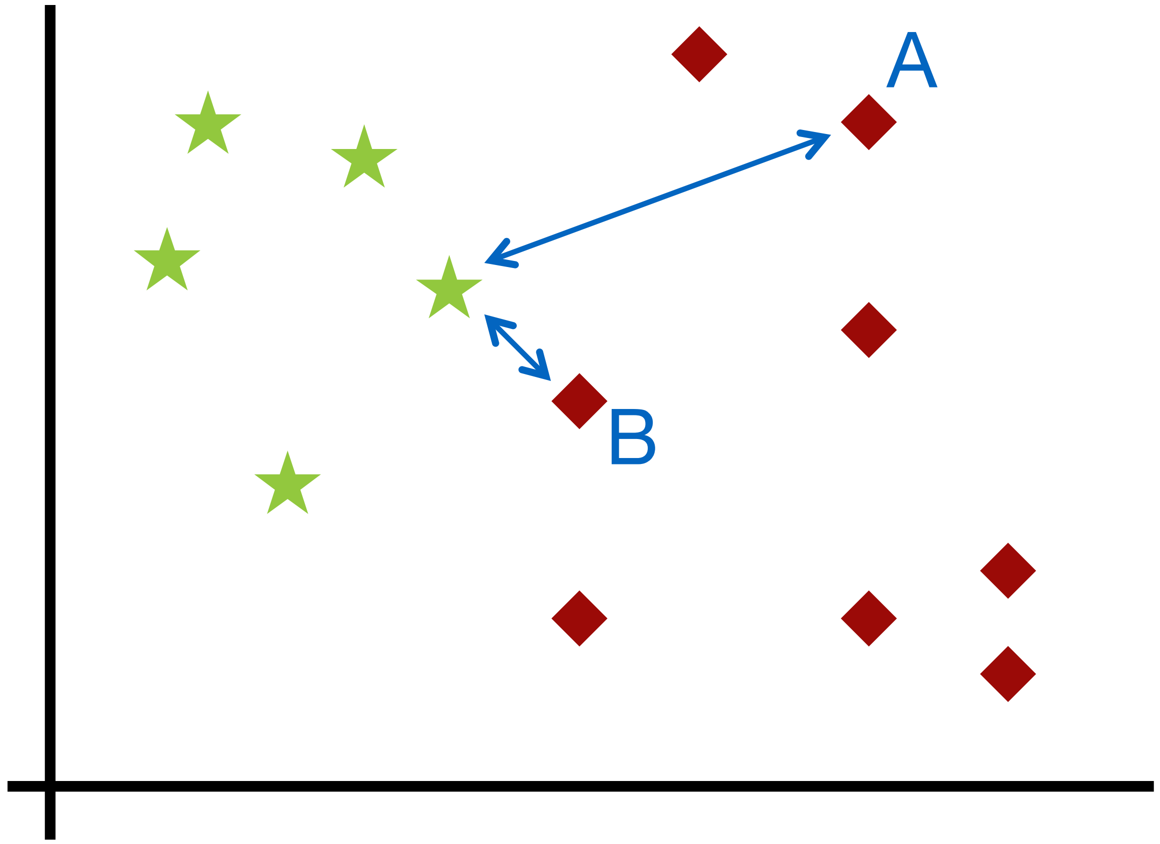
Complete data set

Random Initialisation
⇒ Different Condensed NN solutions



Condensed NN

- Problem: Different outcomes, depending on the data order. *Non-deterministic* → Not a desirable property in an algorithm!
- Improving CNN: Sort examples based on the distance to *nearest unlike neighbour*.



Motivation:

Identify examples near the *decision boundary*.

e.g. example B is more useful than A

Application: Spam Filtering

- Concept drift: 2015 spam does not look like spam from 2006!
- A Lazy Learning system should be able to adapt to the changing nature of spam/real email content.

➡ Simply add more recent examples, remove old examples from the training set.

- **Classifier system design questions…**

  Q. How do we represent our data?

  Q. What are the relevant features and examples?

  Q. How do we measure distance/similarity?

  Q. What are the appropriate parameters for our algorithm?

Text as Bag-of-Words

- Raw email data is textual, not numeric. Requires pre-processing.
- Bag-of-Words Model: Each document is represented by a vector in a *m*-dimensional coordinate space, where *m* is number of unique terms (words) across all documents in the data.

It is the easiest online school on earth.
Here is the information from Computer Science for the next Graduate School Board meeting.
Please find attached the agenda for the Graduate School Board meeting.

Remove “stopwords”

→

easiest	online	school	earth	information	computer	science	graduate	board	meeting	please	find	attached	agenda
1	1	1	1	0	0	0	0	0	0	0	0	0	0
0	0	1	0	1	1	1	1	1	1	0	0	0	0
0	0	1	0	0	0	0	1	1	1	1	1	1	1

Convert email text to sparse vector representation (document-term matrix).

Row is a document, column (feature) is a unique term.

N-Grams

- Bag-of-words model does not preserve sequence information, order of words in a sentence is lost.
- Solution: Build features using sequences of adjacent terms.
- Term Bigrams: Build features from every pair of adjacent terms.

President Barack Obama addresses US congress →

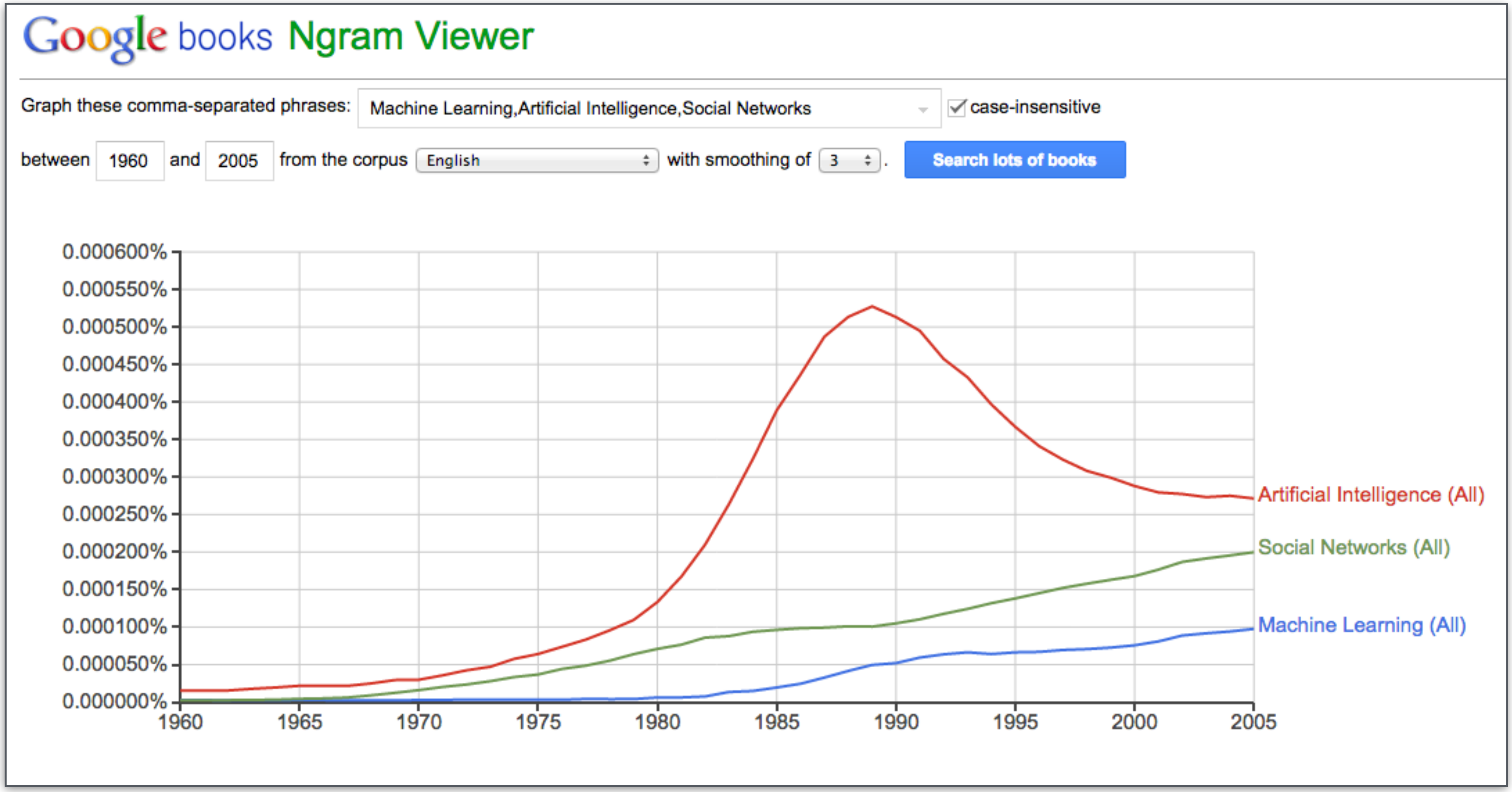
- President Barack
- Barack Obama
- Obama addresses
- addresses US
- US congress

Document can be represented as bag-of-words with 5 term bigram features

- Term *N*-grams: Build features from *N* adjacent terms.
- NB: This approach significantly increases the dimensionality of document vectors → makes document-term matrix more sparse.

N-Grams

- Google Ngram Viewer: Chart years counts of n-gram phrases in 5.2 million books between 1500-2008.



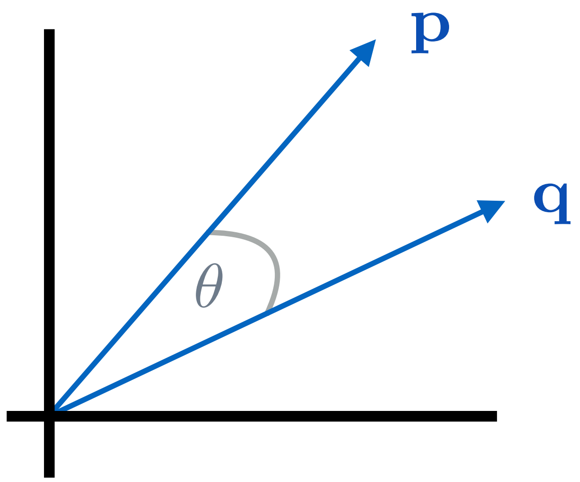
https://books.google.com/ngrams

Text Similarity

- Cosine similarity:
  Bag-of-words model produces highly sparse vectors, mostly containing 0s.

$$\begin{bmatrix} 0 & 1 & 0 & 0 & 0 & 0 & \cdots & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & \cdots & 0 & 1 & 0 \end{bmatrix}$$

- More appropriate to measure similarity based on cosine of the angle between the two vectors.



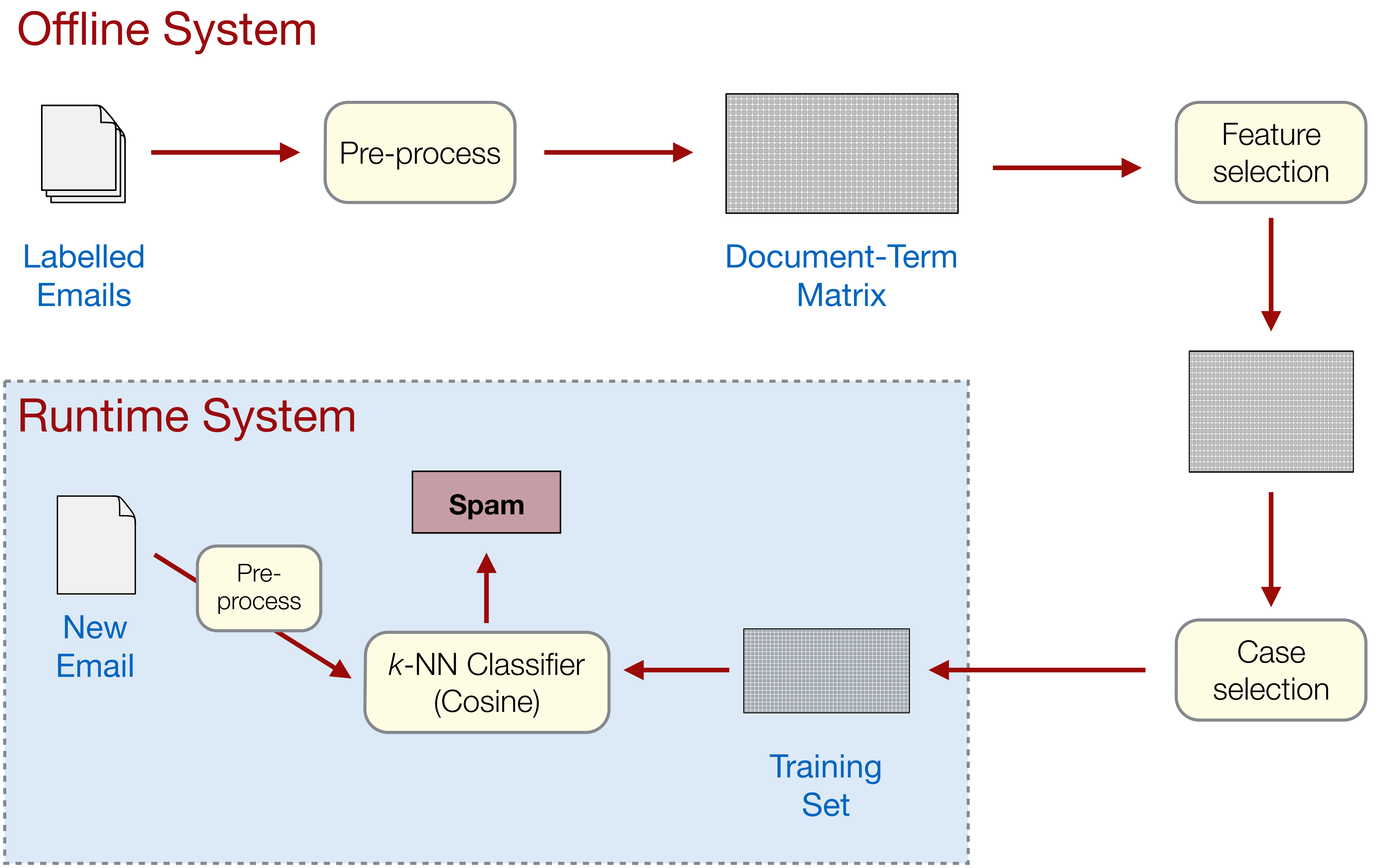
$$\cos(\mathbf{p}, \mathbf{q}) = \frac{\mathbf{p} \cdot \mathbf{q}}{||\mathbf{p}|| \, ||\mathbf{q}||}$$

1 = Same orientation
0 = At 90° to each other
-1 = Diametrically opposed

Convert to a distance metric to use with *k*-NN

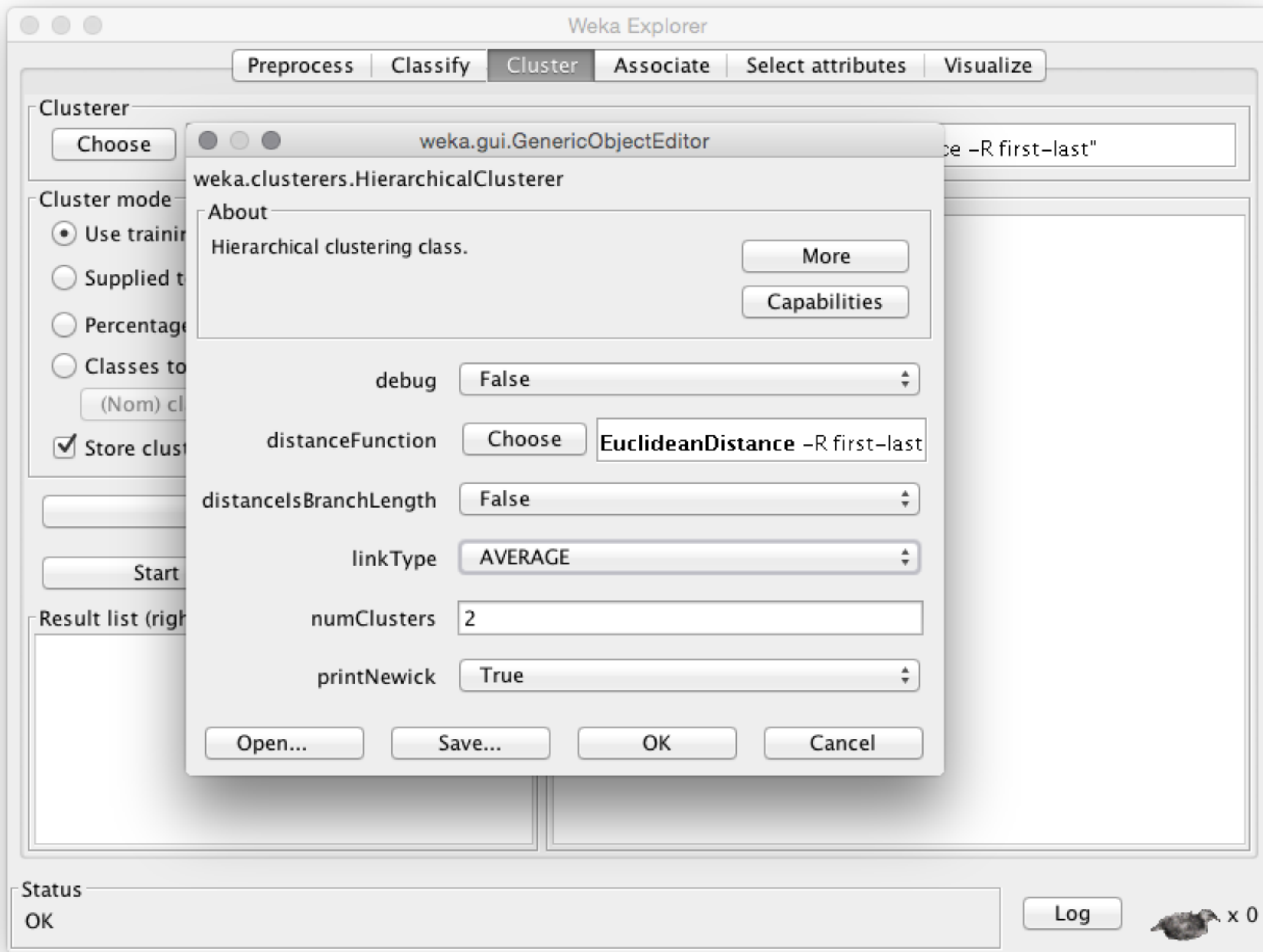
$$\mathrm{D}_{cos}(\mathbf{p}, \mathbf{q}) = 1 - \cos(\mathbf{p}, \mathbf{q})$$

Classifier System Design: Spam Filtering



k-NN in Weka

Install Java *Weka Toolkit (Version: Stable 3.6.12)*



http://www.cs.waikato.ac.nz/ml/weka

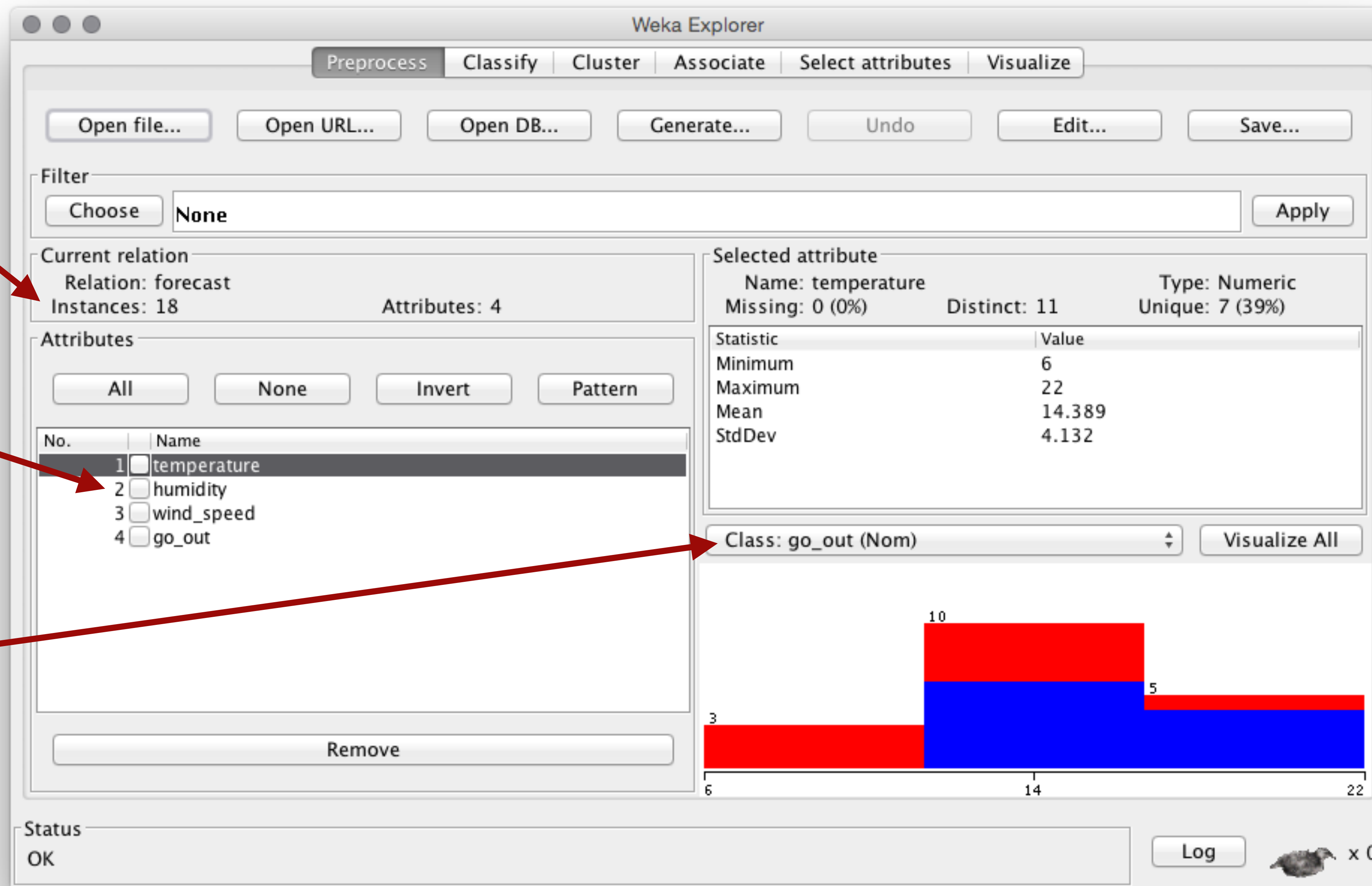
k-NN in Weka

1. Launch the WEKA application and click on the *Explorer* button.
2. Click *Open File* - e.g. forecast.arff (WEKA ARFF dataset format)

18 examples
("instances")

3 numeric features
("attributes")

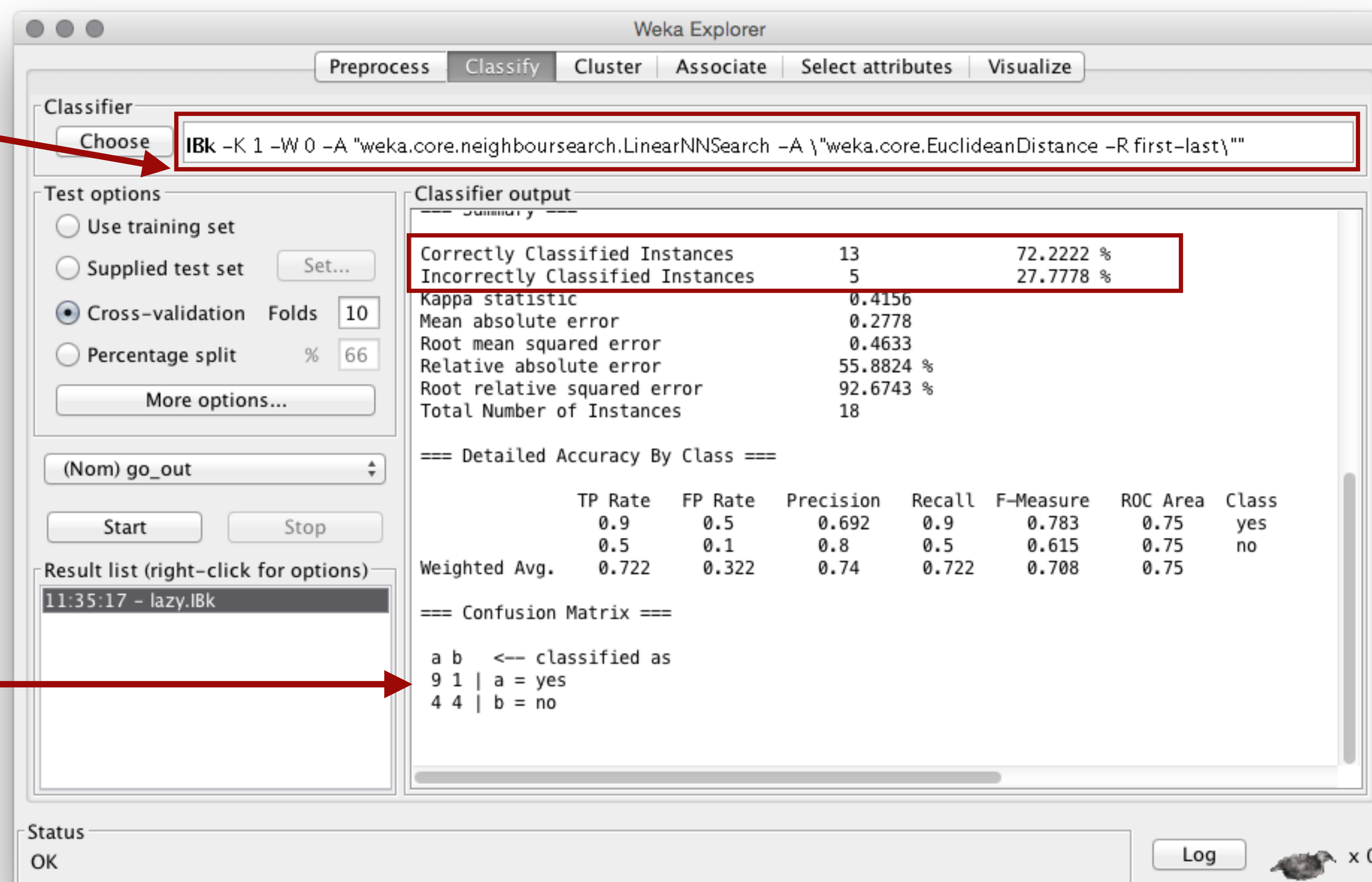
Class label
go_out={yes,no}



k-NN in Weka

3. In *Classify* tab, click *Choose* and find *Lazy→IBk* on the list.
4. Choose *(Nom) go_out* as class label from drop-down list.
5. Click *Start*.

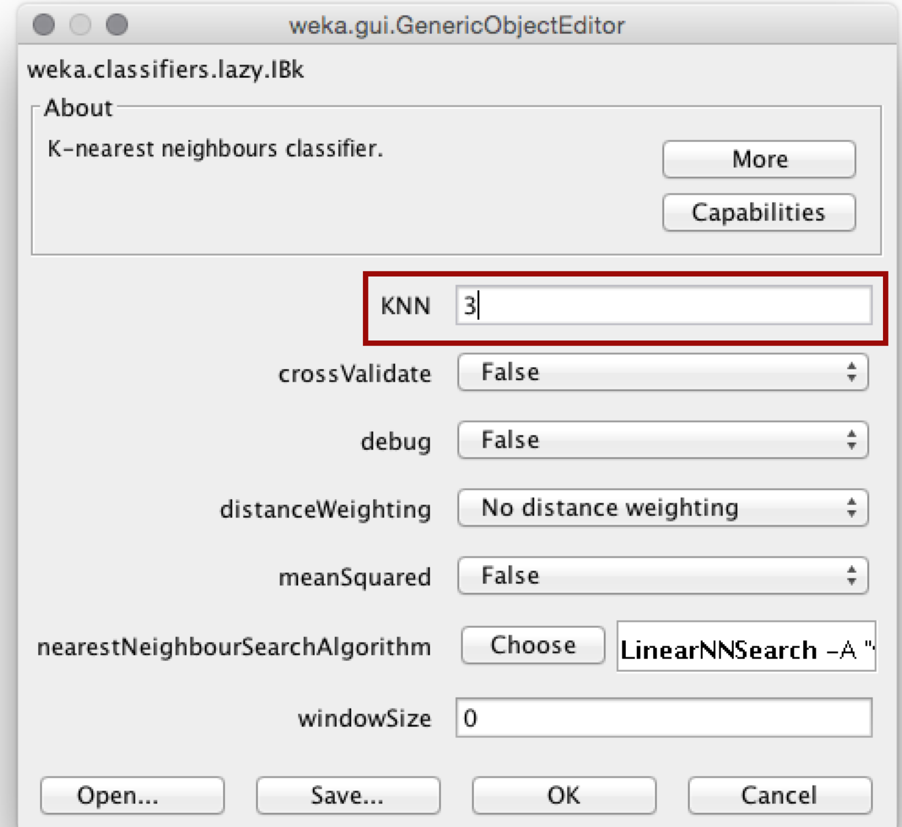
Parameter set:
By default
K=1 neighbours



Output of
classification
process

k-NN in Weka

- To change algorithm parameter values:
    1. Click the parameter set
    2. Enter new value for number of neighbours (KNN) - e.g 3
    3. Click *OK* and re-run process.




```
=== Summary ===

Correctly Classified Instances          15               83.3333 %
Incorrectly Classified Instances         3               16.6667 %
Kappa statistic                          0.6582
Mean absolute error                      0.2156
Root mean squared error                  0.3575
Relative absolute error                 43.3647 %
Root relative squared error             71.5158 %
Total Number of Instances               18

=== Detailed Accuracy By Class ===

               TP Rate   FP Rate   Precision   Recall  F-Measure   ROC Area  Class
                 0.9       0.25      0.818      0.9       0.857      0.9      yes
                 0.75      0.1       0.857      0.75      0.8        0.9      no
Weighted Avg.    0.833     0.183     0.835      0.833     0.832      0.9

=== Confusion Matrix ===

 a b   <-- classified as
 9 1 | a = yes
 2 6 | b = no
```

Summary

- Eager v Lazy Classification
- Similarity-based Learning
  - Feature spaces
  - Measuring similarity/distance
- The *k*-Nearest Neighbour Classifier
  - Lazy classifier based on majority voting
  - Requires an appropriate distance measure
- Improving *k*-NN Performance
  - Feature Selection + Condensed NN
- Classifying Text Documents
  - Bag-of-Words Model + Cosine similarity
- *k*-NN in Weka

COMP41450
Advanced Machine Learning

- 10 credit Level 4 module
- Extended version of COMP30120
  - ➡ Attend all COMP30120 lectures/tutorials
  - ➡ Complete COMP30120 assignments/tests for 5 credits
- Additional 5 credits:
  - ➡ 6 weeks additional lectures in Semester 1
  - ➡ Programming-based assignment (Java/Python/C)
  - ➡ In-class test

Starts 21st October

Wednesdays 2-3pm CS B109