# Classification

Adam Carter, EPCC, The University of Edinburgh
a.carter@epcc.ed.ac.uk

1 September 2020

www.archer2.ac.uk

archer2

epcc

# Reusing this material

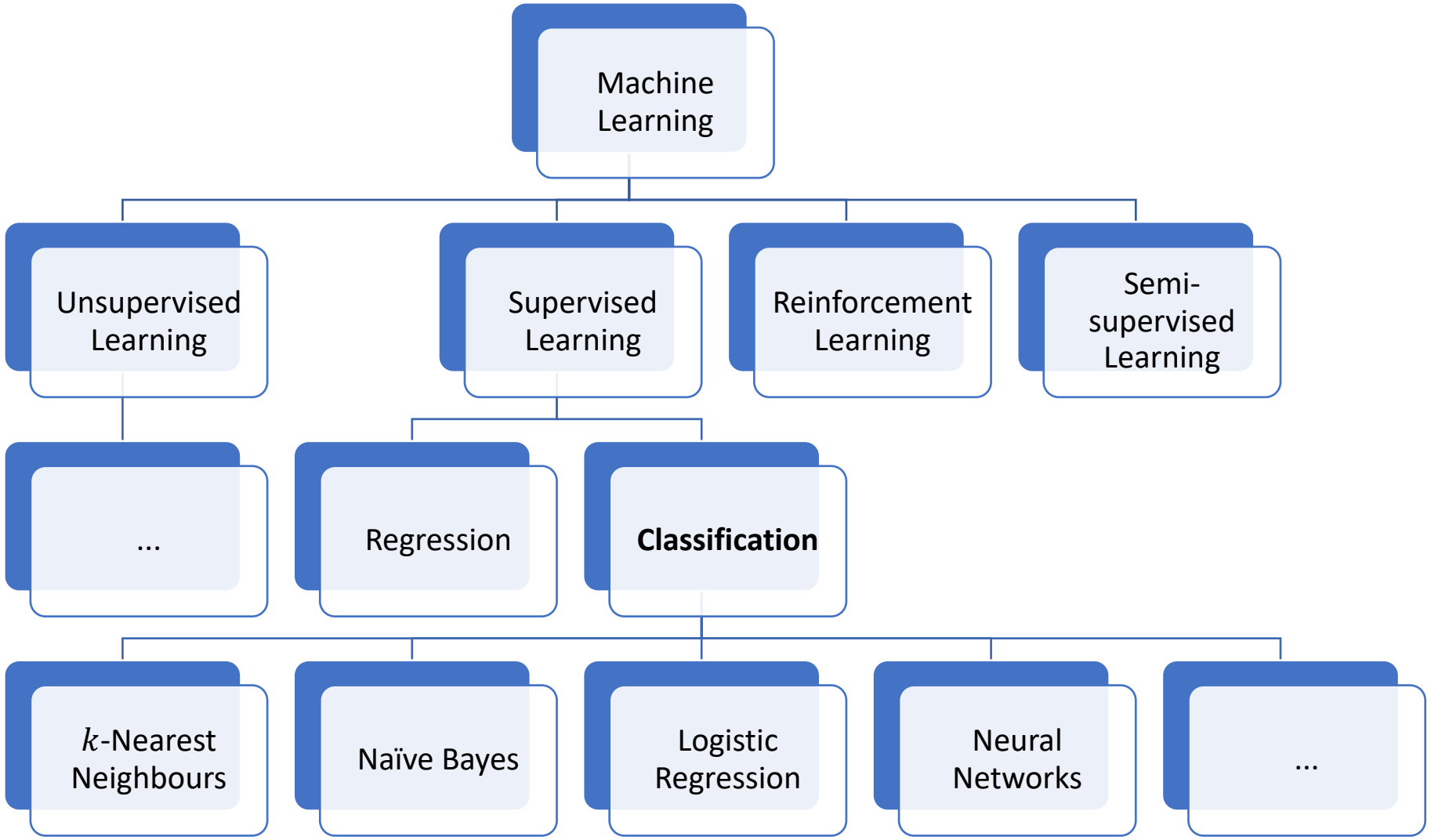
# Partners

epcc


epcc


THE UNIVERSITY of EDINBURGH

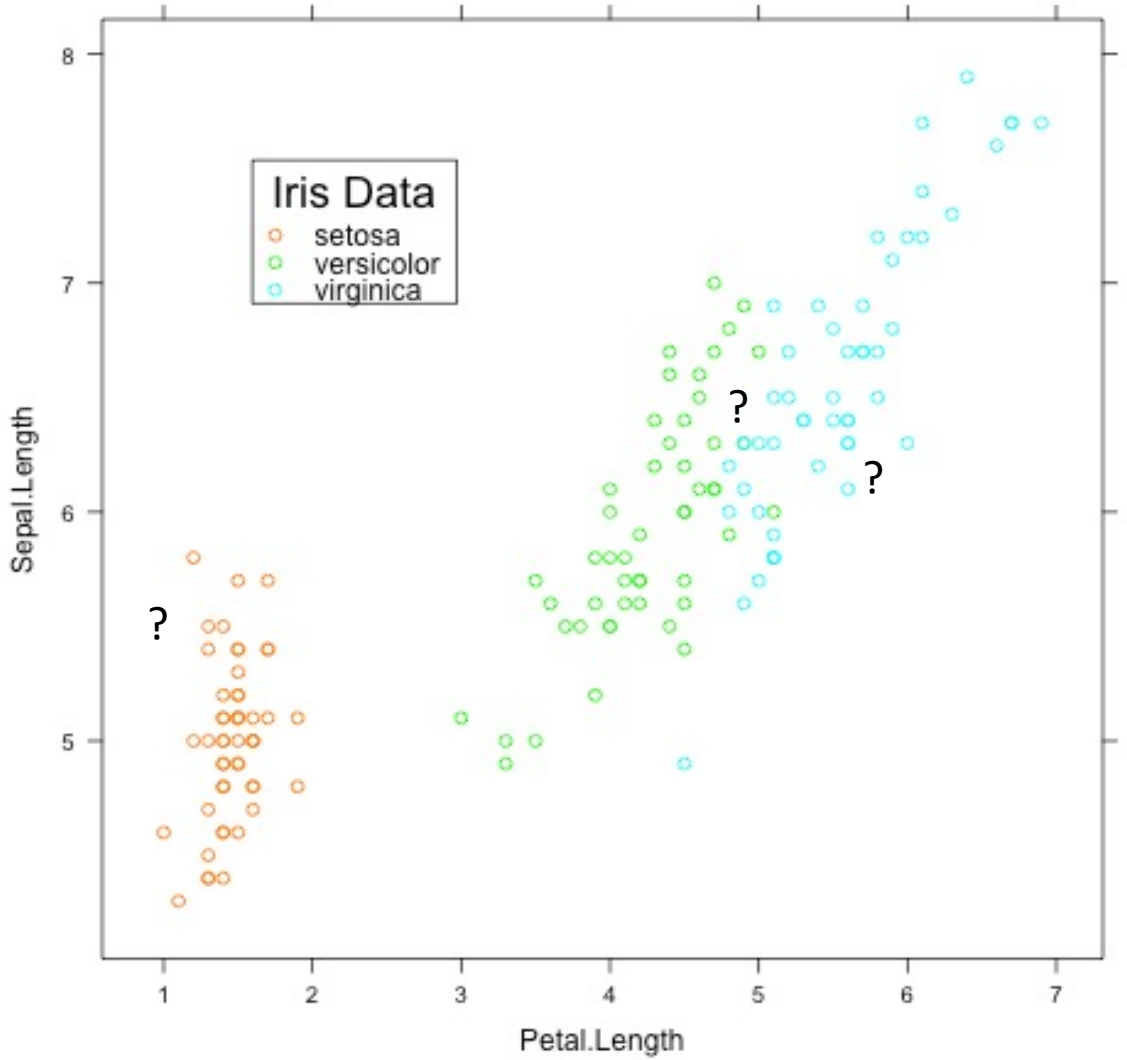
Hewlett Packard Enterprise


Machine Learning
Unsupervised Learning
Supervised Learning
Reinforcement Learning
Semi-supervised Learning
...
Regression
Classification
$k$-Nearest Neighbours
Naïve Bayes
Logistic Regression
Neural Networks
...

# Classification: problem overview

- Input is one or more features
  - numerical or categorical
- Output is a category (or class)

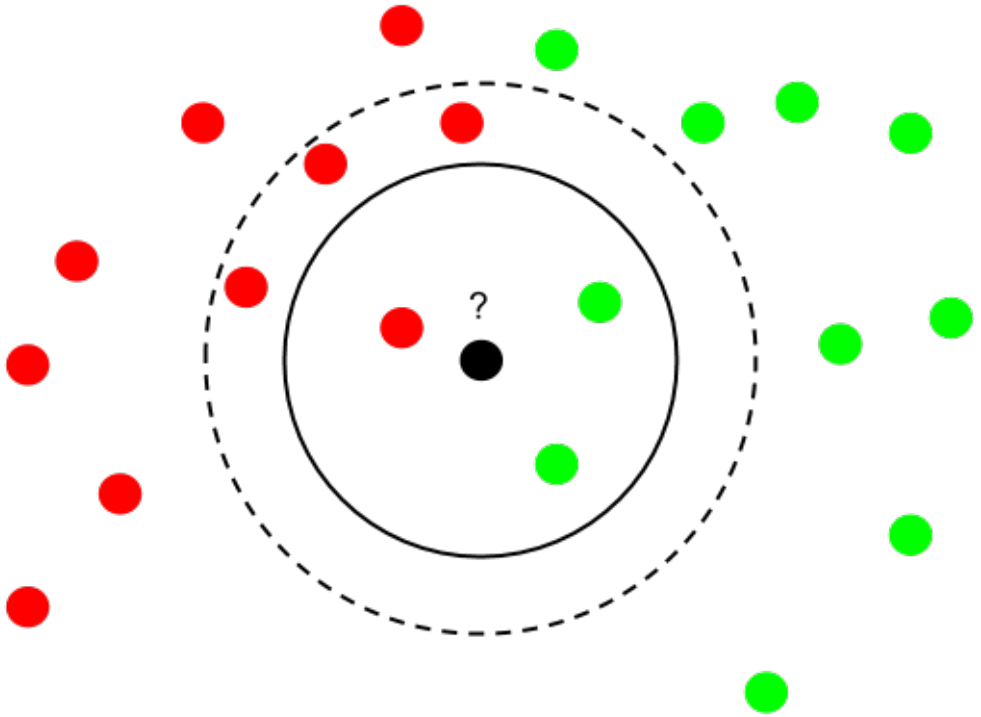
Iris image courtesy of David Berger under a CC-BY licence



# $k$-Nearest Neighbours Classification


epcc

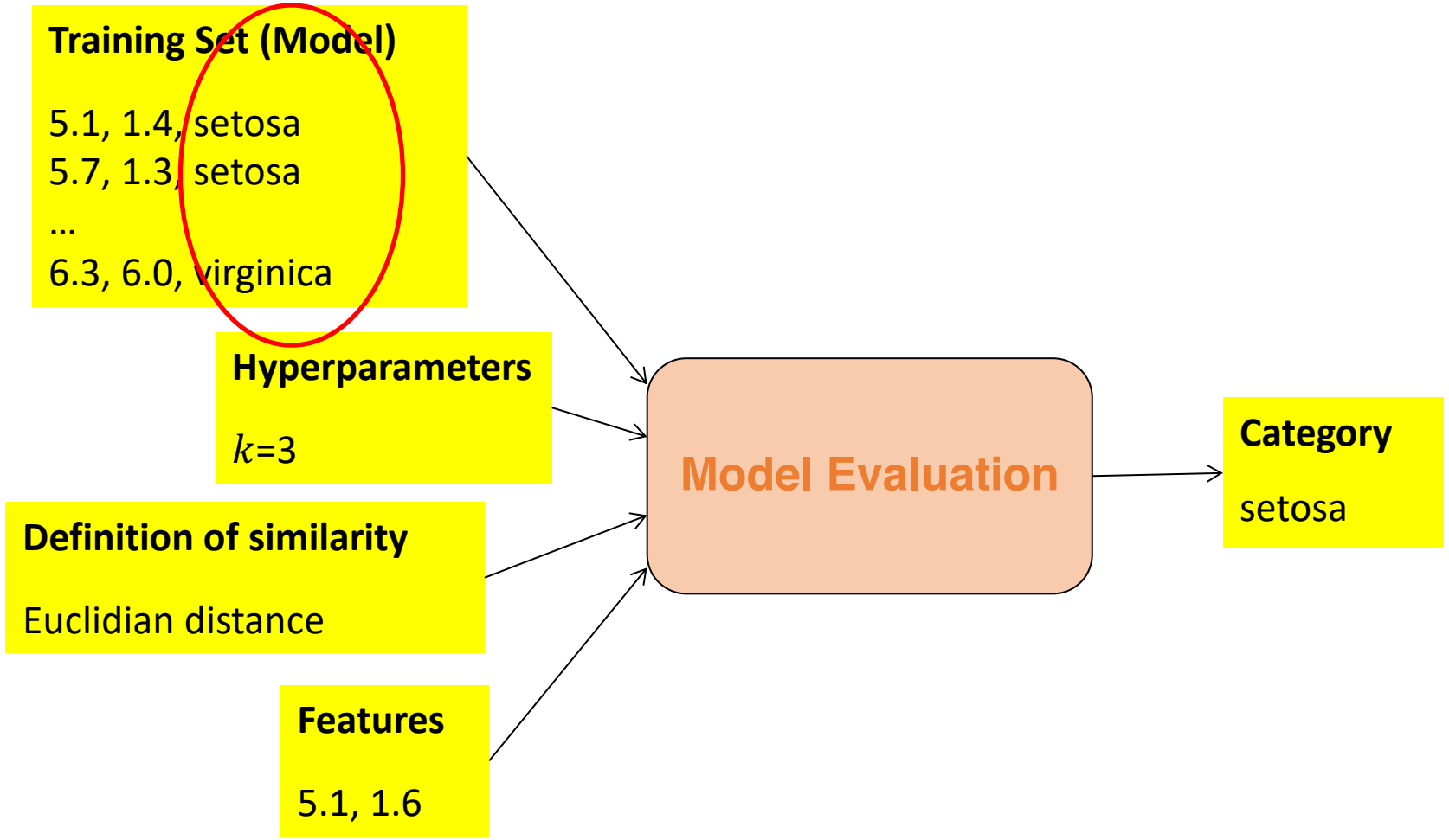
# $k$-Nearest Neighbours




Two decisions to make:

- How to define **similarity** or **closeness**
- How many neighbours ($k$) should vote

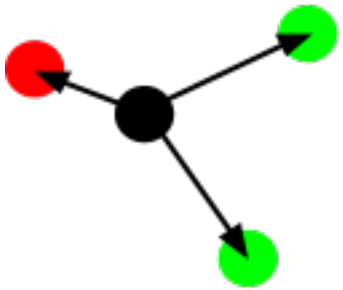
# $k$-Nearest Neighbours: supervised learning



# Euclidian Distance & Feature Scaling

- Euclidean distance works for real valued features

$$D(x, y) = \sqrt{\sum_{i=1}^{n} (x_i - y_i)^2}$$



- For this to be meaningful, it is normally necessary to **normalise** your data (also known as **feature scaling**)
  - Consider (age, salary) feature pair shown here
  - Euclidean distance measure is heavily biased towards salary
  - Solution: Normalise data first
    - $x_i \mapsto \frac{x_i - \bar{x}_i}{\sigma_{x_i}}$
  - It is also good to avoid highly correlated features as these give extra weight to that feature

| age | salary |
|---|---|
| 25 | 34,675 |
| 27 | 42,534 |
| 56 | 41,345 |

# Alternative Distance Functions 1

- Jaccard Similarity
  - How similar two sets are
  - A = {Sheep, Cow, Hen},  B = {Cow, Horse, Hen}

$$J(A,B) = \frac{A \cap B}{A \cup B}$$

- Cosine similarity
  - Measure of angle between vectors
  - 1 identical, -1 exactly opposite, 0 independent
  - Used in text analysis
  - Hence can used to classify documents, tweets etc.

    ("I love", "I hate", "delighted", "broken", "not", …)

    "I love my new tablet" -> (1,0,1,0,0,…)

    "I hate my broken tablet. Not working. Not happy" -> (0,1,0,1,2,..)

$$\cos(x,y) = \frac{x \cdot y}{\|x\|\|y\|}$$

# Alternative Distance Functions 2

- Mahalanobis Distance
  - Real valued vectors
  - Scale invariant
  - Takes correlation into account
  - No need to normalise (accounted for by co-variance matrix, $S$)

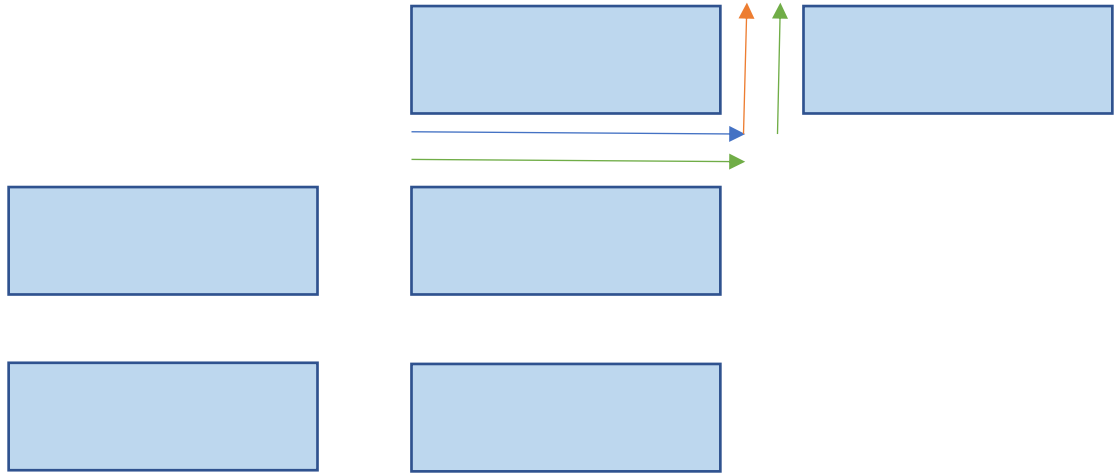
$$D(x, y) = \sqrt{(x - y)^T S^{-1} (x - y)}$$

- Hamming Distance
  - Distance between two sequences of same length
  - Simply count the differences at each point in the sequence
  - D("GGATC", "GAATT") = 2
  - Used in biology for simple matching of short sequences

# Alternative Distance Functions 3

- Manhattan Distance

$$D(x, y) = \sum_i |y_i - x_i|$$

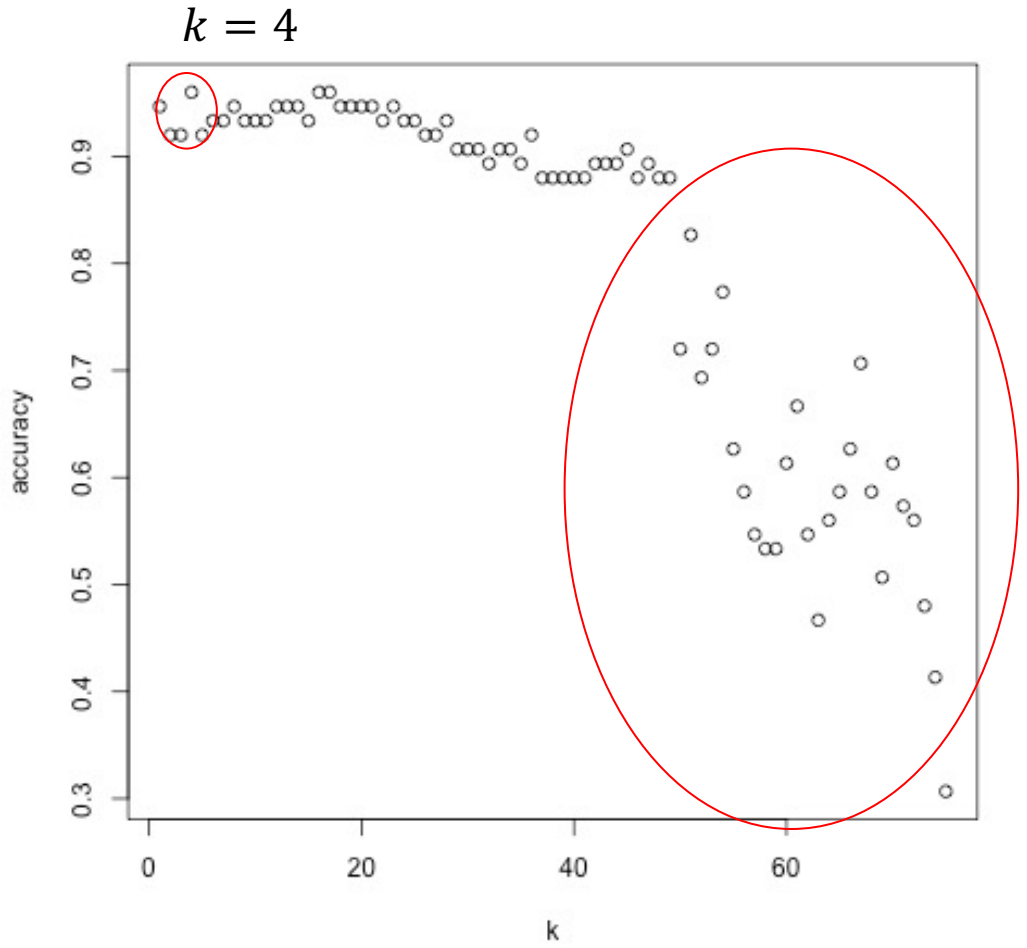


# Algorithm for choosing $k$

epcc

1. Choose similarity or distance metric
2. Divide data set into training set and test set (e.g., 80/20)
3. Choose **evaluation metric**
   - e.g. accuracy – fraction of correctly classified instances
4. Run $k$-NN several times with varying $k$
   - Use the *training* set as the model
   - Evaluate accuracy using the *test* set
5. Optimise $k$ by choosing the one that maximises the accuracy

# Choosing $k$




Model set is 75 instances, so accuracy falls away as $k$ approaches this.

# HPC Implications

- Various dimensions for trivial parallelisation
  - If evaluation set is small and model set is large parallelise on model set
  - If evaluation set is large maybe parallelise on evaluation set
  - Parallelise over candidate $k$ values
- Map Reduce pattern is a candidate
  - Distribute model set
  - Map: model instance -> key:distance value:category
  - Single Reducer: Find k lowest distances and count votes
  - Combiner would significantly reduce data transfer
    - Filter to just the $k$ smallest distances
- GPU
  - Will map onto GPU well if non-branching distance functions

# $k$-NN summary

epcc

- Classification – outputs a class
- Supervised learning
- Features need not be numeric so long a numeric distance measure can be defined
- Model
  - Large – full data set
  - Slow to apply
- Learning
  - Need to choose or learn $k$
  - Normalisation may be required – one time transform of all data

# Evaluating a Classifier

# Training and test set

- Bad practice to evaluate your model on the same data on which you trained it
- Split data into **training set** and **test set**
  - Using 20%-25% of the data as test set is often a sensible default
  - Train using training set
  - Evaluate with test set
- Evaluation will give you a confusion matrix

| | | Gold Standard | | |
|---|---|---|---|---|
| | | A | B | C |
| Test Outcome | A | 324 | 70 | 8 |
| | B | 43 | 939 | 34 |
| | C | 6 | 90 | 104 |

# Evaluation metrics

- Accuracy: fraction of correctly classified cases
- Weighted metric to reflect costs of different errors

| | | Gold Standard | | |
|---|---|---|---|---|
| | | A | B | C |
| Test Outcome | A | 0.8 | 0.4 | -2.0 |
| | B | 0.4 | 0.8 | -2.0 |
| | C | -1.0 | -1.0 | 1.0 |

- For binary outcomes define TP, FP, FN and TN

| | | Gold Standard | |
|---|---|---|---|
| | | True | False |
| Test Outcome | True | True Positive | False Positive |
| | False | False Negative | True Negative |

# Binary classification statistics

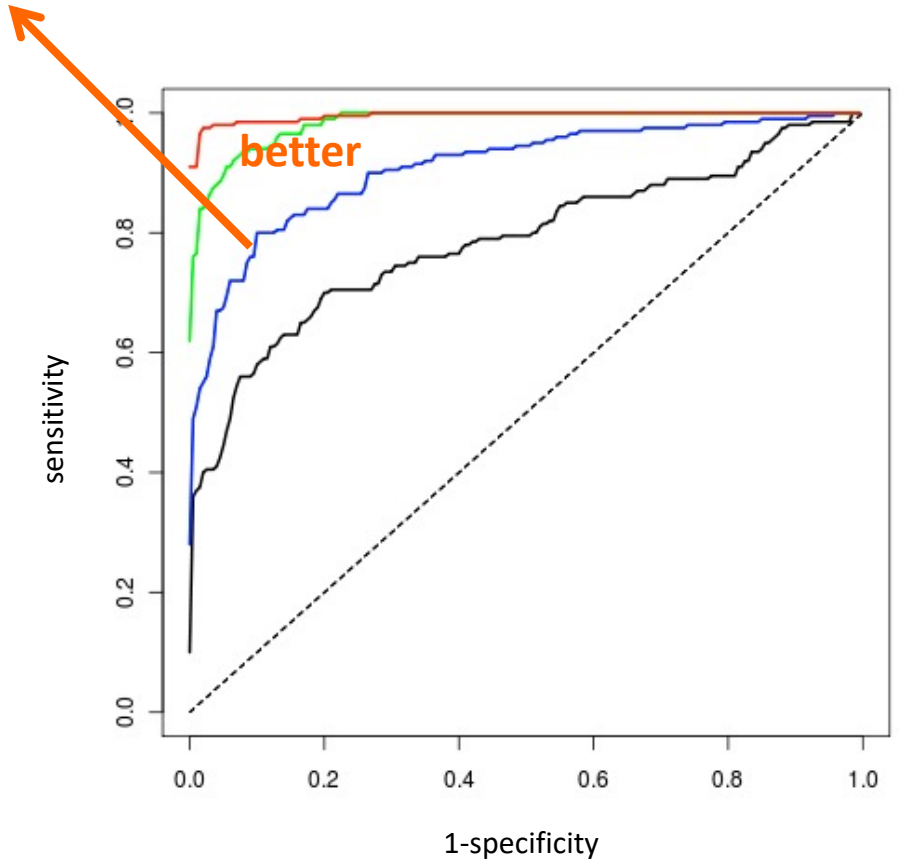
- Accuracy
  - (TP+TN)/(P+N) = (17+55)/100 = 0.72
- Sensitivity (recall, true positive rate)
  - TP/P = TP/(TP+FN) = 17/20 = 0.85
- Specificity (true negative rate)
  - TN/N = TN/(FP+TN) = 55/80 = 0.69
- Precision (positive predictive value)
  - TP/(TP+FP) = 17/(17+25) =17/42 = 0.40
- Negative predictive value
  - TN/(TN+FN) = 55/58 = 0.94
- F-score
  - (2 x precision x recall)/(precision + recall) = 0.548

| | | Gold Standard | |
|---|---|---|---|
| | | True | False |
| Test | True | TP | FP |
| | False | FN | TN |

| | True | False | Total |
|---|---|---|---|
| **True** | 17 | 25 | 42 |
| **False** | 3 | 55 | 58 |
| **Total** | 20 | 80 | 100 |

# ROC curve

- Receiver operating characteristic (ROC) curve
  - Sensitivity against 1-specificity
- Point on curve dependent on tunable parameter in algorithm
  - *i.e.* set thresholds for classifying positive classes
- Would choose to run a system at the appropriate point on graph
- (*N.B.* 1-specificity = false positive rate = FP / FP + TN)




ROC image courtesy of Carl Boettiger under a CC-BY licence.

# AUC (Area under [RO]Curve)

- Plotting a ROC requires calculating many TP vs FP results at different thresholds
- To determine suitable parameter value we can use AUC, an efficient sorting-based algorithm (use, for e.g, sklearn)
- Provides aggregate measure of performance across all classification thresholds
  - Probability of ranking a random positive example more highly than a random negative result
- AUC range from 0 to 1
  - Model with 100% predictions wrong = 0
  - Model with 100% predictions correct = 1

# Acknowledgements


- This lecture was compiled by Adam Carter (EPCC, The University of Edinburgh) from a lecture written by Ally Hume (EPCC) for EPCC's MSc in High Performance Computing with Data Science