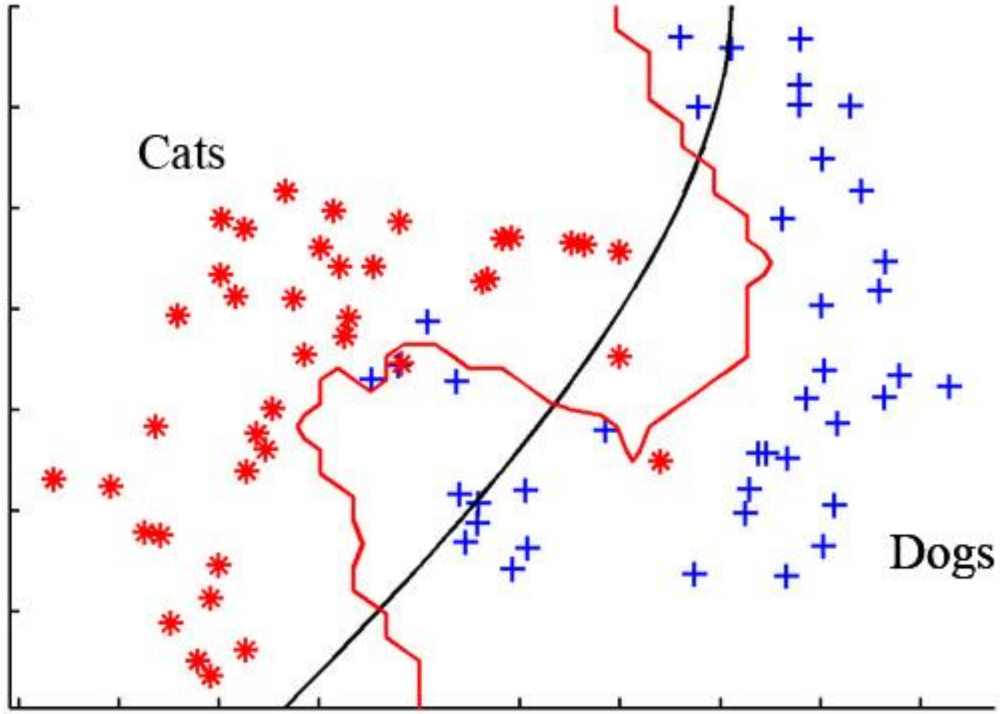
Classification – An introduction



Classification

- In this lecture and lab:
  - Learn what a classifier is
  - Learn about two well-known classifiers
  - How they model the data
  - Their strengths and weaknesses
  - How to build your own classifiers in R

What is Classification?

- In lecture 2 we saw some algorithms for learning “labels” for data based on how similar items are
- Classification attempts to solve the problem:
  - Given some data
  - Where each case in the data has been allocated a class
  - Assign a class to a new unassigned case
- This is Supervised Learning

http://arogozhnikov.github.io/2016/04/28/demonstrations-for-ml-courses.html

Brunel University London

Classification

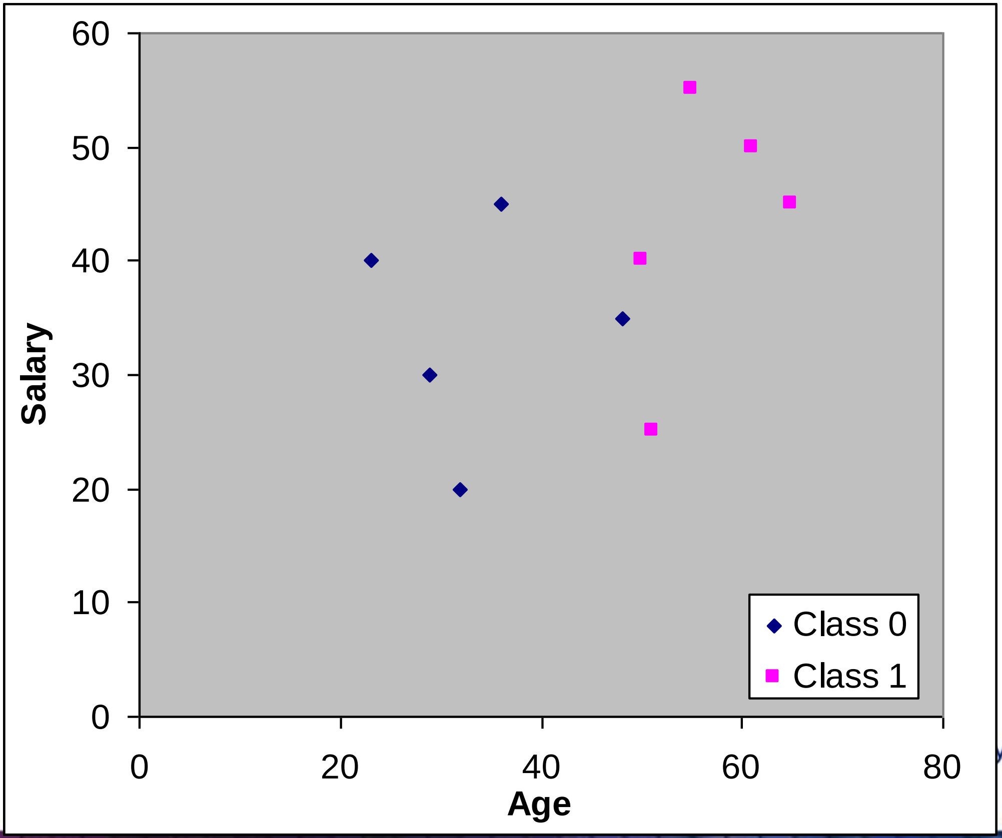
- Example:

Case	Age	Salary	Class
1	50	40	1
2	32	20	0
3	36	45	0
4	55	55	1
5	61	50	1
6	29	30	0
7	48	35	0
8	65	45	1
9	23	40	0
10	51	25	1

Brunel University London

Classification

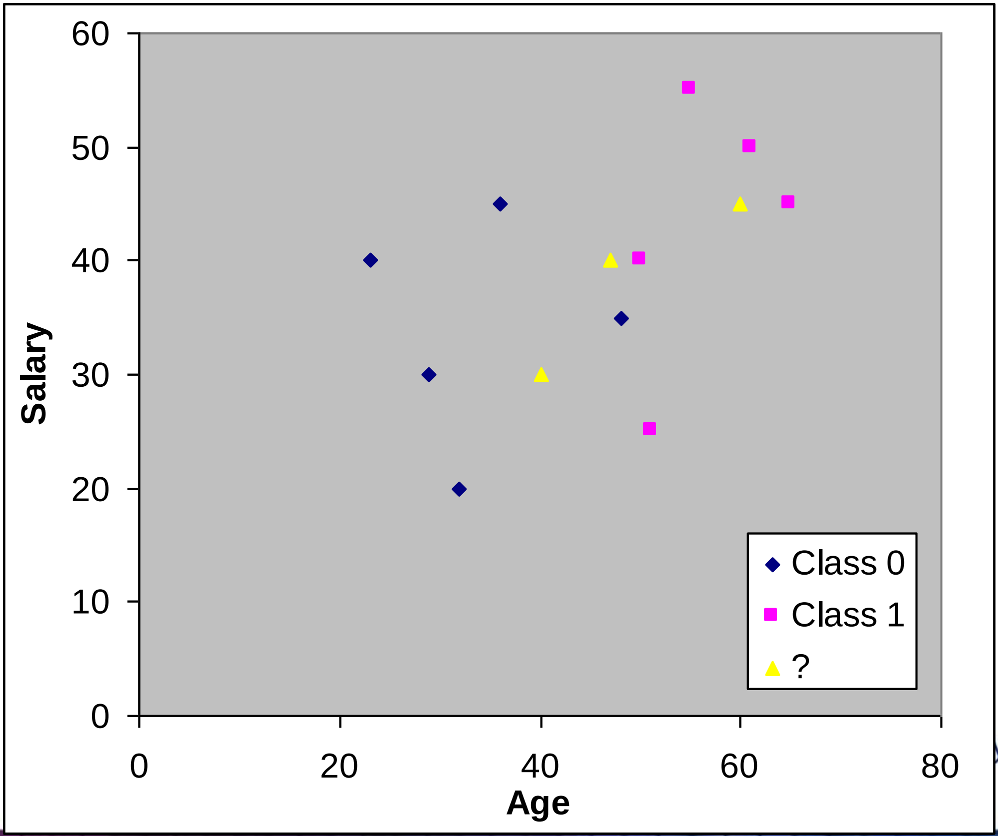
- Easy to visualise with only 2 variables
- 2D scatterplot



Classification

- New data:

Case	Age	Salary	Class
11	60	45	?
12	40	30	?
13	47	40	?

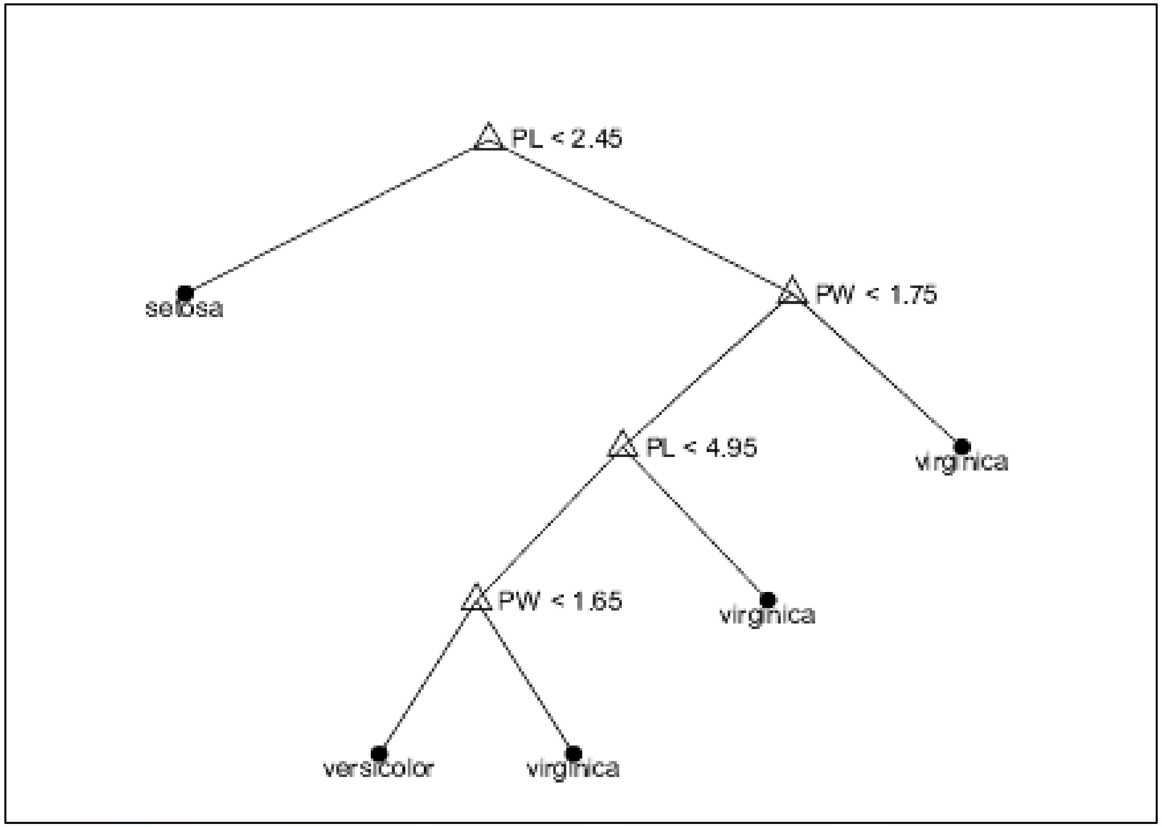


Decision Trees

- Very popular
- *Nodes* represent decisions
- *Arcs* represent possible answers
- *Terminal nodes* represent classification

Decision Trees

- Example from Iris data:

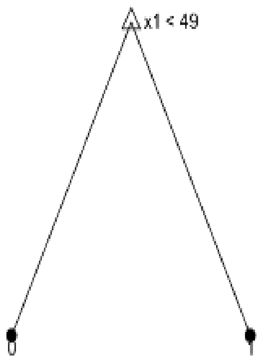


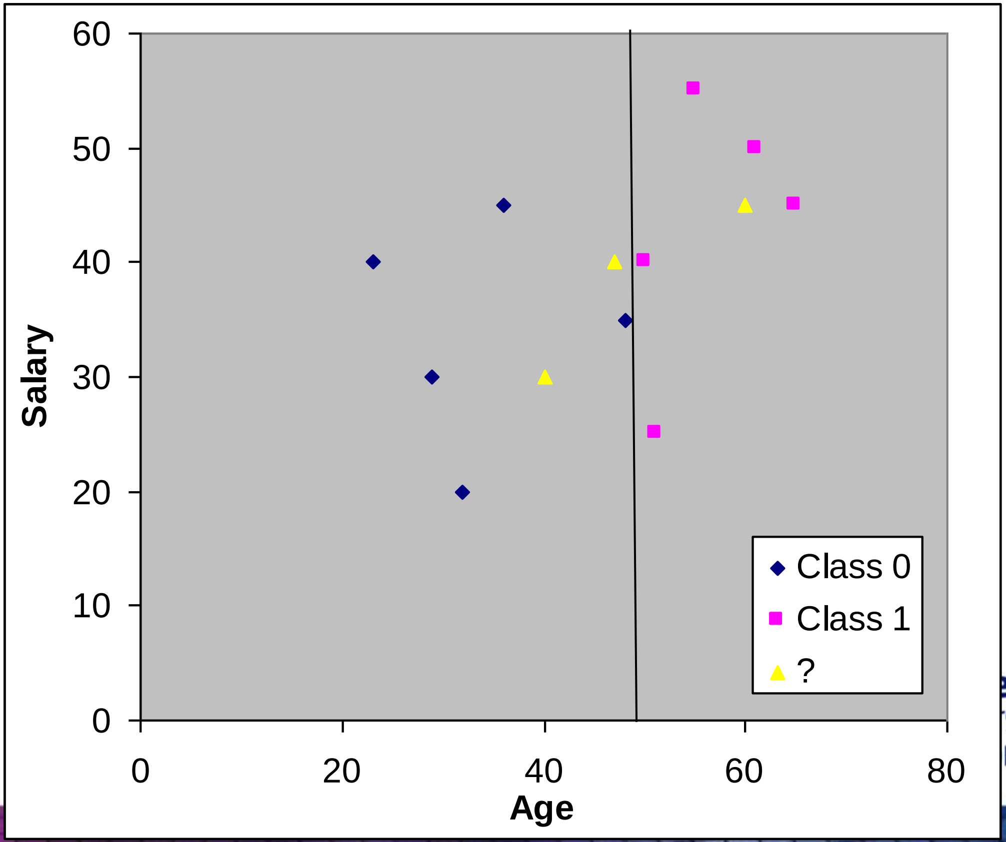
Classifying with a Decision Tree

- Traverse tree starting at the root node
- At each decision node follow the appropriate branch according to the case that you are classifying
- Repeat until you reach a leaf node
- Classify according to the label at the leaf node

Building a Decision Tree

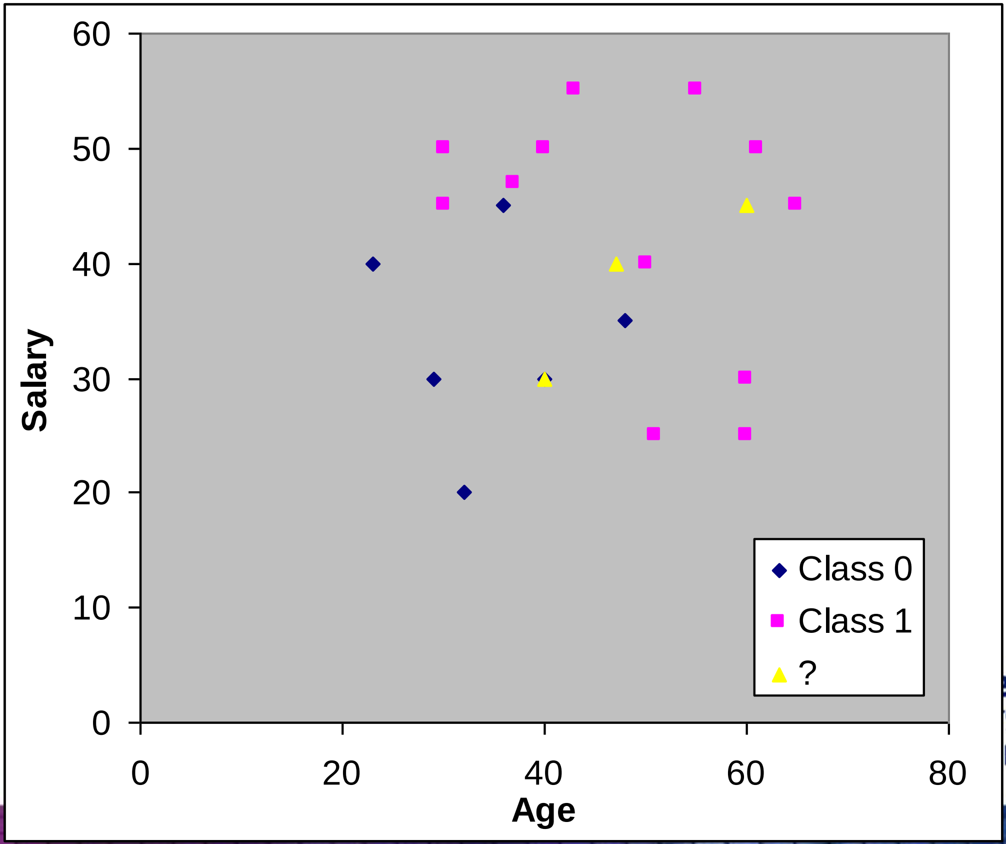
- Now let's use the example data we used before:





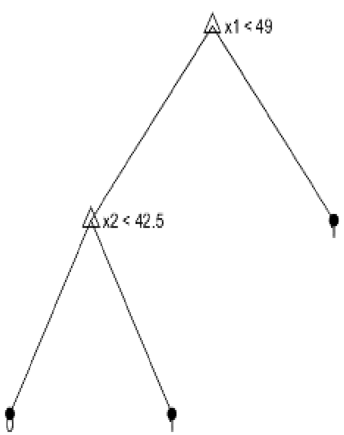
Building a Decision Tree

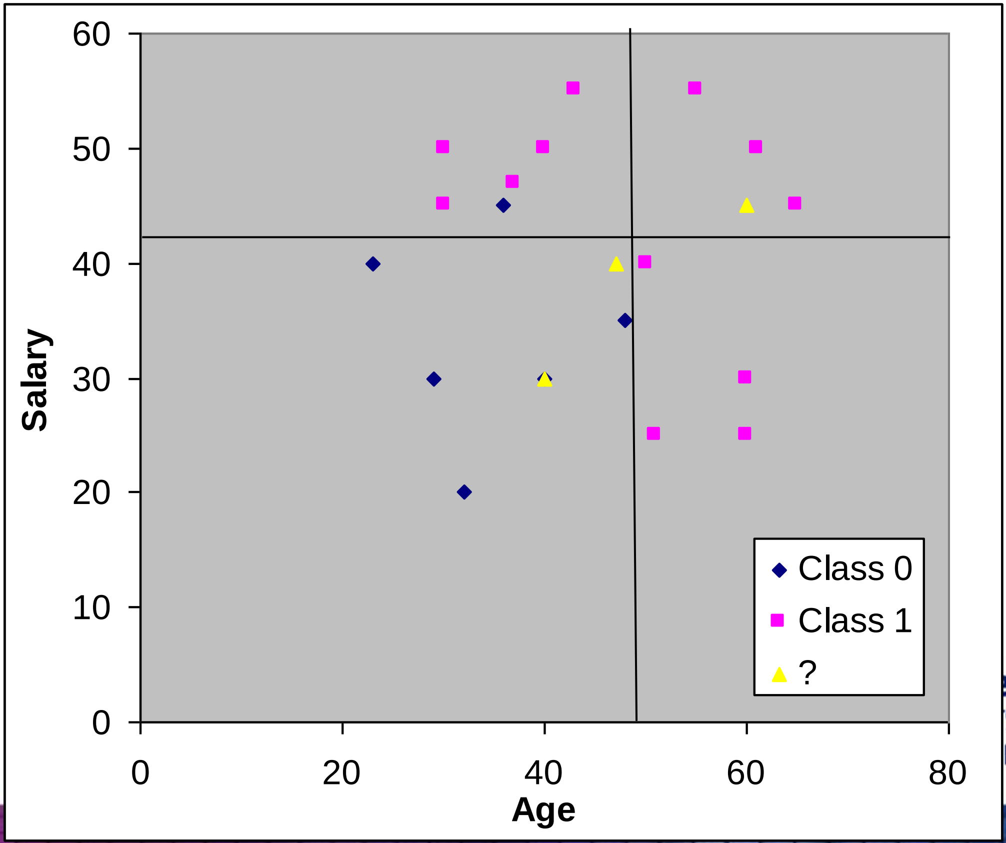
- What happens if we add new data?:



Building a Decision Tree

- What happens if we add new data?:





http://arogozhnikov.github.io/2016/04/28/demonstrations-for-ml-courses.html

Building a Decision Tree

- Often known as *rule induction*
- Nodes are repeatedly *split* until all elements represented belong to one class
- Nodes then become terminal nodes
- Deciding which nodes to split next as well as the evaluation function used to split it depend on the algorithm

Brunel University London

ID3 Algorithm

- ID3(*examples*, *target_classes*)

```
Create root node, root
If all examples are pos. then return single node with label +
If all examples are neg. then return single node with label –
Otherwise
    a   = attribute that best classifies examples
    Set the decision attribute for root to a
    For all values of a, v_i
      add a new node corresponding to a = v_i
      Set examples_vi = all cases in examples where a = v_i
      If examples_vi is empty Then add a leaf node labelled with
      most common value of target in examples
      Else Call ID3(examples_vi , target_classes)
    End For
End
Return root
```

ID3 Algorithm

- ID3(*examples*, *target_classes*)

```
Create root node, root
If all examples are pos. then return single node with label +
If all examples are neg. then return single node with label –
Otherwise
    a   = attribute that best classifies examples
    Set the decision attribute for root to a
    For all values of a, v_i
      add a new node corresponding to a = v_i
      Set examples_vi = all cases in examples where a = v_i
      If examples_vi is empty Then add a leaf node labelled with
      most common value of target in examples
      Else Call ID3(examples_vi , target_classes)
    End For
End
Return root
```

ID3 Algorithm

- ID3(*examples*, *target_classes*)

```
Create root node, root
If all examples are pos. then return single node with label +
If all examples are neg. then return single node with label –
Otherwise
    a   = attribute that best classifies examples
    Set the decision attribute for root to a
    For all values of a, v_i
      add a new node corresponding to a = v_i
      Set examples_vi = all cases in examples where a = v_i
      If examples_vi is empty Then add a leaf node labelled with
      most common value of target in examples
      Else Call ID3(examples_vi , target_classes)
    End For
End
Return root
```

Building a Decision Tree

- How do we decide which attribute best classifies the examples?
- A popular function is known as *entropy*:

$$-\sum_j p_j \log p_j$$

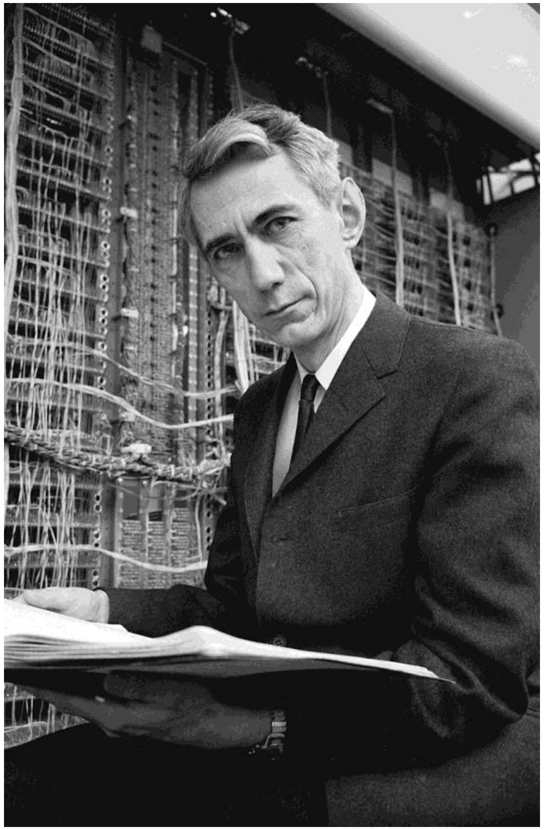
  - Where j is a class and p_j is the proportion of cases that take on the class j
- ID3 uses *information gain* which is the expected reduction in entropy

Brunel University London

Entropy

$$H = -\sum p_i \log_2 p_i$$

where p_i = probability of each message.

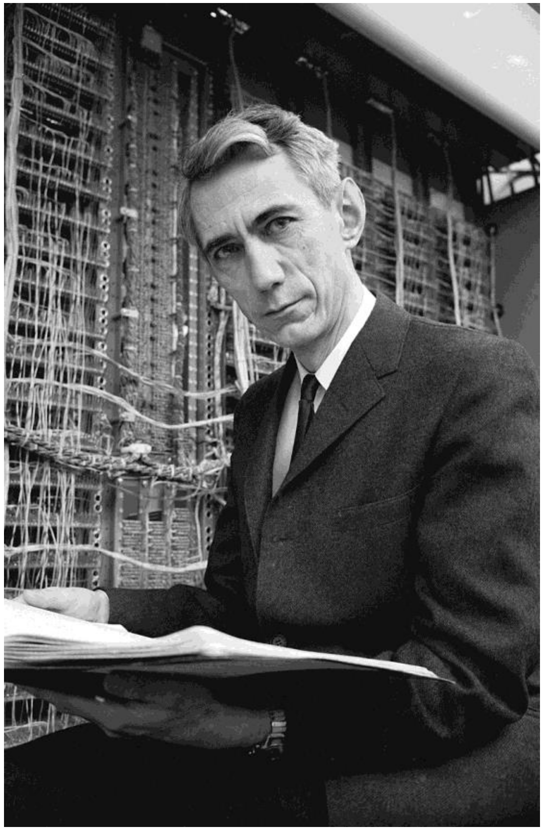


Brunel University London

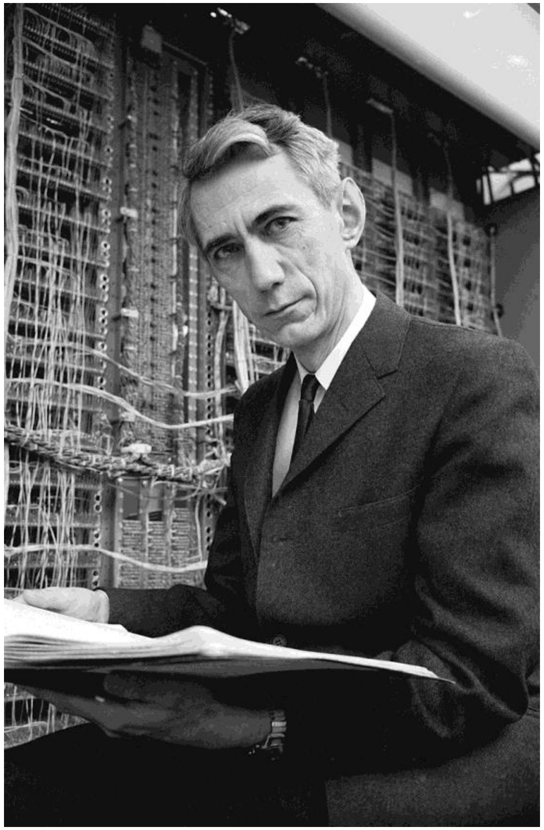
Entropy

$$H = -\sum p_i \log_2 p_i$$

where p_i = probability of each message.



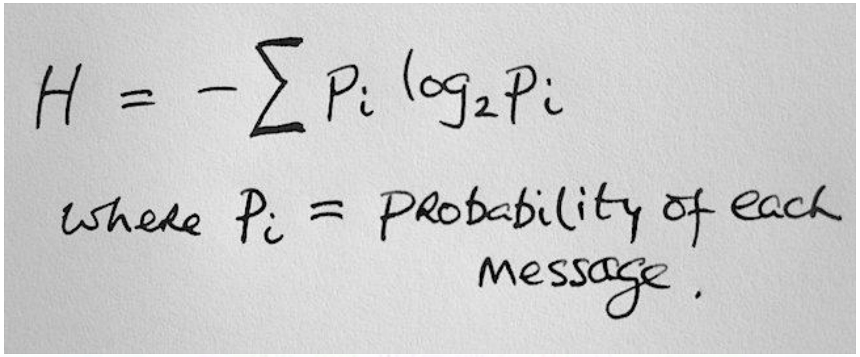
Entropy

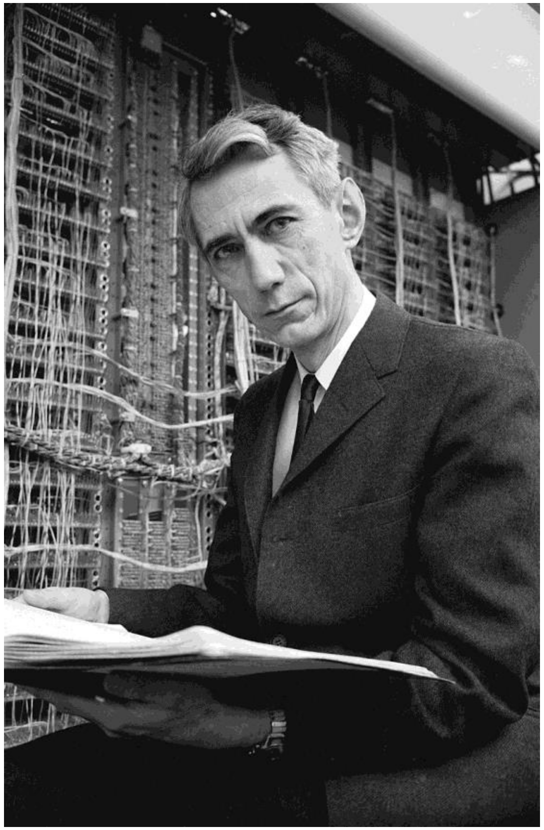


$$H = -\sum p_i \log_2 p_i$$

where p_i = probability of each message.

Entropy

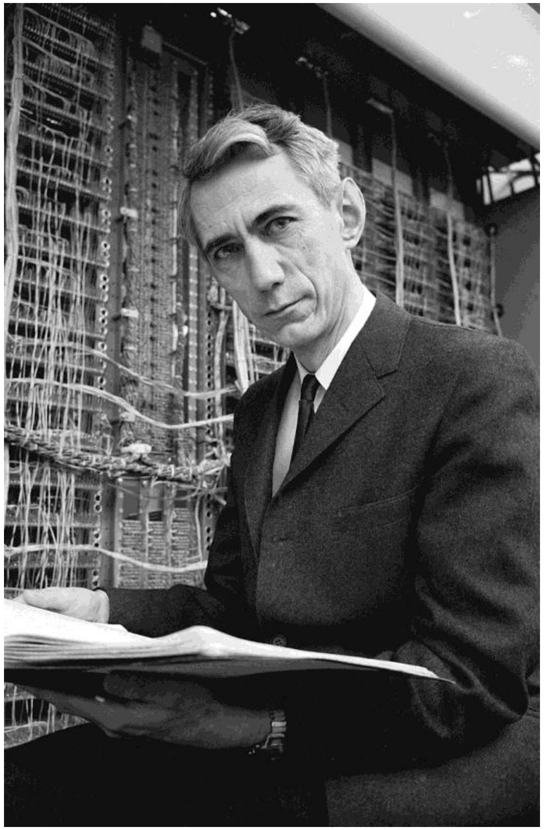




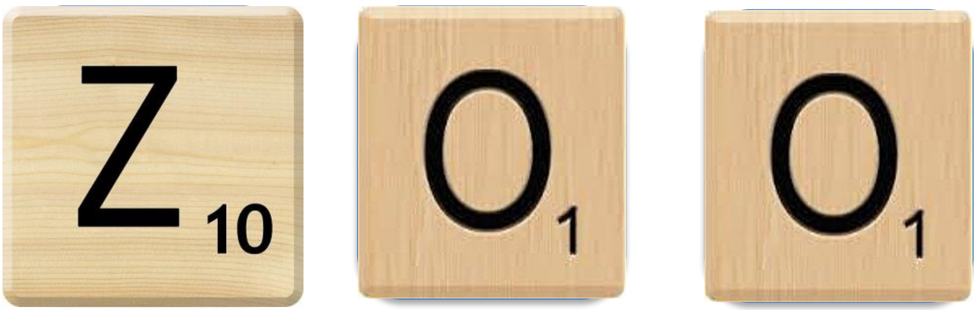
Entropy

$$H = -\sum p_i \log_2 p_i$$

where p_i = probability of each message.

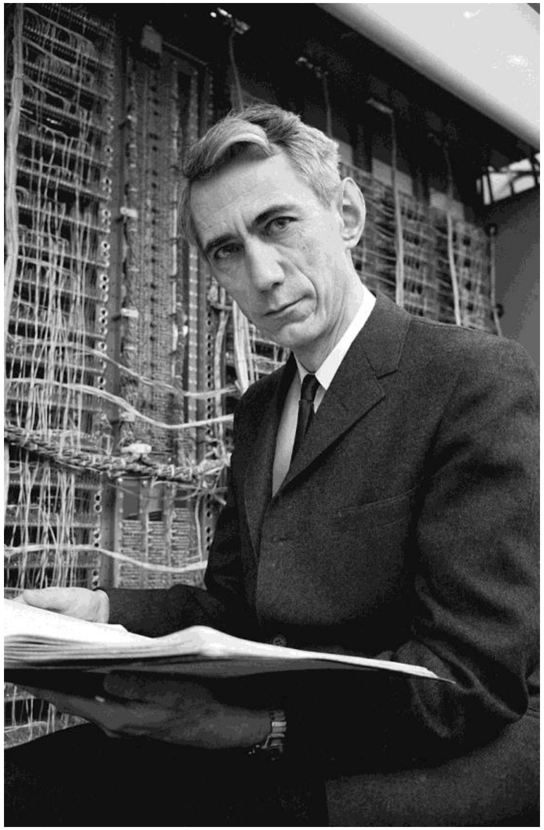


Entropy



$$H = -\sum p_i \log_2 p_i$$

where p_i = probability of each message.



Pruning Decision Trees

- Simpler models are preferred
- Known as “Occam’s razor”
- Prevents overfitting (more on this later)
- Tree pruning can be performed
  - Goes through each decision node
  - Considers converting it into a leaf node
  - If this does not reduce classification accuracy then the pruning is carried out


akinator

K-Nearest Neighbour

- Case based reasoning
- Based on the idea that items that are located "nearby" in the data space will influence how unknown data is classified
- Been around since 1910
- Requires:
  - Distance Metric
  - *k* parameter (no. of neighbours)
  - Weighting function
  - How to combine the info from neighbours

K-Nearest Neighbour

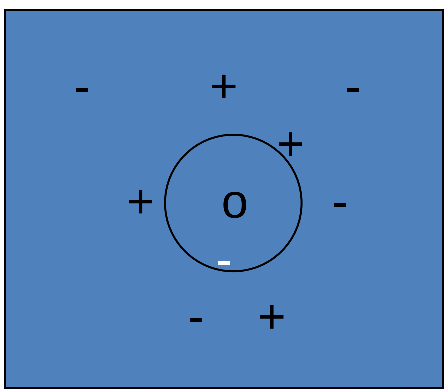
- Distance metrics
  - Euclidian (as we saw in the last lecture)
  - Correlation
  - Mahalanobis

Brunel University London

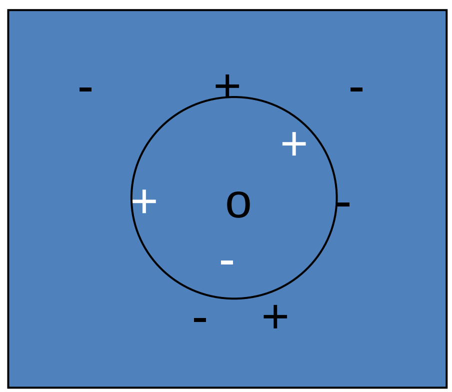
K-Nearest Neighbour

- Example
  - Metric = Euclidian
  - No weighting function
  - Maximum vote of neighbours

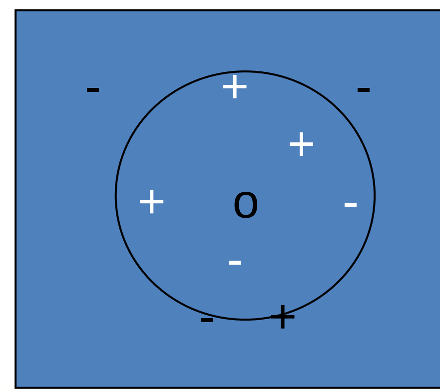
k=1



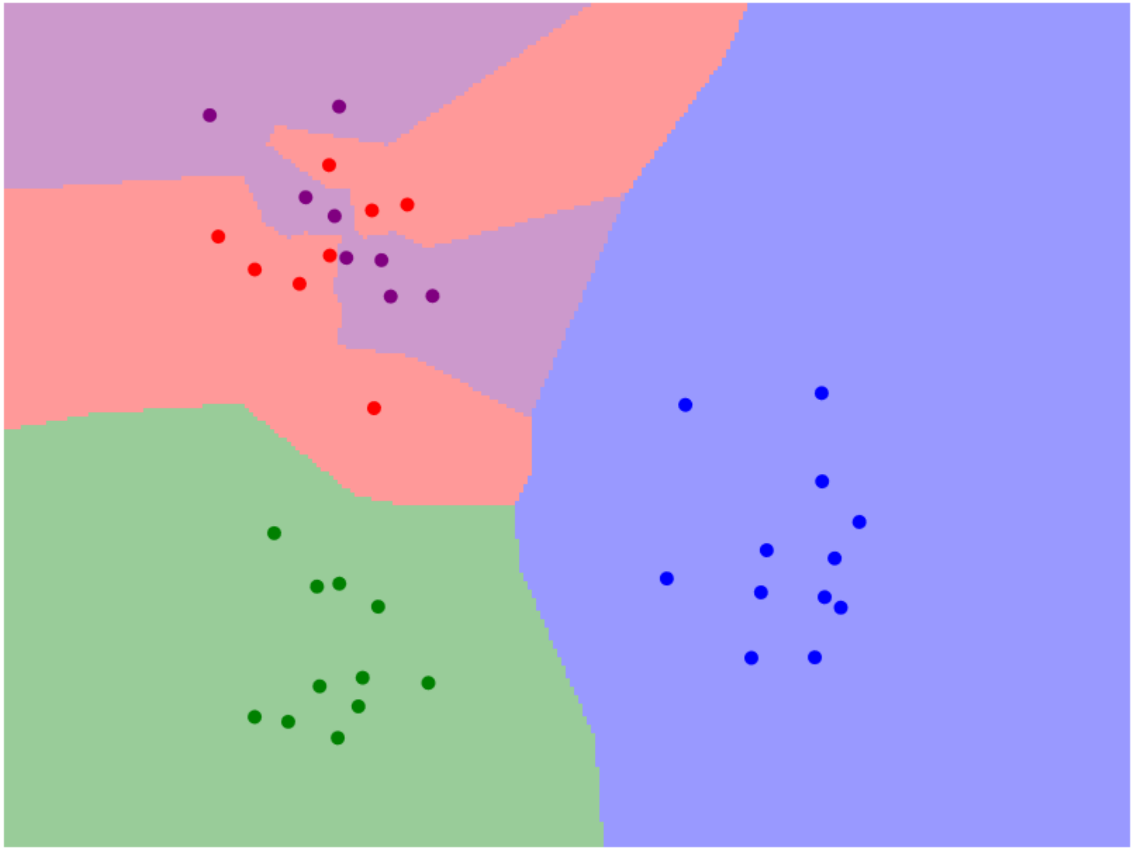
k=3



k=5



KNN Demo



http://vision.stanford.edu/teaching/cs231n-demos/knn/

Brunel University London

Testing Performance

- Aim is to classify new unseen data
- Often look at a simple error rate to assess our classifiers

error = number of errors / number of cases

Testing Performance

- Note *empirical* error rate is not the same as *true* error rate
- Empirical is based on a sample
- True is based on infinite cases
- Is it possible to estimate the true error rate?

Brunel University London

Training and Test Sets

- Obviously a good idea to split data into a *training* set and a *test* set
- Known as the *holdout* method
- Use training set to learn model
- Use test set to score the accuracy
- Ideally the two sets are independent (generated separately)

Brunel University London

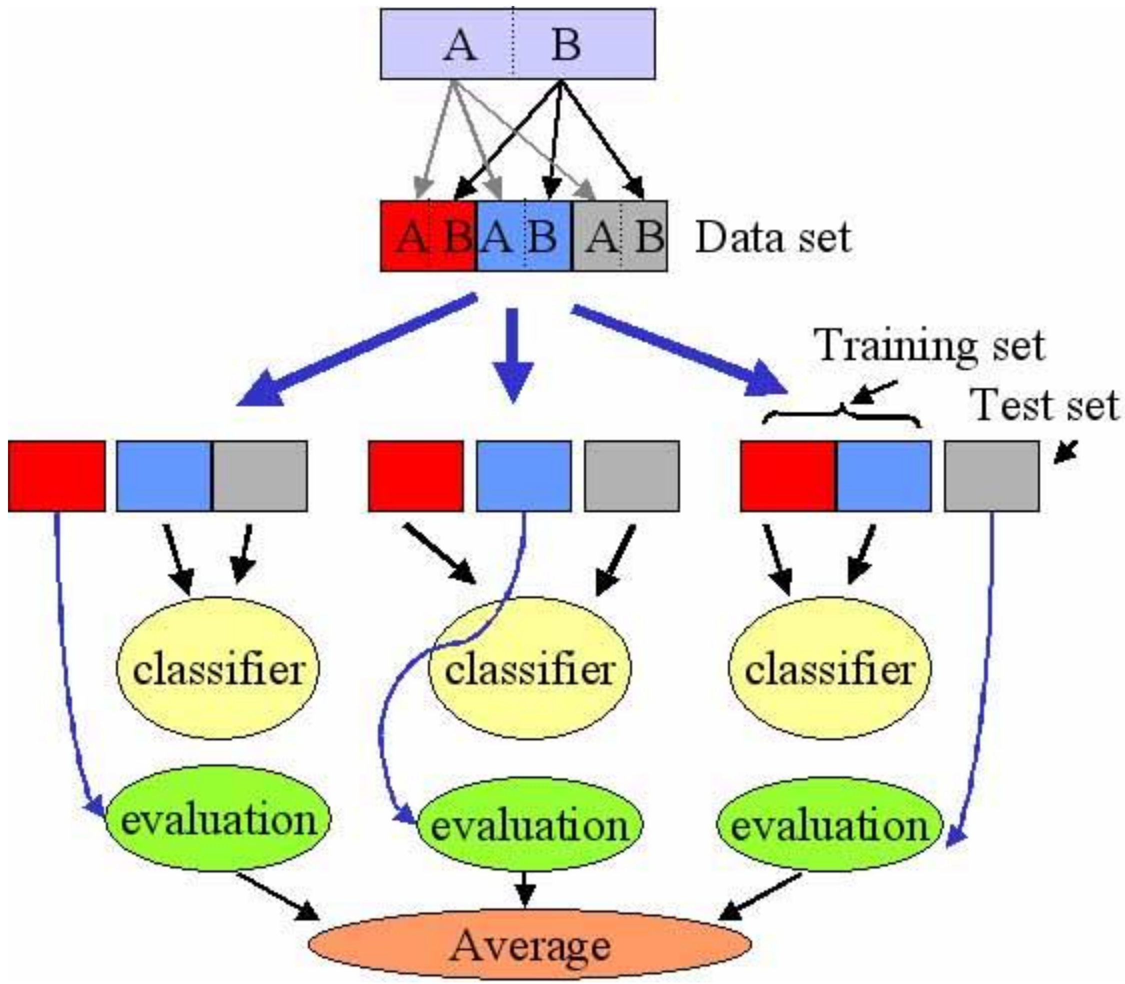
Resampling

- What if the sample of data is small or biassed?
- *Resampling* methods can be used
  - Randomly select training and test sets
  - Repeat for a fixed number of iterations
- Methods include
  - Cross-Validation
  - Bootstrapping

Cross Validation

- K-fold cross validation
  - Randomly split the data up into k subsets of equal size
  - Remove one of the subsets and train classifier on remaining subsets
  - Test on the removed subset
  - Repeat for all subsets
- Cross Validation considered an unbiased estimator of true error

Cross Validation

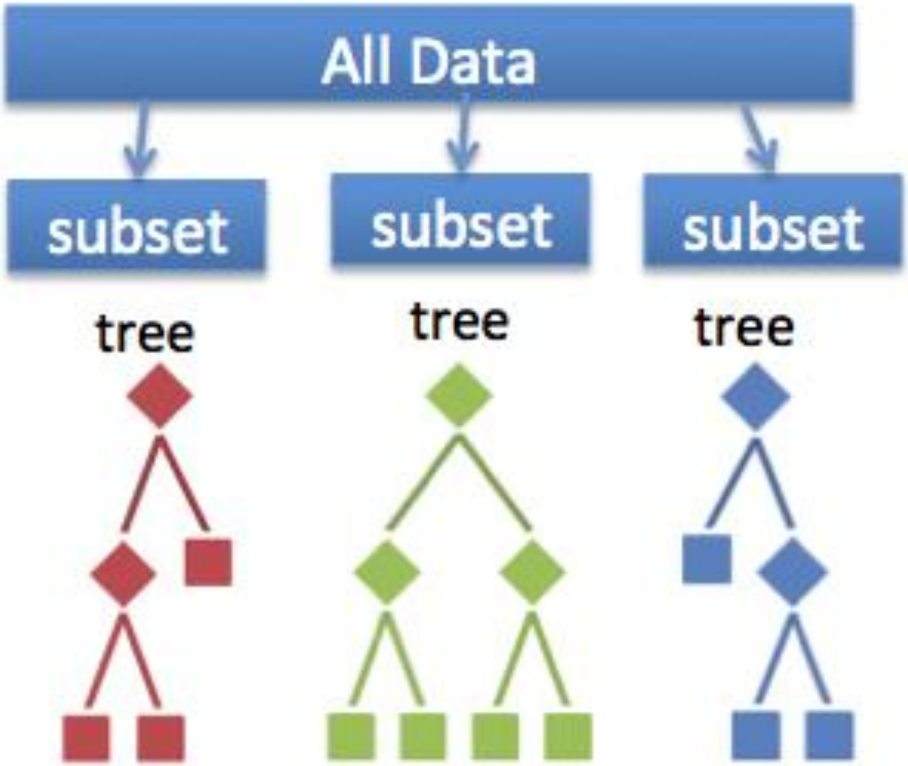


Bootstrapping

- For the *bootstrap n* training data items are sampled **with replacement** from *n* cases
- Cases that are not found in the training set are used for the test set
- Generally produces worse rates than the true error rate (worse case scenario)

Bootstrapping

Resampling & Random Forests



Brunel University London

Confusion Matrix

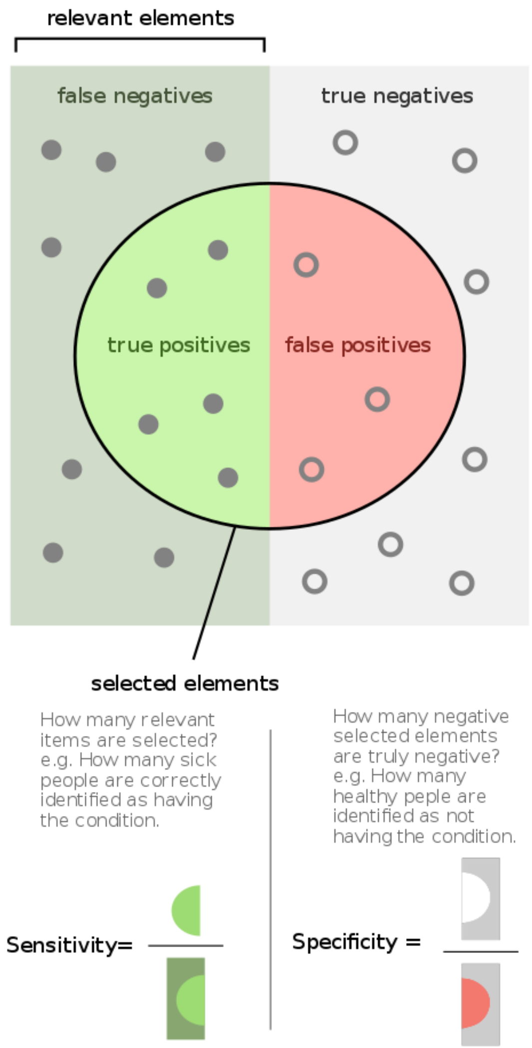
- If errors are of differing importance
  - E.g. failing to diagnose a disease can be more serious than diagnosing one that is not present
- Then use a confusion matrix:
  - Class versus prediction
  - False positives and negatives

Sensitivity & Specificity

Common measures make use of False positives and negatives

	Class Pos. (C+)	Class Neg. (C-)
Predict Pos. (P+)	True Positive	False Positive
Predict Neg. (P-)	False Negative	True Negative

Sensitivity & Specificity



Sensitivity: TP / C+

Specificity: TN / C-

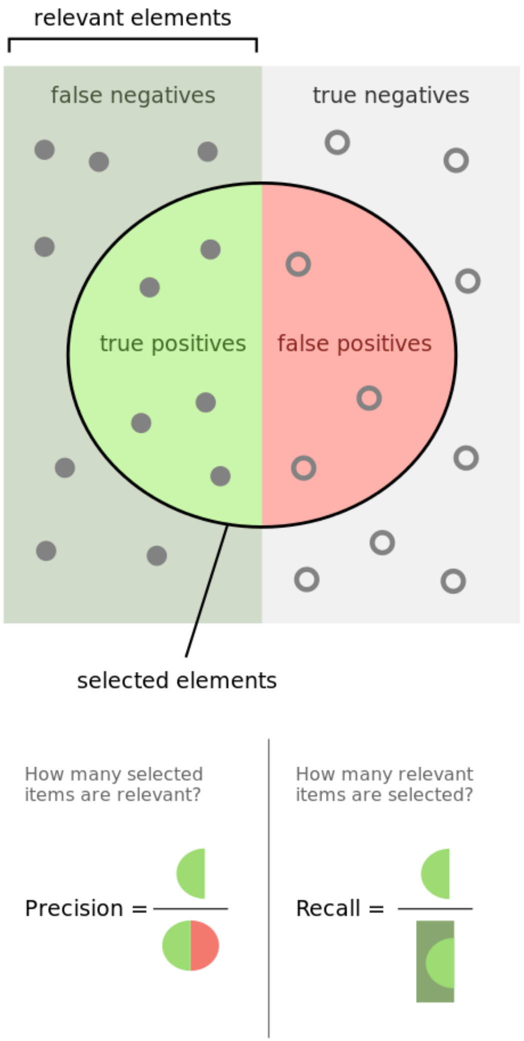
Accuracy: (TP + TN) / ((C+) + (C-))

Sensitivity & Specificity

- For example, a classifier can have high *sensitivity* if it successfully classifies people who have developed cancer but a low *specificity* if it also classifies non-sufferers with cancer.

- But what about data where there are only a few positive cases: imbalanced?

Precision & Recall



- Good for imbalanced data

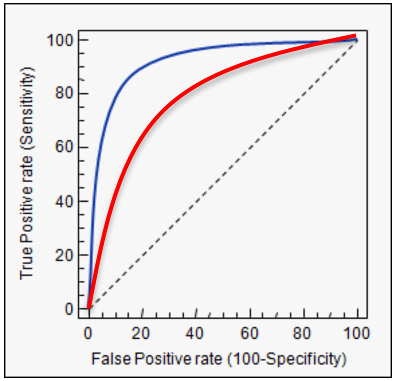
Precision = TP/(TP+FP)

Recall = TP/(TP+FN)

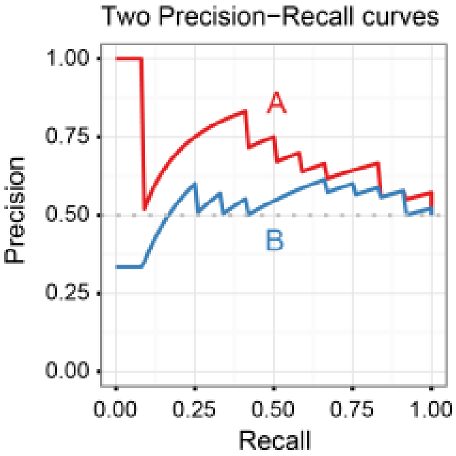
Brunel University London

ROC Curves vs PR Curves

- Receiver Operating Characteristic curve
  - Sensitivity / Specificity tradeoff



- Precision Recall Curves
  - Precision / Recall tradeoff



Brunel University London

Sensitivity & Specificity

- Some examples:

	Class Pos. (C+)	Class Neg. (C-)
Predict Pos. (P+)	15	1
Predict Neg. (P-)	10	24

Sensitivity & Specificity

- Some examples:

	Class Pos. (C+)	Class Neg. (C-)
Predict Pos. (P+)	24	10
Predict Neg. (P-)	1	15

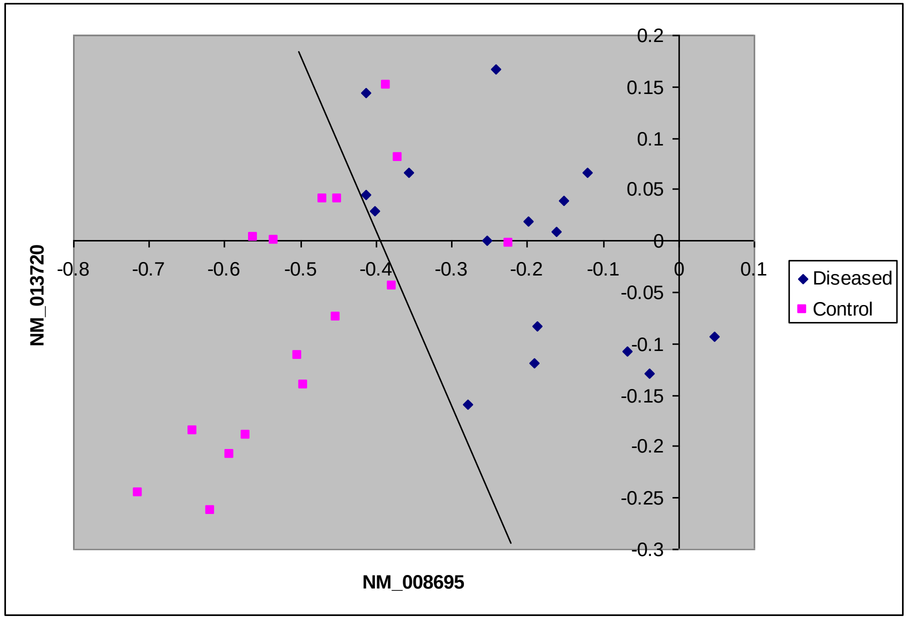
Sensitivity & Specificity

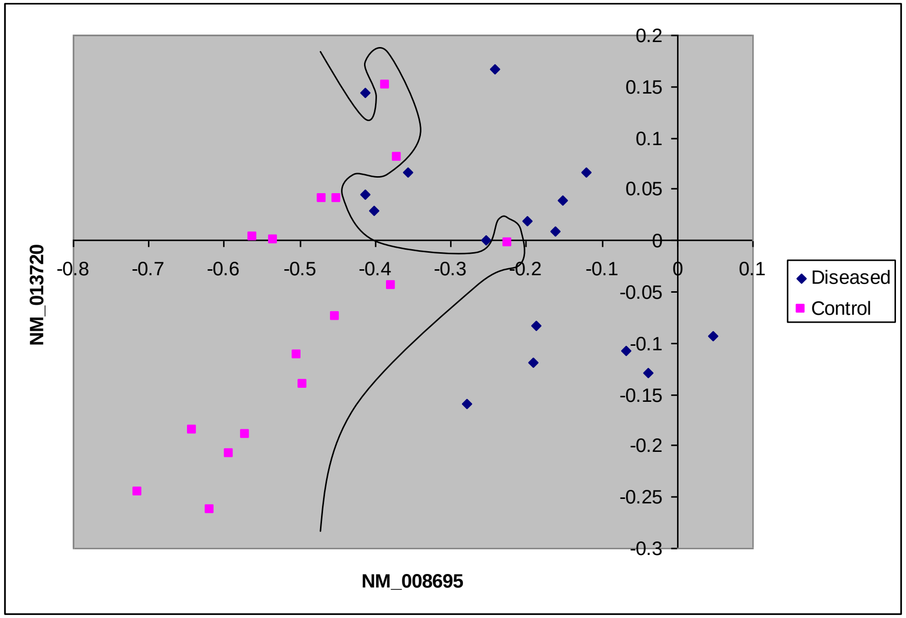
- Which would be most suitable for detecting patients with high-risk of cancer for follow up tests?
- Which would be best for selecting patients for high-risk surgery?
- Which is the best?

Brunel University London

Bias, Variance & Overfitting

- Consider two models:



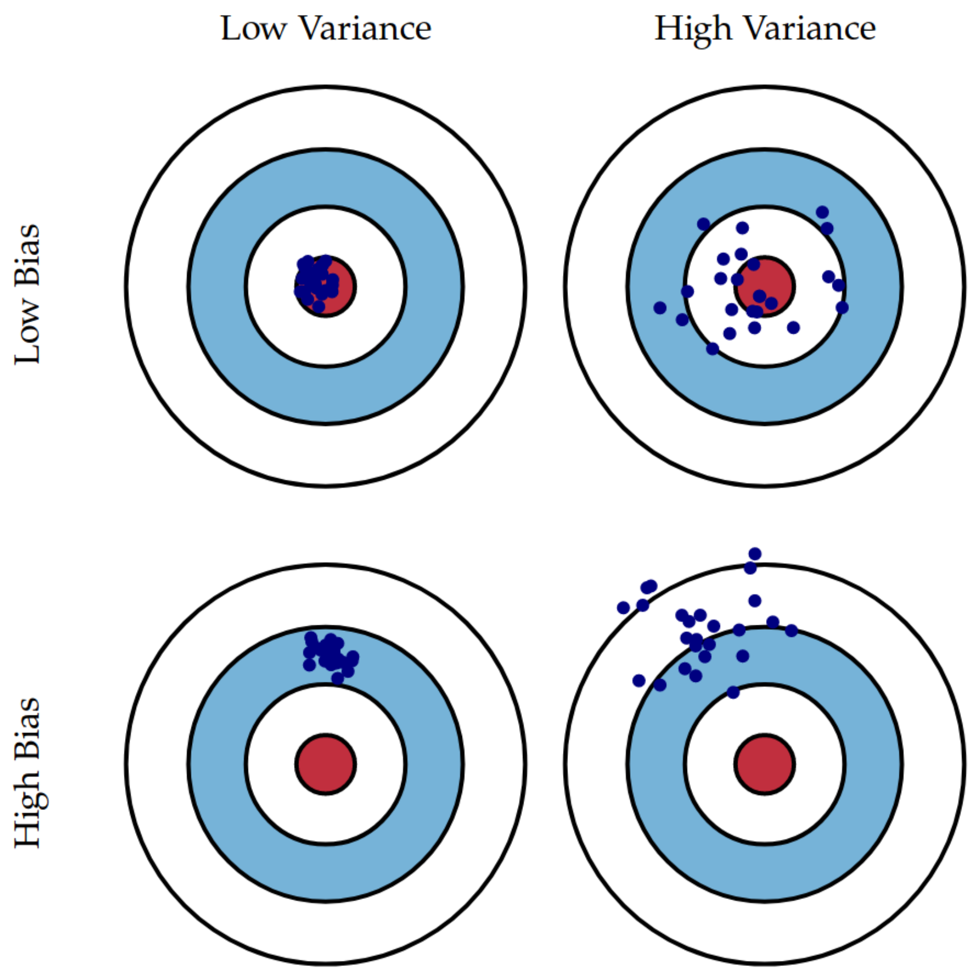


Brunel University London

Bias, Variance & Overfitting

- Bias is some systematic error in the model
- Variance is the difference from one model to the next
- The first model (straight line) has high bias
- The second has high variance
  - It fits the data very well
  - But it will not predict new cases
  - It has *overfit* to the data

Overfitting, Bias and Variance

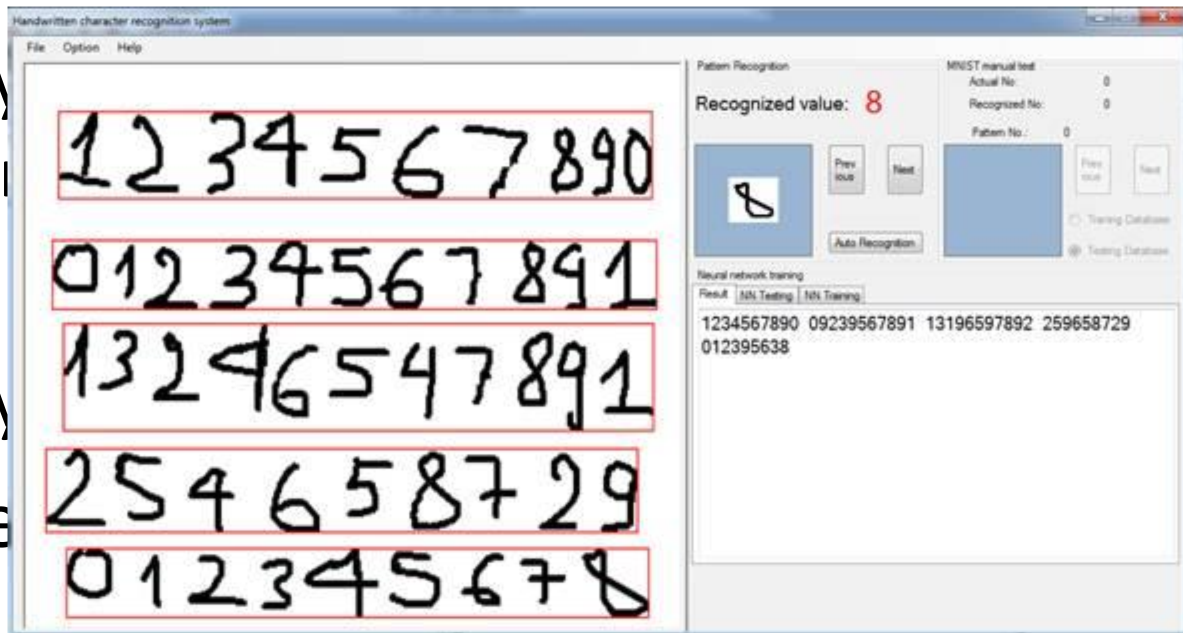


Summary

- Decision trees split the data in lines at each decision node
    - Easy to interpret, prone to overfit but can be pruned
- KNN does not model the data
    - Easy to interpret
- Different approaches to testing a classifier – e.g. sensitivity analysis

Summary

- Decision trees split the data in lines at each decision node
  - Easy … be
    pru…
- KNN d…
  - Easy…
- Differe… …er –
e.g. sen… analysis



Brunel University London

Summary

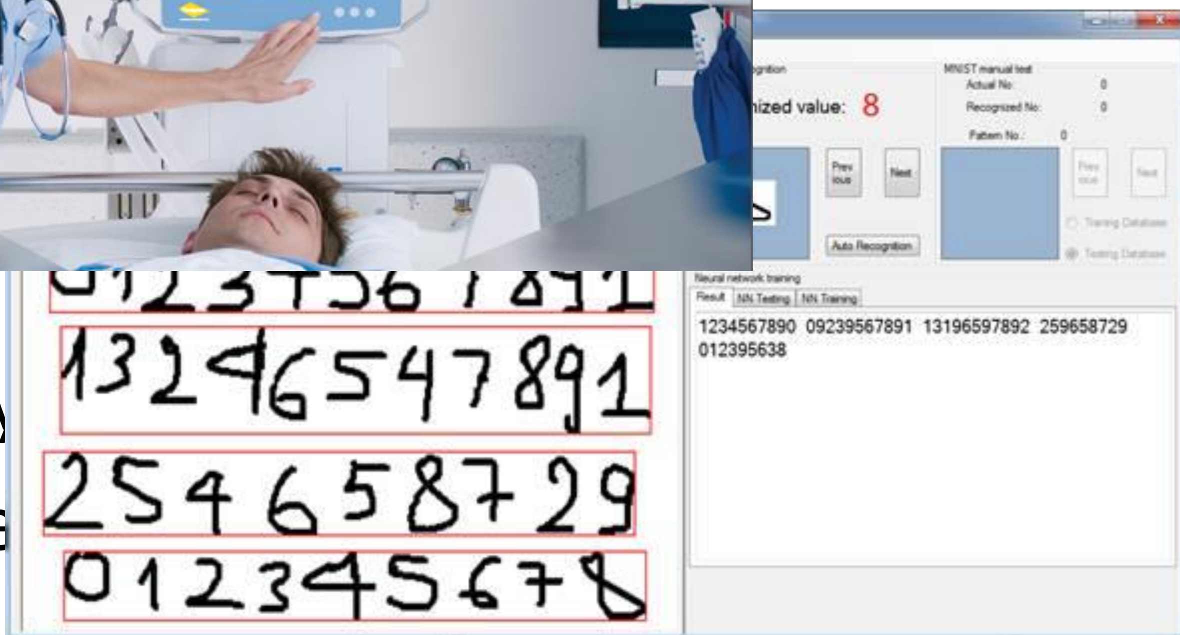
• D… in lines at each de…

…be

• KNN d…

  • Easy…

• Differe… …er – e.g. sen…



Brunel University London

Lab Next

- Explore the use of two classifiers on real data
- THIS IS ASSESSED

Reading

- David Hand: Chapter 10, Sections 2, 4, 5, 6 & 11
- Pang-Ning Tan "Introduction to Data Mining" (Chapter 4):

http://www-users.cs.umn.edu/~kumar/dmbook/index.php

http://vision.stanford.edu/teaching/cs231n-demos/knn/

- Notes on blackboard