

Lecture 7

- Improving the decision tree
- Naïve Bayes' classification
- Evaluating classifier performance

Presenter: Dr Heshan Kumarage

Week-by-week outline

Week Starting	Seminar	Topic	App Ses	A1	A2	Q/P	A3	Due Date
2/3/2026	1	Introduction to Data Science, R, review of basic statistics	-					
9/3/2026	2	Data visualisation	S1					
16/3/2026	3	Data manipulation	S2					
23/3/2026	4	Regression modelling	S3					
30/3/2026	5	Clustering	S4					
6/4/2026	-	Mid-semester Break						
13/4/2026	6	Classification using decision trees	S5					17/4/2026
20/4/2026	7	Improving and evaluating classifiers. Naïve Bayes classification	S6					
27/4/2026	8	Ensemble methods, Artificial Neural Networks	S7					
4/5/2026	9	Network analysis	S8					
11/5/2026	10	Introduction to text analysis	S9					15/5/2026
18/5/2026	11	Text analysis applications	Quiz/Prac					22/5/2026
25/5/2026	12	Text Network Analysis, Review of the unit, Assignment 3	S10,11,12					
1/6/2026		SWOT VAC	-					
8/6/2026		EXAM PERIOD	-					12/6/2026

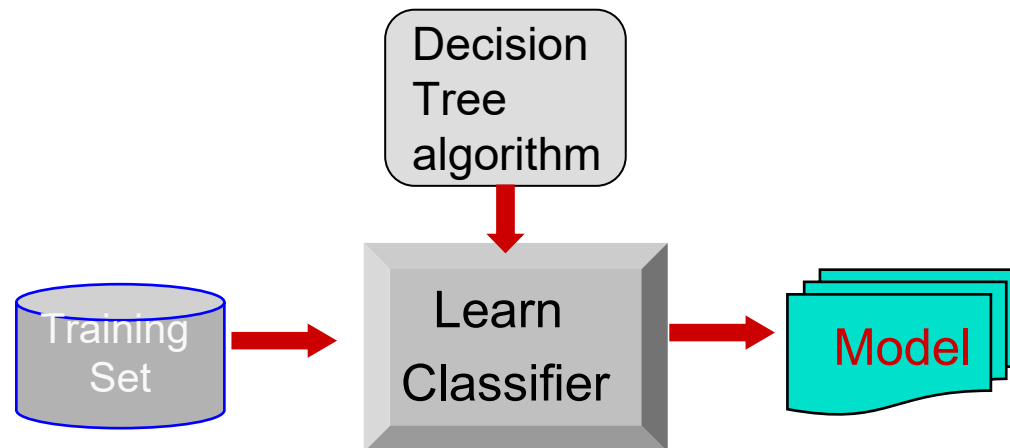
Improving the decision tree

In this section we'll cover:

- Review of classification,
- Performance metrics
- Decision tree size and accuracy
- Pruning the decision tree
- Cross validation
- Examples in R

Review: classification

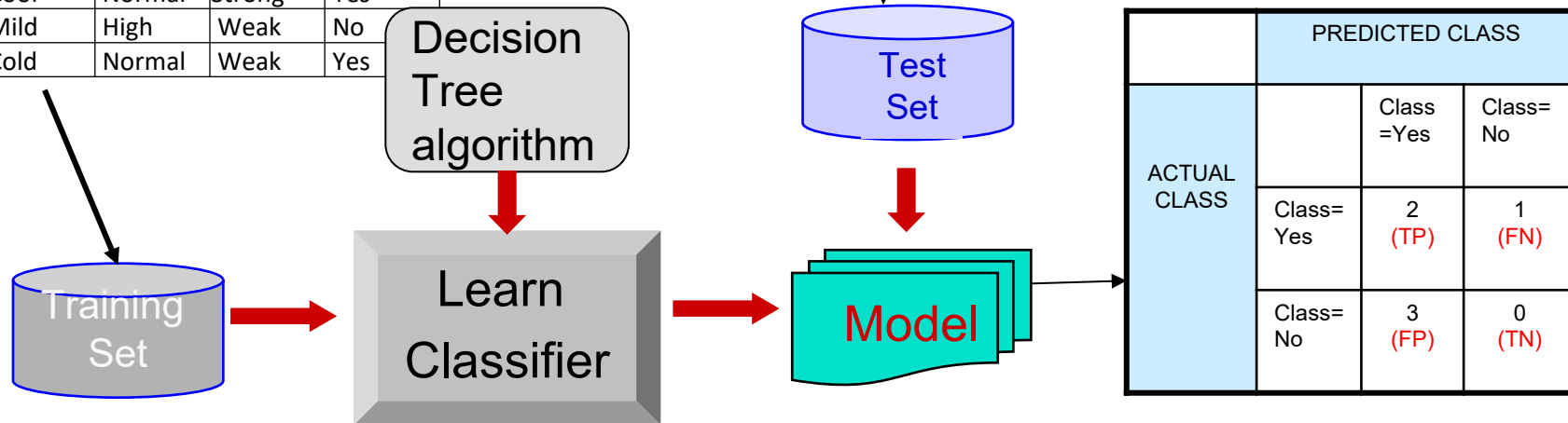
- The objective is to produce a model from a labelled data set.
- The model is used to classify new, unlabelled cases.



Review: Play Tennis example

ID	outlook	temp	humidity	wind	play
D1	Sunny	Hot	High	Weak	No
D10	Rain	Mild	Normal	Weak	Yes
D11	Sunny	Mild	Normal	Strong	Yes
D12	Overcast	Mild	High	Strong	Yes
D13	Overcast	Hot	Normal	Weak	Yes
D14	Rain	Mild	High	Strong	No
D2	Sunny	Hot	High	Strong	No
D3	Overcast	Hot	High	Weak	Yes
D4	Rain	Mild	High	Weak	Yes
D5	Rain	Cool	Normal	Weak	Yes
D6	Rain	Cool	Normal	Strong	No
D7	Overcast	Cool	Normal	Strong	Yes
D8	Sunny	Mild	High	Weak	No
D9	Sunny	Cold	Normal	Weak	Yes

ID	outlook	temp	humidity	wind	play
D15	Sunny	Mild	Normal	Strong	No
D16	Sunny	Hot	High	Weak	Yes
D17	Rain	Hot	High	Weak	No
D18	Overcast	Cool	High	Strong	No
D19	Overcast	Mild	Normal	Weak	Yes
D20	Rain	Mild	Normal	Weak	Yes



Review: performance evaluation

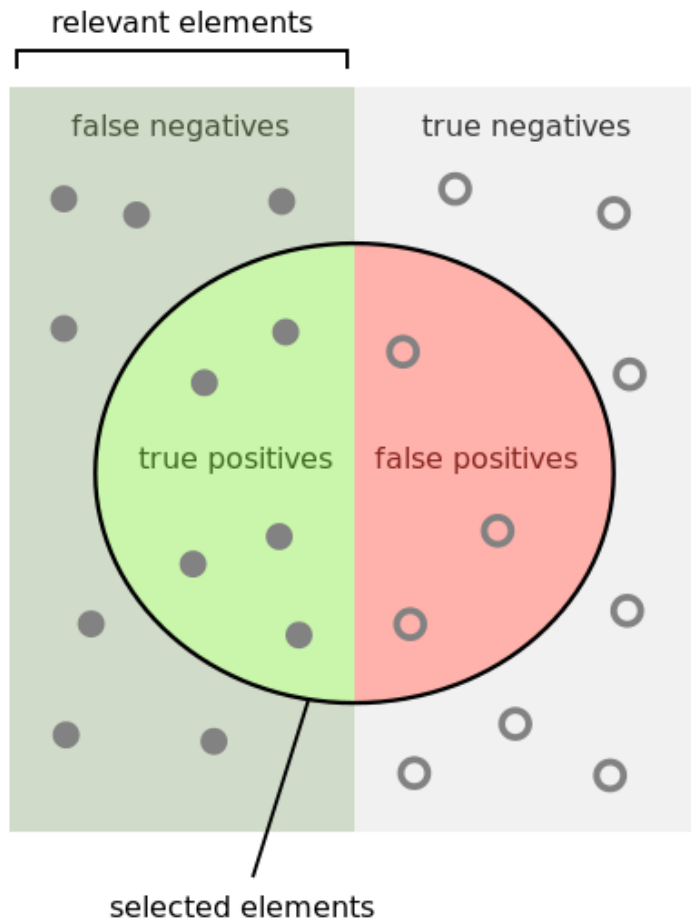
- Summary of test results as a Confusion Matrix

		PREDICTED CLASS	
		Class=Yes	Class=No
ACTUAL CLASS	Class=Yes	2 (TP)	1 (FN)
	Class=No	3 (FP)	0 (TN)

- Most commonly used metric:

$$Accuracy = \frac{a + d}{a + b + c + d} = \frac{TP + TN}{TP + TN + FP + FN} = \frac{2 + 0}{6} = 33.3\%$$

Additional metrics - precision & recall



How many selected items are relevant?

$$\text{Precision} = \frac{\text{true positives}}{\text{true positives} + \text{false positives}}$$

How many relevant items are selected?

$$\text{Recall} = \frac{\text{true positives}}{\text{true positives} + \text{false negatives}}$$

https://en.wikipedia.org/wiki/Precision_and_recall

Additional metrics - precision & recall

Precision is:

$$\frac{\textit{True Positives}}{\textit{True Positives} + \textit{False Positives}}$$

Recall is:

$$\frac{\textit{True Positives}}{\textit{True Positives} + \textit{False Negatives}}$$

Additional metrics – F1–Score

F1-Score:

$$\frac{2 * (Precision * Recall)}{Precision + Recall}$$

- The harmonic mean of precision and recall.
- Good for imbalanced datasets.
- Ranges from 0 – 1.
- One indicates perfect precision and recall while zero indicates either or both is zero.

Metrics: accuracy, precision & recall

		PREDICTED CLASS	
		Class=Yes	Class=No
ACTUAL CLASS	Class=Yes	2 (TP)	1 (FN)
	Class=No	3 (FP)	0 (TN)

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} =$$

$$Precision = \frac{TP}{TP + FP} =$$

$$Recall = \frac{TP}{TP + FN} =$$

$$F1 - Score =$$

Metrics: accuracy, precision & recall

		PREDICTED CLASS	
		Class=Yes	Class=No
ACTUAL CLASS	Class=Yes	2 (TP)	1 (FN)
	Class=No	3 (FP)	0 (TN)

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} = \frac{2 + 0}{6} = 33.3\%$$

$$Precision = \frac{TP}{TP + FP} = \frac{2}{5} = 40\%$$

$$Recall = \frac{TP}{TP + FN} = \frac{2}{3} = 66.7\%$$

$$F1 - Score = \frac{2 * (0.4 * 0.667)}{(0.4 + 0.667)} = 0.5$$

Decision trees

Advantages:

- Explicit rules plus graphical representation make them easy to interpret.
- Can handle mixed input: discrete and continuous.
- Quite robust to outliers, noise and missing values.
- Able to find the most discriminating attributes.
- Accuracy is good in general.

Disadvantages:

- Unstable in that small changes may lead to a completely different tree.
- Can become overly complex.

Decision trees

Overfitting the data:

- Leads to a model that does not generalise well.
- Model is excessively complex.
- Performs well on training data but not on unseen data.
- Has low recall.

Underfitting the data:

- Leads to a model that is too simple.
- Performs poorly on both training and unseen data.
- Has low precision.

Decision tree size

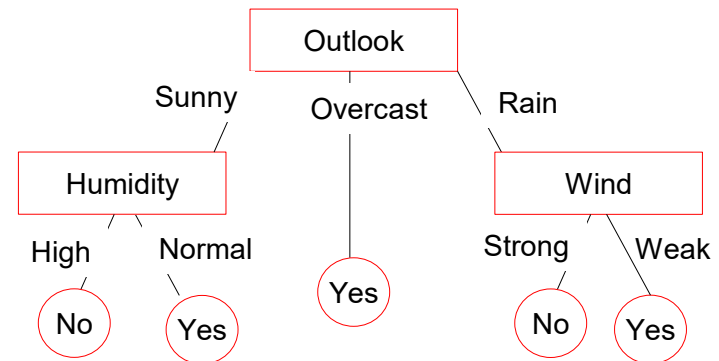
Leaf node count

- Corresponds to the number of rules that are encoded in a decision tree (5 in this example).

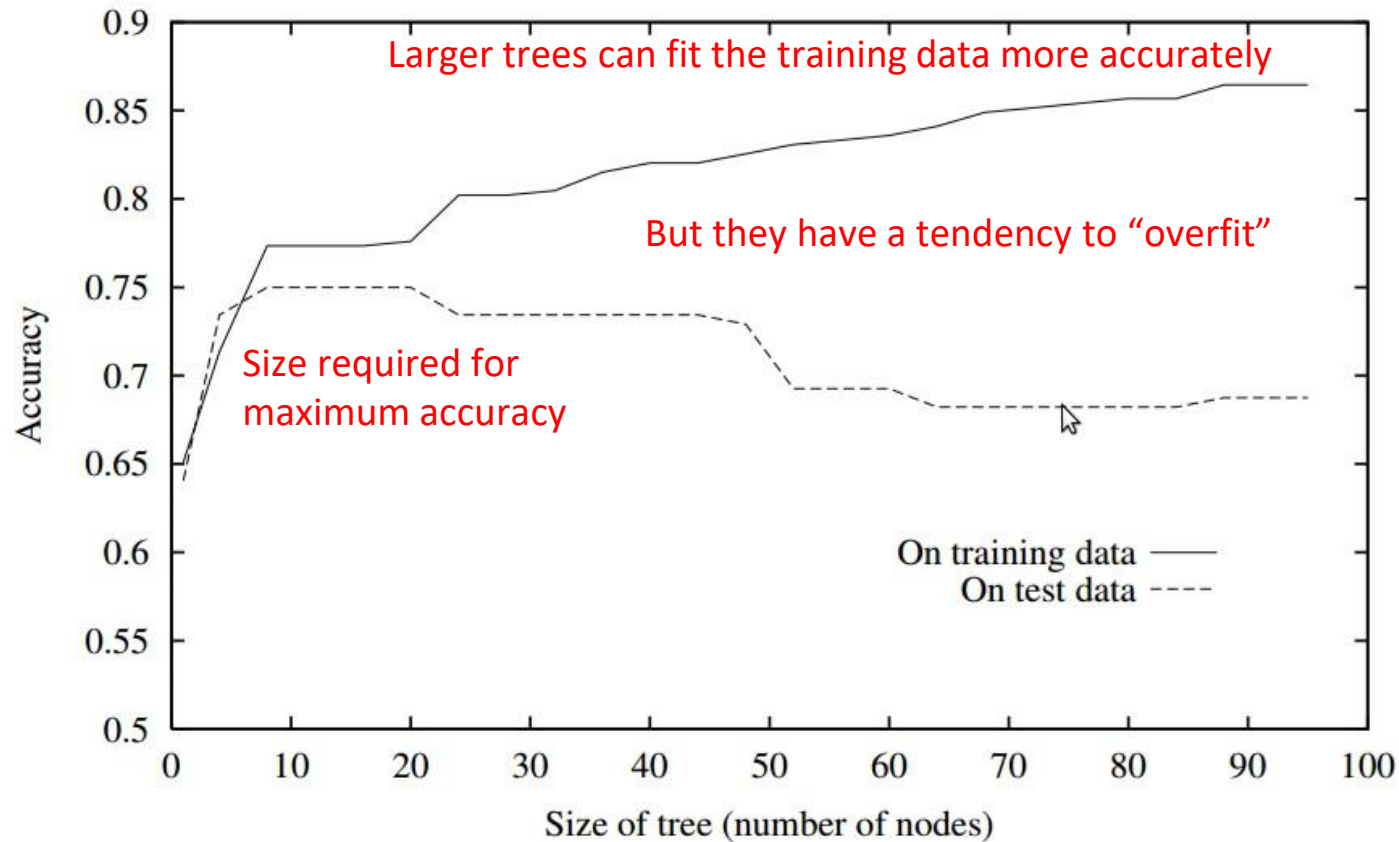
Tree height

- Corresponds to the maximum rule length and number of premises to be evaluated to reach class decision (2 in this example).

If outlook = Overcast, Then Play= Yes
If outlook = Rain And wind = Strong, Then Play= No
If outlook = Rain And wind = Weak, Then Play = Yes
If outlook = Sunny And humidity = High, Then Play = No
If outlook = Sunny And humidity = Normal, Then Play = Yes



Decision tree size vs accuracy



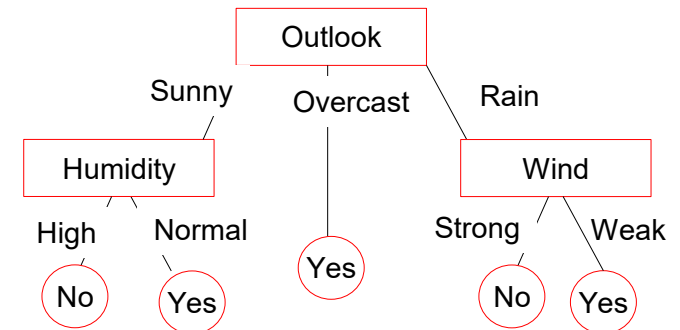
Pruning the decision tree

- Decision trees can be pruned to avoid overfitting the data.
- Pre-pruning aims to stop growing the tree before it over-fits the training data.
- Post-pruning allows the tree to overfit the data. The tree is then pruned. It is more effective in practice and more commonly used.
- Depth of the tree is reduced by replacing sub-trees with leaf nodes.
- The general concept is to remove rules if they have little effect on the error rate.

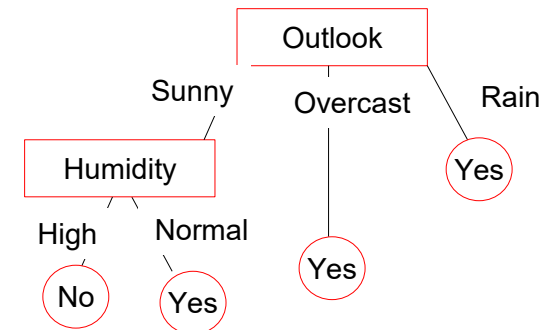
Post pruning example

- Keep a portion of data as validation set.
- Grow tree out to usual size as per no-pruning.
- Prune by replacing sub-trees with leaf nodes, by measuring error against validation set.
- Class label of replacement leaf node determined by majority of labels in the removed sub-tree.
- **3 Yes vs 2 No, so replace node with Yes.**

Original tree



After pruning



Cross validation

How do we make use of testing to guide the training process?

Remember: You cannot test on cases that you have already used as part of the training!

Therefore, you always need distinct *training* and *testing* data sets.

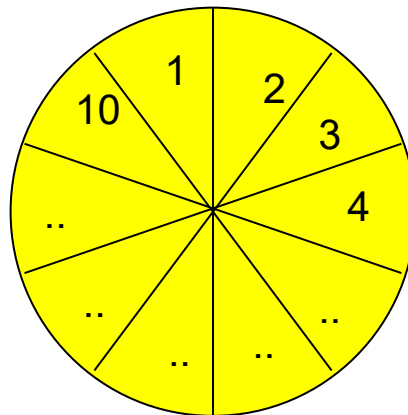
A model can be cross validated by evaluating across multiple train/test sets. This helps overcome the effects of randomness in train/test sets.

Cross validation

k -fold Cross validation steps:

- Partition the training data into k disjoint subsets,
- Train on $k-1$ partitions, test on the remaining one,
- Repeat until all partitions have been used to train/test.

Popular approach is 10-fold cross validation.

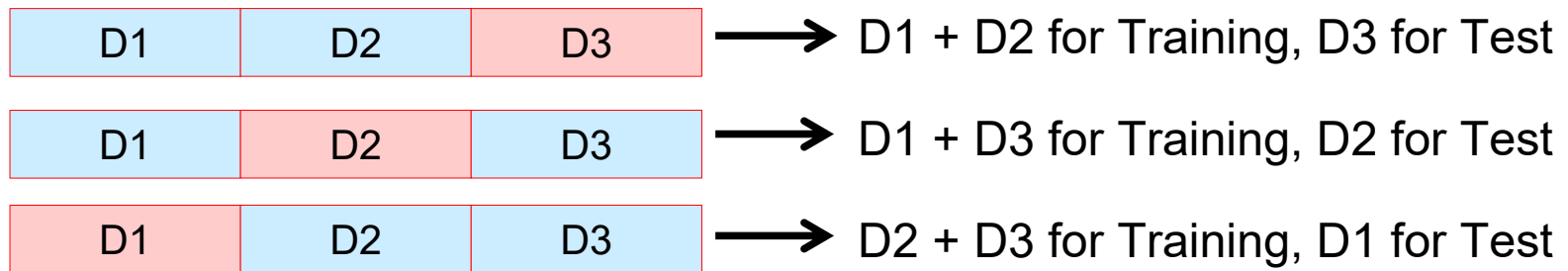


Example: 3-fold cross validation

Partitioning:

- $2/3$ training (making the model)
- $1/3$ testing (measuring performance of model)

Repeat the procedure three times so that every case has been used exactly once for testing.



Average performance across D1, D2 and D3.

Pruning and cross validation in R

Disclaimer:

- The following example shows how to cross validate and prune a decision tree in R. We're using a familiar tree to demonstrate the method.
- Because we're using very simple datasets, the full benefit of this process is not realized.
- Results are also subject to random seed choice.
- However, we will finish with an example using a more complex data set, where the benefit of this method becomes apparent.

Pruning and cross validation in R

Using the ‘playtennis’ example from last lecture:

- > # clean up the environment before starting
- > rm(list = ls())
- > install.packages("tree")
- > library(tree)
- > options(digits=4)

Pruning and cross validation in R

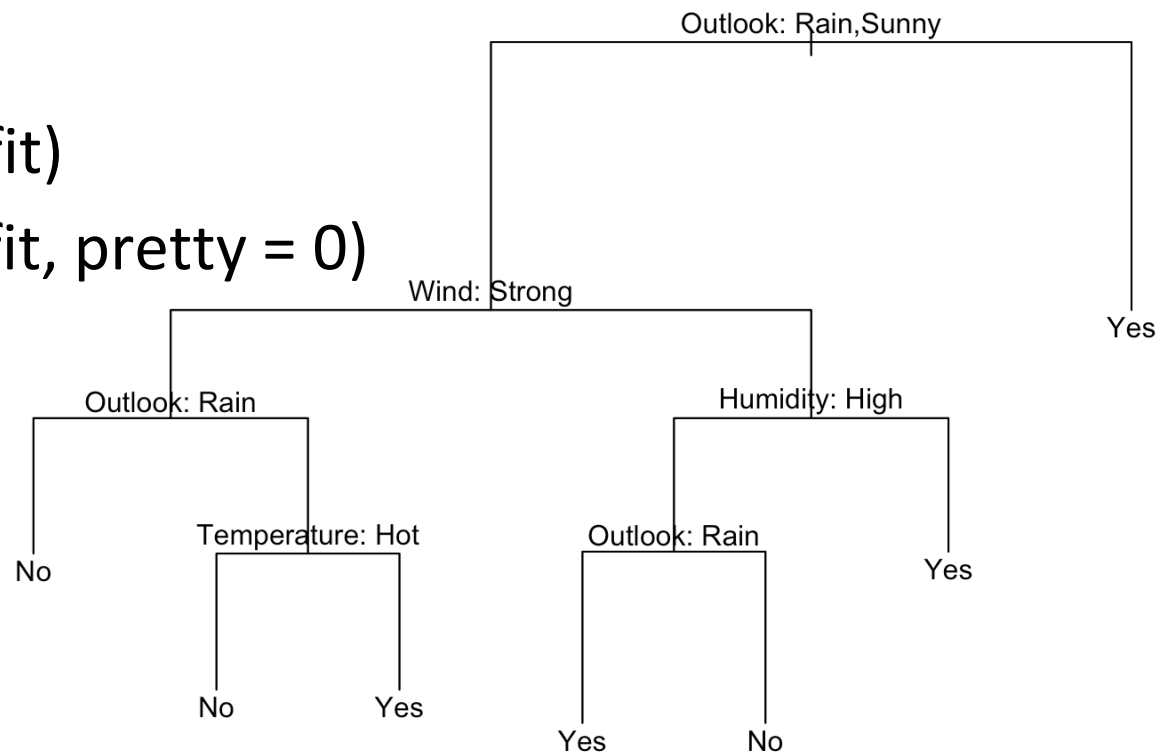
Resampling to create a larger, synthetic sample, and creating the basic tree:

- > `ptt <- read.csv("playtennistrain.csv")`
- > `set.seed(9999) #random seed`
- > `# resampling with replacement`
- > `pttrain = ptt[sample(nrow(ptt), 100, replace = TRUE),]`
- > `# fitting the model`
- > `ptfit = tree(Play ~. -Day, data = pttrain)`
- > `summary(ptfit)`

Pruning and cross validation in R

The original tree with 7 leaves.

- > plot(ptfit)
- > text(ptfit, pretty = 0)



Pruning and cross validation in R

Making predictions from the test data:

- > `pttest <- read.csv("playtennistest.csv")`
- > `tpredict = predict(ptfit, pttest, type = "class")`
- > `table(actual = pttest$Play, predicted = tpredict)`

	predicted	
actual	No	Yes
No	0	3
Yes	1	2

- *Accuracy* = $\frac{2}{6} = 33.3\%$ can we improve on this?

Pruning and cross validation in R

Cross validation test at different tree sizes:

```
> testptfit = cv.tree(ptfit, FUN = prune.misclass)
```

```
> testptfit
```

```
  $size
```

```
[1] 7 5 3 1
```

```
  $dev
```

```
[1] 5 11 22 44 count misclassifications
```

```
  $k
```

```
[1] -Inf 4.0 5.5 10.0
```

```
  $method
```

```
[1] "misclass"
```

```
...
```

Ideally, choose a tree size to minimize deviations/misclassifications. This example suggests Size = 7 is best.

? cv.tree



- Description

Runs a K-fold cross-validation experiment to find the deviance or number of misclassifications as a function of the cost-complexity parameter k .

- Usage

```
cv.tree(object, rand, FUN = prune.tree, K = 10,  
...)
```

FUN The function to do the pruning.

K The number of folds of the cross-validation.

... Additional arguments to FUN.

Pruning and cross validation in R

Pruning the tree: Choosing a smaller tree to illustrate method even though misclassifications suggested Size = 7 best.

```
> prune.ptfit = prune.misclass(ptfit, best = 3)
```

```
> summary(prune.ptfit)
```

```
Classification tree:
```

```
snip.tree(tree = ptfit, nodes = 4:5)
```

```
Variables actually used in tree construction:
```

```
[1] "Outlook" "Wind"
```

```
Number of terminal nodes: 3
```

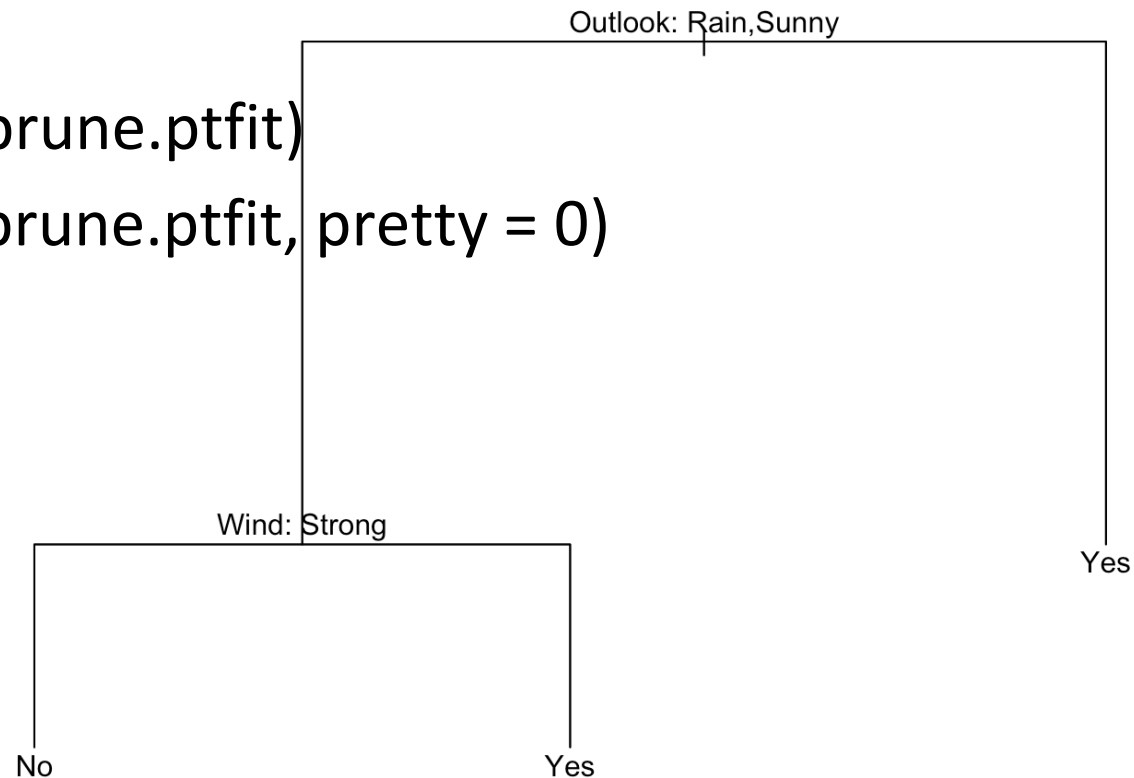
```
Residual mean deviance: 0.834 = 80.9 / 97
```

```
Misclassification error rate: 0.19 = 19 / 100
```

Pruning and cross validation in R

The new tree with 3 leaves.

- > `plot(prune.ptfit)`
- > `text(prune.ptfit, pretty = 0)`



Pruning and cross validation in R

Making predictions from the test data:

- > `ppredict = predict(prune.ptfit, pttest, type = "class")`
- > `table(actual = pttest$Play, predicted = ppredict)`

	predicted	
actual	No	Yes
No	1	2
Yes	0	3

- *Accuracy* = $\frac{4}{6} = 66.7\%$ compared with 33.3% previously!

Confidence level of a classification

How confident was the model of the “No” or “Yes” classifications in the “Play Tennis” example?

- We could obtain the confidence level for each class and use this to make the classification...
 - > `ppredict = predict(prune.ptfit, pttest, type = "vector")`

	No	Yes
1	0.1471	0.8529
2	0.8571	0.1429
3	0.8571	0.1429
4	0.0000	1.0000
5	0.0000	1.0000
6	0.1471	0.8529

Predict as a “vector”.

Pruning and cross validation in R

Example with more complex data: the Japanese Credit Data, from Applied Sessions 7 & 8.

- > JC <- read.csv("JapaneseCredit.csv", stringsAsFactors = T)
- > JC = JC[complete.cases(JC),]
- > set.seed(9999) #random seed
- > train.row = sample(1:nrow(JC), 0.7*nrow(JC))
- > JC.train = JC[train.row,]
- > JC.test = JC[-train.row,]

Pruning and cross validation in R

Fitting basic tree:

- > JC.tree = tree(Class ~., data = JC.train)
- > JC.predtree = predict(JC.tree, JC.test, type = "class")

```
table(actual = JC.test$Class, predicted =  
       JC.predtree)
```

```
          predicted  
actual   -   +  
   -    96  20  
   +     8  76
```

Pruning and cross validation in R

Cross validating:

```
> JC.CV.tree = cv.tree(JC.tree, FUN = prune.misclass, K = 10)
```

```
> JC.CV.tree
```

```
  $size
```

```
[1] 16 13 10  8  2  1
```

```
  $dev
```

```
[1] 73  71  72  71  71 215
```

```
  $k
```

```
[1] -Inf    0.000    1.667    2.000    3.000 152.000
```

Pruning and cross validation in R

Pruning and predicting:

- > `prune.JC.CV.tree = prune.misclass(JC.tree, best = 8)`
- > `JC.CV.predtree = predict(prune.JC.CV.tree, JC.test, type = "class")`
- > `table(actual = JC.test$Class, predicted = JC.CV.predtree)`

```
      predicted
actual -  +
-    95 21
+     4 80
```

- > # misclassifications reduced from 28 to 25

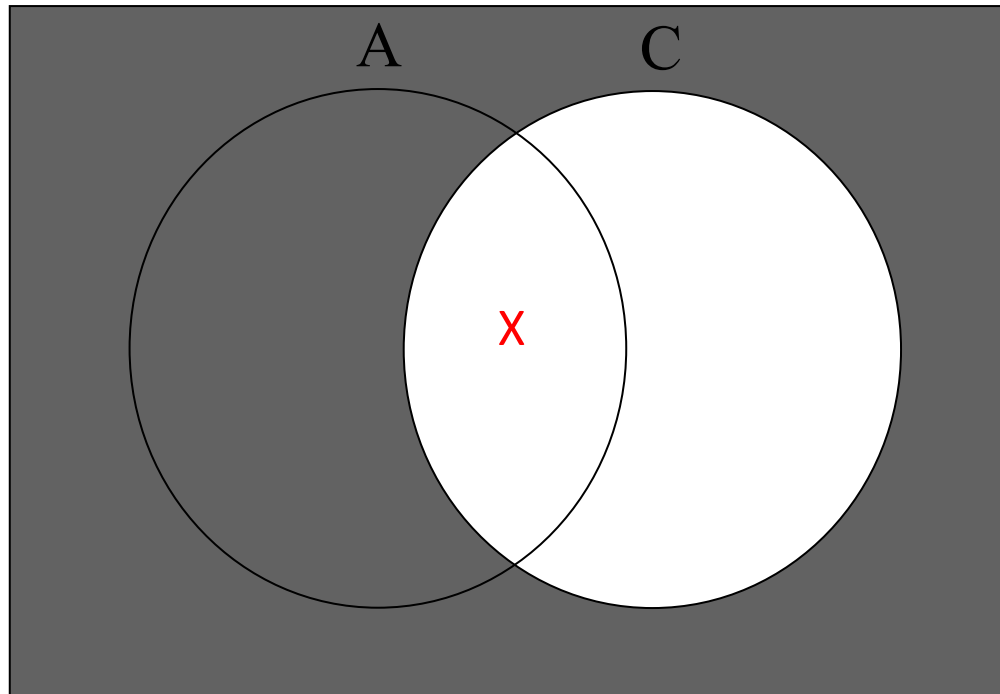
Naïve Bayes' classification

In this section we'll cover:

- Bayes' Theorem
- Bayesian Classifiers
- Independent events
- Naïve Bayes' Classifier
- Hand-worked example
- Naïve Bayes Classification in R

Introduction: conditional probability

What is the conditional probability that x is a member of A given that it is a member of C :



Introduction: conditional probability

More formally, calculate the conditional probability that x has attribute A given it is of class C :

- Using Bayes' Theorem:

$$P(A|C) = \frac{P(A \cap C)}{P(C)}$$

- Which can be rewritten to make $P(A \cap C)$ explicit as:

$$P(A \cap C) = P(C) \cdot P(A|C)$$

Applying Bayes' Theorem

- The *prior* probability of belonging to Class C is $P(C)$.
- Obtain extra information: the probability of having attribute A if a member of C: $P(A|C)$.
- Flipping this around: what is the *posterior* probability of belonging to class C when having attribute, A? That is, $P(C|A)$?

- Using Bayes' Theorem this is:

$$P(C|A) = \frac{P(A \cap C)}{P(A)} = \frac{P(C) \cdot P(A|C)}{P(A)}$$

- We can use this idea for classification...

Example: Bayes' Theorem

Spam detection:

- Incoming emails are classified as spam or not to build up a corpus of examples.
- New emails are assigned a probability of being spam or not by comparing their attributes (words/domain etc.) with the corpus and classified.
- Corpus is updated with new classified examples – this enables it to learn new spam words and techniques.

Bayesian classifiers

General idea:

- Begin with a corpus of classified data.
- Consider each attribute and class label as a random variable.
- Given a record with decision attributes (A_1, A_2, \dots, A_n) the goal is to classify a record as belonging to class C , or not.
- Specifically, we want to find the value of C , C_j that maximizes $P(C_j/A_1, A_2, \dots, A_n)$.
- This can be estimated directly from data.

Bayesian classifiers

Approach:

- Compute the posterior probability for all values of C_j using Bayes theorem

$$P(C_j|A_1 \cap A_2 \cap A_3 \dots \cap A_n) = \frac{P(C_j) \cdot P(A_1 \cap A_2 \cap A_3 \dots \cap A_n|C_j)}{P(A_1 \cap A_2 \cap A_3 \dots \cap A_n)}$$

- Choose the value of C_j that maximises this posterior probability

$$P(C_j|A_1 \cap A_2 \cap A_3 \dots \cap A_n).$$

- Is equivalent to maximising $P(C_j) \cdot P(A_1 \cap A_2 \cap A_3 \dots \cap A_n|C_j)$, since $P(A_1 \cap A_2 \cap A_3 \dots \cap A_n)$ is the same for all C_j .

Independent events

The joint probability of two events is equivalent to the product of their probabilities if and only if they are independent.

Assuming *independence* between attributes lets us use a Naïve Bayes model...

Naïve Bayes classifier

From Wikipedia:

- In machine learning, naïve Bayes classifiers are a family of simple probabilistic classifiers based on applying Bayes' theorem with strong (naive) independence assumptions between the features.
- ... remains a popular (baseline) method for text categorization, the problem of judging documents as belonging to one category or the other (such as spam or legitimate, sports or politics, etc...)

https://en.wikipedia.org/wiki/Naive_Bayes_classifier

Naïve Bayes classifier

- Assume independence among attributes A_i when class is given, thus

$$P(A_1 \cap A_2 \cap A_3 \dots \cap A_n | C_j) = P(A_1 | C_j) \times P(A_2 | C_j) \times \dots \times P(A_n | C_j)$$

- Estimate $P(A_i | C_j)$ for all A_i

- A new point is classified as C_j to maximise

$$P(C_j) \times P(A_1 | C_j) \times P(A_2 | C_j) \times \dots \times P(A_n | C_j)$$

- Classification confidence (probability) is given by

$$P(C_j | A_1 \cap A_2 \cap A_3 \dots \cap A_n) = \frac{P(C_j) \times P(A_1 | C_j) \times P(A_2 | C_j) \times \dots \times P(A_n | C_j)}{P(A_1) \times P(A_2) \times \dots \times P(A_n)}$$

Naïve Bayes example

Given the following data: classify an unknown animal as mammal (M) or non-mammal (N).

Name	A1: Gives Birth	A2: Can Fly	A3: Live in Water	A4: Has Legs	Class
bat	yes	yes	no	yes	M
cat	yes	no	no	yes	M
dolphin	yes	no	yes	no	M
eagle	no	yes	no	yes	N
eel	no	no	yes	no	N
frog	no	no	sometimes	yes	N
gila monster	no	no	no	yes	N
human	yes	no	no	yes	M
komodo	no	no	no	yes	N
leopard shark	yes	no	yes	no	N
owl	no	yes	no	yes	N
penguin	no	no	sometimes	yes	N
pigeon	no	yes	no	yes	N
platypus	no	no	no	yes	M
porcupine	yes	no	no	yes	M
python	no	no	no	no	N
salamander	no	no	sometimes	yes	N
salmon	no	no	yes	no	N
turtle	no	no	sometimes	yes	N
whale	yes	no	yes	no	M
Unknown	yes	no	yes	no	?

Naïve Bayes example

- Recall that: a new point is classified as C_j to maximise: $P(C_j) \times P(A_1|C_j) \times P(A_2|C_j) \times \dots \times P(A_n|C_j)$
- Looking at the non-mammals only. Note $P(N) = 13/20$:

Name	A1: Gives Birth	A2: Can Fly	A3: Live in Water	A4: Has Legs	Class
eagle	no	yes	no	yes	N
eel	no	no	yes	no	N
frog	no	no	sometimes	yes	N
gila monster	no	no	no	yes	N
komodo	no	no	no	yes	N
leopard shark	yes	no	yes	no	N
owl	no	yes	no	yes	N
penguin	no	no	sometimes	yes	N
pigeon	no	yes	no	yes	N
python	no	no	no	no	N
salamander	no	no	sometimes	yes	N
salmon	no	no	yes	no	N
turtle	no	no	sometimes	yes	N

Unknown	yes	no	yes	no	?
---------	-----	----	-----	----	---

- $P(A_1|N) = 1/13$; $P(A_2|N) = 10/13$; $P(A_3|N) = 3/13$; $P(A_4|N) = 4/13$
- $P(N) * P(A|N) = 13/20 * 1/13 * 10/13 * 3/13 * 4/13 = 0.0027$

Naïve Bayes example

- Recall: maximise: $P(C_j) \times P(A_1|C_j) \times P(A_2|C_j) \times \dots \times P(A_n|C_j)$
- Now for the mammals, note $P(M) = 7/20$:

Name	A1: Gives Birth	A2: Can Fly	A3: Live in Water	A4: Has Legs	Class
bat	yes	yes	no	yes	M
cat	yes	no	no	yes	M
dolphin	yes	no	yes	no	M
human	yes	no	no	yes	M
platypus	no	no	no	yes	M
porcupine	yes	no	no	yes	M
whale	yes	no	yes	no	M
Unknown	yes	no	yes	no	?

- $P(A_1|M) = _ / 7$; $P(A_2|M) = _ / 7$; $P(A_3|M) = _ / 7$; $P(A_4|M) = _ / 7$
- $P(M) * P(A|M) = 7/20 * 6/7 * 6/7 * 2/7 * 2/7 = 0.021$

Naïve Bayes example

- Putting it all together. We want j to maximise:

$$P(C_j) \times P(A_1|C_j) \times P(A_2|C_j) \times \dots \times P(A_n|C_j)$$

- When $j = \text{non-mammal}$, $P(N) = 13/20$
- $P(A_1|N) = 1/13$; $P(A_2|N) = 10/13$; $P(A_3|N) = 3/13$; $P(A_4|N) = 4/13$
- $P(N) * P(A|N) = 13/20 * 1/13 * 10/13 * 3/13 * 4/13 = 0.0027$

- When $j = \text{mammal}$, $P(M) = 7/20$
- $P(A_1|M) = 6/7$; $P(A_2|M) = 6/7$; $P(A_3|M) = 2/7$; $P(A_4|M) = 2/7$
- $P(M) * P(A|M) = 7/20 * 6/7 * 6/7 * 2/7 * 2/7 = 0.021$

- $P(M) * P(A|M) > P(N) * P(A|N)$ so classify as Mammal

Naïve Bayes classification probability

- The previous example only used the numerator of the Bayes' calculation to determine the most likely class. If we want the actual classification probability, we need to adapt the complete calculation:

$$P(C_j|A_1 \cap A_2 \cap A_3 \dots \cap A_n) = \frac{P(C_j) \cdot P(A_1 \cap A_2 \cap A_3 \dots \cap A_n|C_j)}{P(A_1 \cap A_2 \cap A_3 \dots \cap A_n)}$$

- When $j = \text{non-mammal}$, $P(N) = 13/20$
- $P(A_1|N) = 1/13$; $P(A_2|N) = 10/13$; $P(A_3|N) = 3/13$; $P(A_4|N) = 4/13$
- $P(N) * P(A|N) = 13/20 * 1/13 * 10/13 * 3/13 * 4/13 = 0.0027$
- $P(A_1 \cap A_2 \cap \dots) = 7/20 * 16/20 * 5/20 * 6/20 = 0.021$
- Classification probability = $0.003/0.021=0.13$

Naïve Bayes classifier

Some considerations. The model is:

- Robust to isolated noise points,
- Can adapt quickly to new instances (new data),
- Handles missing values by ignoring the instance during probability estimate calculations,
- Robust to irrelevant attributes,
- Independence assumption may not hold for some attributes, in which case use another technique such as Bayesian Belief Networks.

Naïve Bayes classification in R

Install package “e1071”, read data and build model (note: resampling not required).

- > `install.packages("e1071")`
- > `library(e1071)`
- > `pttrain <- read.csv("playtennistrain.csv")`
- > `pttest <- read.csv("playtennistest.csv")`
- > `tmodel = naiveBayes(Play ~. -Day, data = pttrain)`

Naïve Bayes

Classification

```
> tbpredict = predict(tmodel, pttest)
```

```
> tbpredict
```

```
[1] No  Yes  Yes  Yes  Yes  No
```

```
Levels: No Yes
```

```
> table(actual = pttest$Play, predicted = tbpredict)
```

```
      predicted
actual No  Yes
No     1   2
Yes    1   2
```

Naïve Bayes

Alternatively create a vector of confidence levels for predicting each class:

- > `tbpredict.r = predict(tmodel, ptest, type = 'raw')`
- > `tbpredict.r`

	No	Yes
<code>[1,]</code>	<code>9.932e-01</code>	<code>0.006812</code>
<code>[2,]</code>	<code>3.381e-02</code>	<code>0.966191</code>
<code>[3,]</code>	<code>1.531e-02</code>	<code>0.984686</code>
<code>[4,]</code>	<code>2.916e-05</code>	<code>0.999971</code>
<code>[5,]</code>	<code>1.822e-06</code>	<code>0.999998</code>
<code>[6,]</code>	<code>9.558e-01</code>	<code>0.044248</code>

Evaluating classifier performance

In this Section we'll cover:

- Evaluating classifier performance
- ROC (Receiver Operating Characteristic)
- AUC (Area Under the Curve)
- Lift

Receiver Operating Characteristic

Developed in the 1950s for signal detection - to analyse noisy signal transmission.

- Characterises the tradeoff between positive hits and false alarms for a binary classifier.
- ROC plots True Positive Rate, TPR, (on y axis) against False Positive Rate, FPR, (on x axis).
- Performance of a single classifier presented as a single point of ROC curve.
- Changing the threshold of algorithm, sample distribution or cost matrix etc. changes that point: this lets a profile of classifier to be developed.

ROC

Calculating TPR and FPR

- True Positive Rate, TPR, also called *sensitivity* (or *recall*), indicates how good a test is for correctly predicting “yes” when it should predict “yes”. (Think of statistical confidence.)

$$\text{True Positive Rate: } TPR = \frac{TP}{TP + FN}$$

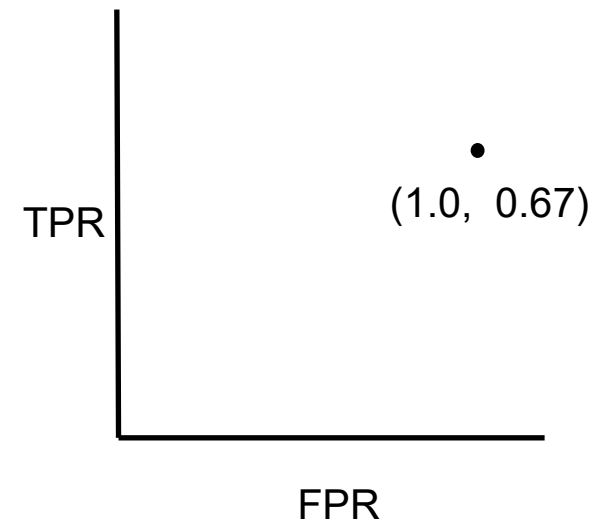
- False Positive Rate, FPR, also known as a ‘*false alarm*’ (or $1 - \text{specificity}$) is:

$$\text{False Positive Rate: } FPR = \frac{FP}{FP + TN}$$

- ROC plots TPR against FPR.

ROC: Play tennis example

		PREDICTED CLASS	
		Class=Yes	Class=No
ACTUAL CLASS	Class=Yes	2 (TP)	1 (FN)
	Class=No	3 (FP)	0 (TN)

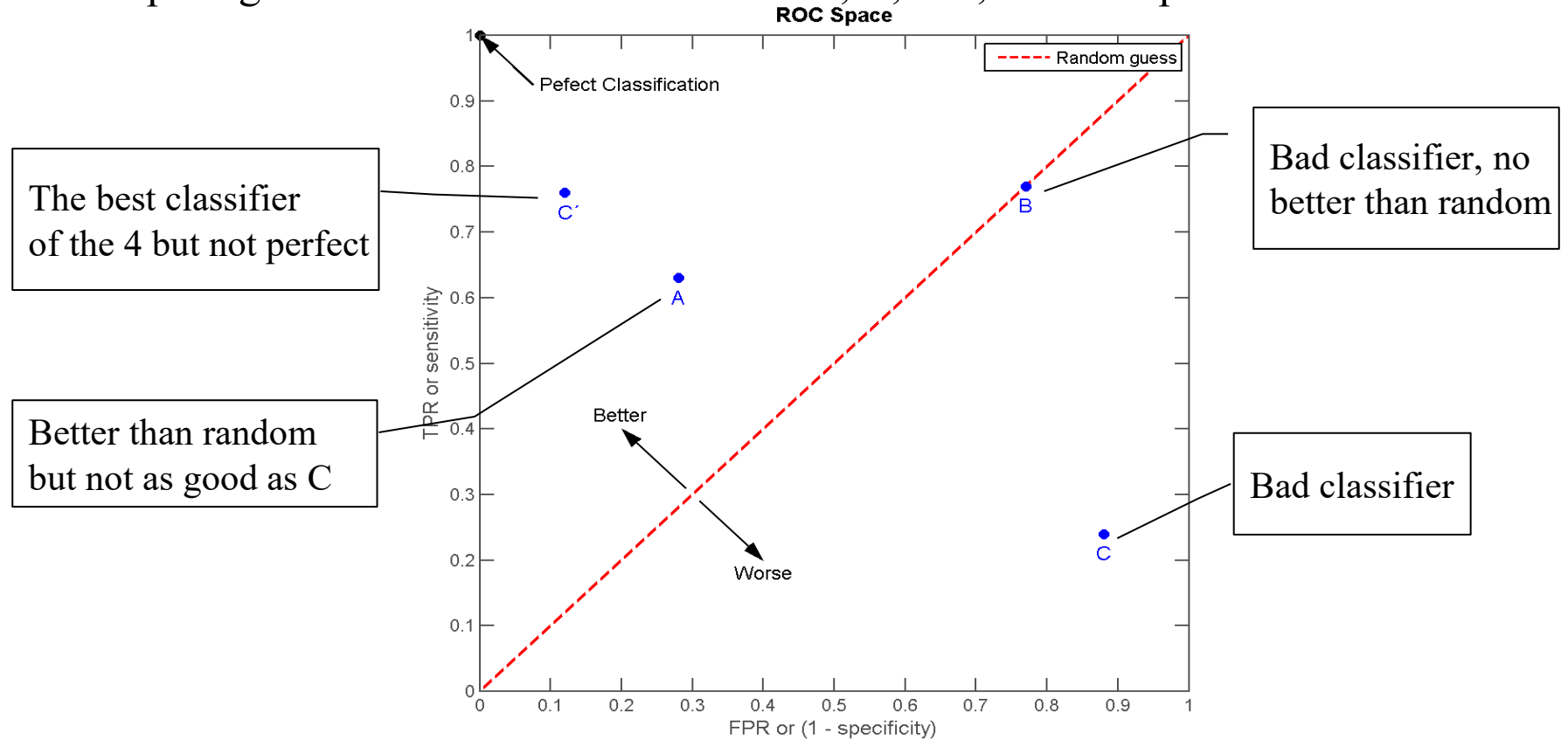


$$\text{True Positive Rate: } TPR = \frac{TP}{TP + FN} = \frac{2}{2 + 1} = 0.67$$

$$\text{False Positive Rate: } FPR = \frac{FP}{FP + TN} = \frac{3}{3 + 0} = 1.0$$

ROC

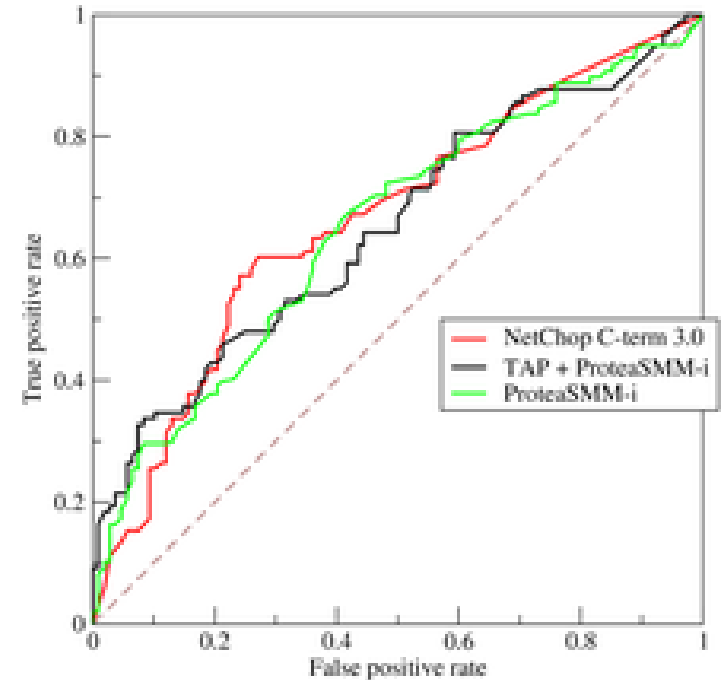
Comparing ROCs for different classifiers A, B, C..., ref. Wikipedia



https://en.wikipedia.org/wiki/Receiver_operating_characteristic

Comparing classifiers using ROC

- The TPR and FPR values on previous slides were calculated assuming a fixed classification confidence (usually 50%).
- ROC curves can be graphed at varying confidence ‘threshold’ levels for a single classifier
- This gives a more comprehensive comparison of several classifiers and is a way of visualising the ‘goodness’ of a classifier overall.



Classifier prediction confidence

Classifiers predict at varying levels of confidence.

Recall Naïve Bayes example:

```
> tbpredict.r = predict(tmodel, ptest, type = 'raw')
> tbpredict.r
```

		No	Yes
[1,]	9.932e-01	0.006812	
[2,]	3.381e-02	0.966191	
[3,]	1.531e-02	0.984686	
[4,]	2.916e-05	0.999971	
[5,]	1.822e-06	0.999998	
[6,]	9.558e-01	0.044248	

The ROC curve can show performance of the classifier at all levels (0 – 1) of confidence.

Example: Constructing ROC curve

- We have a list of several brands of chocolate bars. Chocolate experts have classified them into either: 1 = ‘excellent’ or 0 = ‘OK’ using various attributes such as packaging, cocoa content, smoothness and flavour.

Brand	Actual Class
Aero	0
Bounty	1
Cherry Ripe	0
Flake	1
Kitkat	0
Snickers	0
Violet Crumble	1

<http://fouryears.eu/2011/10/12/roc-area-under-the-curve-explained/>

Example: Constructing ROC curve

- We use a decision tree to calculate the probability (confidence) that the chocolate is excellent:

Brand	Actual Class	Confidence
Aero	0	0.3
Bounty	1	0.6
Cherry Ripe	0	0.4
Flake	1	0.8
Kitkat	0	0.4
Snickers	0	0.7
Violet Crumble	1	1.0

- To be more than 50% sure the chocolate is excellent:

Brand	Actual Class	Confidence	Pred Val
Aero	0	0.3	0
Bounty	1	0.6	1
Cherry Ripe	0	0.4	0
Flake	1	0.8	1
Kitkat	0	0.4	0
Snickers	0	0.7	1
Violet Crumble	1	1.0	1

Example: Constructing ROC curve

- Constructing a confusion matrix at 50% confidence:

Brand	Actual Class	Confidence	Predicted Class
Aero	0	0.3	0
Bounty	1	0.6	1
Cherry Ripe	0	0.4	0
Flake	1	0.8	1
Kitkat	0	0.4	0
Snickers	0	0.7	1
Violet Crumble	1	1.0	1

	Predicted Class Labels		
Actual Class		Class 1	Class 0
Class	Class 1	TP	FN
Labels	Class 0	FP	TN
TPR		1.00	
FPR			0.25

$$TPR = \frac{TP}{TP + FN} \quad FPR = \frac{FP}{FP + TN}$$

- Thus, to predict any any chocolate bar to be Class 1 at a confidence level of 50% or above results in 100% of the ‘excellent’ choc bars being correctly classified and 25% of the ‘OK’ chocolate bars being misclassified.
- But what about other confidence levels...?*

Example: Constructing ROC curve

- Now do at each threshold (confidence level):

Brand	Actual Class	Confidence	$C \geq 0.3$	$C \geq 0.4$	$C \geq 0.6$	$C \geq 0.7$	$C \geq 0.8$	$C \geq 1.0$
Aero	0	0.3						
Cherry Ripe	0	0.4						
Kitkat	0	0.4						
Bounty	1	0.6						
Snickers	0	0.7						
Flake	1	0.8						
Violet Crumble	1	1.0						

$C \geq 0.3$	Predicted Class Labels			$C \geq 0.4$	Predicted Class Labels			$C \geq 0.6$	Predicted Class Labels		
Actual		Class 1	Class 0	Actual		Class 1	Class 0	Actual		Class 1	Class 0
Class	Class 1			Class	Class 1			Class	Class 1		
Labels	Class 0			Labels	Class 0			Labels	Class 0		
TPR	_____	FPR	_____	TPR	_____	FPR	_____	TPR	_____	FPR	_____
$C \geq 0.7$	Predicted Class Labels			$C \geq 0.8$	Predicted Class Labels			$C \geq 1.0$	Predicted Class Labels		
Actual		Class 1	Class 0	Actual		Class 1	Class 0	Actual		Class 1	Class 0
Class	Class 1			Class	Class 1			Class	Class 1		
Labels	Class 0			Labels	Class 0			Labels	Class 0		
TPR	_____	FPR	_____	TPR	_____	FPR	_____	TPR	_____	FPR	_____

Example: Constructing ROC curve

- Now do at each threshold (confidence level):

Brand	Actual Class	Confidence	$C \geq 0.3$	$C \geq 0.4$	$C \geq 0.6$	$C \geq 0.7$	$C \geq 0.8$	$C \geq 1.0$
Aero	0	0.3						
Cherry Ripe	0	0.4						
Kitkat	0	0.4						
Bounty	1	0.6						
Snickers	0	0.7						
Flake	1	0.8						
Violet Crumble	1	1.0						

$C \geq 0.3$	Predicted Class Labels			$C \geq 0.4$	Predicted Class Labels			$C \geq 0.6$	Predicted Class Labels		
Actual		Class 1	Class 0	Actual		Class 1	Class 0	Actual		Class 1	Class 0
Class	Class 1			Class	Class 1			Class	Class 1		
Labels	Class 0			Labels	Class 0			Labels	Class 0		
TPR		FPR		TPR		FPR		TPR		FPR	
$C \geq 0.7$	Predicted Class Labels			$C \geq 0.8$	Predicted Class Labels			$C \geq 1.0$	Predicted Class Labels		
Actual		Class 1	Class 0	Actual		Class 1	Class 0	Actual		Class 1	Class 0
Class	Class 1			Class	Class 1			Class	Class 1		
Labels	Class 0			Labels	Class 0			Labels	Class 0		
TPR		FPR		TPR		FPR		TPR		FPR	

Example: Constructing ROC curve

- Now do at each threshold (confidence level):

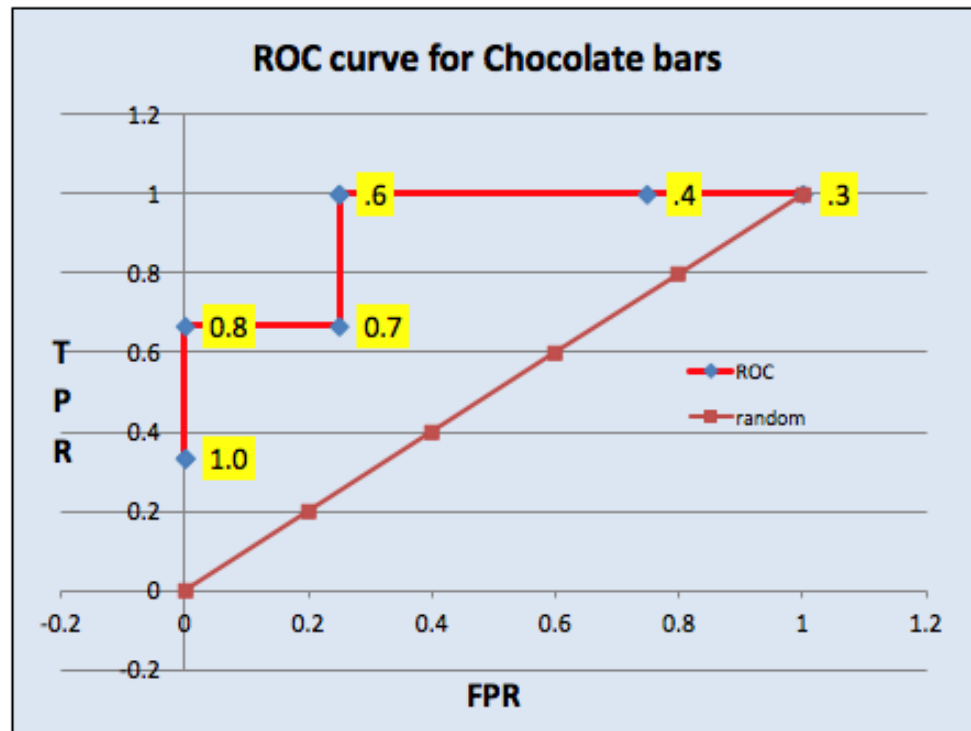
Brand	Actual Class	Confidence	$C \geq 0.3$	$C \geq 0.4$	$C \geq 0.6$	$C \geq 0.7$	$C \geq 0.8$	$C \geq 1.0$
Aero	0	0.3	1	0	0	0	0	0
Cherry Ripe	0	0.4	1	1	0	0	0	0
Kitkat	0	0.4	1	1	0	0	0	0
Bounty	1	0.6	1	1	1	0	0	0
Snickers	0	0.7	1	1	1	1	0	0
Flake	1	0.8	1	1	1	1	1	0
Violet Crumble	1	1.0	1	1	1	1	1	1

$C \geq 0.3$	Predicted Class Labels			$C \geq 0.4$	Predicted Class Labels			$C \geq 0.6$	Predicted Class Labels		
Actual		Class 1	Class 0	Actual		Class 1	Class 0	Actual		Class 1	Class 0
Class	Class 1	3	0	Class	Class 1	3	0	Class	Class 1	3	0
Labels	Class 0	4	0	Labels	Class 0	3	1	Labels	Class 0	1	3
TPR	1.00	FPR	1.00	TPR	1.00	FPR	0.75	TPR	1.00	FPR	0.25
$C \geq 0.7$	Predicted Class Labels			$C \geq 0.8$	Predicted Class Labels			$C \geq 1.0$	Predicted Class Labels		
Actual		Class 1	Class 0	Actual		Class 1	Class 0	Actual		Class 1	Class 0
Class	Class 1	2	1	Class	Class 1	2	1	Class	Class 1	1	2
Labels	Class 0	1	3	Labels	Class 0	0	4	Labels	Class 0	0	4
TPR	0.67	FPR	0.25	TPR	0.67	FPR	0.00	TPR	0.33	FPR	0.00

Example: Constructing ROC curve

- Plot all the TPR/FNR pairs to get the ROC curve

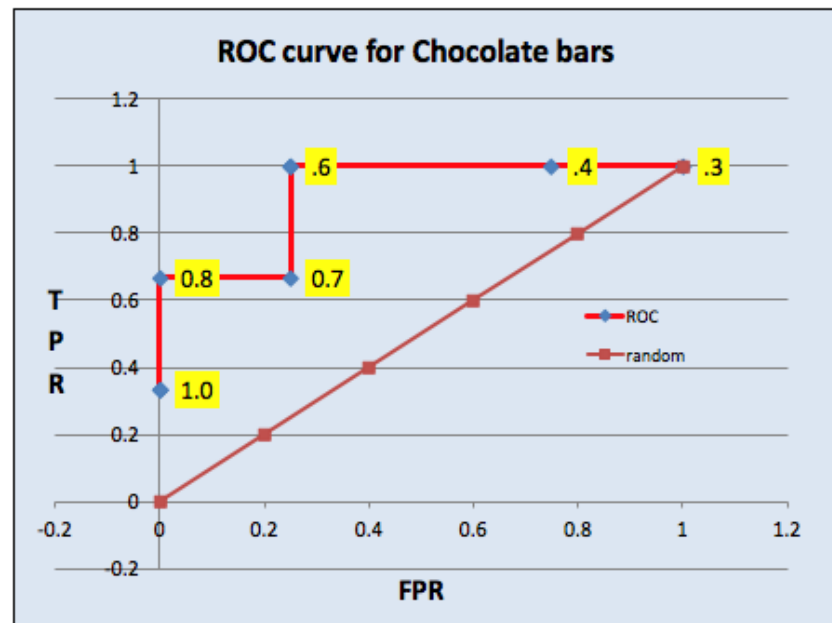
Threshold	FPR	TPR
1.0	0.00	0.33
0.8	0.00	0.67
0.7	0.25	0.67
0.6	0.25	1.00
0.4	0.75	1.00
0.3	1.00	1.00



■ Indicates Confidence

Example: Constructing ROC curve

- What does the ROC curve tell us?
- Correct classifications and ‘*false alarm*’ rates vary with threshold value.
- This classifier is better than a ‘random’ classifier.
- If we don’t want any false positives, then we must use a confidence level equal to or greater than 0.8.



ROC curve interpretation

(TPR,FPR)

- (0,0): Declare everything to be negative class
- (1,1): If declare everything to be positive class

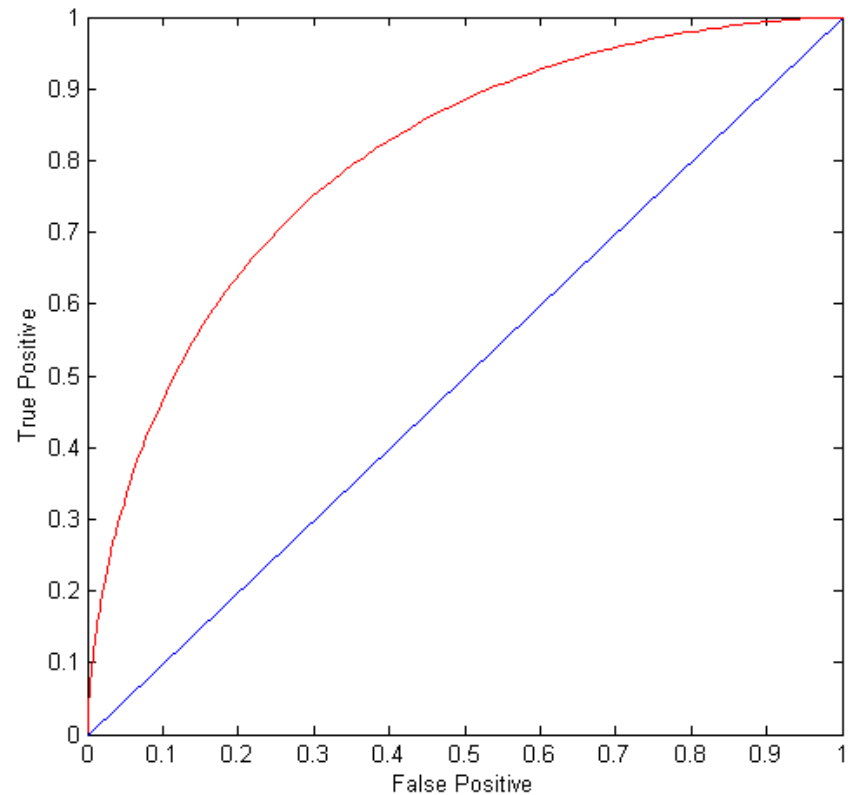
- (1,0): Ideal

Diagonal line:

- Random guessing

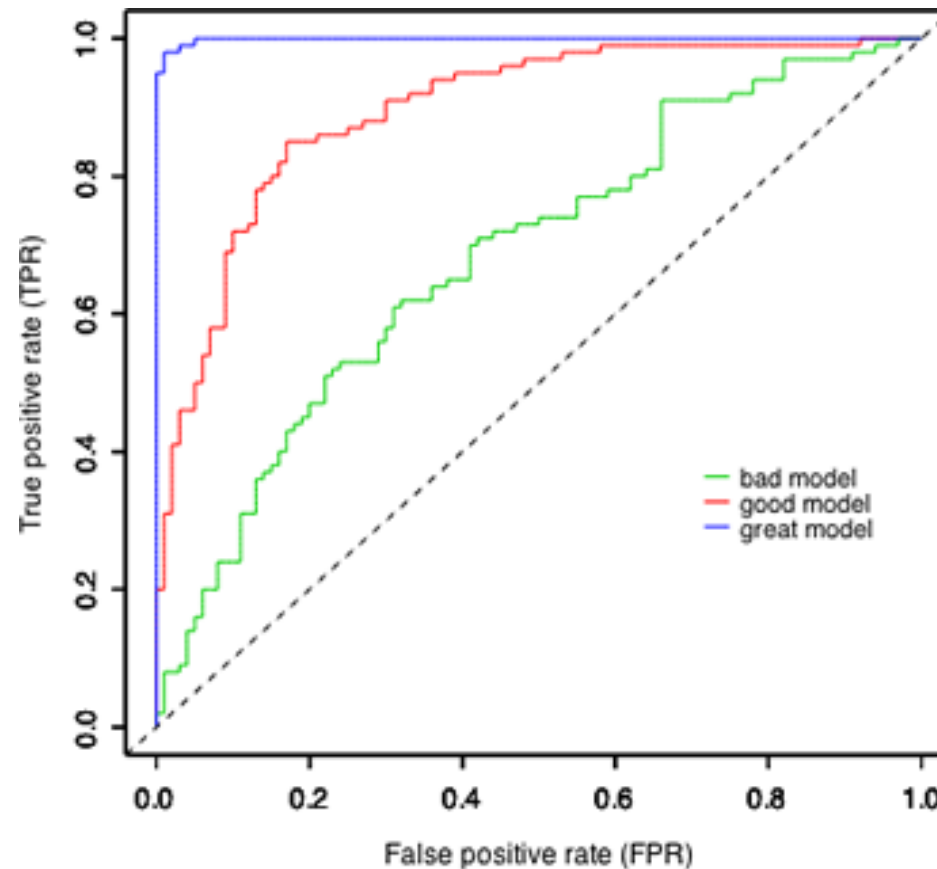
Below diagonal line:

- Prediction is opposite of the true class (worse than guessing).



Using ROC for Model Comparison

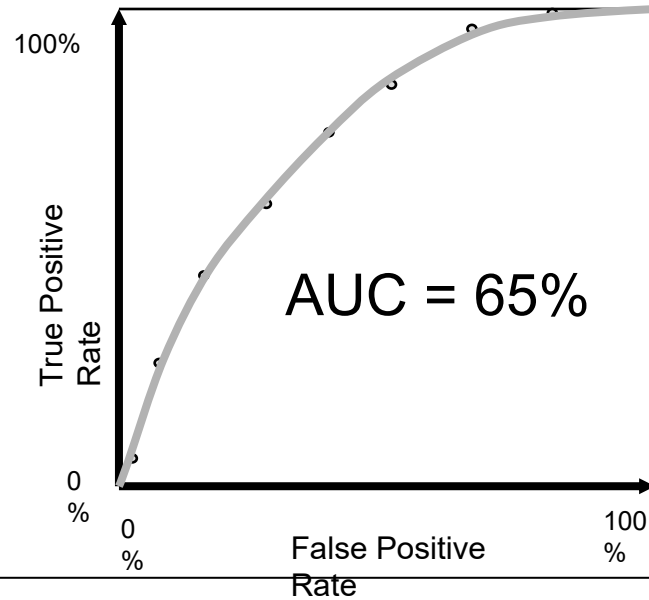
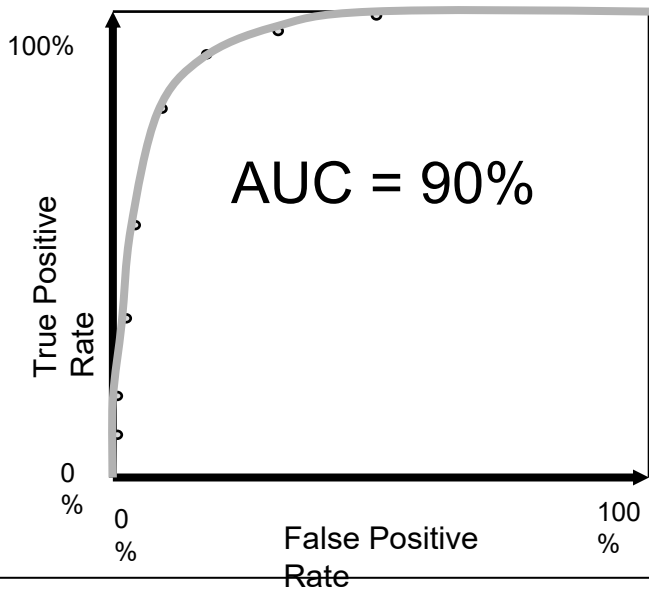
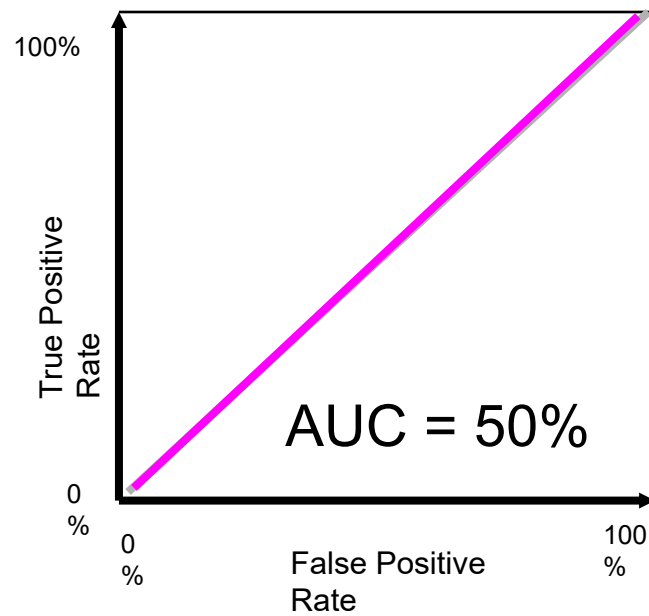
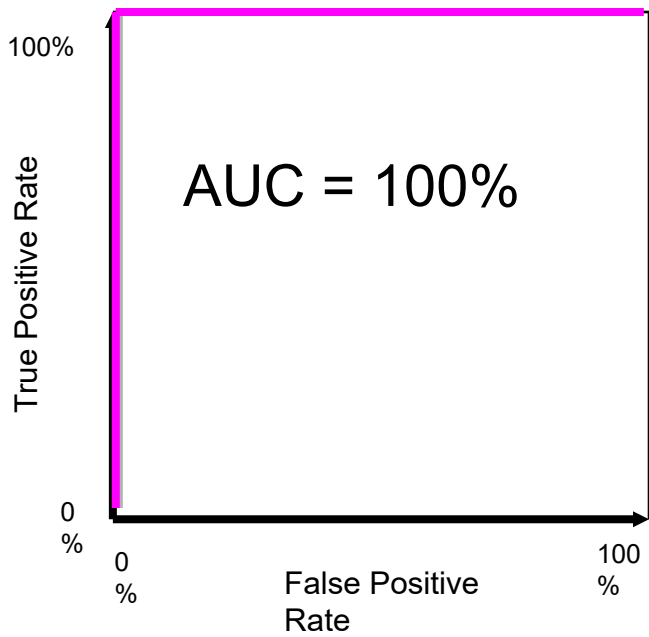
ROC curves for three different classifiers



Area under ROC curve (AUC)

For binary data:

- Overall single measure of test performance
- Comparisons between two tests based on differences between (estimated) AUC
- For continuous data, AUC is equivalent to Mann-Whitney U-statistic
- Some examples of AUC on following slide:



An introduction to ROC analysis.

An ROC curve is a two-dimensional depiction of classifier performance. To compare classifiers we may want to reduce ROC performance to a single scalar value representing expected performance. A common method is to calculate the area under the ROC curve, abbreviated AUC. Since the AUC is a portion of the area of the unit square, its value will always be between 0 and 1.0. However, because random guessing produces the diagonal line between $(0, 0)$ and $(1, 1)$, which has an area of 0.5, no realistic classifier should have an AUC less than 0.5. The AUC has an important statistical property: the AUC of a classifier is equivalent to the probability that the classifier will rank a randomly chosen positive instance higher than a randomly chosen negative instance.

From: Fawcett, T. An introduction to ROC analysis.

ROC in practice

Smartwatch data help detect COVID-19

Data from consumer smartwatches can improve the detection of COVID-19 when combined with symptom self-reporting, and can also detect the disease in pre-symptomatic individuals.

Tingting Zhu, Peter Watkinson and David A. Clifton

Tests for the detection of COVID-19 are typically time consuming, costly and require professional expertise. Improving the frequency, ease and ubiquity of testing for COVID-19 is urgent, particularly when a substantial proportion of patients (40–45%; ref. ¹) may be pre-symptomatic or asymptomatic. Obtaining longitudinal physiological data via commonplace wearable devices², typically worn on the wrist, may offer a convenient means of detection. Self-reported symptoms can be used to construct relatively simple models for the identification of COVID-19 (ref. ³), and data from wearables may similarly be used to identify viral respiratory illnesses^{4,5}. Reporting in *Nature Medicine*, Giorgio Quer and colleagues now show how smartwatch data can be used in conjunction with self-reported symptoms to determine whether an individual has COVID-19 after the onset of symptoms⁶. And in *Nature Biomedical Engineering*,

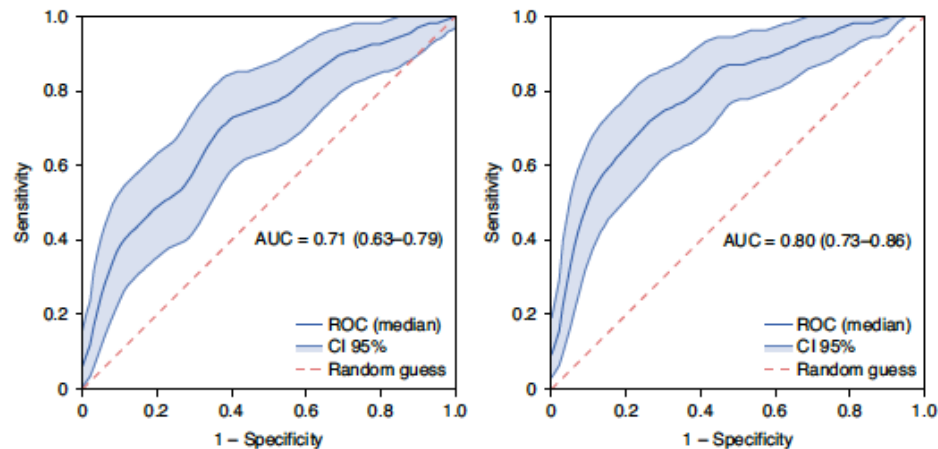










Fig. 1 | Prediction of COVID-19 from self-reported symptoms, and from self-reported symptoms combined with RHR, sleep and activity data from smartwatches. The receiver operating characteristic (ROC) curves for the discrimination of 54 individuals who tested positive for COVID-19 and 279 individuals who tested negative for the disease show an AUC of 0.71 for the symptom-based model (left) and of 0.80 for the model using symptoms and smartwatch data (right). CI, 95% confidence interval. Figure reproduced with permission from ref. ⁶, Springer Nature Ltd.

<https://www.nature.com/articles/s41551-020-00659-9.pdf>

ROC in practice

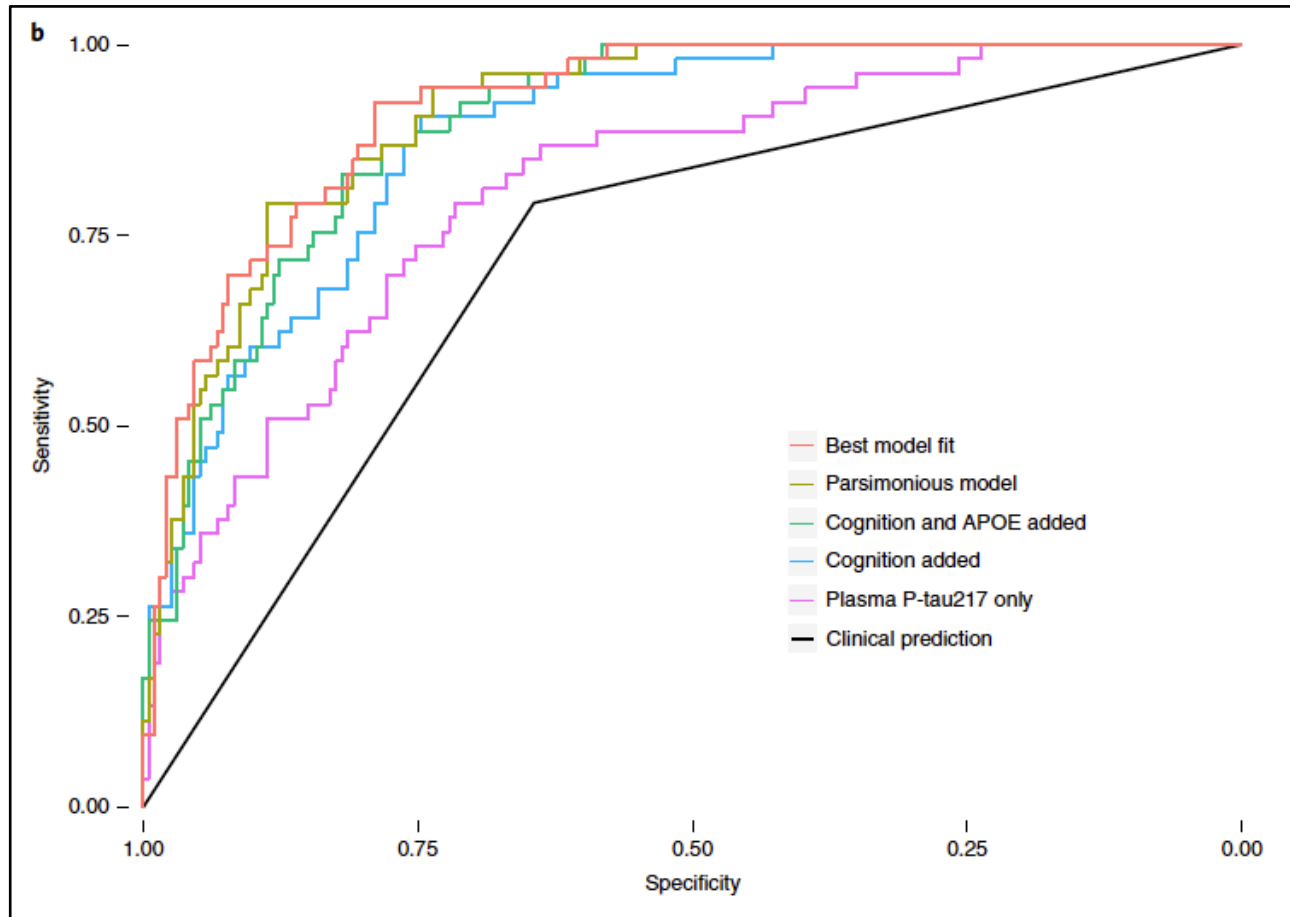
Prediction of future Alzheimer's disease dementia using plasma phospho-tau combined with other accessible measures

Sebastian Palmqvist ^{1,2} , Pontus Tideman^{1,2}, Nicholas Cullen¹, Henrik Zetterberg ^{3,4,5,6}, Kaj Blennow ^{3,4}, the Alzheimer's Disease Neuroimaging Initiative*, Jeffery L. Dage ⁷, Erik Stomrud^{1,2}, Shorena Janelidze ¹, Niklas Mattsson-Carlgren^{1,8,9} and Oskar Hansson ^{1,2} 

A combination of plasma phospho-tau (P-tau) and other accessible biomarkers might provide accurate prediction about the risk of developing Alzheimer's disease (AD) dementia. We examined this in participants with subjective cognitive decline and mild cognitive impairment from the BioFINDER ($n = 340$) and Alzheimer's Disease Neuroimaging Initiative (ADNI) ($n = 543$) studies. Plasma P-tau, plasma A β 42/A β 40, plasma neurofilament light, APOE genotype, brief cognitive tests and an AD-specific magnetic resonance imaging measure were examined using progression to AD as outcome. Within 4 years, plasma P-tau217 predicted AD accurately (area under the curve (AUC) = 0.83) in BioFINDER. Combining plasma P-tau217, memory, executive function and APOE produced higher accuracy (AUC = 0.91, $P < 0.001$). In ADNI, this model had similar AUC (0.90) using plasma P-tau181 instead of P-tau217. The model was implemented online for prediction of the individual probability of progressing to AD. Within 2 and 6 years, similar models had AUCs of 0.90-0.91 in both cohorts. Using cerebrospinal fluid P-tau, A β 42/A β 40 and neurofilament light instead of plasma biomarkers did not improve the accuracy significantly. The clinical predictions by memory clinic physicians had significantly lower accuracy (4-year AUC = 0.71). In summary, plasma P-tau, in combination with brief cognitive tests and APOE genotyping, might greatly improve the diagnostic prediction of AD and facilitate recruitment for AD trials.

<https://www.nature.com/articles/s41591-021-01348-z>

ROC in practice



<https://www.nature.com/articles/s41591-021-01348-z>

ROC analysis in R

The package “ROCR” is designed to calculate and display performance measures of classifiers.

For the chocolate bars example:

- > `install.packages("ROCR")`
- > `library(ROCR)`
- > `pconfidence = c(0.3, 0.6, 0.4, 0.8, 0.4, 0.7, 1)`
- > `plabels = c(0, 1, 0, 1, 0, 0, 1)`
- > `# transform the inputs into a prediction object`
- > `cpred <- prediction(pconfidence, plabels)`

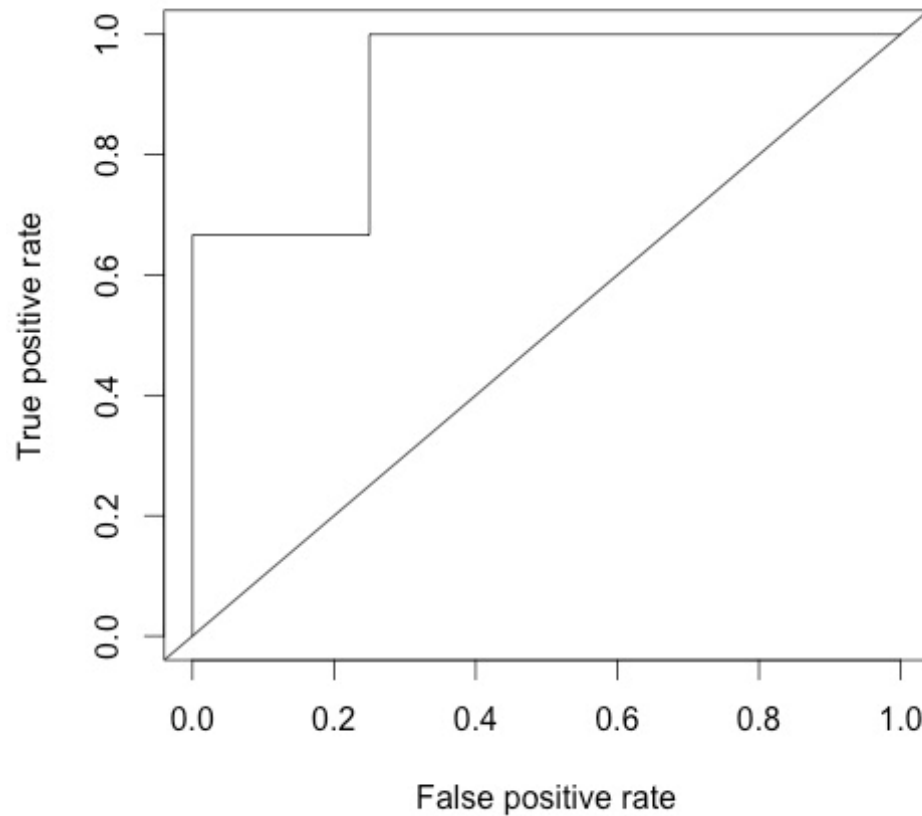
ROC analysis in R

Calculate performance measures and plot:

```
> # calculate the performance functions
> cperf <- performance(cpred,"tpr","fpr")
> plot(cperf)
> abline(0,1)
> # calculate and print auc
> cauc = performance(cpred, "auc")
> print(as.numeric(cauc@y.values))
      [1] 0.9167
```

ROC analysis in R

ROC plot showing $x = y$ line:



ROC analysis in R

Looking at the Naïve Bayes predictions:

```
> pttrain <- read.csv("playtennistrain.csv")
> pttest <- read.csv("playtennistest.csv")
> tmodel = naiveBayes(Play ~. -Day, data = pttrain)
> # outputs as confidence levels
> tbpredict.r = predict(tmodel, pttest, type = 'raw')
> tpred <- prediction( tbpredict.r[,2], pttest$Play)
> tperf <- performance(tpred,"tpr","fpr")
> plot(tperf)
> abline(0,1)
```

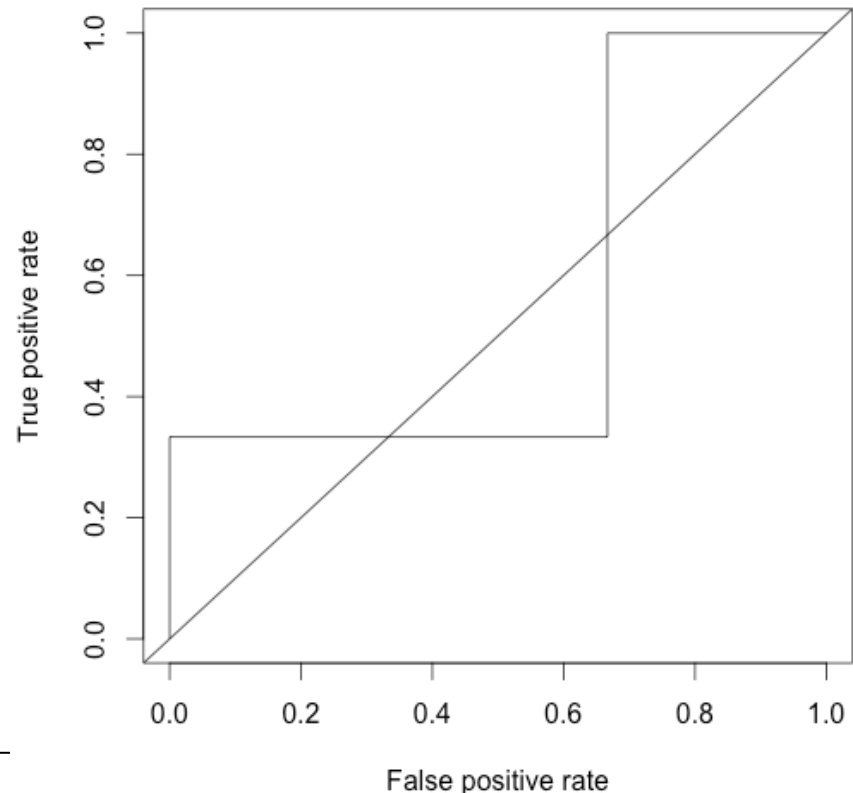
ROC analysis in R

Inputs to the calculations and ROC chart:

```
> inputs = cbind( tbpredict.r[,2], ptttest$Play)
```

```
> inputs
```

	[,1]	[,2]
[1,]	0.006812	1
[2,]	0.966191	2
[3,]	0.984686	1
[4,]	0.999971	1
[5,]	0.999998	2
[6,]	0.044248	2



Lift

For binary classification and prediction models:

- Lift is a measure of the effectiveness of a predictive model calculated as the ratio between the results obtained with and without the predictive model.
- Lift charts are visual aids for measuring model performance.
- Lift factor = success rate with model / success rate without model.

Lift

Scenario:

- Suppose you have a model that outputs the probability of predicted outcomes (e.g. decision tree, naïve Bayes’).
- Your job is to find subsets of instances that have a high proportion of positive instances, higher than in the test set as a whole.
- To do this you can sort instances in descending order of predicted confidence (probability of positive class).
- You can select a sample of a given size having the greatest possible proportion of positive instances.
- The lift factor is calculated as the success proportion for the sample divided by the success proportion for the complete set.

Lift

- For example, if you have 150 instances, of which 50 are positive, an overall success rate of 33.33% if selected randomly.

instance	1	2	3	4	5	6		147	148	149	150
confidence of +	0.37	0.99	0.21	0.70	0.11	0.97	...	0.37	0.56	0.96	0.60
actual value	-	+	-	-	-	+		-	+	-	+

- Sorting the instances in terms of their predicted confidence gives the following top 20 entries:

Confidence of +	0.95	0.94	0.93	0.92	0.9	0.89	0.87	0.84	0.83	0.81	0.8	0.78	0.77	0.76	0.75	0.72	0.71	0.7	0.69	0.66
Order of confidence	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
Actual value	+	+	-	-	+	+	+	+	-	+	-	+	+	-	+	-	+	-	+	+

Lift

Confidence of +	0.95	0.94	0.93	0.92	0.9	0.89	0.87	0.84	0.83	0.81	0.8	0.78	0.77	0.76	0.75	0.72	0.71	0.7	0.69	0.66
Order of confidence	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
Actual value	+	+	-	-	+	+	+	+	-	+	-	+	+	-	+	-	+	-	+	+

- Choosing a sample size of 10 give a lift of $(7/10) / (50/150) = 2.1$
- This means that if you use your model and contact the ‘best’ 10 instances, then your success rate will be 2.1 times higher than if you selected 10 instances at random
- In case you didn’t the have the true class labels, were seeking the most promising sample of size 10, your safest bet is to select top 10 ranking instances as a result of your classifier.

Lift: example



For the chocolate bars example:

- The probability of any chocolate bar chosen at random being “good” is $3/7$.
- If we just sample the top 2 (28%) we’re most confident of then the probability of “good” is $2/2$. What is the lift?

$$Lift = \frac{\binom{2}{2}}{\binom{3}{7}} = \frac{1}{3} = 2.33 \text{ or } 233\%$$

- For top 3 (42%), $P(\text{good}) = 2/3$

$$Lift = \frac{\binom{2}{3}}{\binom{3}{7}} = \frac{14}{9} = 1.56 \text{ or } 156\%$$

Brand	Actual Class	Conf
Aero	0	0.3
Cherry Ripe	0	0.4
Kitkat	0	0.4
Bounty	1	0.6
Snickers	0	0.7
Flake	1	0.8
Violet Crumble	1	1.0

Lift analysis in R



To calculate and plot lift using chocolate bar data from previous example:

```
> clift = performance(cpred, "lift") #ROCR package
```

```
> plot(clift)
```

```
> print(clift@y.values)
```

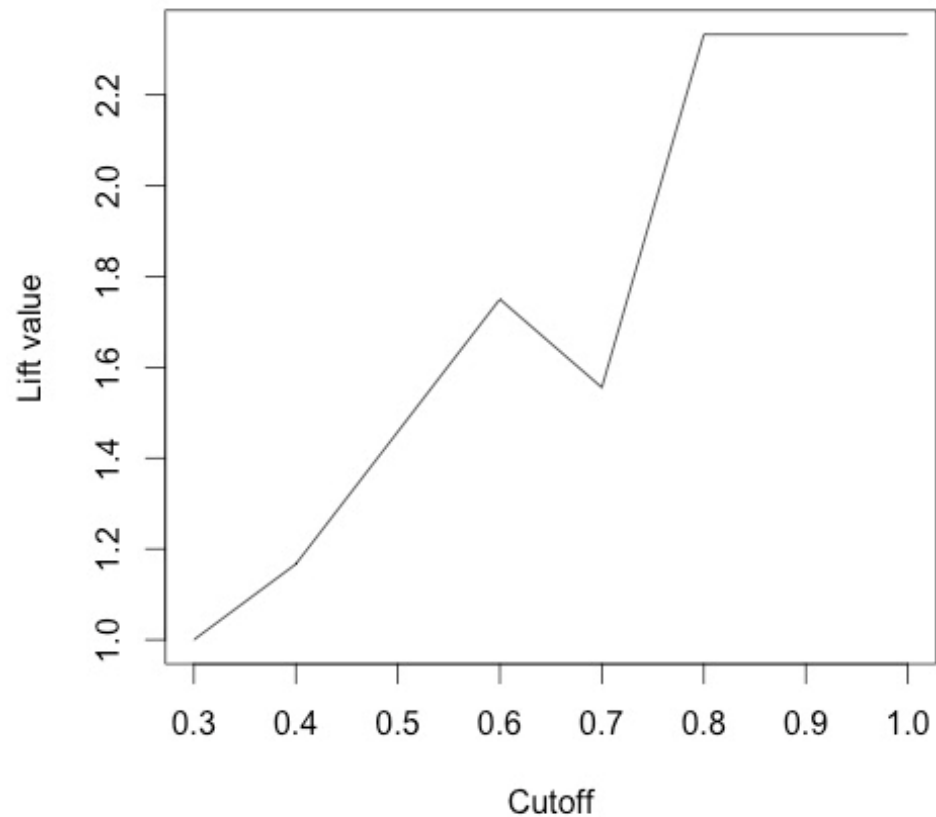
```
[[1]]
```

```
[1] NaN 2.333 2.333 1.556 1.750 1.167 1.000
```

Lift analysis in R



Lift chart:



References for this lecture

- R packages: “tree”, “e1071” and “ROCR” and reference manuals.
- Fawcett, An introduction to ROC analysis, Pattern Recognition Letters, 27, 2009 861 – 874.
- James et al., An Introduction to Statistical Learning with Applications in R, 2nd Ed. Springer. Chapter 8.
- Tan, Steinbach & Kumar.
- Provost & Fawcett, Data Science for Business.

Notes on the presentation

This presentation contains slides created to accompany:
Introduction to Data Mining, Tan, Steinbach, Kumar.
Pearson Education Inc., 2006.

This presentation includes material originally created by
Dr. Sue Bedingfield.