FIT3152 Data analytics– Lecture 8

Classification continued:

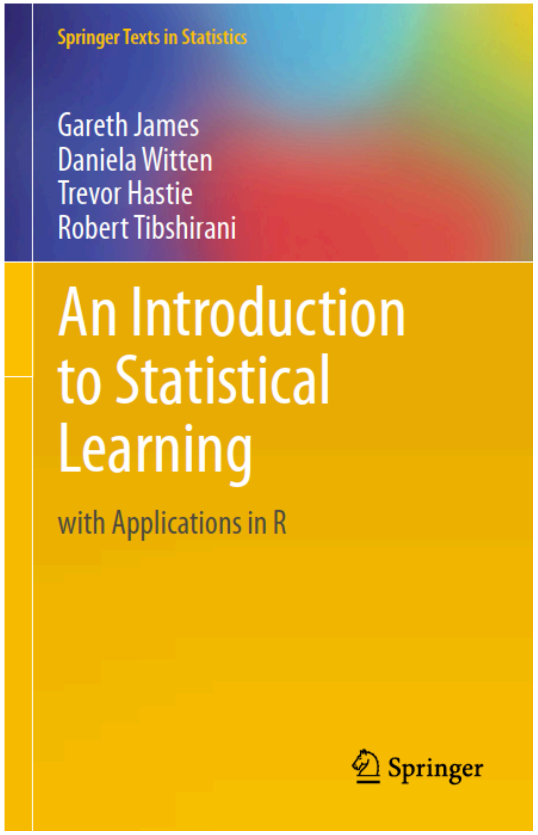
- Improving the basic decision tree: pruning and cross-validation,
- Naïve Bayes classification,
- Classifier model evaluation: ROC curves,
- Lift.

References for this lecture

- R packages: tree, e1071, ROCR.
- I have used the reference manual for each package. Many class examples have been drawn from them.
- 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. Springer. Chapter 8.
- Tan, Steinbach & Kumar.
- Provost & Fawcett, Data Science for Business.

…

James et al., An Introduction to Statistical Learning with Applications in R. Springer. Chapter 8. Access for free via the Monash Library.



Week-by-week

Week Starting	Lecture	Topic	Tutorial	A1	A2
2/3/21	1	Intro to Data Science, review of basic statistics using R	...		
9/3/21	2	Exploring data using graphics in R	T1		
16/3/21	3	Data manipulation in R	T2	Released	
23/3/21	4	Data Science methodologies, dirty/clean/tidy data, data manipulation	T3		
30/3/21	5	Network analysis	T4		
6/4/21		Mid-semester Break			
13/4/21	6	Regression modelling	T5		
20/4/21	7	Classification using decision trees	T6	Submitted	
27/4/21	8	Naïve Bayes, evaluating classifiers	T7		Released
4/5/21	9	Ensemble methods, artificial neural networks	T8		
11/5/21	10	Clustering	T9		
18/5/21	11	Text analysis	T10		Submitted
25/5/21	12	Review of course, Exam preparation	T11		

SETU

Student Evaluation of Teaching and Units (SETU) has opened for Semester 1.

- All students are encouraged to participate. Your feedback is very important.
- You will see a block in Moodle linking you to the survey.
- There are 100, $50 vouchers for students to win.
- The University will email students a weekly reminder with links to the survey.

Assignment 2 has been released!

The objective of this assignment is to gain familiarity with classification models using R.

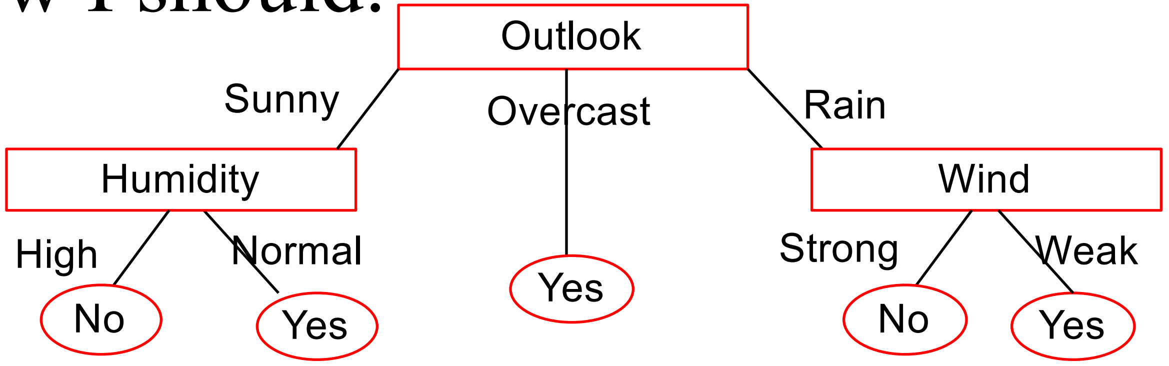
- You will be using a modified version some Kaggle competition data, to predict cloud cover in Australia.
- The data contains a number of meteorological observations as attributes, and the class attribute “CloudTomorrow”.
- Parts 1 – 7 will be familiar from tutorials.
- Parts 8 – 11 are a bit more challenging and will require some independent learning and initiative.

Quick revision from last week:

Please put your answers in the Zoom chat…

Question 1

Outlook = Sunny, Temp = Mild, Humidity = Normal, Wind = Strong. Based on the decision tree below I should:

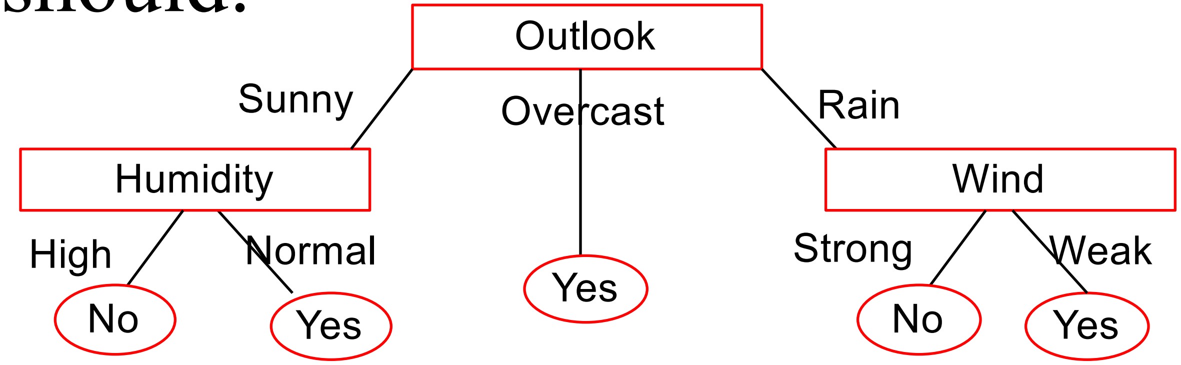


a. Play

b. Not Play

Question 2

Outlook = Sunny, Temp = Mild, Humidity = Normal, Wind = Weak. Based on the decision tree below I should:



a. Play

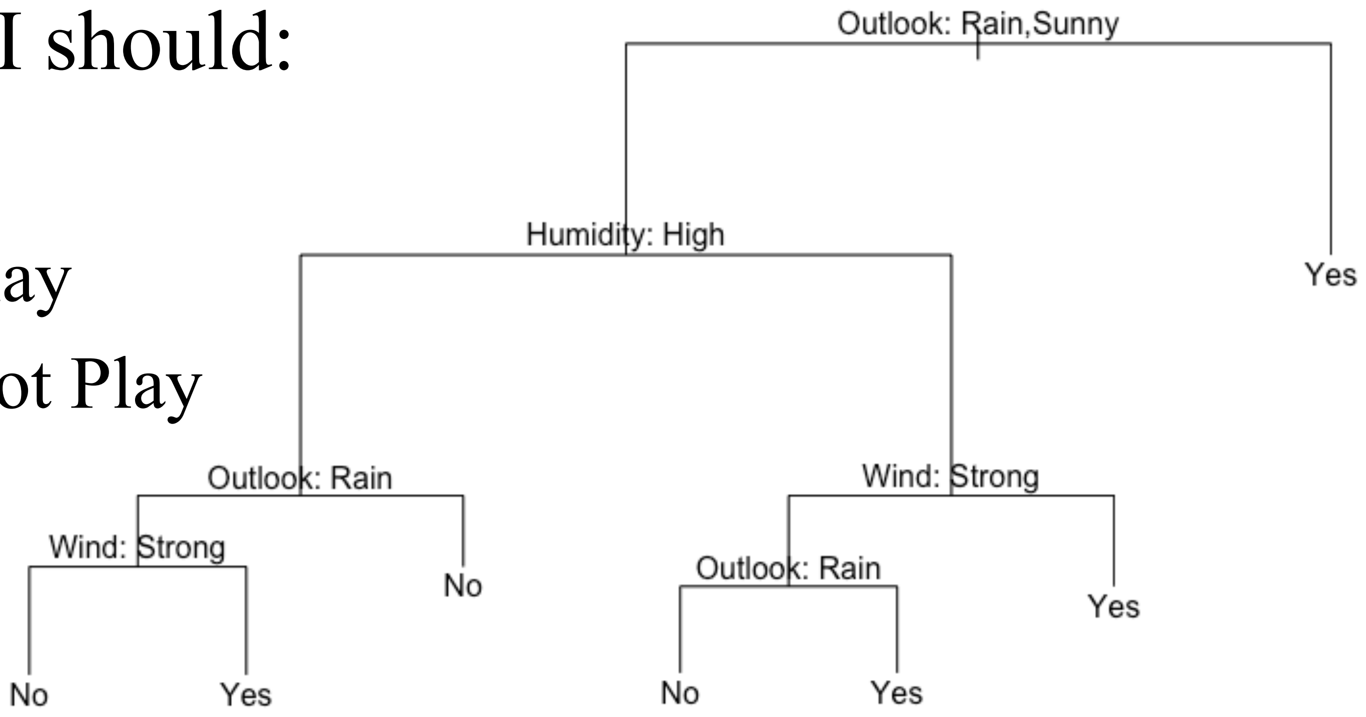
b. Not Play

Question 3

Outlook = Overcast, Temp = Mild, Humidity = Normal, Wind = Weak. Based on the decision tree below I should:

a. Play

b. Not Play



Question 4

Based on this confusion matrix, accuracy is:

a. 2

b. 4

c. $\frac{1}{6}$

d. $\frac{2}{6}$

e. $\frac{3}{6}$

f. $\frac{4}{6}$

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

Improving the decision tree

Performance metrics

Tree size and accuracy

Pruning

Cross validation

Examples in R

Pros and cons of decision trees

Pros:

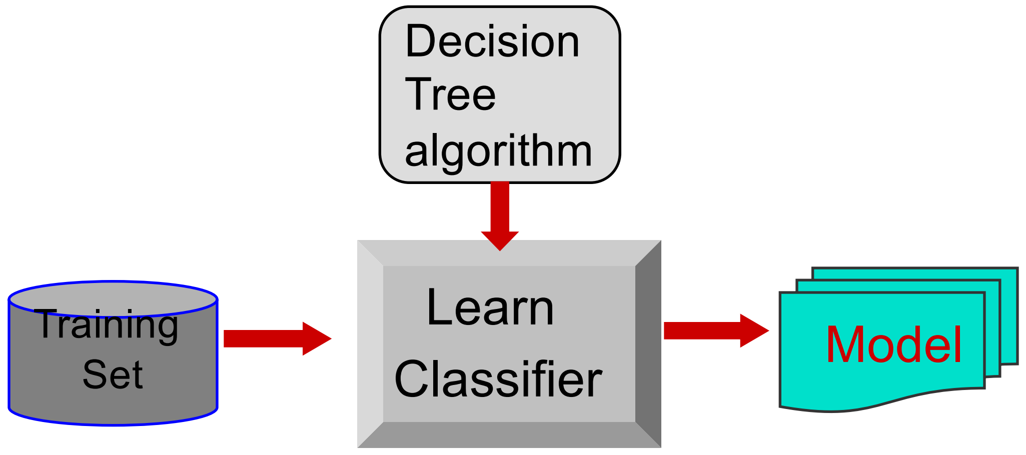
- Easy to interpret – explicit rules plus graphical representation
- Can handle mixed input (discrete and continuous)
- Robust - to outliers (noise), missing values
- Able to find the most discriminating attributes
- Accuracy good – in general

Cons:

- Unstable: small changes may lead to a completely different tree
- Can become overly complex

Review: classification algorithms

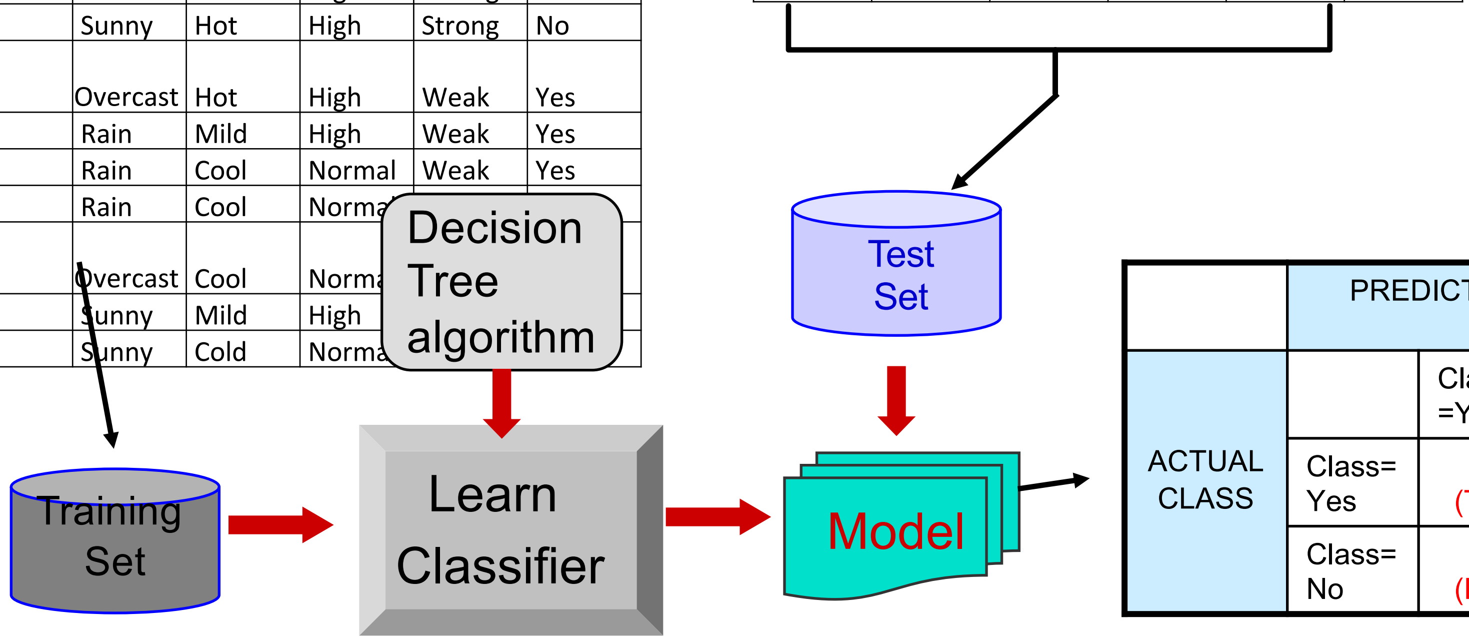
- Objective is to produce a model from a labelled data set.
- Model is used to label new, unlabelled cases.



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	Norma		
D7	Overcast	Cool	Norma		
D8	Sunny	Mild	High		
D9	Sunny	Cold	Norma		

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



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

Review: performance evaluation…

Summary of test results as a Confusion Matrix

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

Most commonly used metric:

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

Metrics - precision & recall

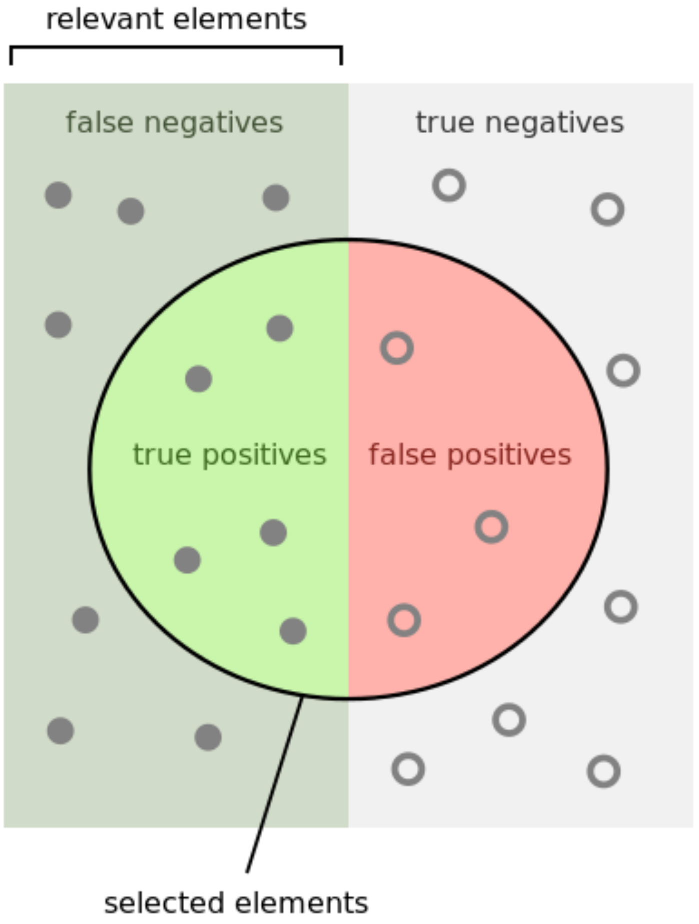
Precision is:

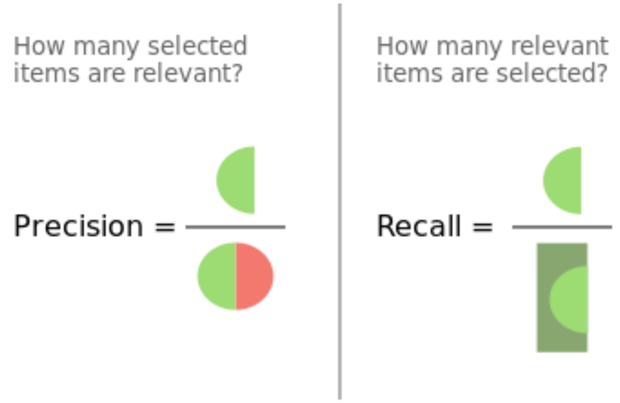
$$\frac{True\ Positives}{True\ Positives + False\ Positives}$$

Recall is:

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

Metrics - precision & recall




How many selected items are relevant?
Precision =
How many relevant items are selected?
Recall =

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

Review: performance evaluation…

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

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

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

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

Review: performance evaluation…

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

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

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

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

Over/underfitting

- Overfitting – model that does not generalise well
  - Model is excessively complex
  - Performs well on training data but not on unseen data
  - Low Recall
- Underfitting – model that is too simple
  - Model is too simple to give accurate labels
  - Performs poorly on both training and unseen data
  - Low Precision

Tree size

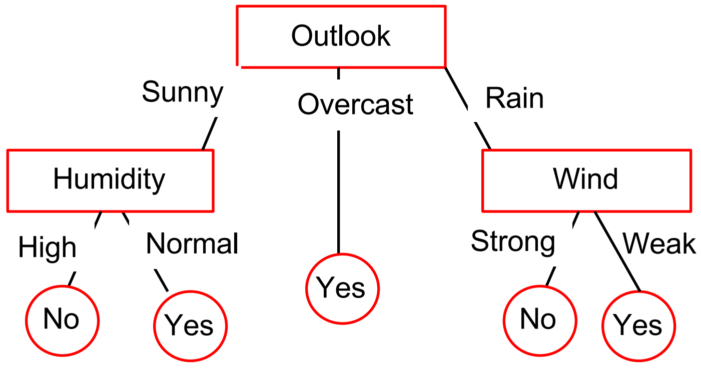
Leaf node count

- 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



Question:

Q: Is a tree with only pure leaves always the best classifier you can have?

A: No.

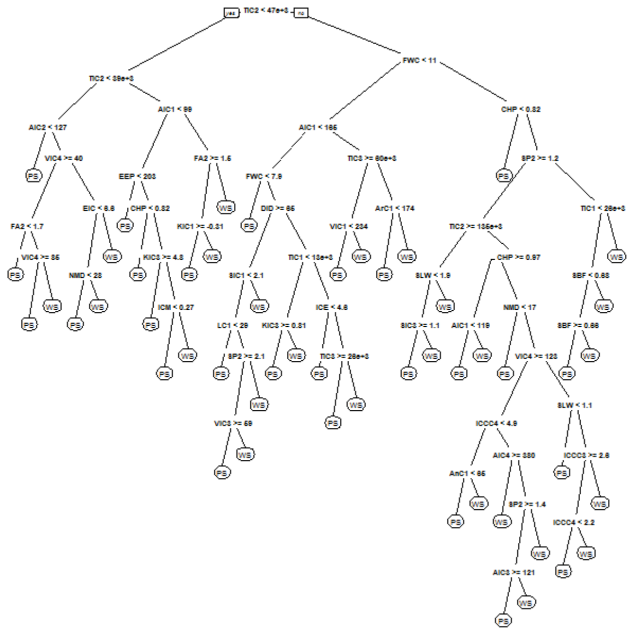
This tree is the best classifier on the training set, but possibly not on new and unseen data.

Because of *overfitting*, the tree may not generalize very well.

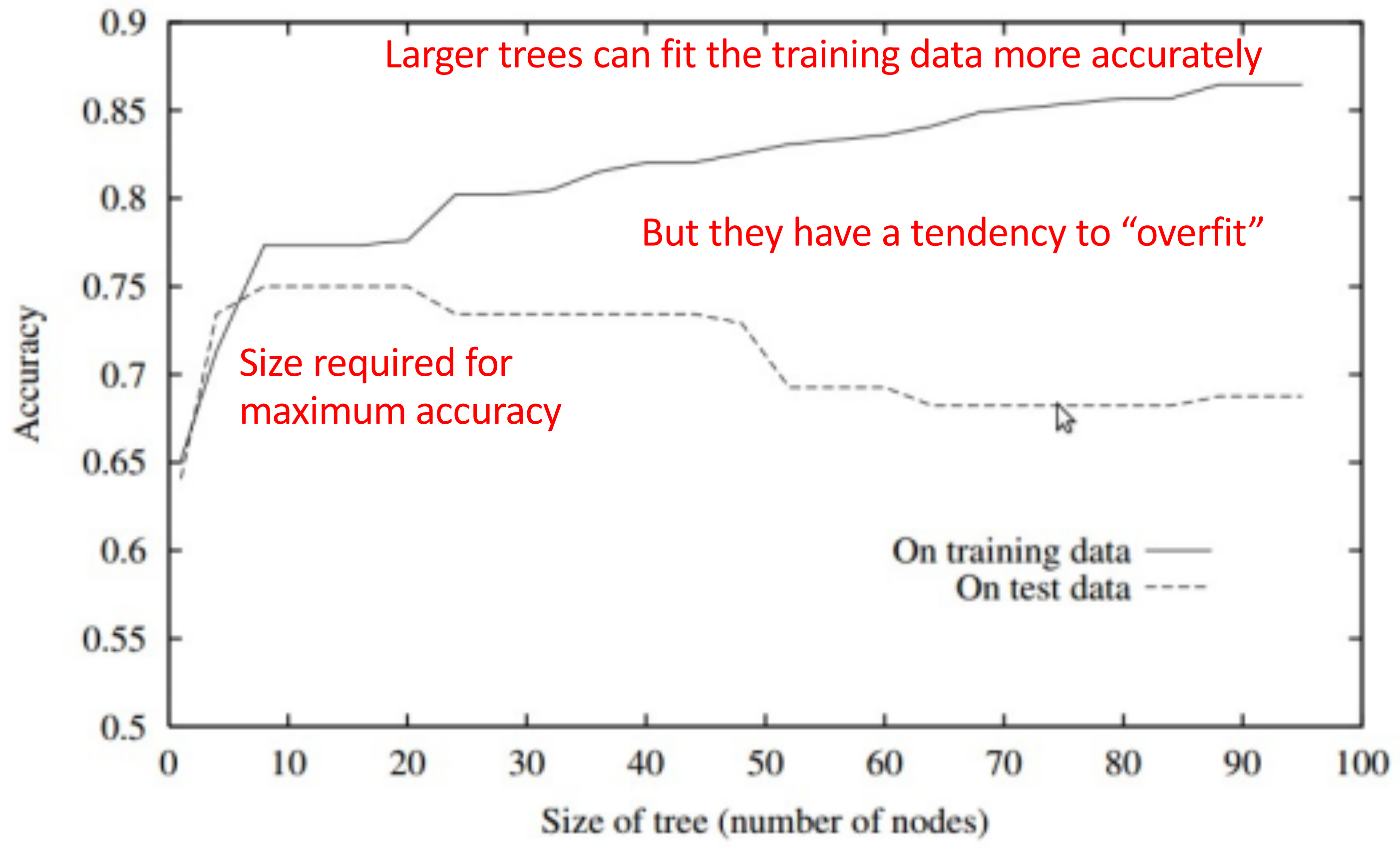
Over-fitting in decision trees

You can grow each branch of the tree deeply enough to perfectly classify the training examples.

But what about noise in the data, or simply irrelevant features?



Example: size of tree vs accuracy



To avoid over-fitting

Two approaches:

- Pre-pruning: stop growing the tree earlier, before it reaches the point where it over-fits on the training data.

- Post-pruning: allow the tree to overfit the data, and then post-prune the tree. (more effective in practice).

DT Post-pruning

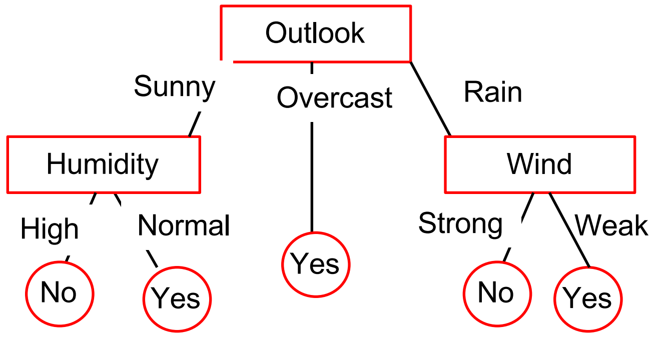
Post-pruning: the decision tree is first grown out as usual.

Depth of the tree is reduced by replacing sub-trees with leaf nodes.

The general concept is to remove rules if they have little or negligible effect on the error rate.

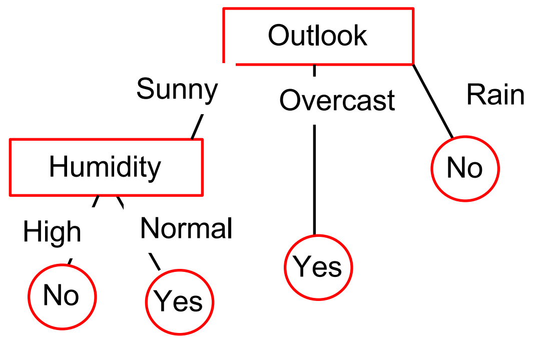
DT Post-pruning example

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



DT Post-pruning example

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



Cross validation

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

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

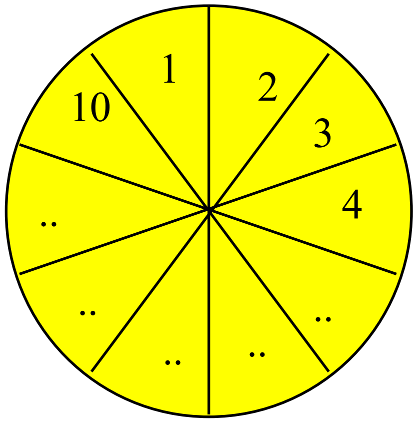
Therefore you always need distinct *training* and *testing* data sets.

Cross validation

k-fold Cross validation steps:

- Partition 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.



Cross validation cont...

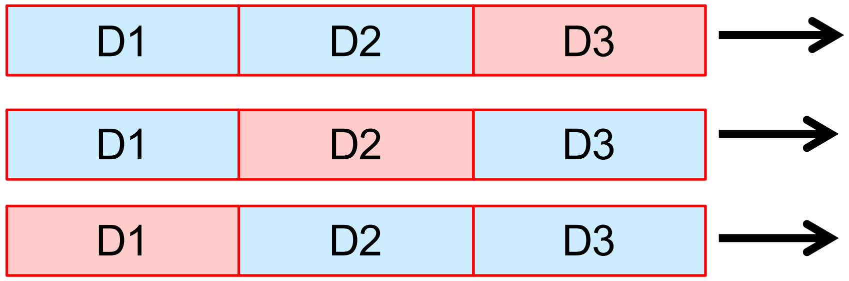
- Make sure that all folds have same distribution of classes.
- ‘Stratification’ (as in stratified sampling) ensures that classes are properly represented across partitions.
- Another variant is leave-one-out cross-validation. Each case is left out, and the model is trained on all the remaining instances. The results across all *n* samples are later averaged to get the final error estimate.

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.



D1 + D2 for Training, D3 for Test

D1 + D3 for Training, D2 for Test

D2 + D3 for Training, D1 for Test

Average performance across D1, D2 and D3.

Pruning the decision tree 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 the decision tree in R

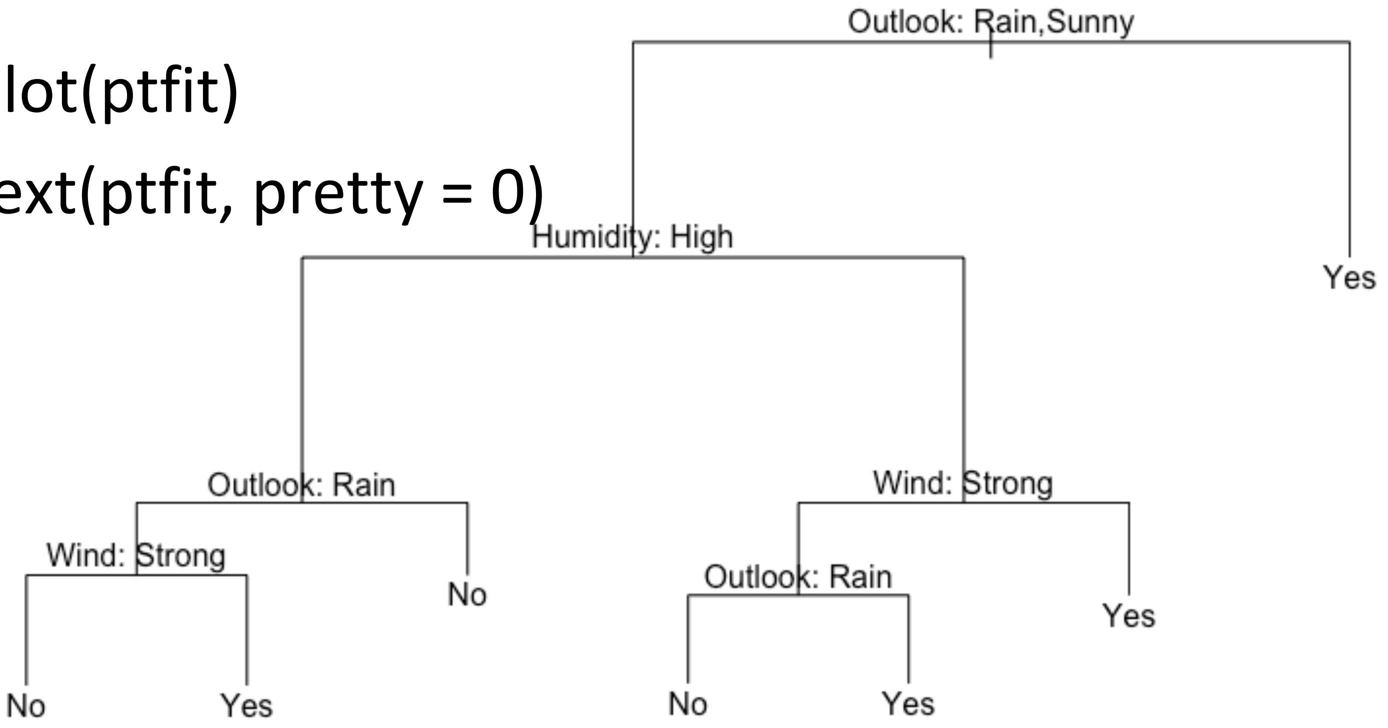
Resampling to create a larger, synthetic sample, and creating the basic tree:

```
>   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 the decision tree in R

The original tree with 7 leaves.

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



Pruning the decision tree 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 the decision tree in R

Cross validation test at different tree sizes:

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

*Note cv.tree problem in R3.6.3, later versions of R seem to be ok.

```
$size
[1] 7 3 1
$dev
[1] 12 10 37 – count misclassifications at each
    size
$k
[1] -Inf  2.5 12.5  – cost-complexity parameter
$method
[1] "misclass"
...
```

? 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 the decision tree in R

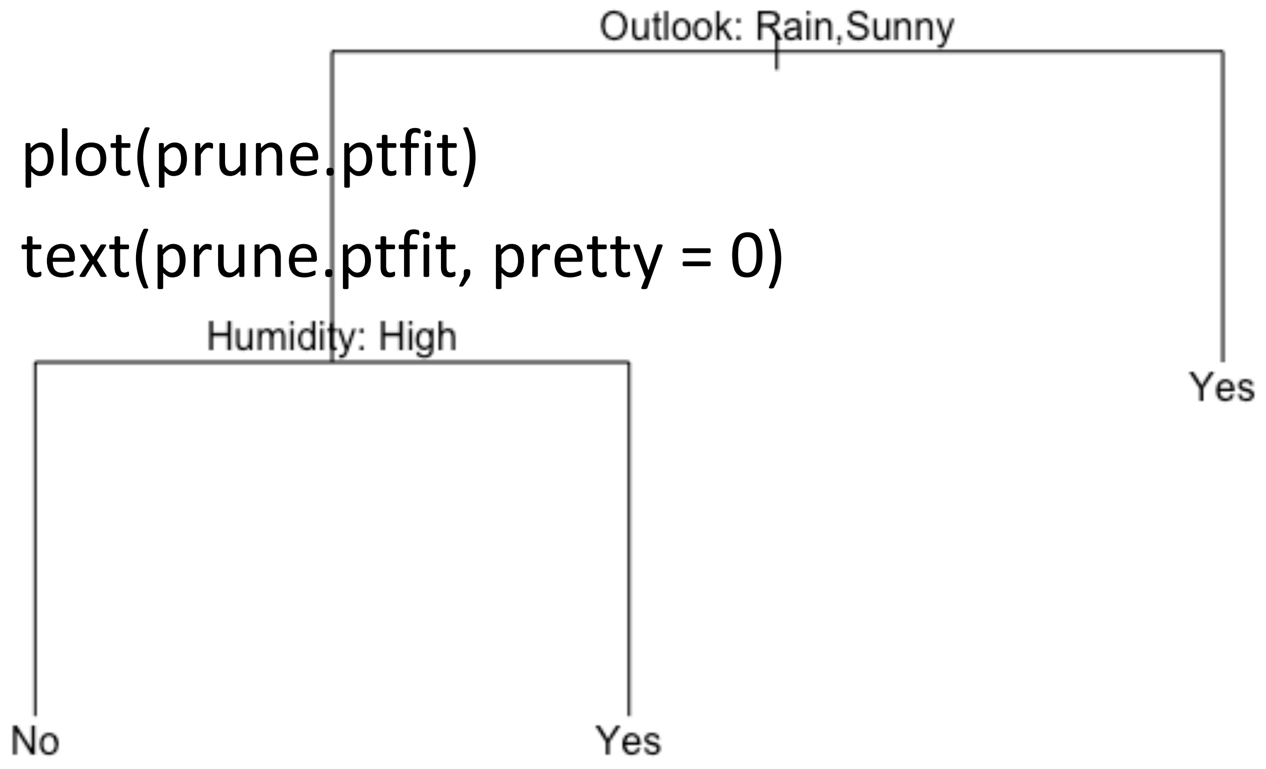
Pruning the tree:

```
> 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"  "Humidity"
Number of terminal nodes:  3
Residual mean deviance:  0.589 = 57.1 / 97
Misclassification error rate: 0.1 = 10 / 100
```

Pruning the decision tree in R

The new tree with 3 leaves.

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



Pruning the decision tree 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  1   2
```

- $Accuracy = \frac{3}{6} = 50.0\%$ compared with 33.3% previously!

Confidence level of a classification

Note: how reliable was the "No" or "Yes" classification in the previous example?

- We can obtain the confidence level for each class, and use this to make the classification, and to evaluate the classifier… (more on this later).
- From previous example:

```
> 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
```

Bayesian classifiers

Bayes’ Theorem

Bayesian Classifiers

Independent events

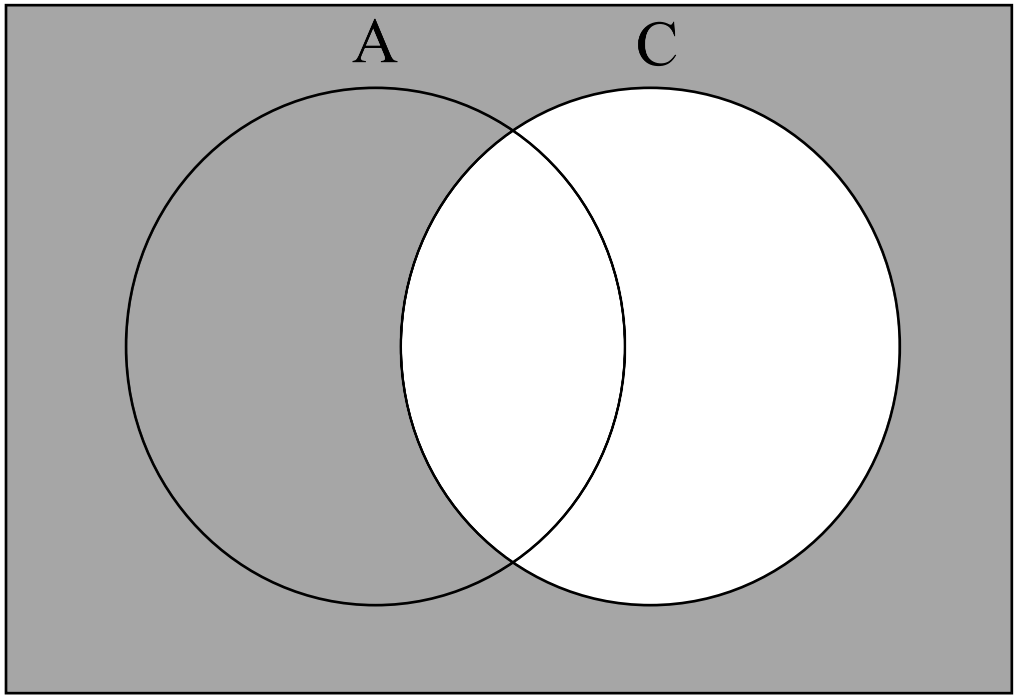
Naïve Bayes’ Classifier

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 as:

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

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 $(A_1, A_2, \ldots, A_n)$ the goal is to predict value class C.
- Specifically, we want to find the value of C, C_j that maximizes P $(C_j|A_1, A_2, \ldots, A_n)$.
- This can be estimated directly from data.

Bayesian classifiers

Approach:

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

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

- Choose the value of C_j that maximises this posterior probability $P(C_j|A_1 \cap A_2 \cap A_3 \ldots \cap A_n)$

- Is equivalent to maximising $P(C_j) \cdot P(A_1 \cap A_2 \cap A_3 \ldots \cap A_n|C_j)$, since $P(A_1 \cap A_2 \cap A_3 \ldots \cap A_n)$ 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, naive 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 to the C_j which maximises

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

- Classification confidence (probability) is given by

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

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 to the C_j which maximises: $P(C_j) \times P(A_1|C_j) \times P(A_2|C_j) \times \dots \times P(A_n|C_j)$
- 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	?

- $P(A_1|N) = 1/13$; $P(A_2|N) = 10/13$; $P(A_3|N) = 3/13$; $P(A_4|N) = 4/13$
- $P(N) * P(A|N) = 13/20 * 1/13 * 10/13 * 3/13 * 4/13 = 0.003$

Naïve Bayes example

- Recall: maximise: $P(C_j) \times P(A_1|C_j) \times P(A_2|C_j) \times \ldots \times P(A_n|C_j)$
- 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:

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

- When j = 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 = 0.003

- When j = 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 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, pttest, 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
```

Classifier model evaluation

How do we decide how ‘good’ our model is? How do we decide which is the ‘best’ model to use on our data? Several methods:

- Confusion matrix, accuracy and other performance measures,
- ROC (Receiver Operating Characteristic) for binary classifiers,
- AUC (Area Under the Curve),
- Lift Charts.

Receiver Operating Characteristic

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

- Characterises the tradeoff between positive hits and false alarms.
- 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.

Receiver Operating Characteristic

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.)

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

- False Positive Rate, FPR, also known as a ‘*false alarm*’ (*or 1 – specificity)* is:

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

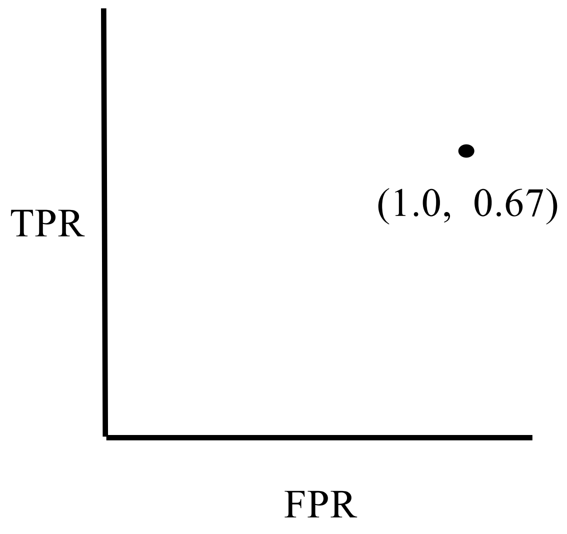
- ROC plots TPR against FPR.

For play tennis example…

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

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

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



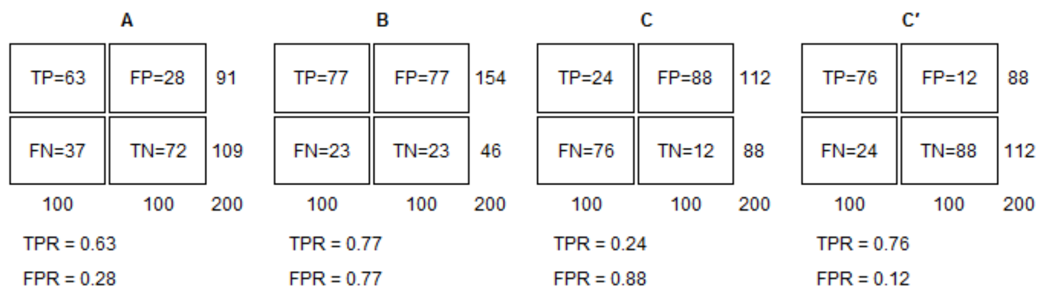
ROC - receiver operating characteristic

Classifiers can be compared using their confusion matrixes.

But this only provides comparison at a specified confidence threshold.

e.g. comparing ROCs for different classifiers A, B, C, C'

ref. Wikipedia.



A

TP=63	FP=28	91
FN=37	TN=72	109
100	100	200

TPR = 0.63

FPR = 0.28

B

TP=77	FP=77	154
FN=23	TN=23	46
100	100	200

TPR = 0.77

FPR = 0.77

C

TP=24	FP=88	112
FN=76	TN=12	88
100	100	200

TPR = 0.24

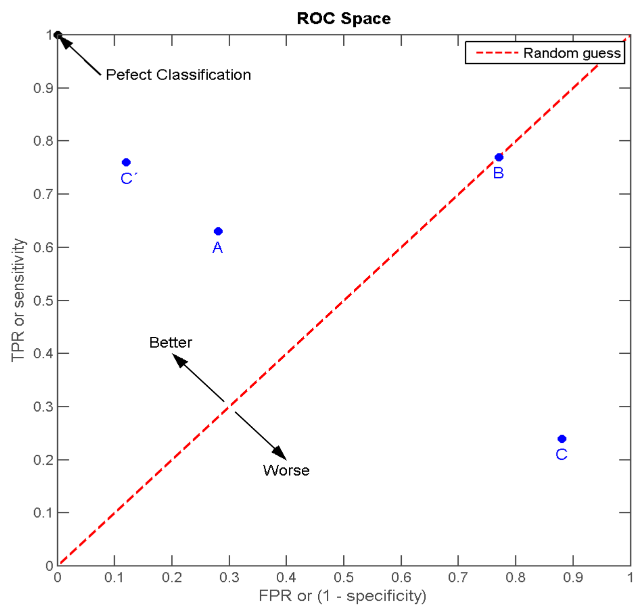
FPR = 0.88

C′

TP=76	FP=12	88
FN=24	TN=88	112
100	100	200

TPR = 0.76

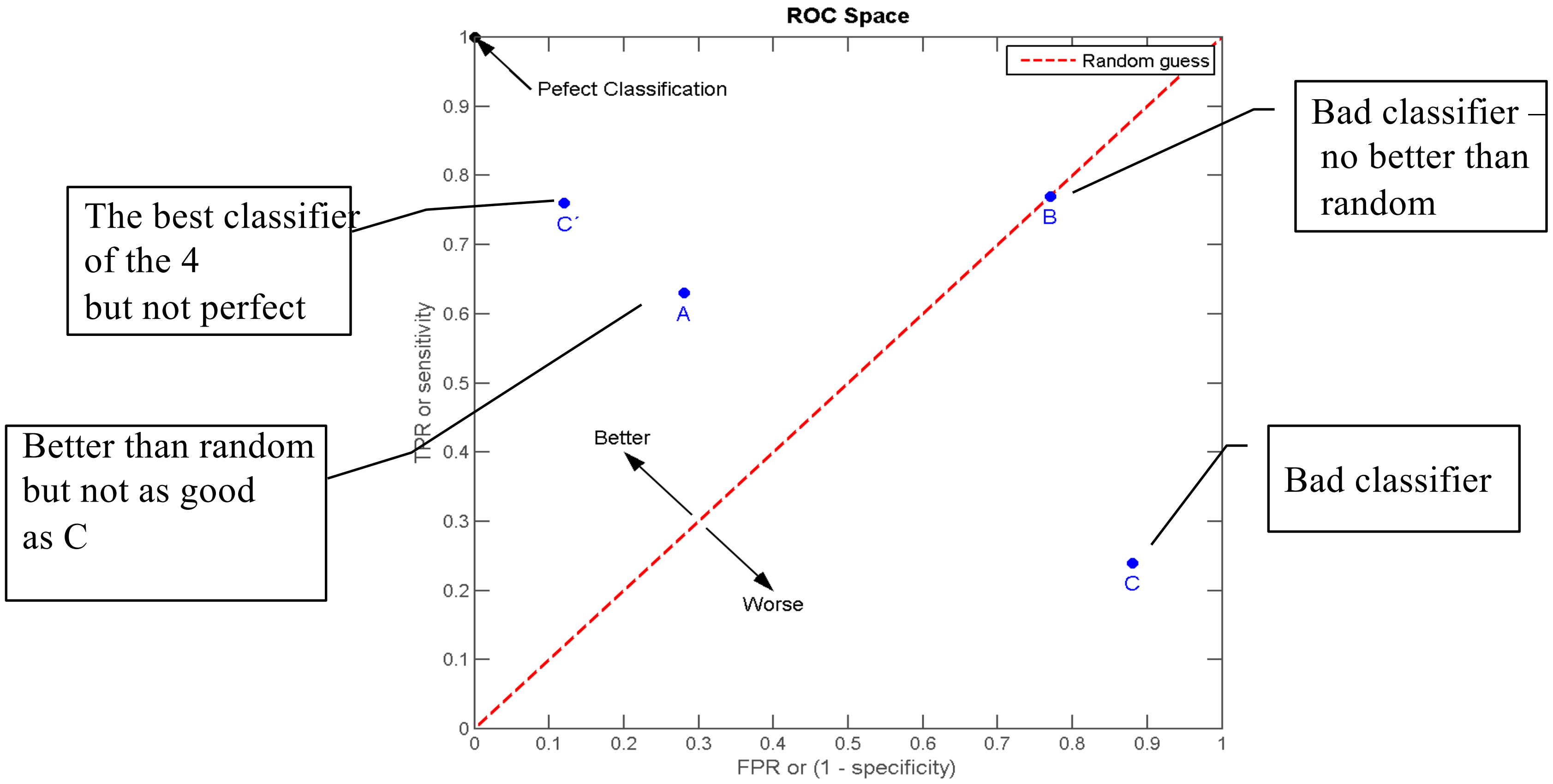
FPR = 0.12



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

ROC - receiver operating characteristic

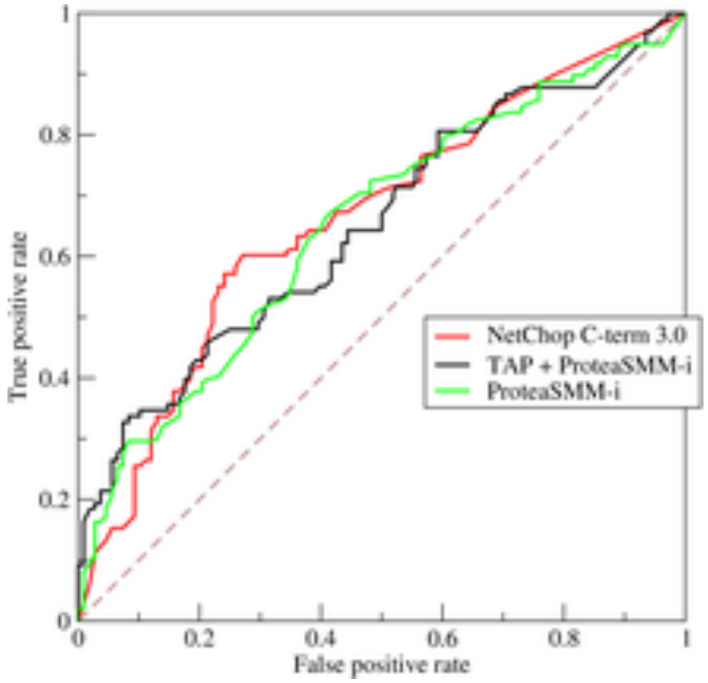
E.g. comparing ROCs for different classifiers A, B, C..., ref. Wikipedia



Note: Specificity = TN/(TN + FP) = 1 - FPR

Comparing classifiers using ROC

- The TPR and FPR values on previous slides have been 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:

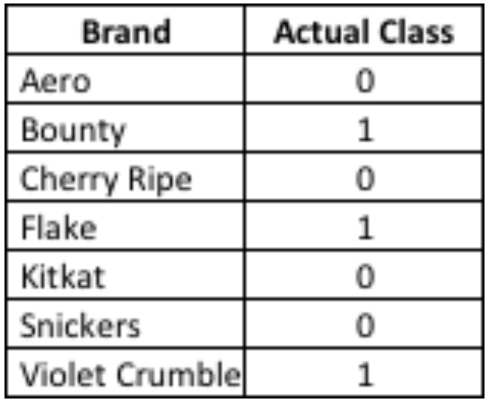
```
>   tbpredict.r = predict(tmodel, pttest, type = 'raw')
>   tbpredict.r
```

```
                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 classify, and get the following classification that the chocolate is excellent:

Brand	Actual Class	Confidence pred class = 1
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 class = 1	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 pred class = 1	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

	Predicted Class Labels		
Actual		Class = 1	Class = 0
Class	Class = 1	TP	FN
Labels	Class = 0	FP	TN
TPR =	_____	FPR =	_____

	Predicted Class Labels		
Actual		Class = 1	Class = 0
Class	Class = 1	3	0
Labels	Class = 0	1	3
TPR =	1.00	FPR =	0.25

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

- 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 pred class = 1**	**C >= 0.3**	**C >= 0.4**	**C >= 0.6**	**C >= 0.7**	**C >= 0.8**	**C >= 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 >= 0.3	Predicted Class Labels		
Actual		Class = 1	Class = 0
Class	Class = 1		
Labels	Class = 0		
TPR =	_____	FPR =	_____

C >= 0.4	Predicted Class Labels		
Actual		Class = 1	Class = 0
Class	Class = 1		
Labels	Class = 0		
TPR =	_____	FPR =	_____

C >= 0.6	Predicted Class Labels		
Actual		Class = 1	Class = 0
Class	Class = 1		
Labels	Class = 0		
TPR =	_____	FPR =	_____

C >= 0.7	Predicted Class Labels		
Actual		Class = 1	Class = 0
Class	Class = 1		
Labels	Class = 0		
TPR =	_____	FPR =	_____

C >= 0.8	Predicted Class Labels		
Actual		Class = 1	Class = 0
Class	Class = 1		
Labels	Class = 0		
TPR =	_____	FPR =	_____

C >= 1.0	Predicted Class Labels		
Actual		Class = 1	Class = 0
Class	Class = 1		
Labels	Class = 0		
TPR =	_____	FPR =	_____

Example: Constructing ROC curve

- Now do at each threshold (confidence level):

Brand	Actual Class	Confidence pred class = 1	C >= 0.3	C >= 0.4	C >= 0.6	C >= 0.7	C >= 0.8	C >= 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 >= 0.3	Predicted Class Labels		
Actual		Class = 1	Class = 0
Class	Class = 1	3	0
Labels	Class = 0	4	0
TPR =	1.00	FPR =	1.00

C >= 0.4	Predicted Class Labels		
Actual		Class = 1	Class = 0
Class	Class = 1	3	0
Labels	Class = 0	3	1
TPR =	1.00	FPR =	0.75

C >= 0.6	Predicted Class Labels		
Actual		Class = 1	Class = 0
Class	Class = 1	3	0
Labels	Class = 0	1	3
TPR =	1.00	FPR =	0.25

C >= 0.7	Predicted Class Labels		
Actual		Class = 1	Class = 0
Class	Class = 1	2	1
Labels	Class = 0	1	3
TPR =	0.67	FPR =	0.25

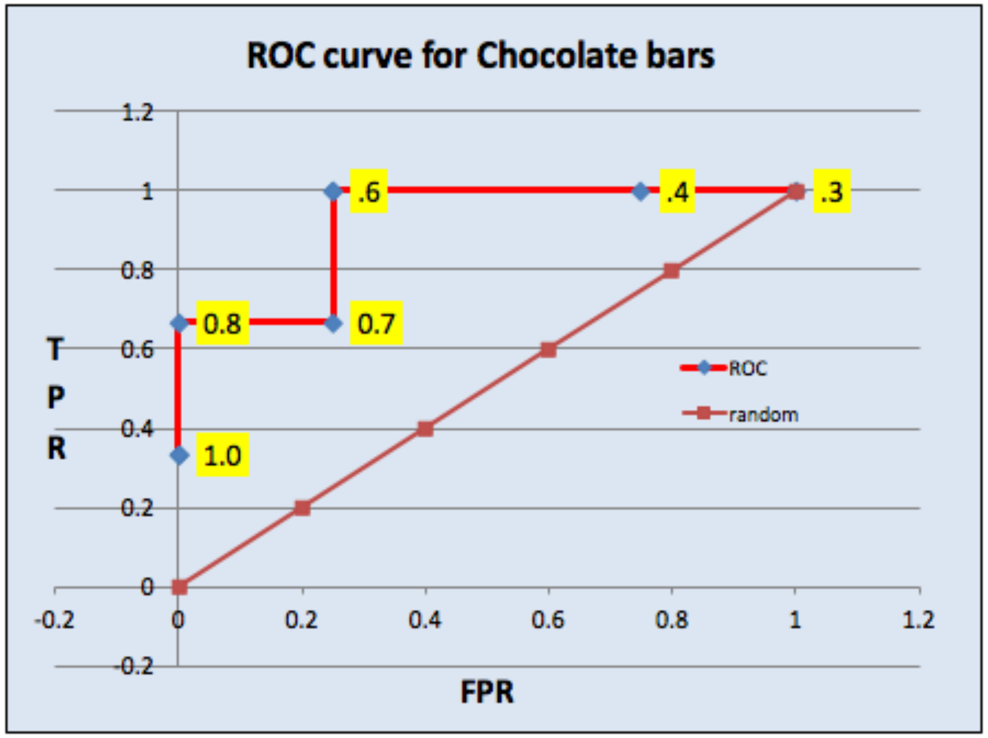
C >= 0.8	Predicted Class Labels		
Actual		Class = 1	Class = 0
Class	Class = 1	2	1
Labels	Class = 0	0	4
TPR =	0.67	FPR =	0.00

C >= 1.0	Predicted Class Labels		
Actual		Class = 1	Class = 0
Class	Class = 1	1	2
Labels	Class = 0	0	4
TPR =	0.33	FPR =	0.00

Example: Constructing ROC curve

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

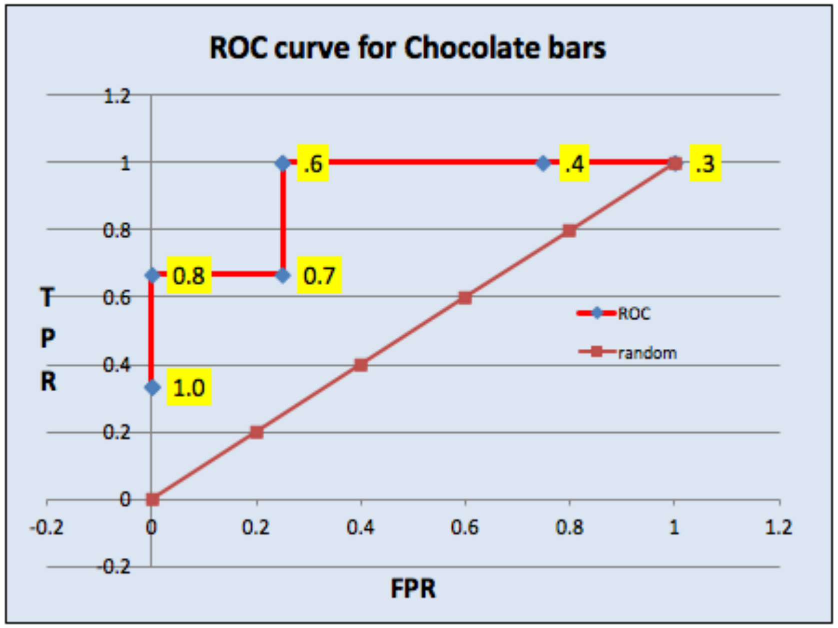
Threshold	FPR	TPR
1.0	0.00	0.33
0.8	0.00	0.67
0.7	0.25	0.67
0.6	0.25	1.00
0.4	0.75	1.00
0.3	1.00	1.00



🟨 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)

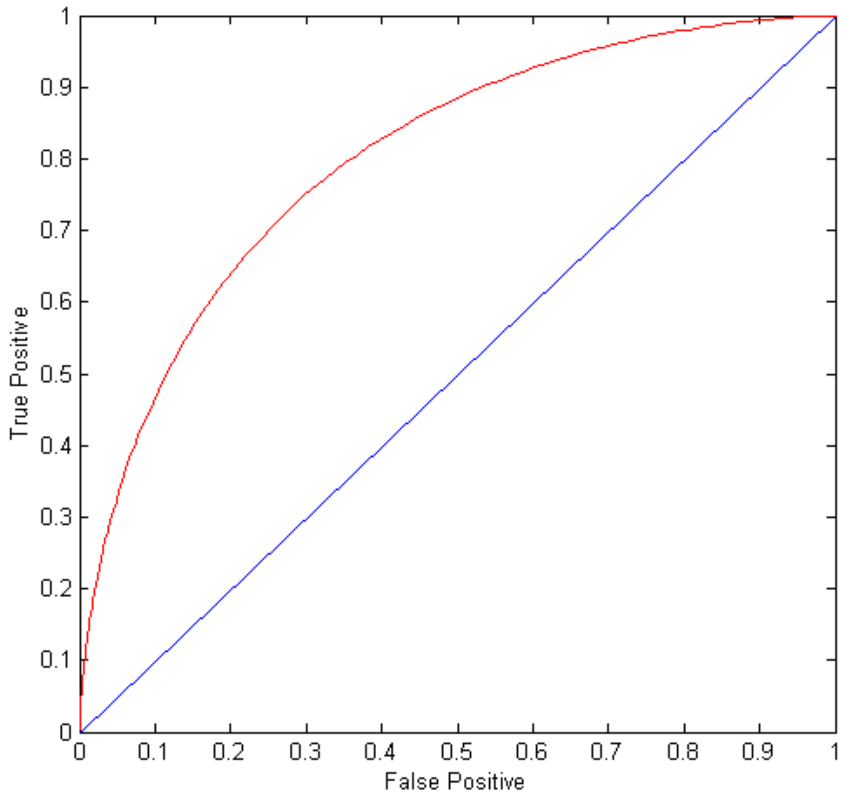
- (0,0): Declare everything to be negative class
- (1,1): If declare everything to be positive class
- (1,0): Ideal

Diagonal line:

- Random guessing

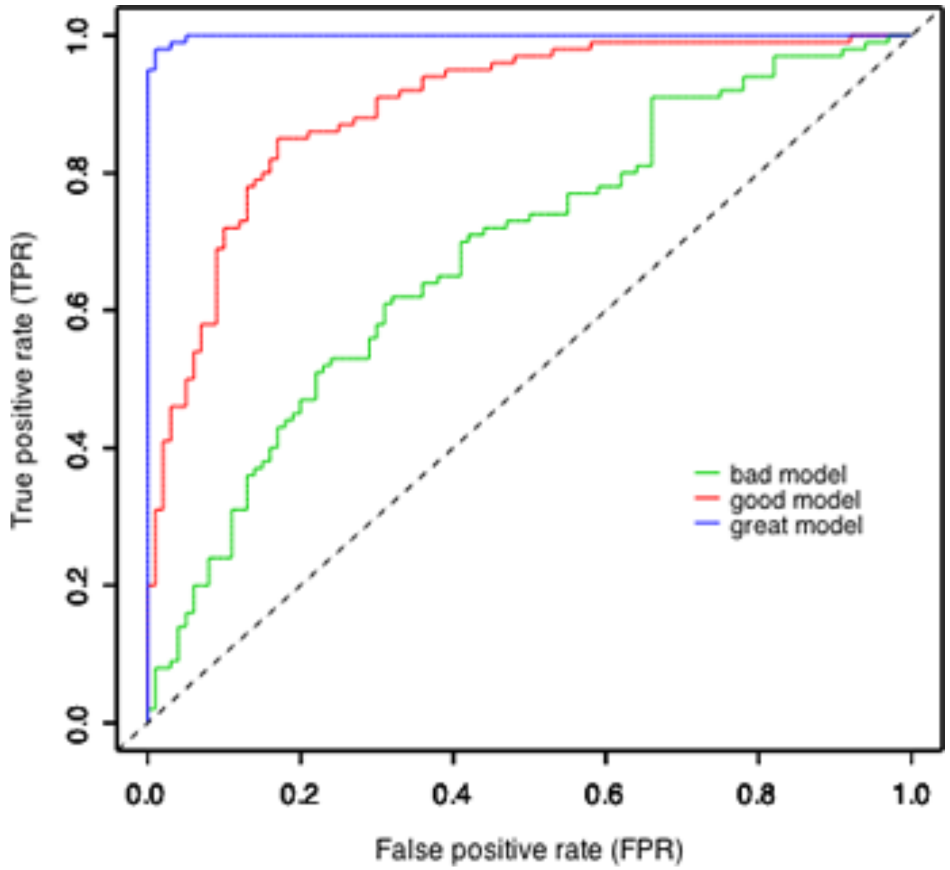
Below diagonal line:

- Prediction is opposite of the true class



Using ROC for Model Comparison

ROC curves for three different classifiers



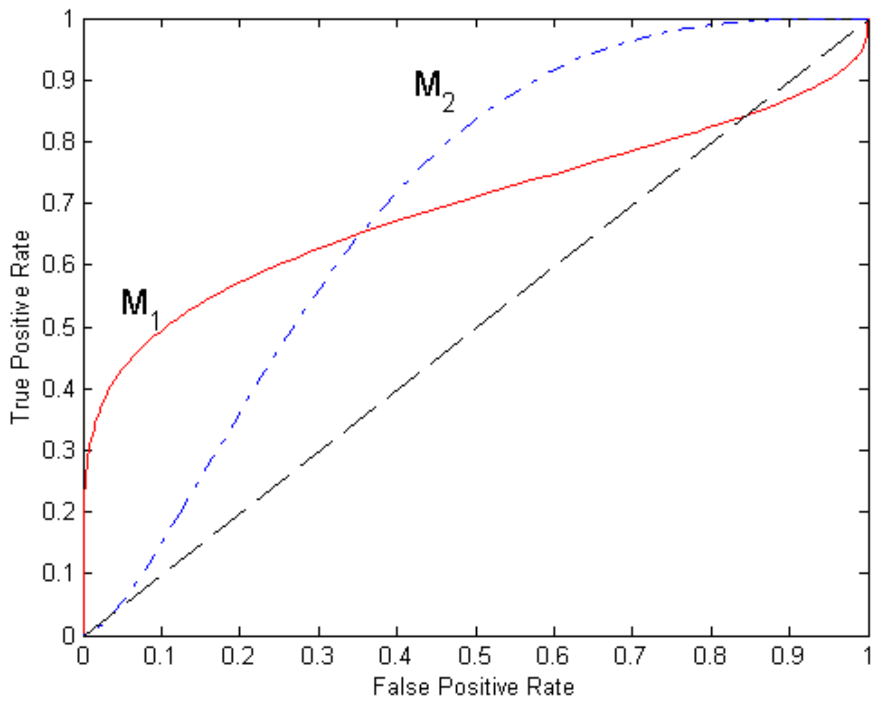
Using ROC for model comparison

No model consistently outperforms the other

- M1 better for small FPR
- M2 better for large FPR

Area Under ROC Curve

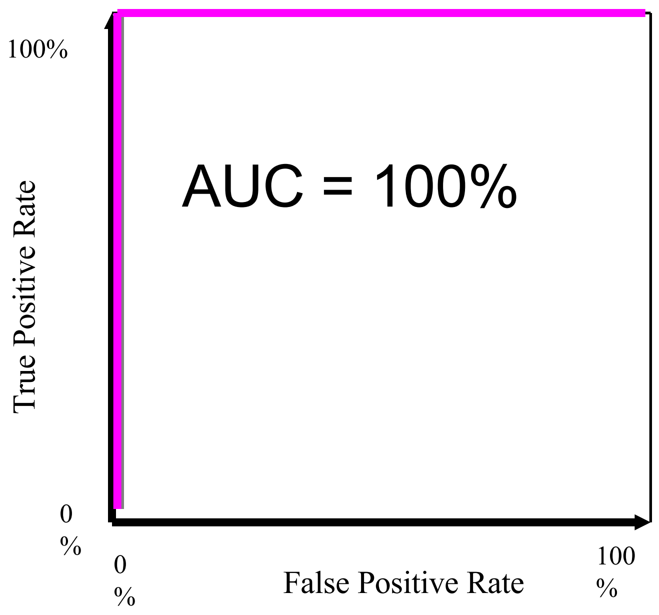
- Ideal: Area = 1
- Random guess: Area = 0.5



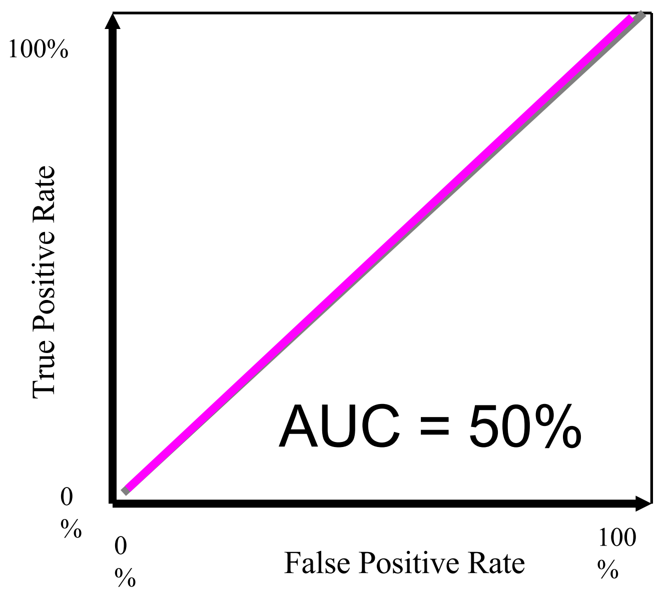
Area under ROC curve (AUC)

For binary data:

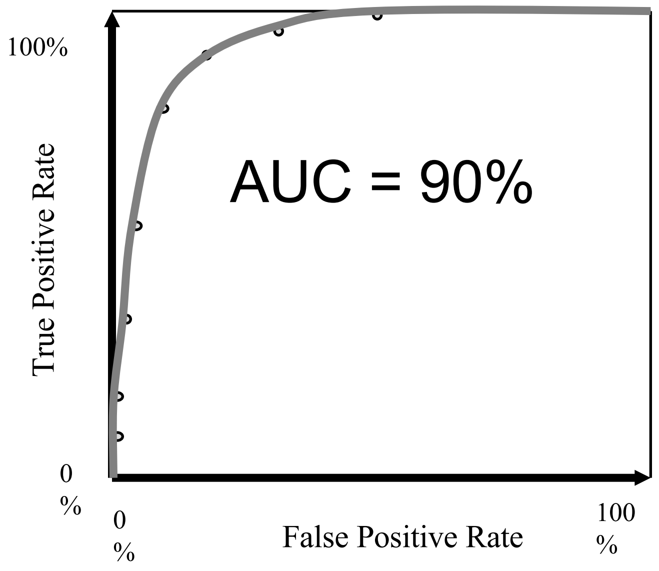
- Overall single measure of test performance
- Comparisons between two tests based on differences between (estimated) AUC
- (For continuous data, AUC equivalent to Mann-Whitney U-statistic)
- Some examples of AUC on following slide:



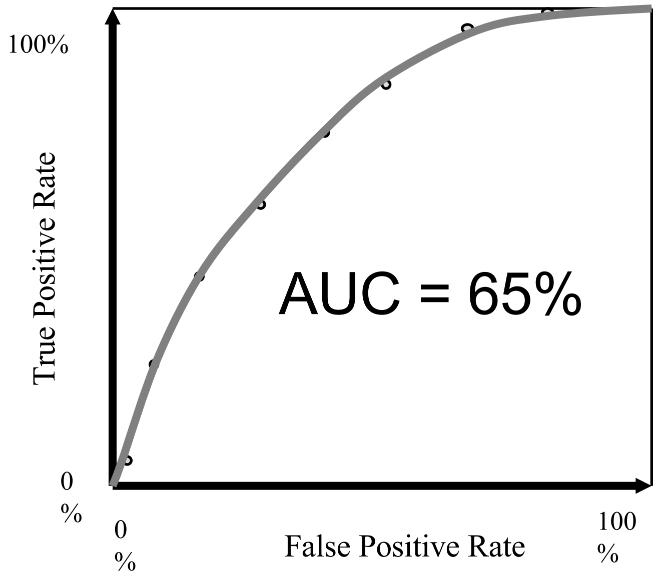
100%
True Positive Rate
0 %
AUC = 100%
0 %
False Positive Rate
100 %



100%
True Positive Rate
0 %
AUC = 50%
0 %
False Positive Rate
100 %



100%
True Positive Rate
0 %
AUC = 90%
0 %
False Positive Rate
100 %



100%
True Positive Rate
0 %
AUC = 65%
0 %
False Positive Rate
100 %

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.

Interpretation of AUC

Some guidelines for interpreting AUC:

Range	**Interpretation**
AUC = 0.5	No discrimination (i.e., might as well flip a coin)
$0.7 \leq \text{AUC} < 0.8$	Acceptable discrimination
$0.8 \leq \text{AUC} < 0.9$	Excellent discrimination
$\text{AUC} \geq 0.9$	Outstanding discrimination (but extremely rare)

Reference: D.W. Hosmer and S. Lemeshow (2000). Applied Logistic Regression. 2nd ed. John Wiley & Sons, Inc. pp. 156-164.

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 devices[2], 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 symptoms[6]. And in *Nature Biomedical Engineering*,

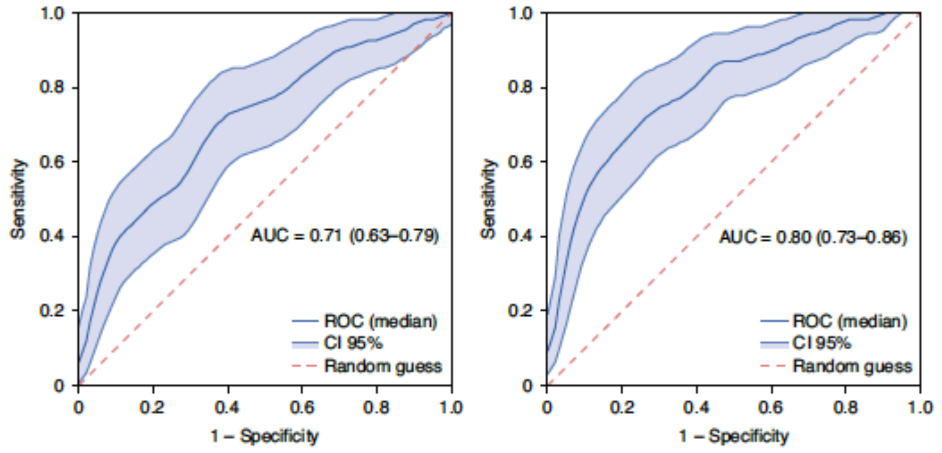


Fig. 1 | **Prediction of COVID-19 from self-reported symptoms, and from self-reported symptoms combined with RHR, sleep and activity data from smartwatches.** The receiver operating characteristic (ROC) curves for the discrimination of 54 individuals who tested positive for COVID-19 and 279 individuals 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 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)
```

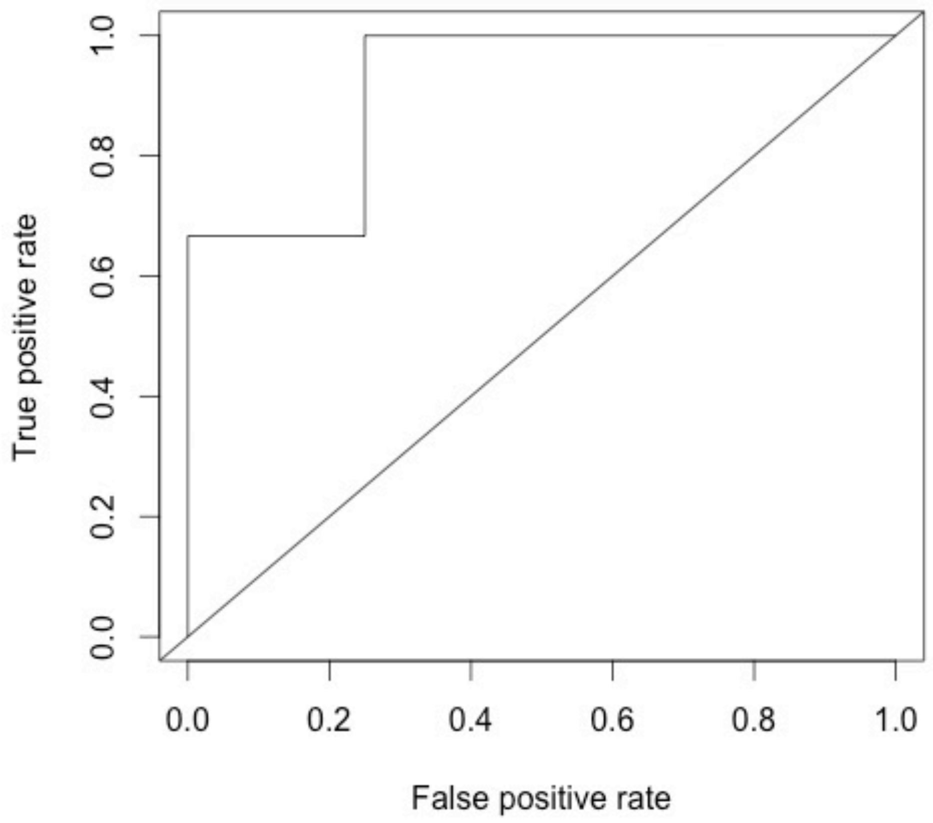
ROC analysis in R

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:

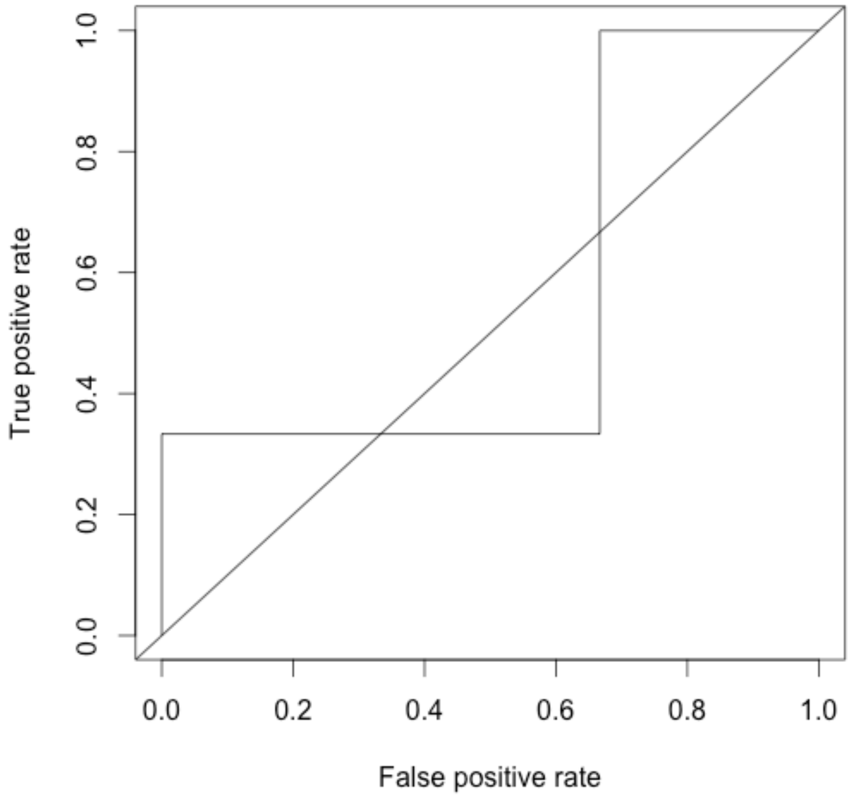
```
>   pttrain <- read.csv("playtennistrain.csv")
>   pttest <- read.csv("playtennistest.csv")
>   tmodel = naiveBayes(Play ~. -Day, data = pttrain)
>   # outputs as confidence levels
>   tbpredict.r = predict(tmodel, pttest, type = 'raw')
>   tpred <- prediction( tbpredict.r[,2], pttest$Play)
>   tperf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

Creating lift charts

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.

Creating lift charts

- 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	-	+	-	-	-	+		-	+	-	+

- Suppose that sorting the instances in terms of their predicted confidence/probability leads to following 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	+	+	-	-	+	+	+	+	-	+	-	+	+	-	+	-	+	-	+	+

Creating lift charts

- Suppose that sorting the instances in terms of their predicted confidence/probability leads to following 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	+	+	-	-	+	+	+	+	-	+	-	+	+	-	+	-	+	-	+	+

- Choosing a sample size of 10 will give you a lift factor 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, and 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.

$$Lift = \frac{\left(\frac{2}{2}\right)}{\left(\frac{3}{7}\right)} = \frac{7}{3} = 2.33\ or\ 233\%$$

- For top 3 (42%), P(good) = 2/3

$$Lift = \frac{\left(\frac{2}{3}\right)}{\left(\frac{3}{7}\right)} = \frac{14}{9} = 1.56\ or\ 156\%$$

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

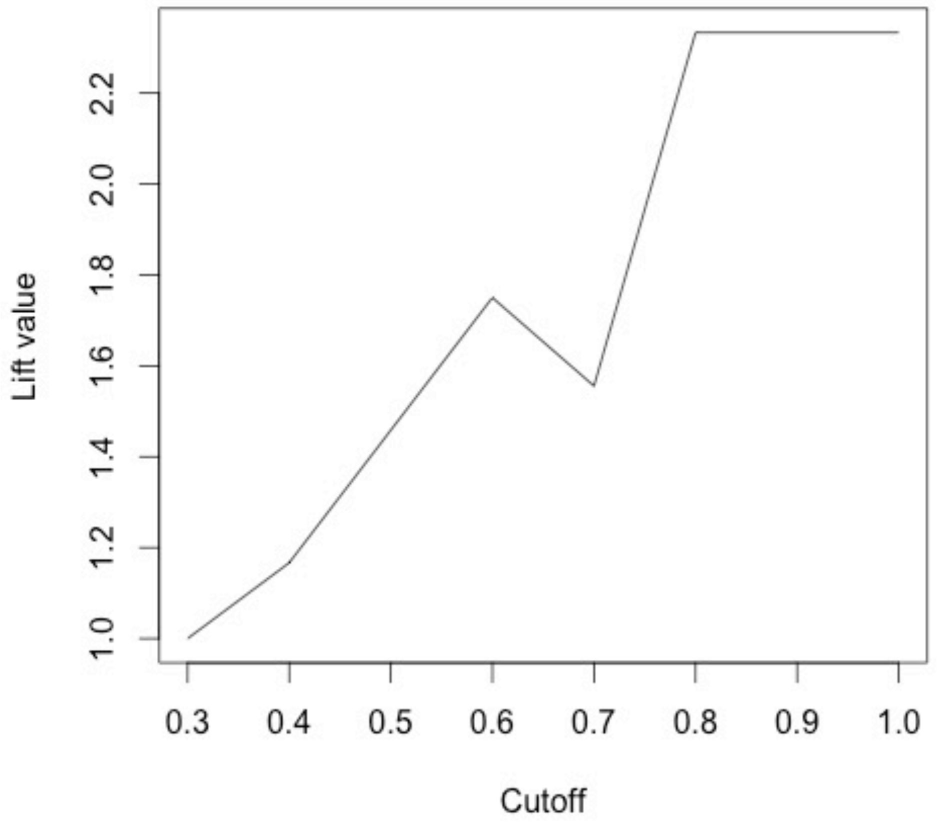
Lift analysis in R

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 plot:



Revision questions answers

1. A
2. A
3. A
4. D

Notes on the presentation

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

Presentation originally created by Dr. Sue Bedingfield.