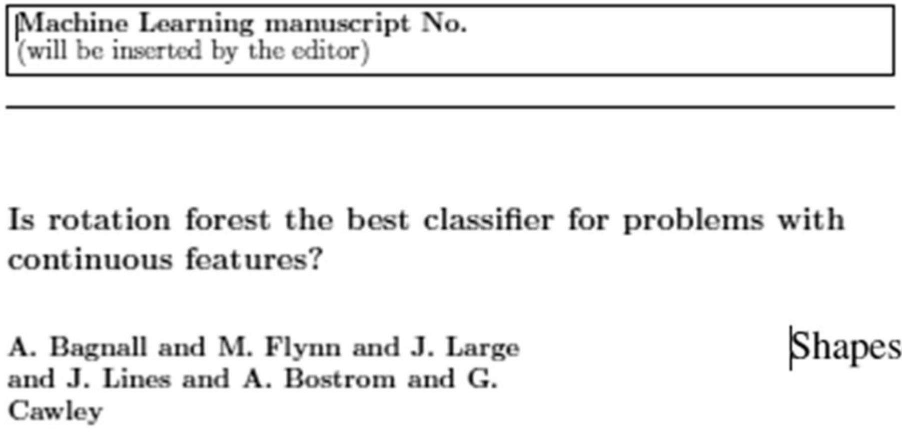
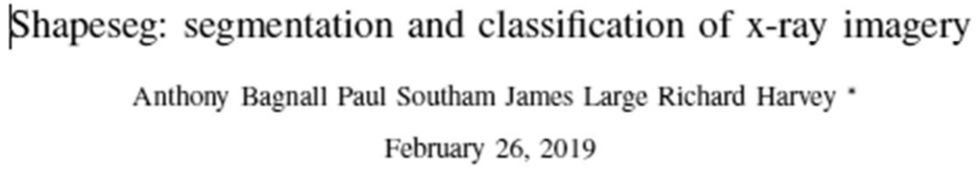
Machine Learning
Classifier Evaluation



Machine Learning manuscript No.
(will be inserted by the editor)

Is rotation forest the best classifier for problems with continuous features?

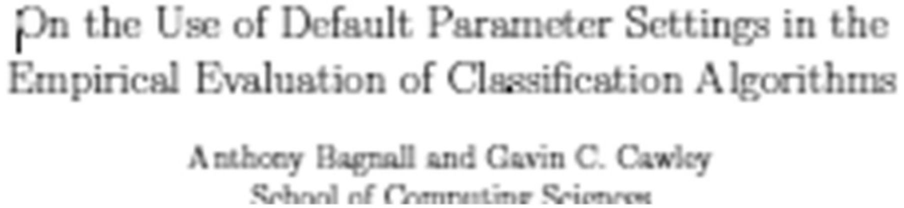
A. Bagnall and M. Flynn and J. Large and J. Lines and A. Bostrom and G. Cawley



Shapeseg: segmentation and classification of x-ray imagery

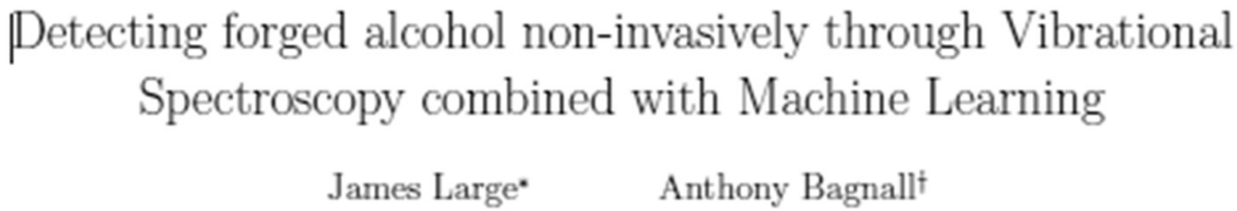
Anthony Bagnall Paul Southam James Large Richard Harvey *

February 26, 2019



On the Use of Default Parameter Settings in the Empirical Evaluation of Classification Algorithms

Anthony Bagnall and Gavin C. Cawley
School of Computing Sciences



Detecting forged alcohol non-invasively through Vibrational Spectroscopy combined with Machine Learning

James Large* Anthony Bagnall†

Classifier Evaluation

- How good is the model for data not seen before? This is often called the models ability to **generalize**
- How does the model built with one algorithm compare to another?
- What do we mean by "good"? What performance metric do we use?
- Are we interested in performance on a single problem or over problems generally?

Classifier Evaluation

- How do we assess a classifier on a single problem?
- How do we compare two classifiers on a single problem?
- How do we compare two classifiers on a test bed of problems?
- How do we compare multiple classifiers on a test bed of problems?

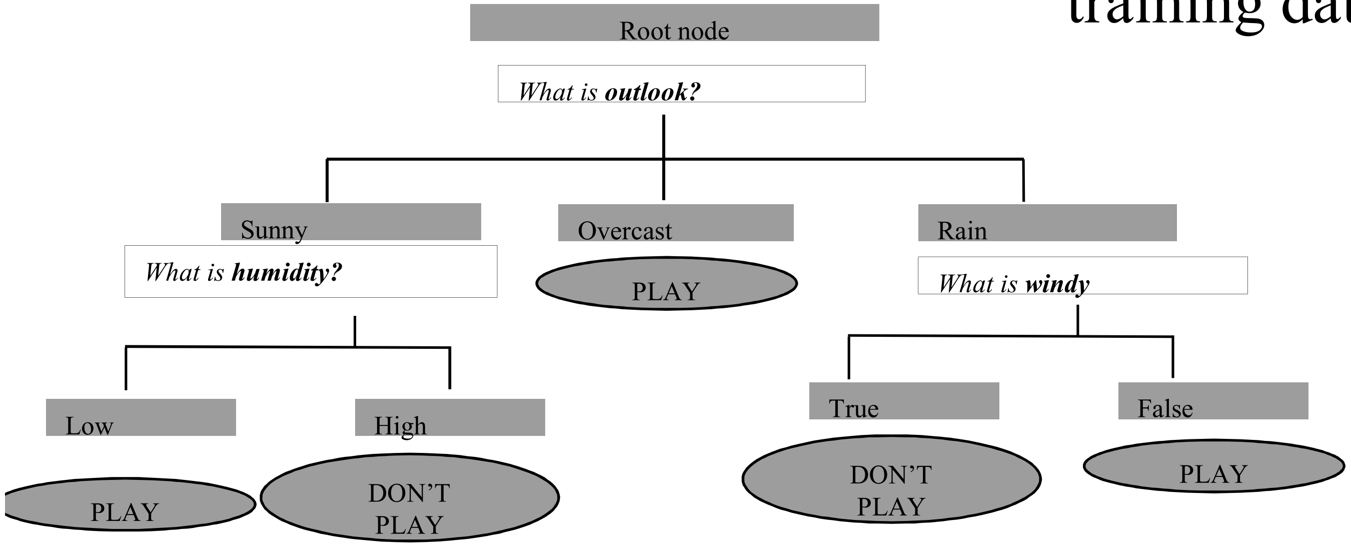
There is evaluation code in Weka and our own is part of the repo
https://github.com/uea-machine-learning/tsml

What data do we use to evaluate a classifier?

It is **vital** we assess classifiers on data that was not used in the building process, because the build process usually tries to minimize the error

Consider, for example, the decision tree we built in the last lecture.

This is 100% accurate on the training data.



That does not mean it will be 100% accurate on unseen data

Train set accuracy is often used in model fitting**, but should not figure in model evaluation**

Sampling Prior to Evaluation

Suppose we have a single data set. To evaluate a classifier we need to split it into training and testing data sets, build on the train and then predict on the test

```
evaluateClassifier(DataSet D, Classifier c)
    {train,test}=sampleData(allData)
    c.buildClassifier(train)
    ResultsFormat res
    for Instance d:test
        double[]p=c.distributionForInstance(d)
        int pred=c.classifyInstance(d)
        res.add(new Triple(d.actualClass,pred,p))
    return res
```

We revisit how to sample later. For now, just assume it produces two data sets

Results Format

We output all results in a standard format text file

File testFold1.csv

Line 1, problem name, classifier name

Line 2, bespoke parameter info if present

Line 3, test accuracy

Line 4, predictions for test case 1

```
balloons,C4.5
No Parameter Info
0.6667
0,1,,0.276,0.724
0,0,,0.873,0.127
0,0,,0.969,0.031
0,0,,0.982,0.018
0,0,,0.970,0.03
1,1,,0.09,0.91
1,0,,0.6,0.4
1,1,,0.099,0.901
1,0,,0.873,0.127
```

Actual Class

Predicted Class

Probability class 0

0,1,,0.276,0.724

Probability class 1

Test Set Accuracy

The most obvious assessment criteria is accuracy (or error)

```
measureAccuracy(Results res)
    correct=0
    n=res.numberInstances ()
    for Triple t:res
        if(t.predictedClass==t.actualClass)
            correct=correct+1
    accuracy=correct/n
    error=1-accuracy
```

File testFold1.csv

6 correct out of 9

Accuracy =6/9, error =3/9

Accuracy can be misleading

I have a rule based algorithm that can predict with 99.9% accuracy whether you will reply to a scam email or not

The rule is **always predict that you will not respond**

This is 99.9% accurate because only 1 in 1000 people respond to scam emails

Whilst my claim of accuracy is true, it is not very insightful into what causes people to respond

Confusion Matrix

Accuracy does not always give the whole story, especially if the class is unbalanced.

More information is conveyed in the **confusion matrix**, which is the counts of correct and false split by class

```
formConfusionMatrix(Results res)
        c = res.numberClasses();
        cTable= new int[c][c]
        for Triple t:res
                c[t.predictedClass][t.actualClass]+=1
```

File testFold1.csv

	Actual 0	Actual 1
Predicted 0	4	2
Predicted 1	1	2

Stats from Confusion Matrix

Suppose we call one class Positive and the other Negative

		True Condition (Actual)	
Predicted		Positive	Negative
Condition	Positive	True positive (a)	False positive (b)
	Negative	False negative (c)	True negative (d)

a,b,c and d are the counts of each occurrence

$$\boxed{\text{Accuracy} =(a+d)/(a+b+c+d)}$$

Rates from the Confusion

	Positive	**Negative**
Positive	True positive (a)	False positive (b)
Negative	False negative (c)	True negative (d)

True positive rate (TPR) is the proportion of positive cases **correctly** classified

TPR=a/(a+c)

The true negative rate (TNR) is the proportion of negatively cases **correctly** classified

TNR=d/(b+d)

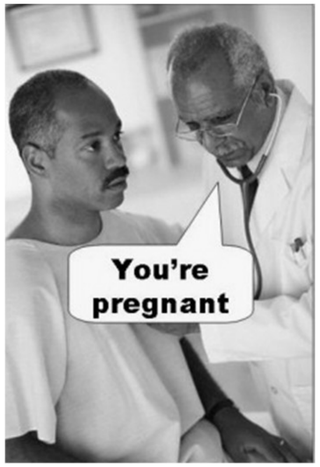
- We are only considering two class problems at the moment
- We assume there is some notion of "positive" and "negative". If unclear, we call the minority class the "positive" class.

Rates from the Confusion

	Positive	**Negative**
Positive	True positive (a)	False positive (b)
Negative	False negative (c)	True negative (d)

The false positive rate (FPR) is the proportion of negative cases **incorrectly** classified as positive. Also called Type I Error

$\boxed{FPR=b/(b+d)}$



The false negative rate (FNR) is the proportion of positive cases **incorrectly** classified as negative. Also called Type II Error

$\boxed{FNR=c/(a+c)}$

Balanced Accuracy

	Positive	Negative
Positive	True positive (a)	False positive (b)
Negative	False negative (c)	True negative (d)

Balanced Accuracy: average accuracy over all classes

$$\boxed{\text{Balanced Accuracy} = (TPR+TNR)/2}$$

It give equal importance to each class irrespective of how many are in each class

Balanced Accuracy can easily be generalised to multi class problems

	0	1	2	3
0	4	100	3	0
1	5	700	10	1
2	0	10	1010	5
3	0	0	10	3

Acc= 0.923

BalAcc=
(0.444+0.864+0.977+0.33)/
4
=0.655

Sensitivity vs Specificity

	Positive	Negative
Positive	True positive (a)	False positive (b)
Negative	False negative (c)	True negative (d)

from wiki

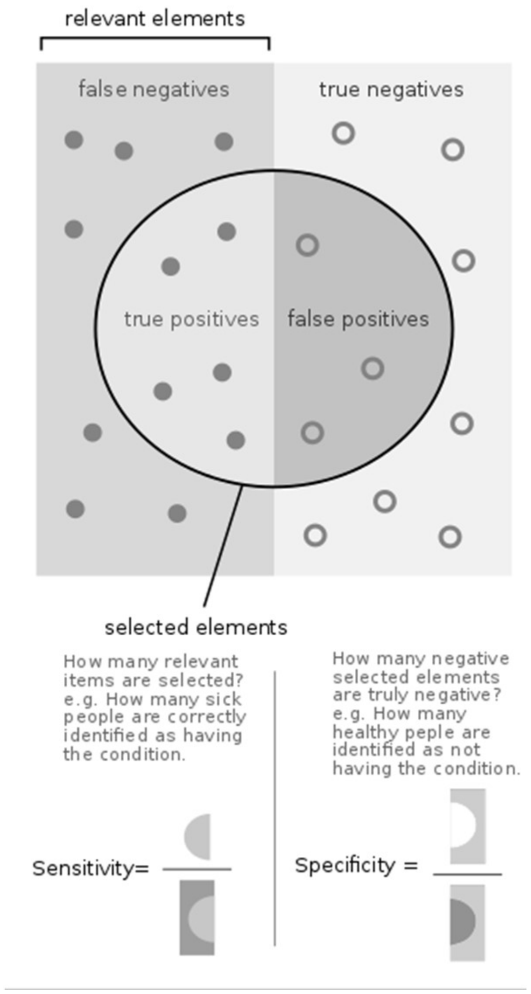
TPR is also often called the **sensitivity** of the classifier

TPR:**Sensitivity**=a/(a+c)

The True Negative Rate (TNR) is also often called the **specificity**

TNR:**Specificity**=d/(b+d)

Sensitivity and specificity are terms commonly used in medical tests


relevant elements
false negatives
true negatives
true positives
false positives
selected elements
How many relevant items are selected? e.g. How many sick people are correctly identified as having the condition.
How many negative selected elements are truly negative? e.g. How many healthy peple are identified as not having the condition.
Sensitivity=
Specificity =

Recall, Precision and F1

	Positive	Negative
Positive	True positive (a)	False positive (b)
Negative	False negative (c)	True negative (d)

TPR is also often called the **recall** of the classifier

The precision

$\boxed{\text{TPR:Sensitivity:}\textbf{Recall}\text{=a/(a+c)}}$

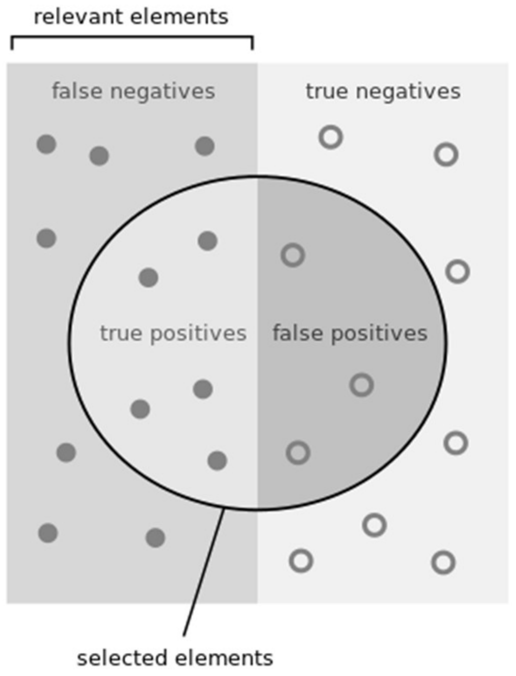
The **precision** of the classifier is the proportion of predicted positive cases that actually are positive

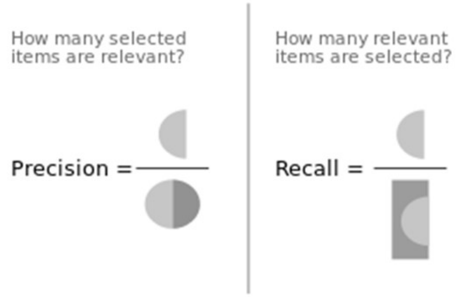
$\boxed{\textbf{Precision}\text{=a/(a+b)}}$

The **F measure** or **F1 score** is

$\boxed{\textbf{F1}\text{=2(precision*recall)/(precision+recall)}}$

These terms originate from **information retrieval**




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

Scam Email Example

- Suppose I am trying to predict whether an individual will respond to a specific form of spam so I can analyse what makes good spam.
- Being a spammer, I have a history of respondents based on spam features (keywords etc) and personal info (age etc). These are my features.
- I build three classifiers to predict respondents to scam emails

Classifier 1	Actual Respond	Actual ignore
Predict respond	0	0
Predict ignore	100	9900

Accuracy =(0+9900)/(10000)=0.99

TPR:Sensitivity:Recall =0

Balanced Accuracy =0.5

TNR:Specificity =1

Precision =??

F1 =??

Classifier 2	Actual Respond	Actual ignore
Predict respond	50	50
Predict ignore	50	9850

Accuracy =(50+9850)/(1000)=0.99

TPR:Sensitivity:Recall =0.5

Balanced Accuracy =0.7475

TNR:Specificity =9850/9900=0.995

Precision =0.5

F1 =0.5

Classifier 3	Actual Respond	Actual ignore
Predict respond	90	90
Predict ignore	10	9810

Accuracy =(90+9810)/(1000)=0.99

TPR:Sensitivity/Recall =0.9

Balanced Accuracy =0.945

TNR:Specificity =0.991

Precision =0.5

F1 =0.643

Scam Email Example Summary

	Classifier 1	Classifier 2	Classifier 3
Accuracy	0.99	0.99	0.99
Balanced Accuracy	0.5	0.7475	0.9495
TPR:Sensitivity:Recall	0	0.5	0.99
TNR:Specificity	1	0.995	0.991
Precision	0	0.5	0.5
F1	0	0.5	0.643

- All three classifiers have the same accuracy.
- Classifier 2 and 3 are superior to Classifier 1 on all other measures except TNR
- Classifier 3 is arguably better than Classifier 2 because it is more accurate on the minority class. This is shown in the balanced accuracy and F1 statistic

Assessing Probabilities

Predictions contain less information than probability estimates

```
balloons,C4.5
No Parameter Info
0.6667
0,1,,0.01,0.99
0,0,,0.95,0.05
```

```
balloons,CART
No Parameter Info
0.6667
0,1,,0.45,0.55
0,0,,0.6,0.4
```

- Both classifiers get the first prediction wrong, but CART was less sure about the prediction (prob=0.55 instead of 0.99), so is in some way better
- Conversely, both classifiers get the second prediction right, but C4.5 was more confident, and hence it could be considered better on this case

Likelihood

The likelihood is the probability of observing data given a model. $L(M|D) = p(D|M) = \prod_{d \in D} p(d|M)$

In this case, by the model we mean the probabilities estimated for each test data

What is the probability of having observed the test data assuming our probabilities are correct?

0,1,,$\boxed{0.276}$,0.724
0,0,,$\boxed{0.873}$,0.127
0,0,,$\boxed{0.969}$,0.031
1,1,,0.09,$\boxed{0.91}$
1,0,,0.6,$\boxed{0.4}$

$p(d1|M)=0.276$
$p(d2|M)=0.873$
$p(d3|M)=0.979$
$p(d4|M)=0.91$
$p(d5|M)=0.4$

$L(M|D) = p(d1|M) * p(d2|M) * p(d3|M) * p(d4|M) * p(d5|M)$
$= 0.085$

Negative Log Likelihood

The large number of multiplications can lead to numeric errors.

It is standard instead to calculate the *log likelihood*

$L(M|D)$ =p(D|M)=$\prod_{d \in D} p(d|M)$

$\log(L(M|D))$ $=\log(\prod_{d \in D} p(d|M)) = \sum_{d \in D} \log(p(d|M))$

Because probabilities are always between 0 and 1, the *log likelihood* is always negative. Hence, when comparing two models, we look at the **negative log likelihood**, and prefer **smaller values**

$$NLL(M|D)) = -\sum_{d \in D} \log(p(d|M))$$

NLL Example

```
balloons,C4.5
No Parameter Info
0.6667
0,1,,0.276,0.724
0,0,,0.873,0.127
0,0,,0.969,0.031
0,0,,0.982,0.018
0,0,,0.970,0.03
1,1,,0.09,0.91
1,0,,0.6,0.4
1,1,,0.099,0.901
1,0,,0.873,0.127
```

Predicted Class 0 Probabilities
0.276,0.873,0.969,0.982, 0.970, 0.91,0.4,0.901,0.127

Predicted Class 0 Log Probabilities
-1.86 -0.20 -0.05 -0.03 -0.04 -0.14 -1.32 -0.15 -2.98

NLL=6.75

Likelihood=2^-6.75=0.009263

```
balloons,CART
No Parameter Info
0.6667
0,0,,0.54,0.45
0,0,,0.93,0.17
0,0,,0.87,0.13
0,0,,0.75,0.25
0,1,,0.2 ,0.8
1,1,,0.1 ,0.9
1,1,,0.05,0.95
1,0,,0.73,0.27
1,0,,0.51,0.49
```

Predicted Class 0 Probabilities
0.54,0.93,0.87,0.75,0.2,0.9,0.95,0.27,0.49

Predicted Class 0 Log Probabilities
-0.89 -0.10 -0.20 -0.42 -2.32 -0.15 -0.07 -1.89 -1.03

NLL=7.07

Likelihood=2^-7.07=0.007413

The probability of observing the data is lower with CART than C4.5, so under this measure, C4.5 is **better**

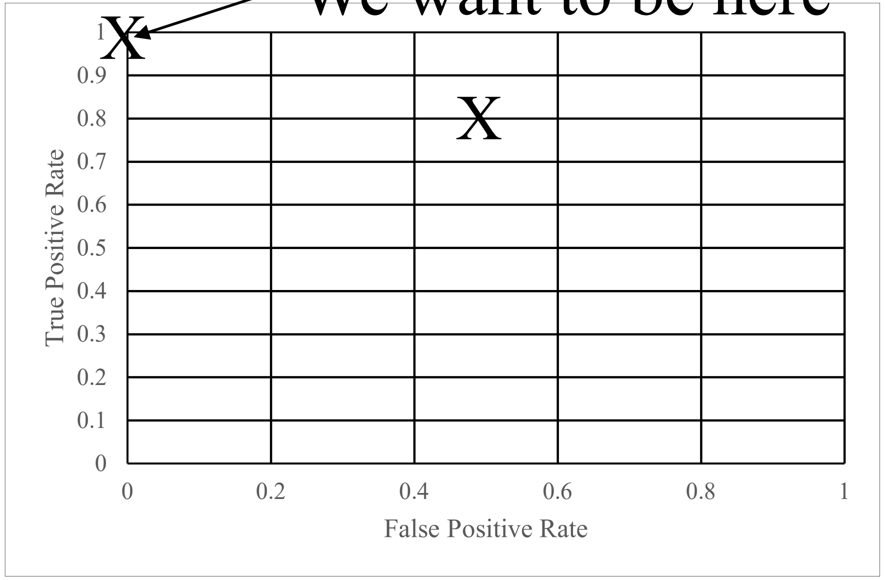
The Receiver Operator Curve

Actual	Predicted	Prob	
+ve	+ve	0.982	0.018
+ve	+ve	0.97	0.03
+ve	+ve	0.969	0.031
+ve	+ve	0.873	0.127
-ve	+ve	0.87	0.13
-ve	+ve	0.6	0.4
+ve	-ve	0.276	0.724
-ve	-ve	0.099	0.901
-ve	-ve	0.09	0.91

When we make a prediction, we use a decision boundary to make it *(assume 0 is "positive" or a "success")*

P(class=0)>0.5 then class 0

We want to be here



This leads to a particular contingency table and TPR, FPR

	Actual	
Predicted	+	-
+	**4**	**2**
-	**1**	**2**

TPR=0.8 FPR=0.5

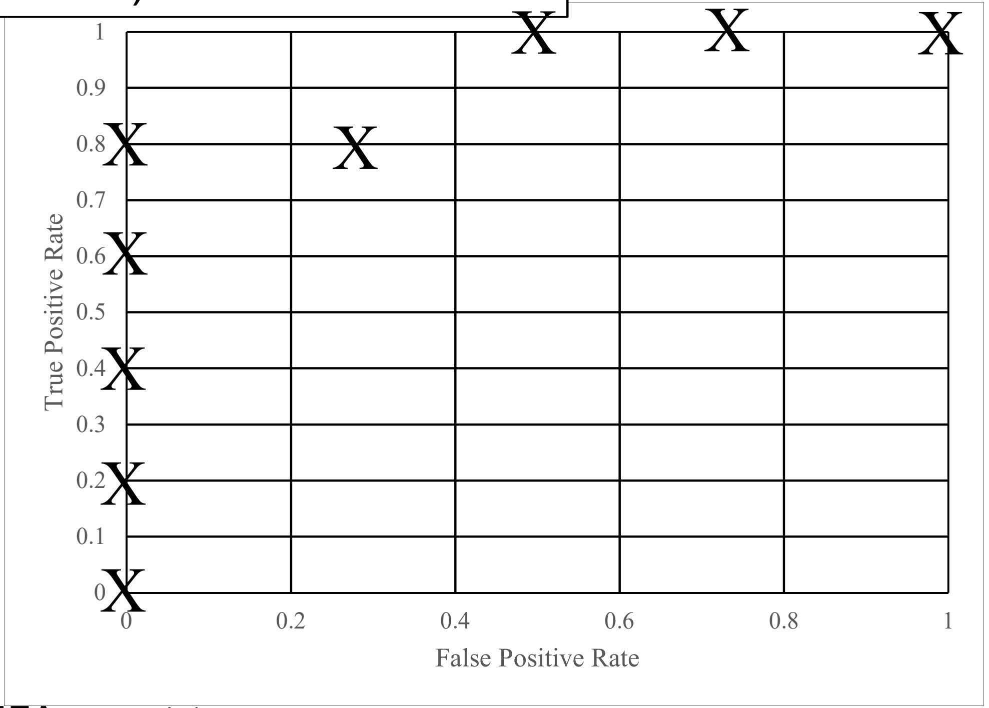
If we changed the boundary, we would get a different TPR and FPR

ROC

Actual	Predicted	Prob	
+ve	+	0.982	0.018
+ve	+	0.97	0.03
+ve	+	0.969	0.031
+ve	+	0.873	0.127
-ve	+	0.87	0.13
-ve	+	0.6	0.4
+ve	+	0.276	0.724
-ve	+	0.099	0.901
-ve	+	0.09	0.91

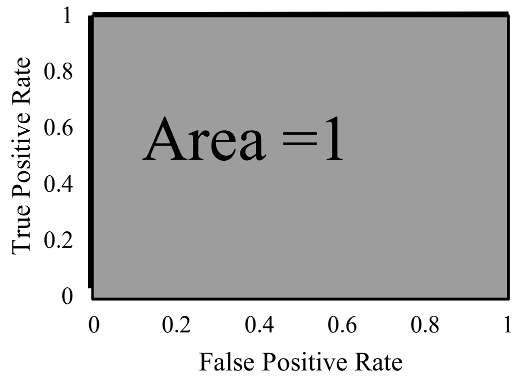
P(class=+ve)>1 then class +ve	TPR=0	FPR=0
P(class=+ve)>0.98 class +ve	TPR=0.2	FPR=0
P(class= +ve)>=0.97 class +ve	TPR=0.4	FPR=0
P(class= +ve)>=0.96 class +ve	TPR=0.6	FPR=0
P(class= +ve)>=0.873 class +ve	TPR=0.8	FPR=0
P(class= +ve)>=0.87 class +ve	TPR=0.8	FPR=0.25
P(class= +ve)>=0.6 class +ve	TPR=0.8	FPR=0.5
P(class= +ve)>=0.27 class +ve	TPR=1	FPR=0.5

	Actual	
Pred	+ve	-ve
+ve	**5**	**4**
-ve	**0**	**0**



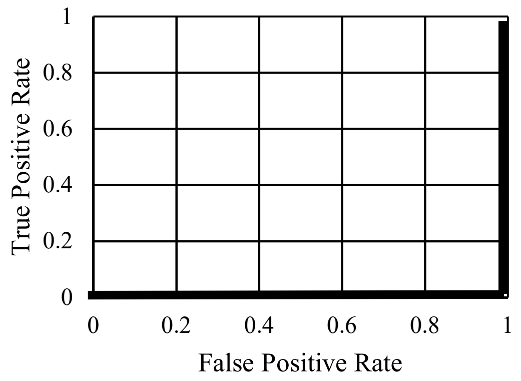
Area Under the ROC Curve

A Perfect ROC Curve (100% accuracy)

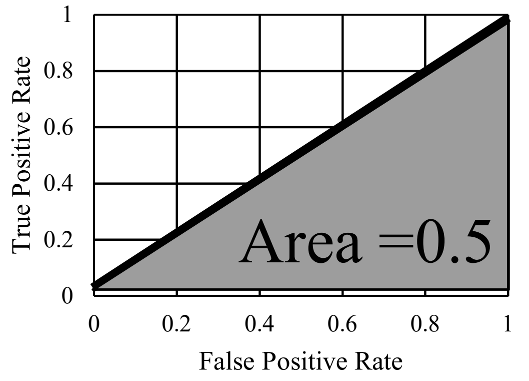


We can quantify how good the curve is by working out the area under the curve

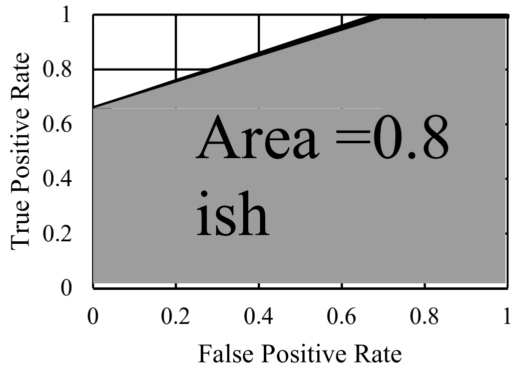
A Completely Incorrect ROC Curve (100% error)



A Random ROC (50% accuracy)



A "Good" ROC Curve



Classifier Evaluation Summary

Prediction Based:

Accuracy/Error:

- This should always be your starting point. There is no reason **not** to report it, even if you include other results

Balanced Accuracy/Error:

- There is no harm in reporting this, although it is only really relevant if you have a large number of classes or large class imbalance

Specificity/Sensitivity/Recall/F1:

- If you have a two class problem with class imbalance, then sure, use them.
- Useful for problems such as disease diagnosis where there is an obvious "success" class. However, many problems do not have this characteristic.
- In some fields (e.g. deep learning) too much credence is placed in them. F1 is IMO just weird. It has no easy interpretation and completely ignores performance on "negative" class.

Classifier Evaluation Summary

Probability Based:

NLL:

Useful, but there are some caveats.

1. Not all classifiers produce good probability estimates (e.g. 1-NN). They are not meant to. If you compare performance by probabilities you are biasing against them.
2. If the probability of the true class is zero for just one case, then the NLL is infinite. We hack to avoid this, but it can skew the overall results: one bad prediction does not mean the classifier is overall bad

AUC/AUROC

Popular, and useful, but assumes a two class problem. We average over 1-vs-all for multi-class problems, but that can be problematic (for example, should we weight by number of instances?)

Comparing Classifiers

1. Load single data set into Instances.
2. Create train/test split
3. Build classifiers on train *(including any model selection)*
4. Predict all test
5. Save all results in standard format

1. Work out stats from results file and report

Working example. Which decision tree classifier is best?

Some of the DTs in Weka: ID3, J48, SimpleCart, BFTree, LMT, NBTree

Evaluation Example

1. Load single data set into Instances.
2. Randomly partition into train/test

```
Instances all=DatasetLoading.loadData(basePath+problem+"//"+problem);
Instances[] split= InstanceTools.resampleInstances(all,fold,0.5);
```

3. Build classifiers on train

```
 Classifier c45=new J48();
C45.buildClassifier(split[0]);
```

4. Predict all test

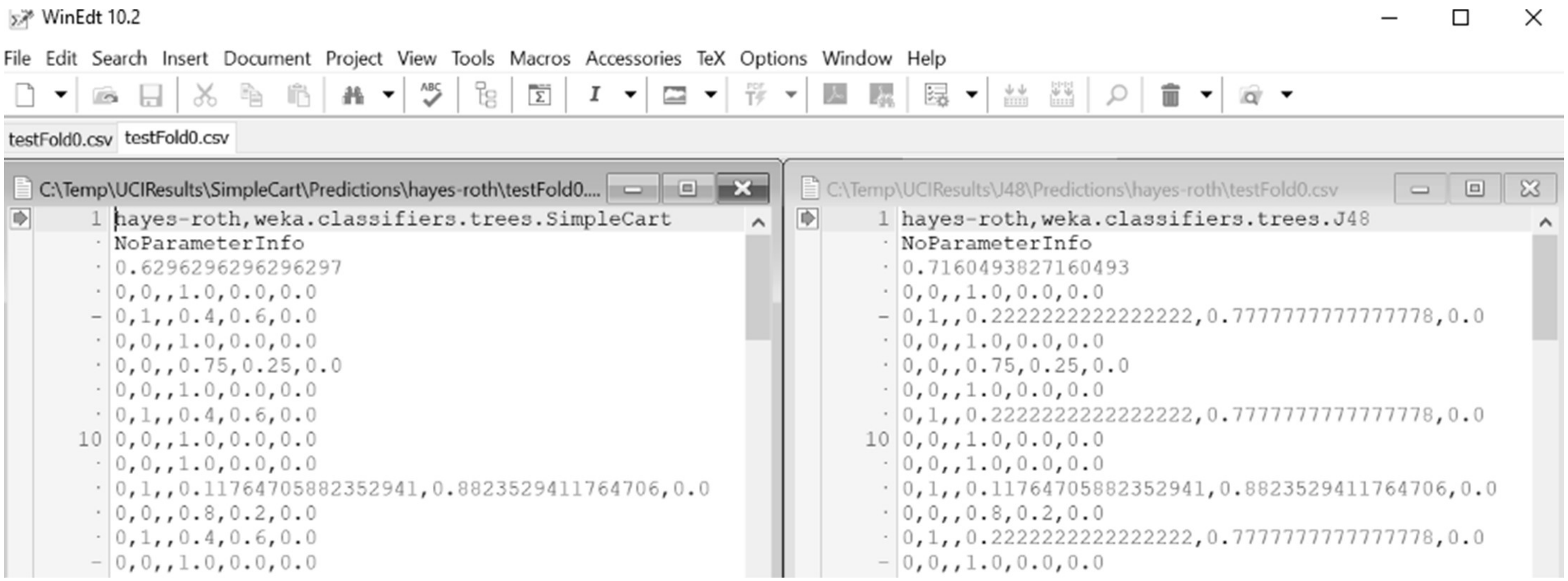
```
 for(int i=0;i<split[1].numInstances();i++){
            Instance tr=split[1].instance(i);
            actual[i]=(int)tr.classValue();
            predicted[i]=(int)a.classifyInstance(tr);
            probs[i]=a.distributionForInstance(tr);

        }
```

5. Save to file

Example

Compare C4.5 and CART on a single problem, hayes-roth


WinEdt 10.2
File Edit Search Insert Document Project View Tools Macros Accessories TeX Options Window Help
testFold0.csv testFold0.csv

C:\Temp\UCIResults\SimpleCart\Predictions\hayes-roth\testFold0....

```
hayes-roth,weka.classifiers.trees.SimpleCart
NoParameterInfo
0.6296296296296297
0,0,,1.0,0.0,0.0
0,1,,0.4,0.6,0.0
0,0,,1.0,0.0,0.0
0,0,,0.75,0.25,0.0
0,0,,1.0,0.0,0.0
0,1,,0.4,0.6,0.0
0,0,,1.0,0.0,0.0
0,0,,1.0,0.0,0.0
0,1,,0.11764705882352941,0.8823529411764706,0.0
0,0,,0.8,0.2,0.0
0,1,,0.4,0.6,0.0
0,0,,1.0,0.0,0.0
```

C:\Temp\UCIResults\J48\Predictions\hayes-roth\testFold0.csv

```
hayes-roth,weka.classifiers.trees.J48
NoParameterInfo
0.7160493827160493
0,0,,1.0,0.0,0.0
0,1,,0.2222222222222222,0.7777777777777778,0.0
0,0,,1.0,0.0,0.0
0,0,,0.75,0.25,0.0
0,0,,1.0,0.0,0.0
0,1,,0.2222222222222222,0.7777777777777778,0.0
0,0,,1.0,0.0,0.0
0,0,,1.0,0.0,0.0
0,1,,0.11764705882352941,0.8823529411764706,0.0
0,0,,0.8,0.2,0.0
0,1,,0.2222222222222222,0.7777777777777778,0.0
0,0,,1.0,0.0,0.0
```

```
ClassifierResults results=new ClassifierResults();
results.loadFromFile("fullPathForAResultFile");
results.findAllStats();
```

C4.5 vs CART on Single hayes-roth split

Statistic	C45	CART
Accuracy	0.716	0.6296
BalancedAccuracy	0.7544	0.6086
NLL	0.9883	1.4809
MeanAUROC	0.8206	0.7477

C45 is better on every measure (remember we want small NLL)

C4.5	0	1	2
0	19	6	1
1	18	24	2
2	1	2	15

CART	0	1	2
0	19	6	6
1	13	24	2
2	1	2	8

C45 does better at predicting class 2 than CART, hence the greater difference in balanced accuracy than in accuracy

Comparing Classifiers on a Single Data Set

C4.5 *seems* to be better than CART on this particular problem, but what can we actually infer from these results?

We get more information if we split the data multiple times and get multiple measurements of performance

This gives us the potential to test whether there is a significant difference between the classifiers

Given a finite amount of data, how do we create multiple train/test splits?

C4.5 vs CART on single split

Suppose we claim c4.5 is better than CART

1. Perhaps it was just unlucky with the particular split we chose from the data
2. Perhaps it is just this problem that CART does badly on, but over many problems it may be better on average (and hence be a better algorithm)

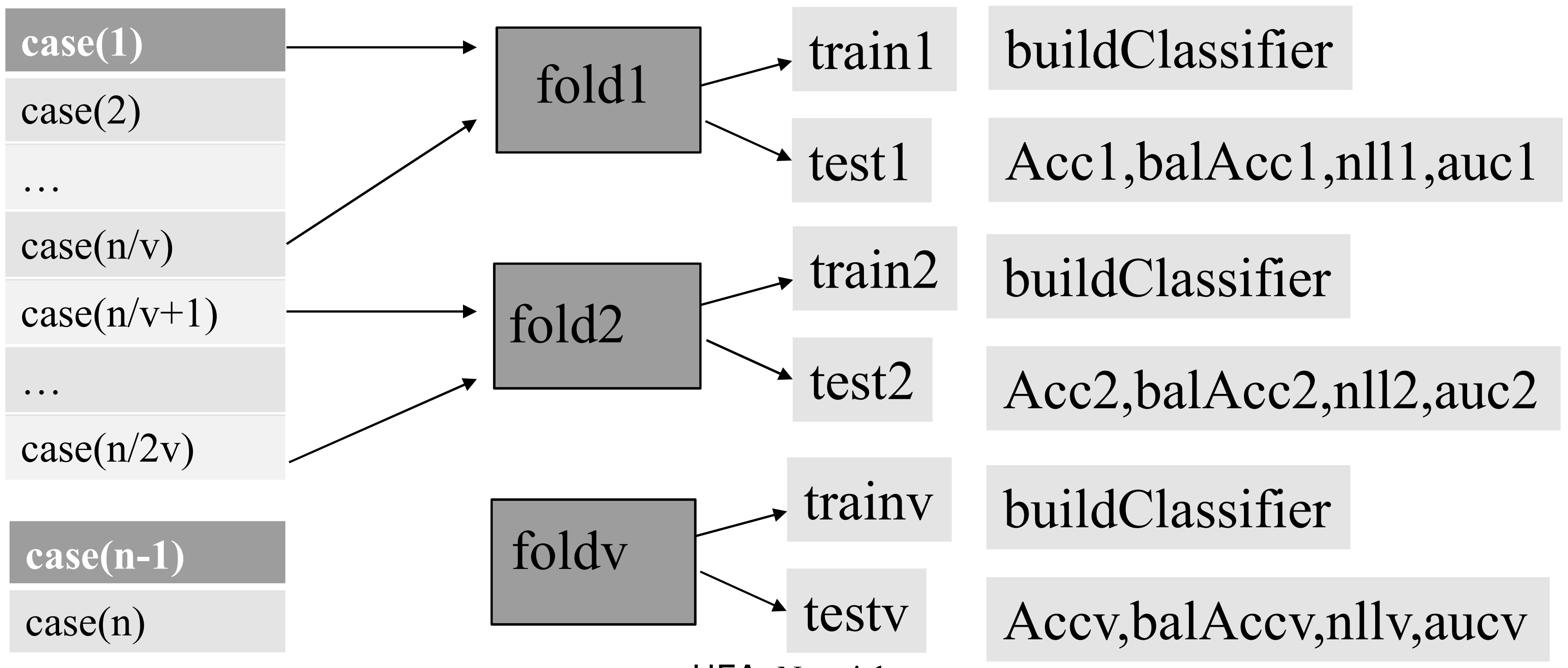
There are different ways of generating multiple train/test splits:

1. Partition into independent data sets
2. Cross validation
3. Sampling without replacement
4. *Sampling with replacement (bootstrapping) see next lecture on ensembles*

Partitioning the data

Suppose we want to create *v* separate train/test splits

We could split D into *v* disjoint sets, then further split each disjoint set into train/test splits



Partitioning Pros/Cons

Pros:

- No case appears in more than one training fold or testing fold.
- **This means each fold is independent of all others**.
- This means the assumptions behind tests of significance are valid

Cons:

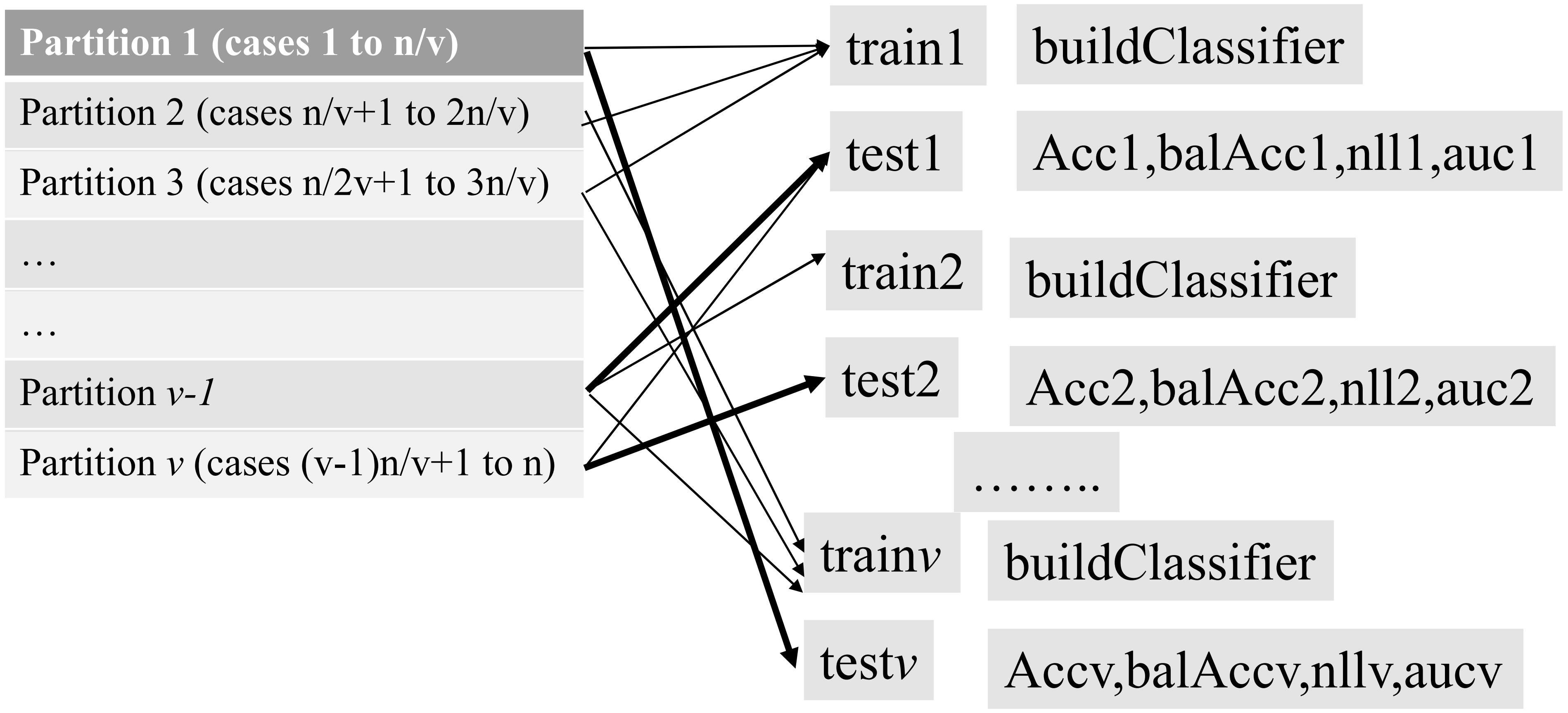
- **Each train fold is much smaller than the totality of data available.**
- As a rule, more data means higher accuracy and less variability.

This is only feasible with very large data and in practice is almost never done. The alternative is to use each data in more than one fold. **The crucial thing to ensure is no data appears in the same train and test fold**

Cross Validation

Suppose we want to create *v* separate train/test splits

1. Partition the data into *v* sets
2. Use *v-1* partitions as training data for each fold
3. Use the left out partition as test data for each fold



Cross Validation for a Single Problem

- For a single problem, it is sensible to cross validate, then aggregate the test folds
- Remember, if you are performing any model selection, it must be done on every single train fold
- The number of folds to use, *v,* is a pragmatic decision. The most extreme version is to perform a **leave-one-out cross validation** (use n folds, where n is the number of instances)
- A reasonable default is to use ten fold CV
- The problem itself may define a sensible fold structure (leave one bottle out, leave one bag out etc)
- If you want to perform statistical tests for a single problem, resampling is probably better

Cross Validation for a Single Problem

Classifier 1

test1

test2

test*v*

$\longrightarrow$

Accuracy, Balanced Accuracy, Contingency Table, NLL, ROC Curve, Area Under the ROC curve

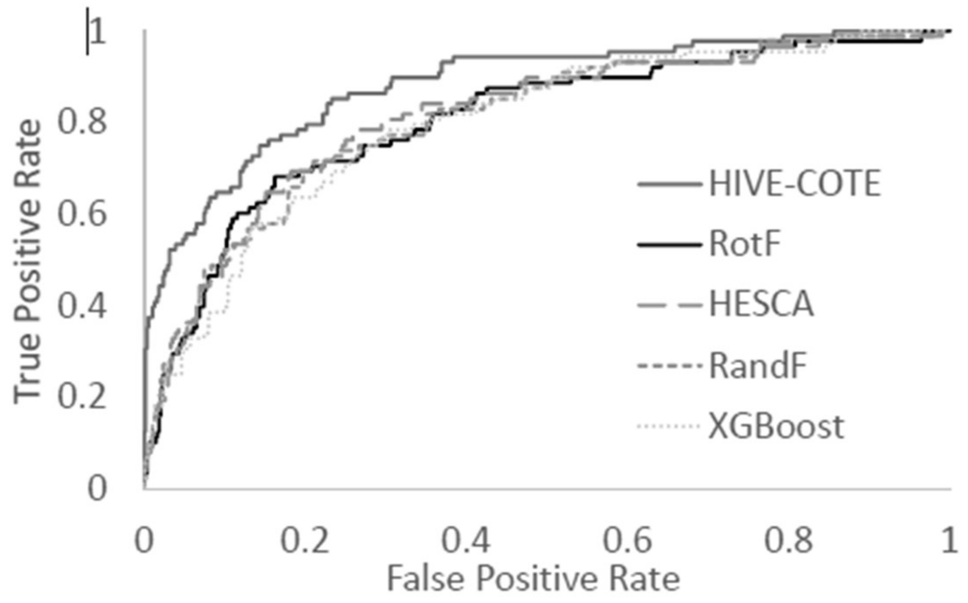
$\longleftarrow$

Classifier 2

test1

test2

test*v*



See detecting-forged-alcohol.pdf and finding-electric-devices.pdf on blackboard for examples of single problem analysis

Cross Validation Pros/Cons

Pros:

- More training data will give a better measure of the overall quality of the classifier
- Pretty standard approach, especially for model selection (see later)

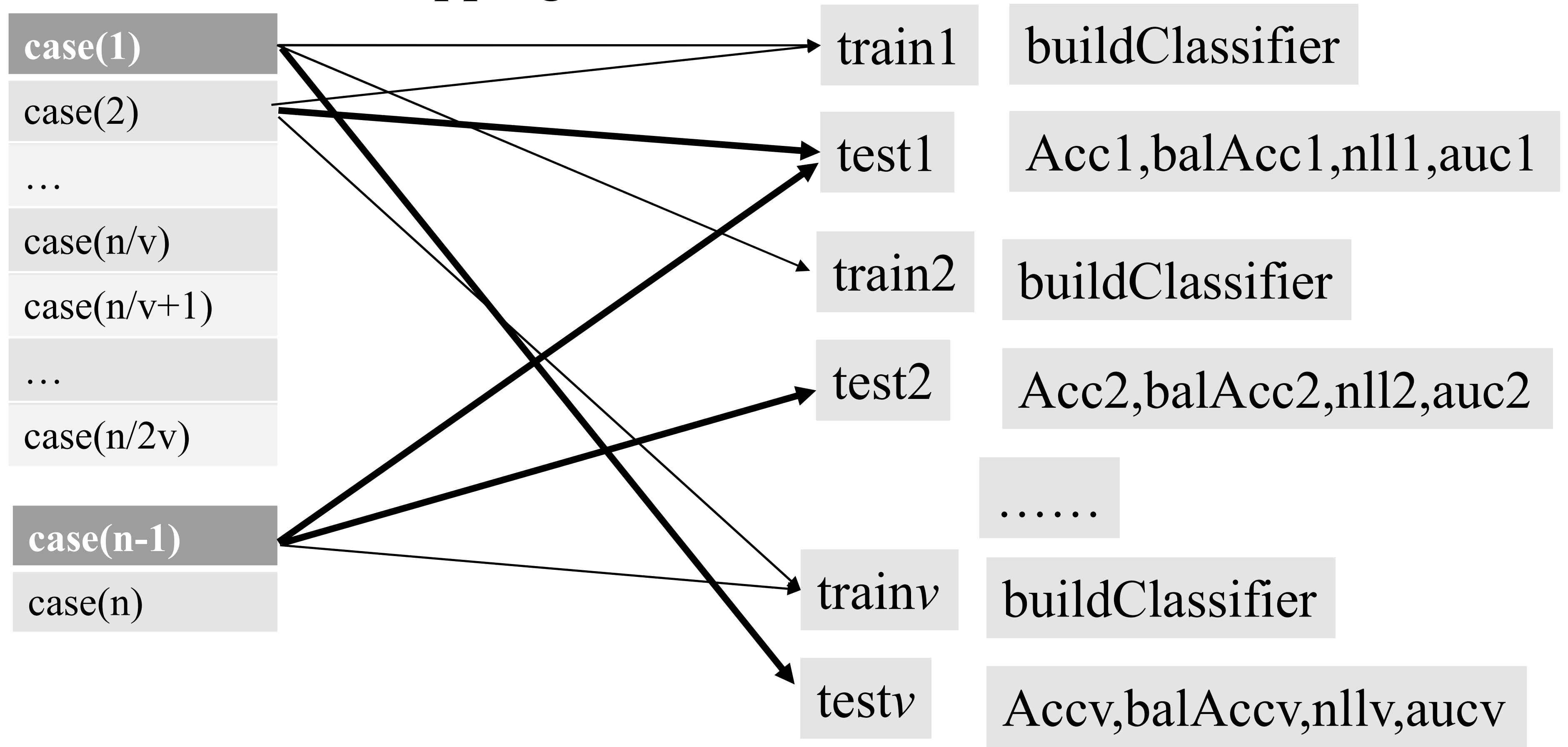
Cons:

- Folds no longer independent, therefore hypothesis tests on dodgy ground
- As a rule, more data means higher accuracy and less variability.
- With small data you cannot do many samples. This makes it hard to standardise the approach over different data
- Fiddly to implement (different size for the last partition)

Sampling without Replacement

Suppose we want to create *v* separate train/test splits

1. Randomly split the data into distinct sets *v* times, no overlapping train and test data sets



Sampling without Replacement

Pros:

Simple to implement and easy to generalise over multiple problems
Better for comparing classifiers

Cons:

If comparing classifiers it is important to use the same splits for each classifier
Samples not independent
Cannot aggregate into a single set of results

Comparing Classifiers

- We have a mechanism for creating splits and statistics for measuring performance, we need to clarify how to compare classifiers

1. Two classifiers on a single problem
2. Multiple classifiers on a single problem
3. Two classifiers on multiple problems
4. Multiple classifiers on multiple problems

1.Two classifiers on a single problem

- If we cross validate or resample a single problem *v* times, we can then generate *v* test files and calculate *v* statistics for each classifier
- Since they are evaluated on the same test data, we can consider the differences

Fold/ Resample	Classifier 1	Classifier 2	Difference
1	a_1	b_1	a_1-b_1=d_1
2	a_2	b_2	d_2
3	a_3	b_3	d_3
V-1	a_v-1	b_v-1	d_v-1
v	a_v	b_v	d_v

The question: is the average difference significantly different to zero?

Two Sample Paired Tests

- The null hypothesis is that the average difference is zero
- The data is the series of differences

d1	**d2**	**d3**							**dv**

- Two relevant statistical tests:

1. Paired-sample t-test. This is a parametric test based on the sample mean

2. Wilcoxon signed rank test. This is a non parametric test based on the ranks of the two classifiers

Paired 2-sample Student t-test

Null hypothesis H0: population mean difference is zero

Alt Hypothesis H1 : population mean difference is not zero

Test statistic $$t = \frac{\bar{d}}{s/\sqrt{v}}$$

$$\bar{d} = \frac{\sum_{i=1}^{v} d_i}{v}$$ Sample mean

$$s = \sqrt{\frac{\sum_{i=1}^{v}(d_i - \bar{d})^2}{v-1}}$$ Sample variance

- If the null hypothesis is true we are unlikely to observe large positive or negative values of *t*

T-Test Decision

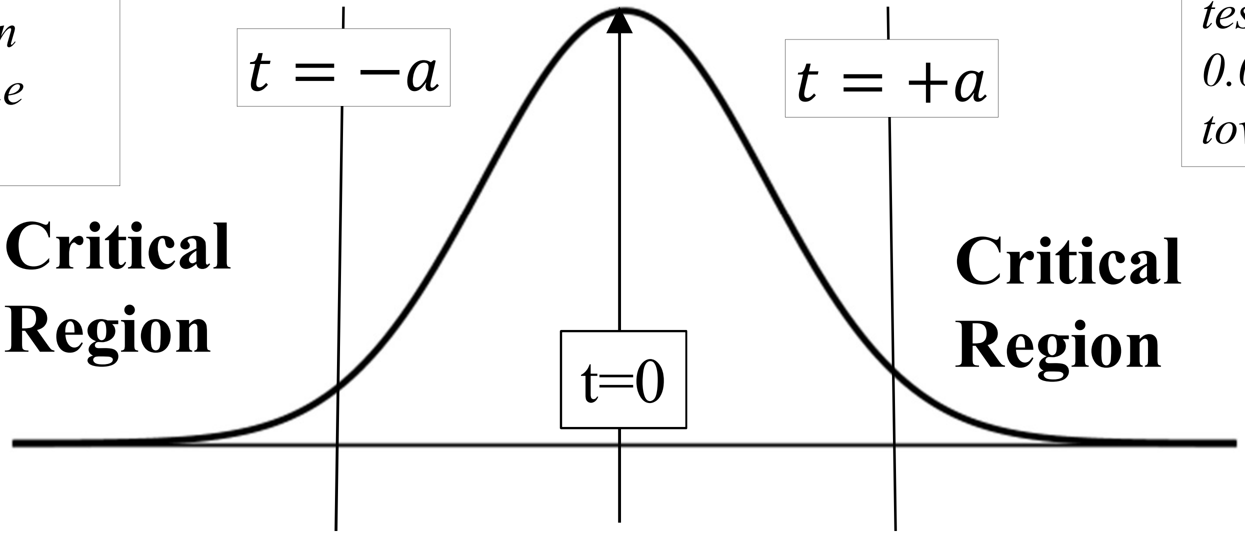
Null Hypothesis H0: population mean difference is zero

Alt Hypothesis H1: population mean difference is not zero

Critical Region: If the null hypothesis is true, and we repeated the experiment multiple times on independent data, then only 5% of these experiments would give a t statistic that lies in the these regions.

a is dependent on alpha, the level of the test (5% in above) and the sample size v

For a two tailed test and alpha 0.05, t tends towards 1.96



If we observe a statistic in the critical region we can say there is evidence that the mean difference is not zero.

https://archive.ics.uci.edu/ml/datasets/Hayes-Roth

H0: there is no difference in mean accuracy between CART and C4.5 on hayes-roth

H1: there is a difference in mean accuracy between CART and C4.5 on hayes-roth

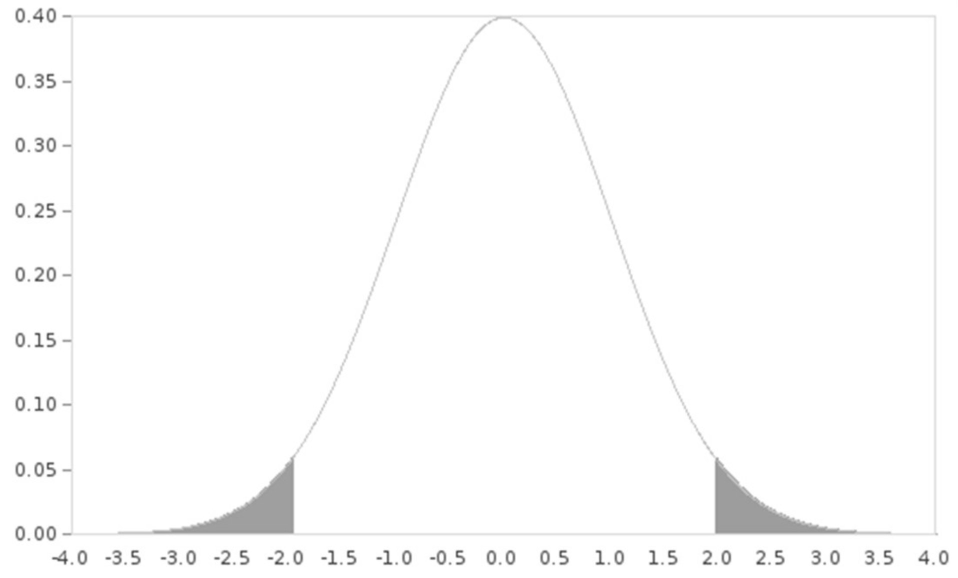
Experiment: resample 30 times, set the level of the test to 0.05, perform two tailed test (do not assume a priori one is better than the other), find differences in test accuracy (assume CART-C4.5, so positive means CART better)

https://mathcracker.com/t_critical_values.php

tc=−2.04 and *tc*=2.04

If we observe *t* less than -2.04, there is evidence C4.5 is better

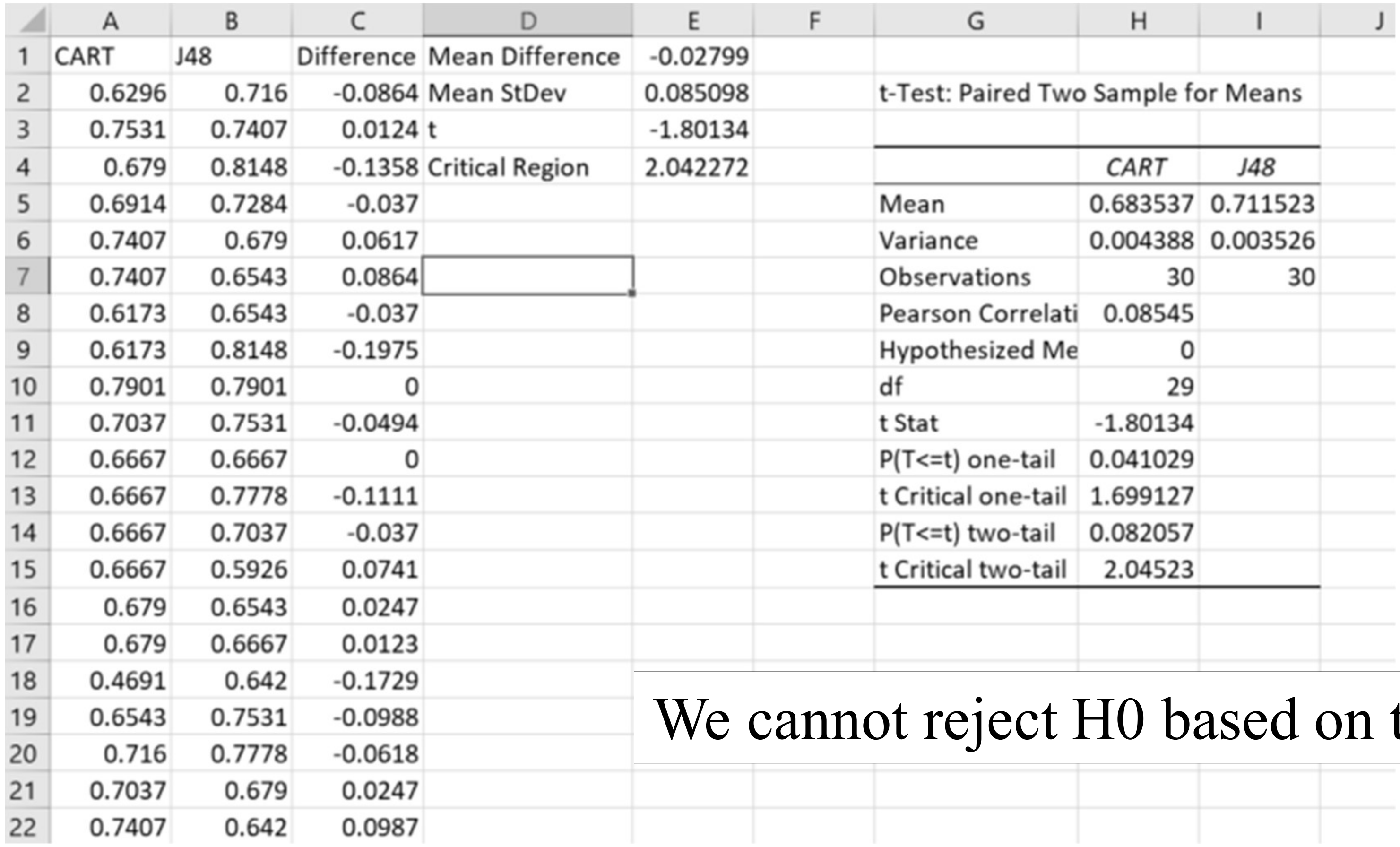
If we observe *t* greater than 2.04, there is evidence CART is better



hayes-roth Experiment

<u>H0</u>: there is no difference in mean accuracy between CART and C4.5 on hayes-roth

<u>H1</u>: there is a difference in mean accuracy between CART and C4.5 on hayes-roth



	A	B	C	D	E	F	G	H	I	J
1	CART	J48	Difference	Mean Difference	-0.02799					
2	0.6296	0.716	-0.0864	Mean StDev	0.085098		t-Test: Paired Two Sample for Means			
3	0.7531	0.7407	0.0124	t	-1.80134					
4	0.679	0.8148	-0.1358	Critical Region	2.042272			*CART*	*J48*	
5	0.6914	0.7284	-0.037				Mean	0.683537	0.711523	
6	0.7407	0.679	0.0617				Variance	0.004388	0.003526	
7	0.7407	0.6543	0.0864				Observations	30	30	
8	0.6173	0.6543	-0.037				Pearson Correlati	0.08545		
9	0.6173	0.8148	-0.1975				Hypothesized Me	0		
10	0.7901	0.7901	0				df	29		
11	0.7037	0.7531	-0.0494				t Stat	-1.80134		
12	0.6667	0.6667	0				P(T<=t) one-tail	0.041029		
13	0.6667	0.7778	-0.1111				t Critical one-tail	1.699127		
14	0.6667	0.7037	-0.037				P(T<=t) two-tail	0.082057		
15	0.6667	0.5926	0.0741				t Critical two-tail	2.04523		
16	0.679	0.6543	0.0247							
17	0.679	0.6667	0.0123							
18	0.4691	0.642	-0.1729							
19	0.6543	0.7531	-0.0988							
20	0.716	0.7778	-0.0618							
21	0.7037	0.679	0.0247							
22	0.7407	0.642	0.0987							

We cannot reject H0 based on this evidence

Wilcoxon Signed Rank Test Example

Null hypothesis H0: population median difference is zero

Alt Hypothesis H1 : population median difference is not zero

Procedure.

1. Remove all with zero difference
2. Rank all differences by the absolute value. Ties are assigned average ranks
3. Sum the ranks for all cases with a positive difference and all cases with a negative difference

If the null hypothesis is true, we would expect the ranks of the positive values to be about the same as those with a negative difference

http://www.socscistatistics.com/tests/signedranks/Default2.aspx

2. Multiple classifiers on a single problem

Fold	Classifier A	Classifier B	Classifier C	…			Classifier X
1	a_1	b_1					
2	a_2	b_2					
3	a_3	b_3					
v-1	a_v-1	b_v-1					
v	a_v	b_v					

The question: is there *any* difference between the classifiers?

The follow up question: is the best classifier significantly better than the others?

Multiple Classifier Tests on a Single Problem

Null hypothesis H0: no difference in averages between classifiers

Alt Hypothesis H1 : there is a difference somewhere

Parametric Test: Analysis of Variance via F-Test

Non-Parametric Test: Friedman Test

- However, we on dodgy ground because our samples are not independent.
- Pairwise tests on a single dataset are reasonable in an exploratory sense, but you must not conclude too much from them unless you are partitioning the data.
- We are generally more interested in classifier performance over multiple data sets

Comparing Classifiers over Multiple Datasets

- For each dataset, resample and average. This will reduce the variance in the estimate
- Form a big table of results of these averages

	J48	SimpleCart	FT	HoeffdingTree	LADTree	NBTree	REPTree	DecisionStump	J48graft
abalone	0.60	0.63	0.63	0.58	0.62	0.62	0.62	0.57	0.59
acute-inflammation	1.00	0.98	1.00	0.82	0.99	1.00	0.95	0.80	0.99
hayes-roth	0.71	0.68	0.55	0.40	0.69	0.56	0.62	0.43	0.71
bank	0.89	0.90	0.88	0.43	0.89	0.89	0.89	0.88	0.88
car	0.95	0.95	0.89	0.56	0.90	0.92	0.93	0.70	0.95
monks-1	0.81	0.79	0.81	0.50	0.76	0.75	0.79	0.69	0.94
breast-cancer	0.70	0.70	0.69	0.71	0.69	0.70	0.70	0.69	0.69

- Rather than average over problems, we look instead at the **ranks**

Comparing Classifiers over Multiple Datasets

	0.81	0.81	0.78	0.57	0.79	0.78	0.79	0.68	0.83
	J48	SimpleCart	FT	HoeffdingTree	LADTree	NBTree	REPTree	DecisionStump	J48graft
abalone	0.60	0.63	0.63	0.58	0.62	0.62	0.62	0.57	0.59
acute-inflammation	1.00	0.98	1.00	0.82	0.99	1.00	0.95	0.80	0.99
hayes-roth	0.71	0.68	0.55	0.40	0.69	0.56	0.62	0.43	0.71
bank	0.89	0.90	0.88	0.43	0.89	0.89	0.89	0.88	0.88
car	0.95	0.95	0.89	0.56	0.90	0.92	0.93	0.70	0.95
monks-1	0.81	0.79	0.81	0.50	0.76	0.75	0.79	0.69	0.94
breast-cancer	0.70	0.70	0.69	0.71	0.69	0.70	0.70	0.69	0.69

	3.00	**3.14**	**5.14**	**7.57**	**4.86**	**4.57**	**4.43**	**8.14**	**4.14**
	J48	SimpleCart	FT	HoeffdingTree	LADTree	NBTree	REPTree	DecisionStump	J48graft
abalone	6	1	2	8	3	5	4	9	7
acute-inflammation	2	6	2	8	5	2	7	9	4
hayes-roth	1	4	7	9	3	6	5	8	2
bank	5	1	6	9	3	2	4	8	7
car	1	3	7	9	6	5	4	8	2
monks-1	2	5	3	9	6	7	4	8	1
breast-cancer	4	2	9	1	8	5	3	7	6

J48 is ranked 6th on abalone

The average rank of NBTree is 4.57

Critical Difference Diagrams

Demsar* proposed a procedure for testing multiple classifiers

1. Is there a significant difference between the classifiers?

Perform a Friedman test to find if there is any difference

2. If so, where do these differences occur?

The performance of two classifiers is significantly different if the corresponding average ranks differ by at least the critical difference

For k classifiers over N problems

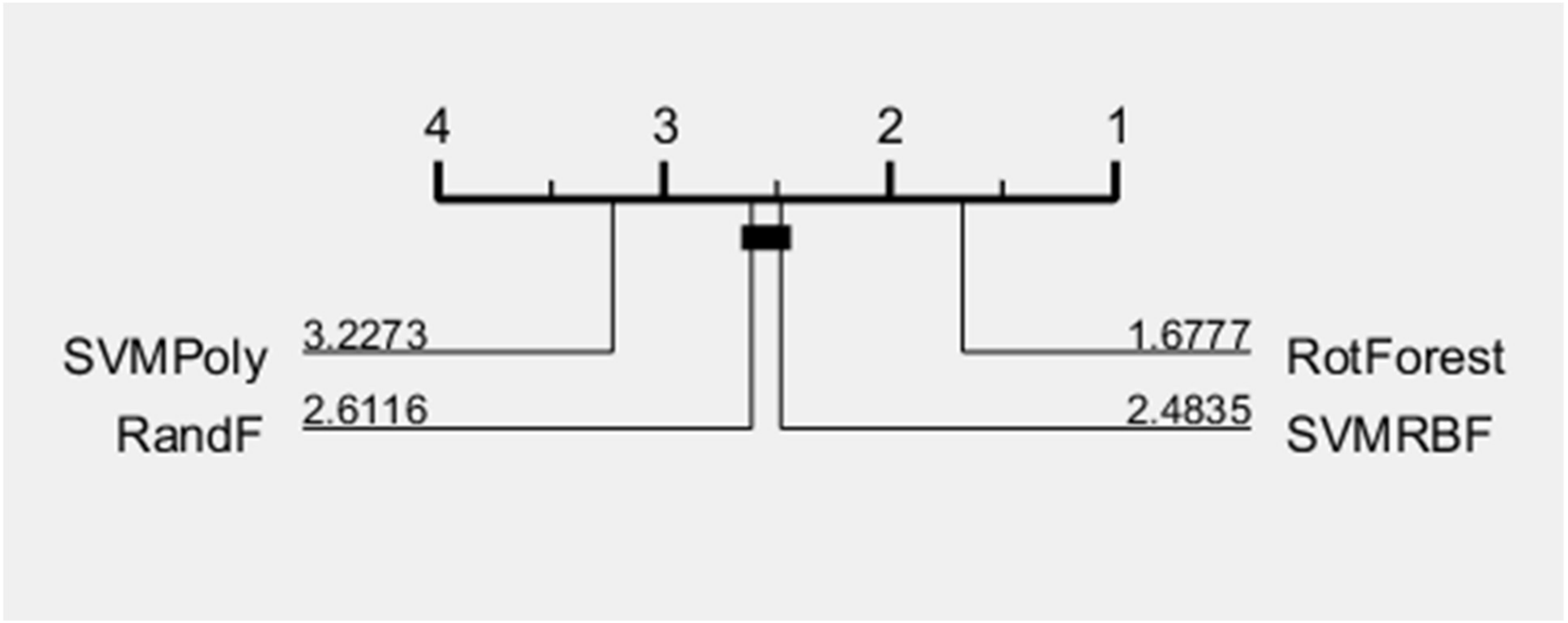
$$CD = q_\alpha \sqrt{\frac{k(k+1)}{6n}}$$

where q is the critical value

*J. Demsar, Statistical Comparisons of Classifiers over Multiple Data Sets, Journal of Machine Learning Research, 7 (2006), 1-30

Critical Difference Diagrams

Plot the average ranks on a number line, and form cliques of classifiers within which there is no significant difference

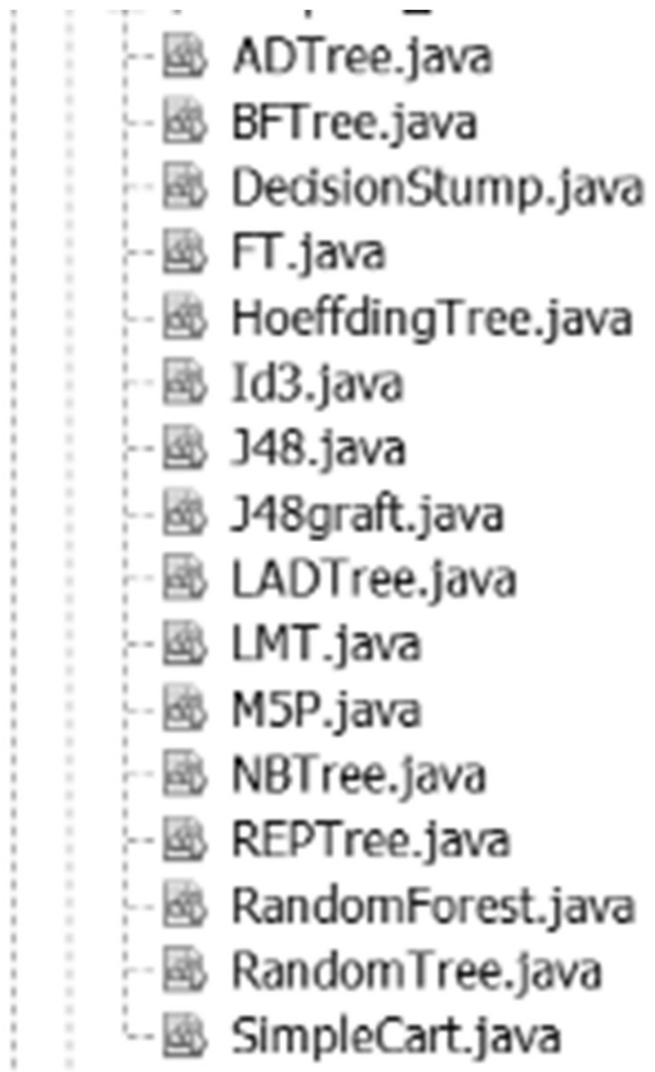


The performance of two classifiers is significantly different if the corresponding average ranks differ by at least the critical difference

*J. Demsar, Statistical Comparisons of Classifiers over Multiple Data Sets, Journal of Machine Learning Research, 7 (2006), 1-30

Comparing Decision Trees

Weka offers the following Decision Trees



```
ADTree.java
BFTree.java
DecisionStump.java
FT.java
HoeffdingTree.java
Id3.java
J48.java
J48graft.java
LADTree.java
LMT.java
M5P.java
NBTree.java
REPTree.java
RandomForest.java
RandomTree.java
SimpleCart.java
```

M5P is for regression only

ID3 only works with discrete data

NBTree and BFTree seem buggy

RandomForest and RotationForest are ensembles we cover next week

ADTree and LMT only work with two class problems

Which is the best of *"J48","SimpleCart","FT","HoeffdingTree","LADTree","REPTree","DecisionStump","J48graft"};*

Experimental Design

Compare 8 classifiers on 100 UCI problems formatted by Delgado *et al.**

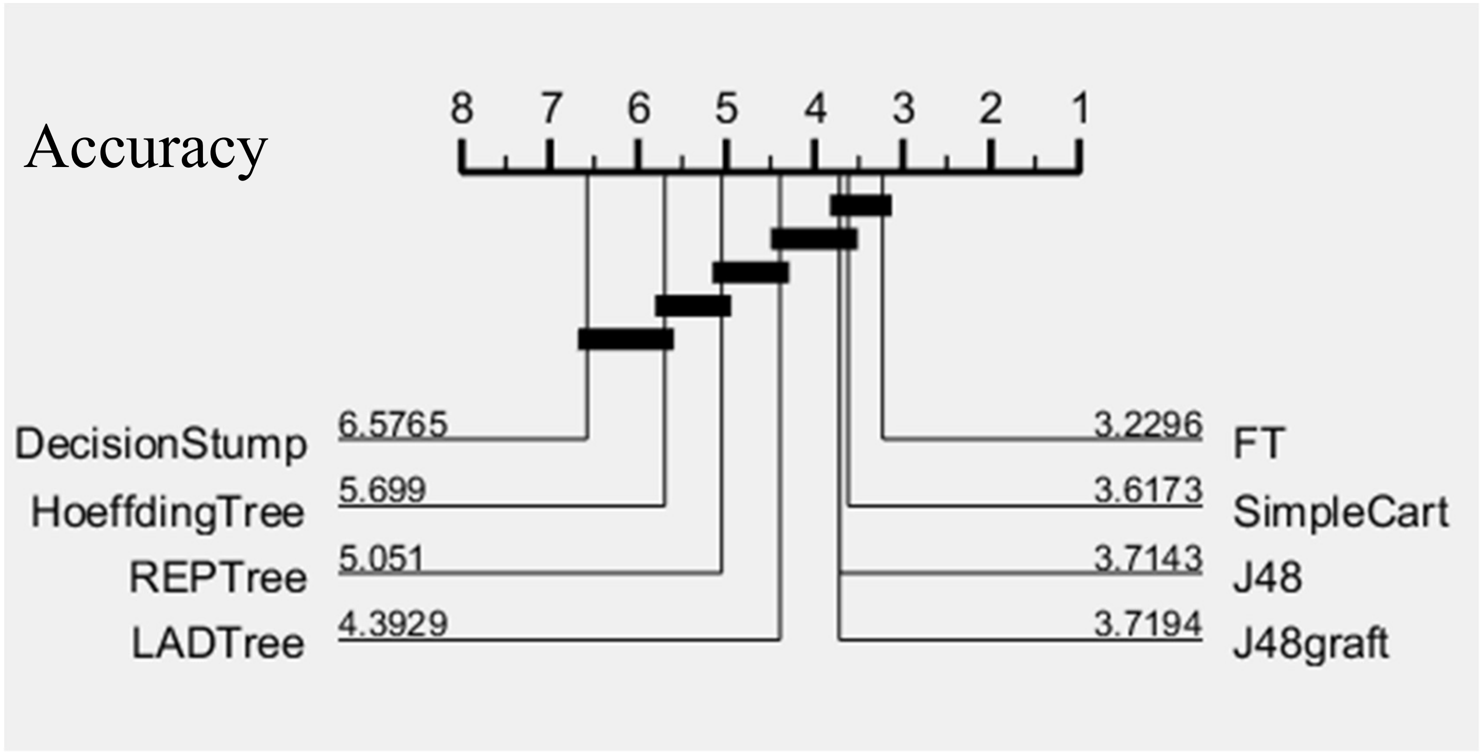
These problems only have continuous attributes and have no missing attributes

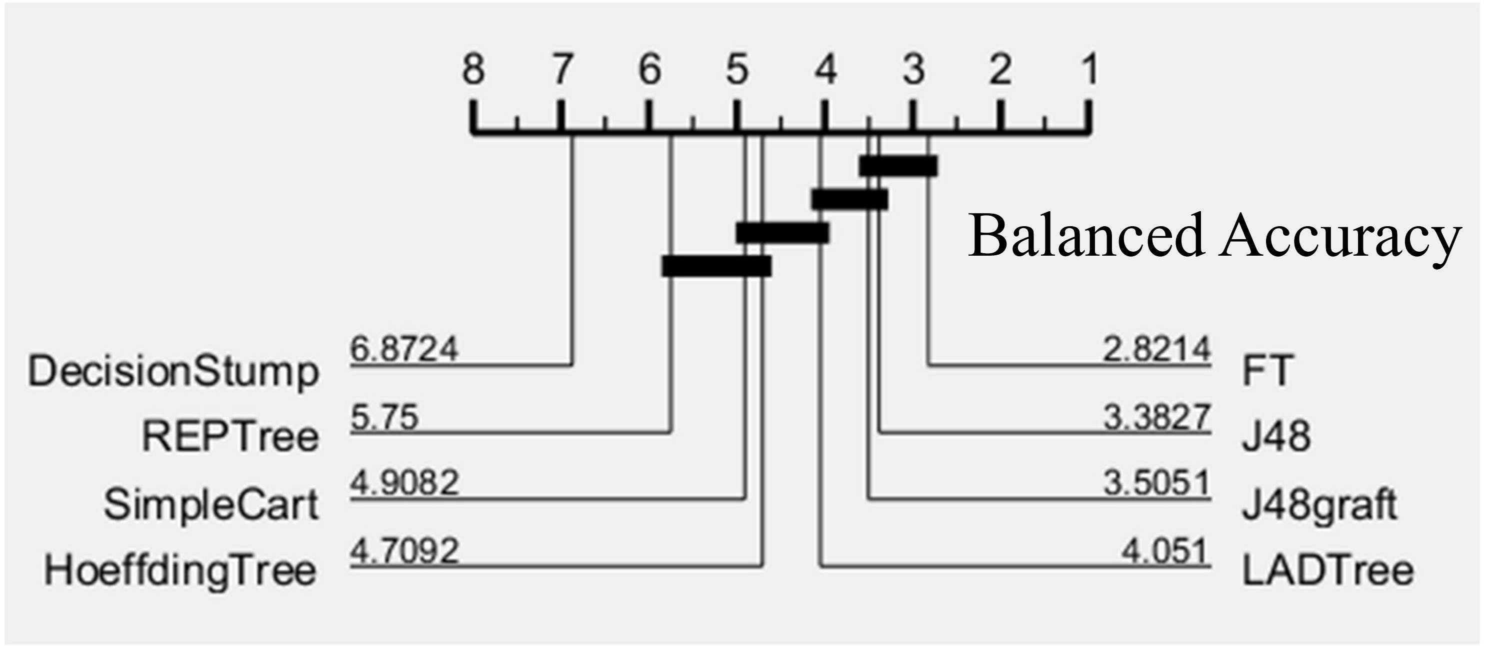
On each sample we measure Accuracy, Balanced Accuracy, NLL and AUROC

We average these statistics over 30 resamples on each data set with 50% train and 50% test. The same resamples used for each classifier

A priori, I expected few overall significant differences but suspected that C4.5 would be best.

*M. Fernandez-Delgado et al. , Do we need hundreds of classifiers to solve real world classification problems?, 15 (2014), 3133-3181
http://jmlr.org/papers/v15/delgado14a.html


Accuracy
8
7
6
5
4
3
2
1
DecisionStump 6.5765
HoeffdingTree 5.699
REPTree 5.051
LADTree 4.3929
FT 3.2296
SimpleCart 3.6173
J48 3.7143
J48graft 3.7194

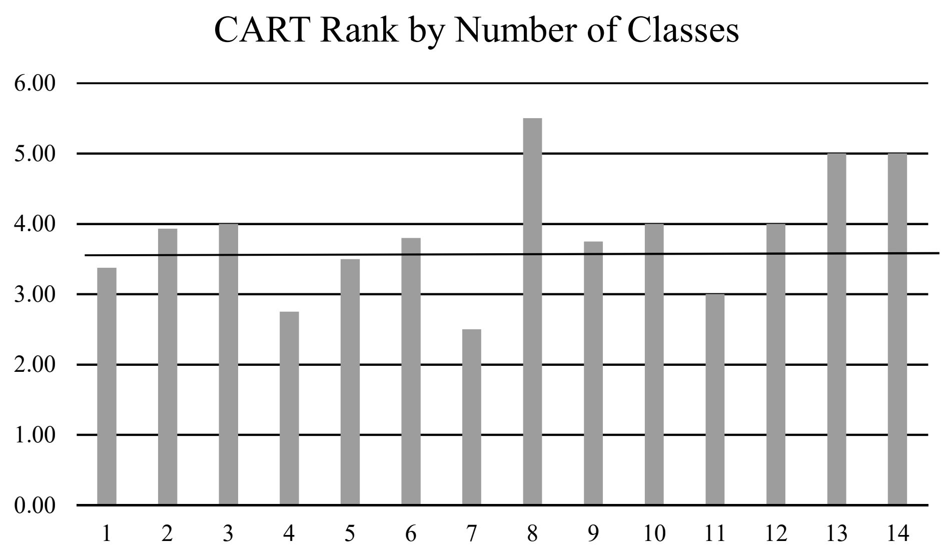

Balanced Accuracy
8
7
6
5
4
3
2
1
DecisionStump 6.8724
REPTree 5.75
SimpleCart 4.9082
HoeffdingTree 4.7092
FT 2.8214
J48 3.3827
J48graft 3.5051
LADTree 4.051

Discussion of Prediction Results

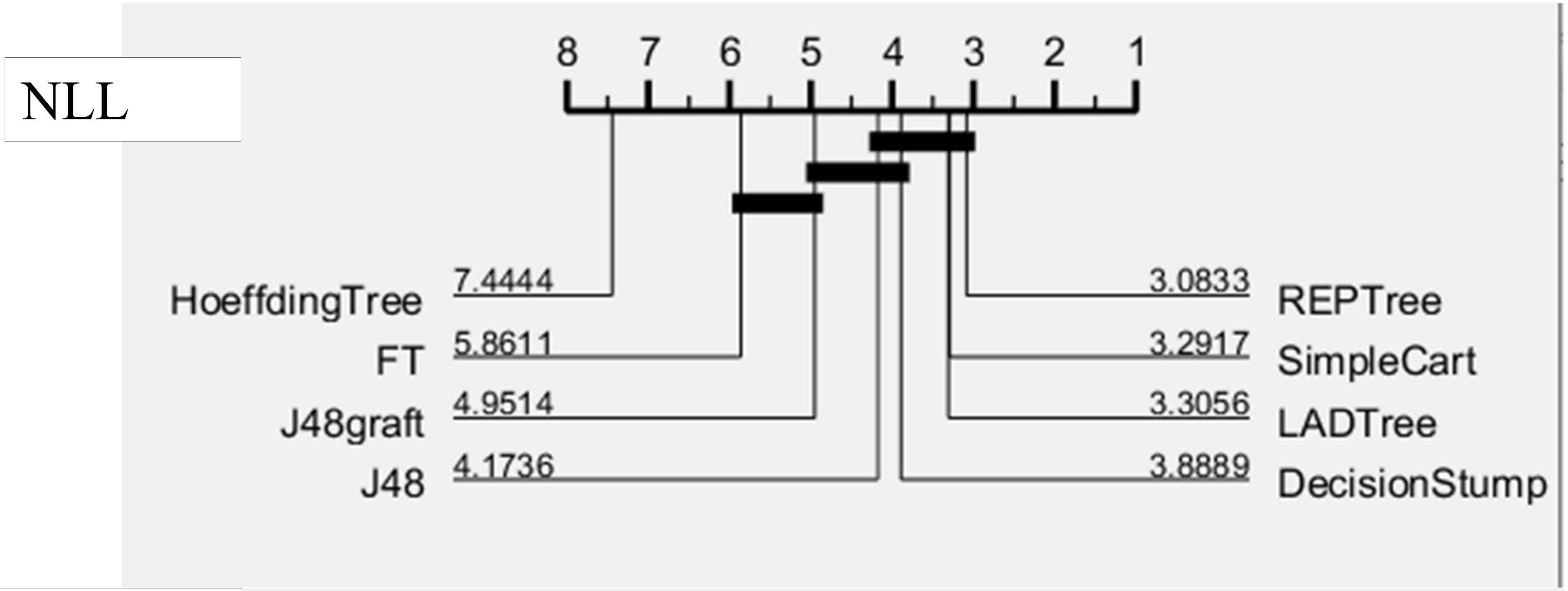
- The FT algorithm (Functional Trees) has the highest average rank, but is not significantly better than SimpleCART, J48, J48Graft
- Decision stump, RepTree and HoeffdingTree are significantly worse than the top clique. We could make a case for the former being worse …
- SimpleCART seems to do relatively worse on Balanced Accuracy, dropping from second place to fifth place. This indicates it may be less useful for problems with many classes or class imbalance. Does the data back this up?

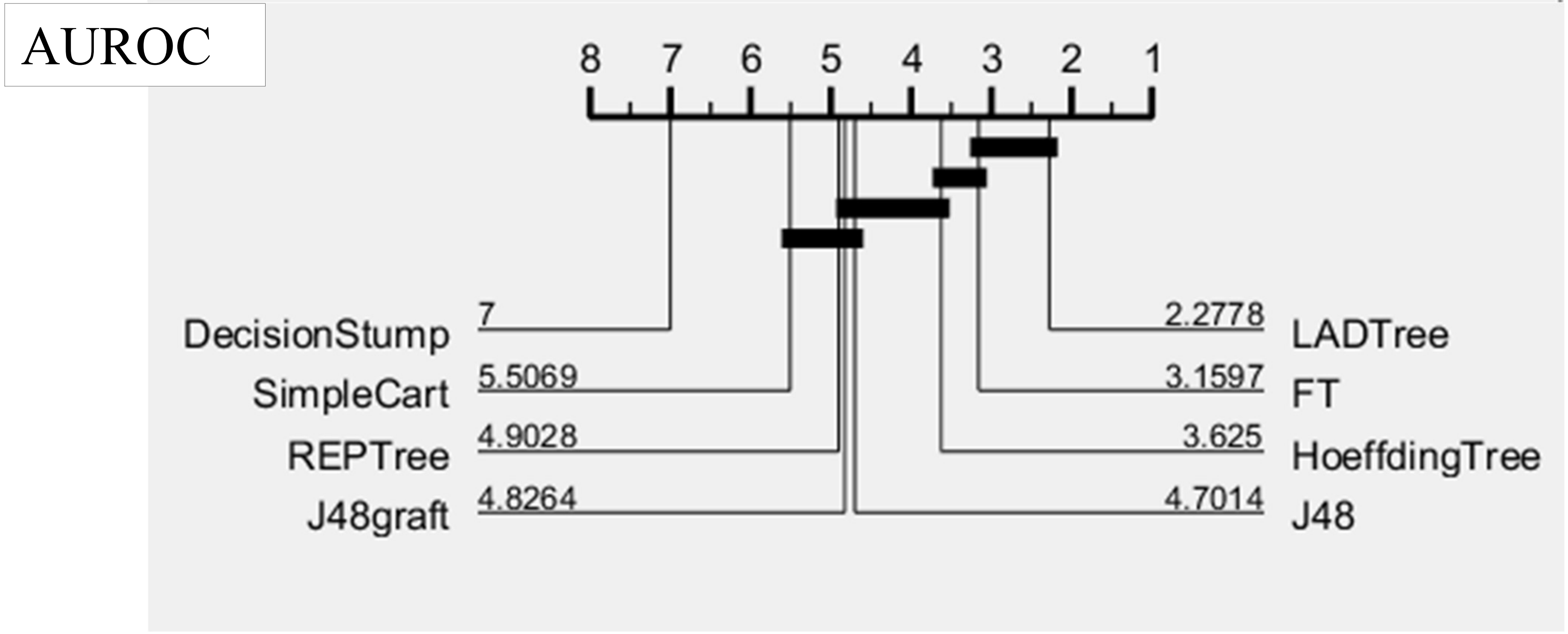
SimpleCART vs number of classes

	Overall	3.61
NumClasses	Rank	Count
2	3.38	44
3	3.93	21
4	4.00	3
5	2.75	4
6	3.50	6
7	3.80	5
8	2.50	2
9	5.50	1
10	3.75	4
11	4.00	1
13	3.00	1
15	4.00	2
18	5.00	2
100	5.00	2



No obvious pattern ……


NLL
8
7
6
5
4
3
2
1
HoeffdingTree 7.4444
FT 5.8611
J48graft 4.9514
J48 4.1736
REPTree 3.0833
SimpleCart 3.2917
LADTree 3.3056
DecisionStump 3.8889


AUROC
8
7
6
5
4
3
2
1
DecisionStump 7
SimpleCart 5.5069
REPTree 4.9028
J48graft 4.8264
LADTree 2.2778
FT 3.1597
HoeffdingTree 3.625
J48 4.7014

Discussion of Probability Results

- The FT algorithm is bad at producing probability estimates (NLL), but is good at ranking (AUROC). This could have implications if it were used in an ensemble
- Apart from that, the top clique is the same as with predictions
- LADTree does well on AUROC, whereas J48 drops down the ranking
- DecisionStump is in the top clique for NLL. This indicates the general principle that decision trees are bad at making probability predictions.

Overall Ranks

	Acc	BalAcc	NLL	AUROC	#Top Cliques
FT	**1**	**1**	7	**2**	3 out of 4
CART	**2**	6	**2**	7	2 out of 4
J48	**3**	**2**	**5**	4	3 out of 4
J48graft	**4**	**3**	6	5	2 out of 4
LADTree	5	4	**3**	**1**	2 out of 4
RepTree	6	7	**1**	6	1 out of 4
HoeffTree	7	5	8	3	0 out of 4
Stump	8	8	**4**	8	1 out of 4

FT results are interesting, because it is not that well known an algorithm. However, overall, J48 does very well and these experiments support its use as a defacto standard decision tree

Tuning Classifiers/Model Selection

- Many classifiers are particularly susceptible to the parameter values. We can get a better classifier by trying to select the best parameters.
- **The rule is simple: Tuning must occur independently on each train set and never involve any test set**
- **Tuning on the test data in any way is cheating. You would be amazed how often it happens**
- The neatest way of tuning a classifier is simply to implement a wrapper, then do the tuning in buildClassifier

A. Bagnall and G. Cawley, On the Use of Default Parameter Settings in the Empirical Evaluation of Classification Algorithms
https://arxiv.org/abs/1703.06777

Tuned Support Vector Machines

- Support vector machines are particularly susceptible to their parameters
- To tune the SVM, we perform a grid search of parameter values
- For each possible set of parameter values, we perform a 10xfold cross validation on the train set to measure the quality
- We pick the parameters with highest accuracy thus measured
- This is hugely computationally expensive

A. Bagnall and G. Cawley, On the Use of Default Parameter Settings in the Empirical Evaluation of Classification Algorithms
https://arxiv.org/abs/1703.06777

TunedSVM

```
public class TunedSVM extends SMO{
 @Override
    public void buildClassifier(Instances
train){
/*Evaluate all possible parameter values and
choose the best. Then set this classifiers
parameters*/
     tuneSVM(train);
//Build final classifier
     super.buildClassifier(train);
   }
}
```

Setting the parameter range is difficult and classifier specific

```
public void tuneSVM(Instances train){
  int folds=10;
  double acc=0,bestAcc=0;
  CrossValidator cv = new CrossValidator();
  cv.setNumFolds(folds);
  cv.buildFolds(train);
  for(double p1:paraSpace1){// parameter C
    SMO model = new SMO();
    model.setC(p1);
    acc=cv.crossValidateWithStats(model,train);
    if(acc>bestAcc){
      bestC=p1;
      bestC=c;
    }
  }
  this.setC(bestC);
```

The actual implementation is more complex, but this is the gist of it

Tuned vs Untuned SVM

Question: Does tuning significantly improve SVM with an TBF kernel?

Experiment:

For each data set:

- Perform 30 resamples, tune independently on each train set
- Evaluate on test (Acc,BalAcc, NLL and AUROC)
- Average over 30 resamples

Analysis:

H0: no significant difference in SVM and TunedSVM

H1: TunedSVM significantly better

We test this hypothesis for each statistic with a paired t-test and a Wilcoxon signed rank test

Tuned vs Untuned SVM

	Tuned	Untuned
Mean Acc	0.7936	0.7711
Mean BalAcc	0.7215	0.6621
Mean NLL	0.989	1.641
Mean AUROC	0.8581	0.6666

WINS	Tuned	Untuned	Draw
Acc	72	48	1
BalAcc	86	34	1
NLL	95	16	10
AUROC	115	5	1

It is well worthwhile reporting this summary data, as it gives an overview. It seems that tuned is much better than untuned. **But is it significant?**

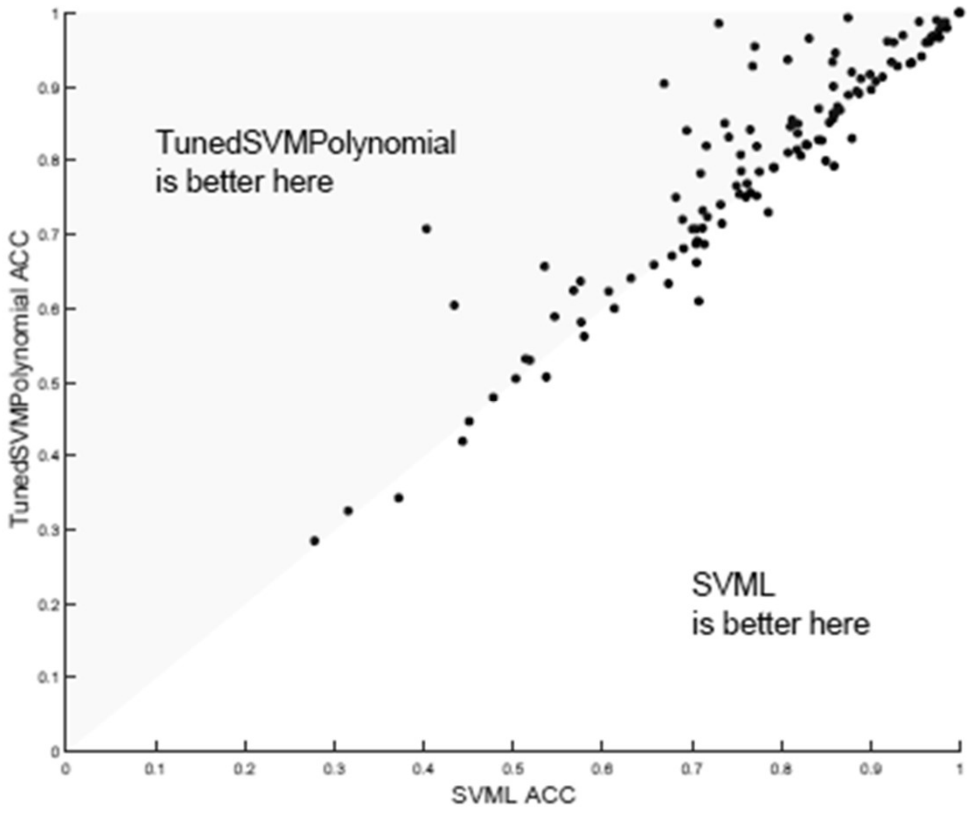
Tuned vs Untuned SVM p-values

	T-test	Wilcoxon
Acc	0.00009	0.00172
BalAcc	0	0
NLL	0	0
AUROC	0	0

Results reported to 6 significant figures

We can reject the null hypothesis at the 5% level for all of these statistics

We can conclude that tuning makes SVM significantly better



Comparing Classifiers: Summary

- We can compare classifiers on a single problem or over multiple problems using any of the evaluation metrics
- On a single problem, we need to perform repeated samples, then perform t-Test or sign-rank test on the differences.
- This is problematic, because the samples are not independent. Do not just rely on hypothesis tests for a single problem
- Over multiple problems, we use the average value over resamples for each problem
- We can then compare two classifiers with a paired test as before, or compare multiple classifiers using a critical difference diagram