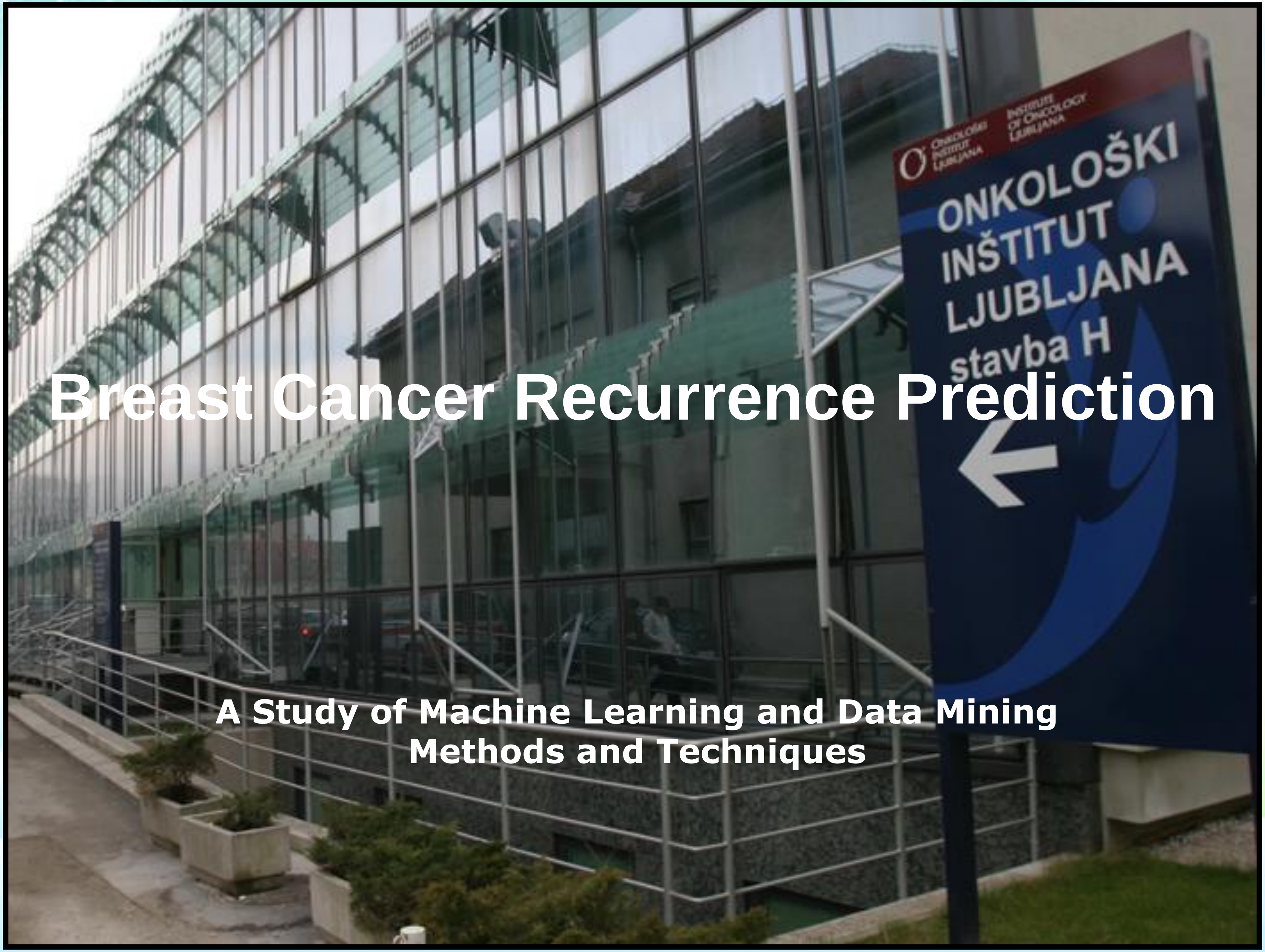
Machine Learning...

"It guessed your PIN number and cleaned out your bank account. Your move."

Breast Cancer Recurrence Prediction

A Study of Machine Learning and Data Mining Methods and Techniques



The Data

Provided by the Institute of Oncology, Ljubljana

Post-surgery data for about 1000 breast cancer patients.

+

Recurrence and time of recurrence.

The Data

	class1	class2	menop	stage	grade	hType	PgR	inv	nLymph	cTh	hTh	famHist	LVI	ER	maxNode	posRatio	age
300	11.82	0	1	2	2	1	0	0	1	1	0	3	0	1	2	3	2
301	4.89	1	0	1	2	1	0	0	2	1	0	0	0	2	1	4	3
302	14.63	0	1	1	4	2	0	0	0	0	0	1	0	1	1	1	3
303	21.83	0	0	1	4	2	1	0	1	0	0	9	0	4	1	2	2
304	19.87	0	0	1	2	1	0	0	0	0	0	0	0	1	2	1	2
305	7.54	0	1	2	3	1	9	2	1	0	1	1	0	3	3	3	4
306	15.15	0	0	1	4	2	1	0	0	0	0	2	0	4	1	1	2
307	0.30	1	0	2	2	1	0	0	3	0	0	9	0	1	1	4	2
308	12.49	0	1	2	2	3	1	0	0	0	0	0	0	4	1	1	5
309	1.77	1	0	2	3	1	1	2	2	1	0	9	1	3	3	3	2

Each patient is described with 17 values:

- 15 patient’s features
- 2 values, which describe the outcome

1 instance = 1 patient

	class1	class2	menop	stage	grade	hType	PgR	inv	nLymph	cTh	hTh	famHist	LVI	ER	maxNode	posRatio	age
300	11.82	0	1	2	2	1	0	0	1	1	0	3	0	1	2	3	2
301	4.89	1	0	1	2	1	0	0	2	1	0	0	0	2	1	4	3
302	14.63	0	1	1	4	2	0	0	0	0	0	1	0	1	1	1	3
303	21.83	0	0	1	4	2	1	0	1	0	0	9	0	4	1	2	2
304	19.87	0	0	1	2	1	0	0	0	0	0	0	0	1	2	1	2
305	7.54	0	1	2	3	1	9	2	1	0	1	1	0	3	3	3	4
306	15.15	0	0	1	4	2	1	0	0	0	0	2	0	4	1	1	2
307	**0.30**	**1**	**0**	**2**	**2**	**1**	**0**	**0**	**3**	**0**	**0**	**9**	**0**	**1**	**1**	**4**	**2**
308	12.49	0	1	2	2	3	1	0	0	0	0	0	0	4	1	1	5
309	1.77	1	0	2	3	1	1	2	2	1	0	9	1	3	3	3	2

- **Menopause?**
- **Tumor stage**
- **Tumor grade**
- **Histological type**
- **Progesterone receptor lvl.**
- **Invasive tumor type**
- **Number of positive lymph nodes**

- **Hormonal therapy?**
- **Chemotherapy?**
- **Family medical history**
- **Lymphovascular invasion?**
- **Estrogen receptor lvl.**
- **Size of max. removed node**
- **Ratio of positive lymph nodes**
- **Age group**

Prognostic Features

	class1	class2	menop	stage	grade	hType	PgR	inv	nLymph	cTh	hTh	famHist	LVI	ER	maxNode	posRatio	age
300	11.82	0	1	2	2	1	0	0	1	1	0	3	0	1	2	3	2
301	4.89	1	0	1	2	1	0	0	2	1	0	0	0	2	1	4	3
302	14.63	0	1	1	4	2	0	0	0	0	0	1	0	1	1	1	3
303	21.83	0	0	1	4	2	1	0	1	0	0	9	0	4	1	2	2
304	19.87	0	0	1	2	1	0	0	0	0	0	0	0	1	2	1	2
305	7.54	0	1	2	3	1	9	2	1	0	1	1	0	3	3	3	4
306	15.15	0	0	1	4	2	1	0	0	0	0	2	0	4	1	1	2
307	**0.30**	**1**	**0**	**2**	**2**	**1**	**0**	**0**	**3**	**0**	**0**	**9**	**0**	**1**	**1**	**4**	**2**
308	12.49	0	1	2	2	3	1	0	0	0	0	0	0	4	1	1	5
309	1.77	1	0	2	3	1	1	2	2	1	0	9	1	3	3	3	2

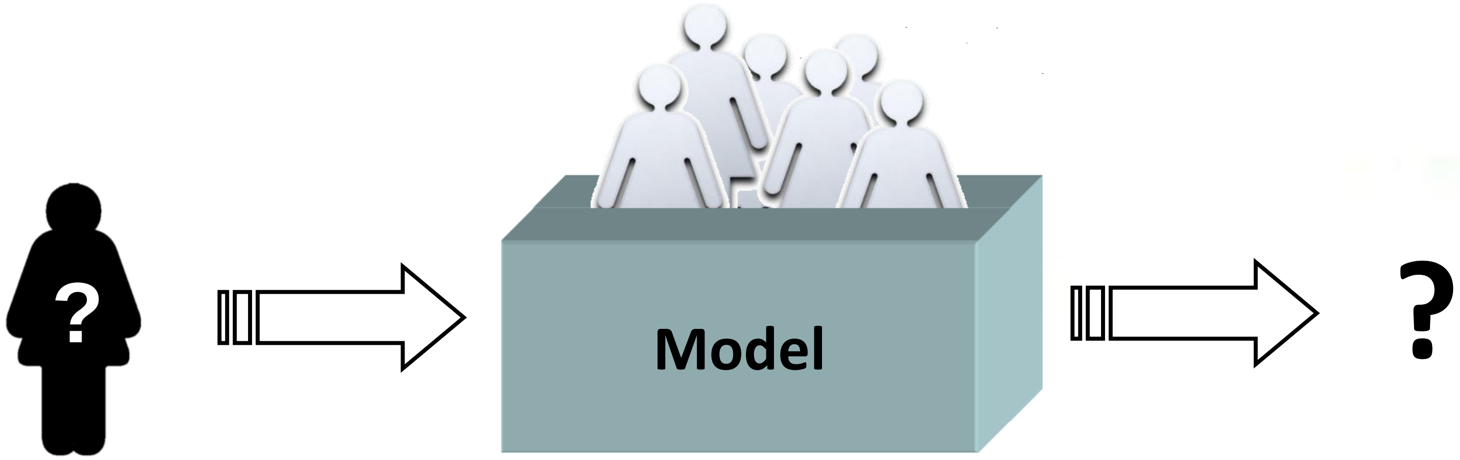
- Menopause?
- Tumor stage
- Tumor grade
- Histological type
- Progesterone receptor lvl.
- Invasive tumor type
- Number of positive lymph nodes

- Hormonal therapy?
- Chemotherapy?
- Family medical history
- Lymphovascular invasion?
- Estrogen receptor lvl.
- Size of max. removed node
- Ratio of positive lymph nodes
- Age group

Oncologists use these attributes for prognosis in every-day medical practice.

Basic Task in ML

We want to learn from past examples,with known outcomes.



To predict the outcome for a new patient.

Let's Use a Decision Tree

How should we define the outcome?
(we have 2 sub-problems)

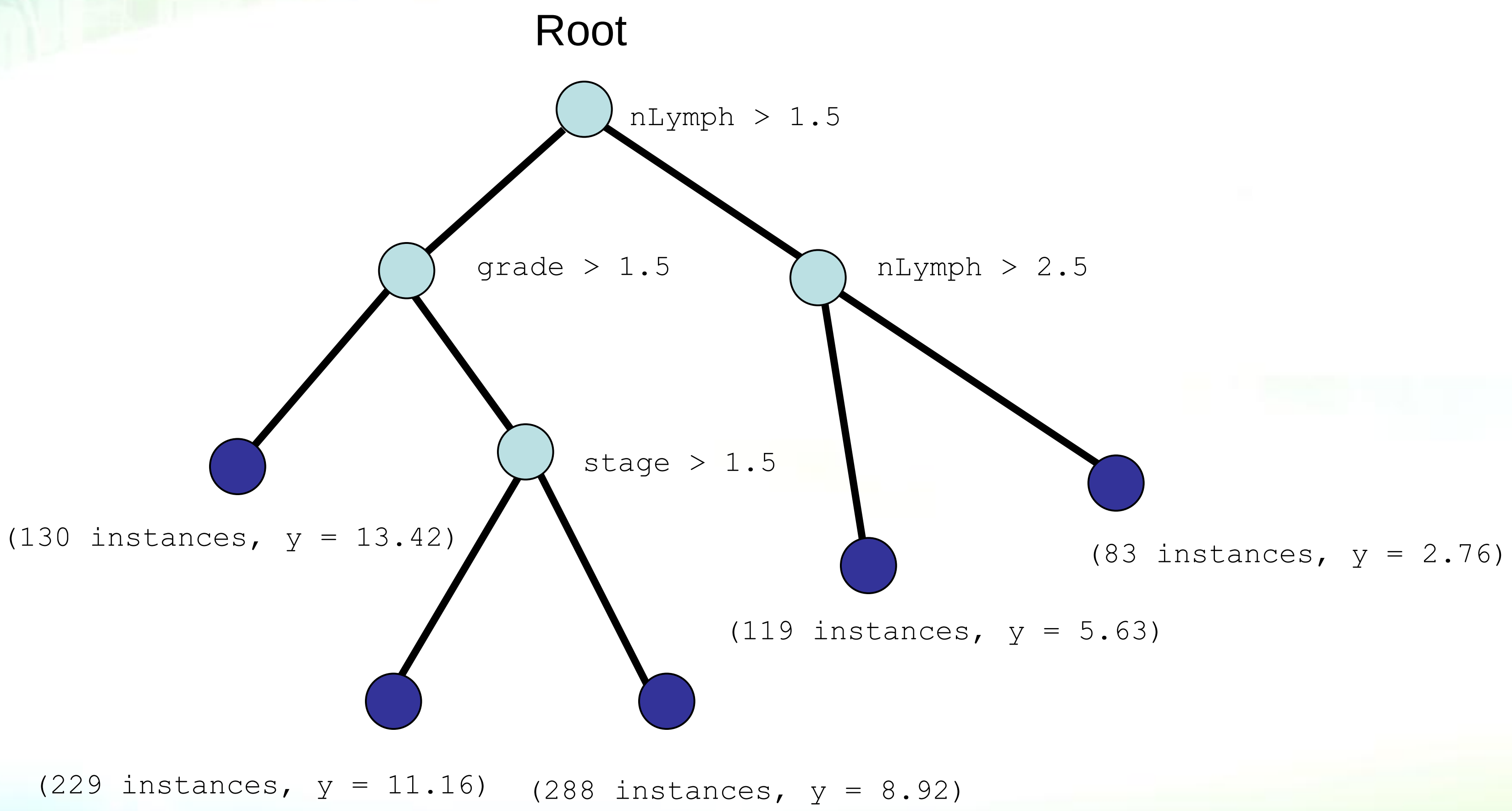
Recurrence?
(yes / no)

CLASSIFICATION TREE

Time to recurrence?
(continuous value)

REGRESSION TREE

A Regression Tree



Basic Evaluation of a Classifier

Our model predicts some instances correctly some incorrectly.

```
   ...
[40,] 0   0
[41,] 0   0
[42,] 0   0
[43,] 0   0
[44,] 0   0
[45,] 1   1
[46,] 0   0
[47,] 0   0
[48,] 1   0
[49,] 0   0
[50,] 1   0
[51,] 1   1
[52,] 1   0
[53,] 1   1
[54,] 1   1
[55,] 1   1
[56,] 0   1
   ...
```

$$\text{accuracy} = \frac{\text{number of correctly predicted instances}}{\text{total number of test instances}} = \mathbf{72\%}$$

Accuracy of predicting with the **majority class value** (0) is approximately **62%.**

information score = **0.22 bit**

$$\text{relative inf. score} = \frac{\text{information score}}{\text{class entropy}} = \mathbf{0.23}$$

Further Evaluation

Confusion Matrix

		predicted 0	predicted 1
actual value	0	57	5
	1	23	15

False Positives

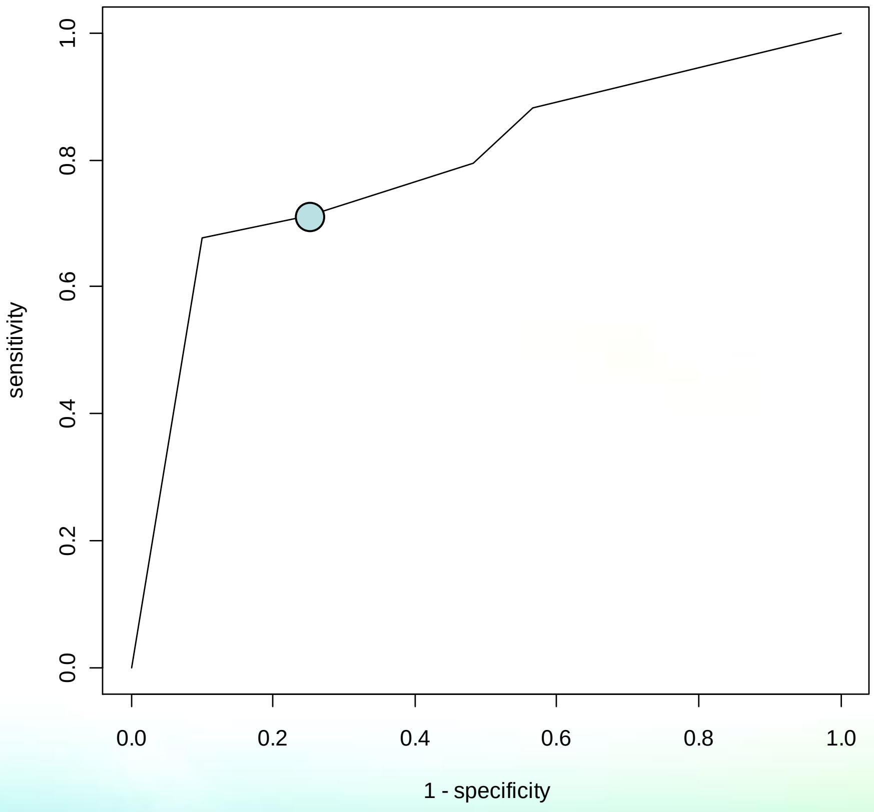
False Negatives

$$\text{sensitivity} = \frac{\text{True Positives}}{\text{True Positives + False Positives}} = \mathbf{0.75}$$

$$\text{specificity} = \frac{\text{True Negatives}}{\text{True Negatives + False Negatives}} = \mathbf{0.712}$$

ROC Curve

Receiver Operating Characteristic (ROC) curve.



Evaluating Regression Models

How close are these predicted values (blue) to the actual values (black)?

…

39	16.30116359	8.920783
40	8.95550992	13.420565
41	17.83436003	8.920783
42	17.18275154	13.420565
43	14.54346338	11.157686
44	11.08829569	13.420565
45	1.70841889	5.625156
46	16.58316222	13.420565
47	6.25872690	11.157686
48	4.13689254	13.420565
49	7.34017796	11.157686
50	1.84257358	11.157686
51	1.95208761	5.625156
52	3.50171116	8.920783
53	0.98015058	2.756620

…

mean absolute error (MAE) = **5.01**

(on average, the model misses by 5 years)

mean squared error (MSE) = **34.81**

Further Evaluation

mean absolute error (MAE) = **5.01**

mean squared error (MSE) = **34.81**

How good are these results compared to simply predicting with the mean value across training instances?

$$\text{relative mean absolute error (RMAE)} = \frac{\text{MAE}}{\text{MAE of predicting with mean value}} = \mathbf{0.86}$$

$\boxed{5.82}$

$$\text{relative mean squared error (RMSE)} = \frac{\text{MSE}}{\text{MSE of predicting with mean value}} = \mathbf{0.84}$$

$\boxed{41.46}$

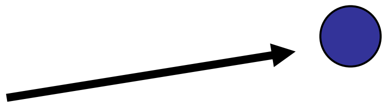
Values lower than 1 indicate that the model is useful.

correlation coefficient = **0.087**

Growing a Regression Tree

Leaves: 1, MSE = 41.46

Root



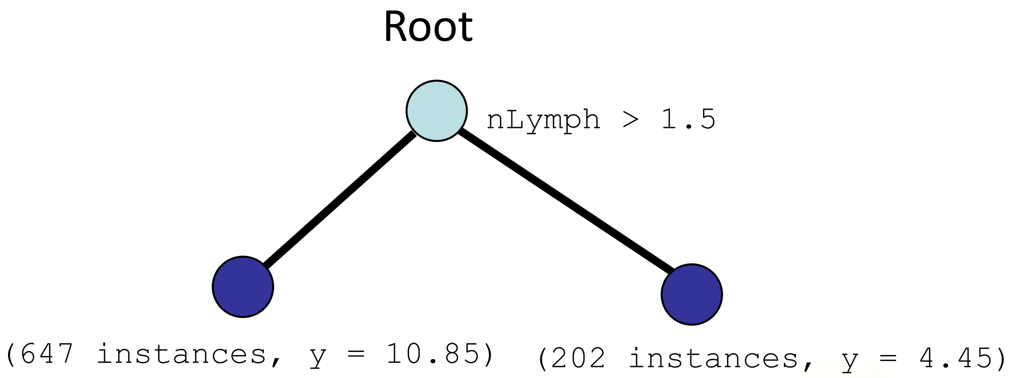
(849 instances, y = 9.15)

The most simple tree.

Growing a Regression Tree

Leaves: 1, MSE = 41.46

Leaves: 2, MSE = 36.32

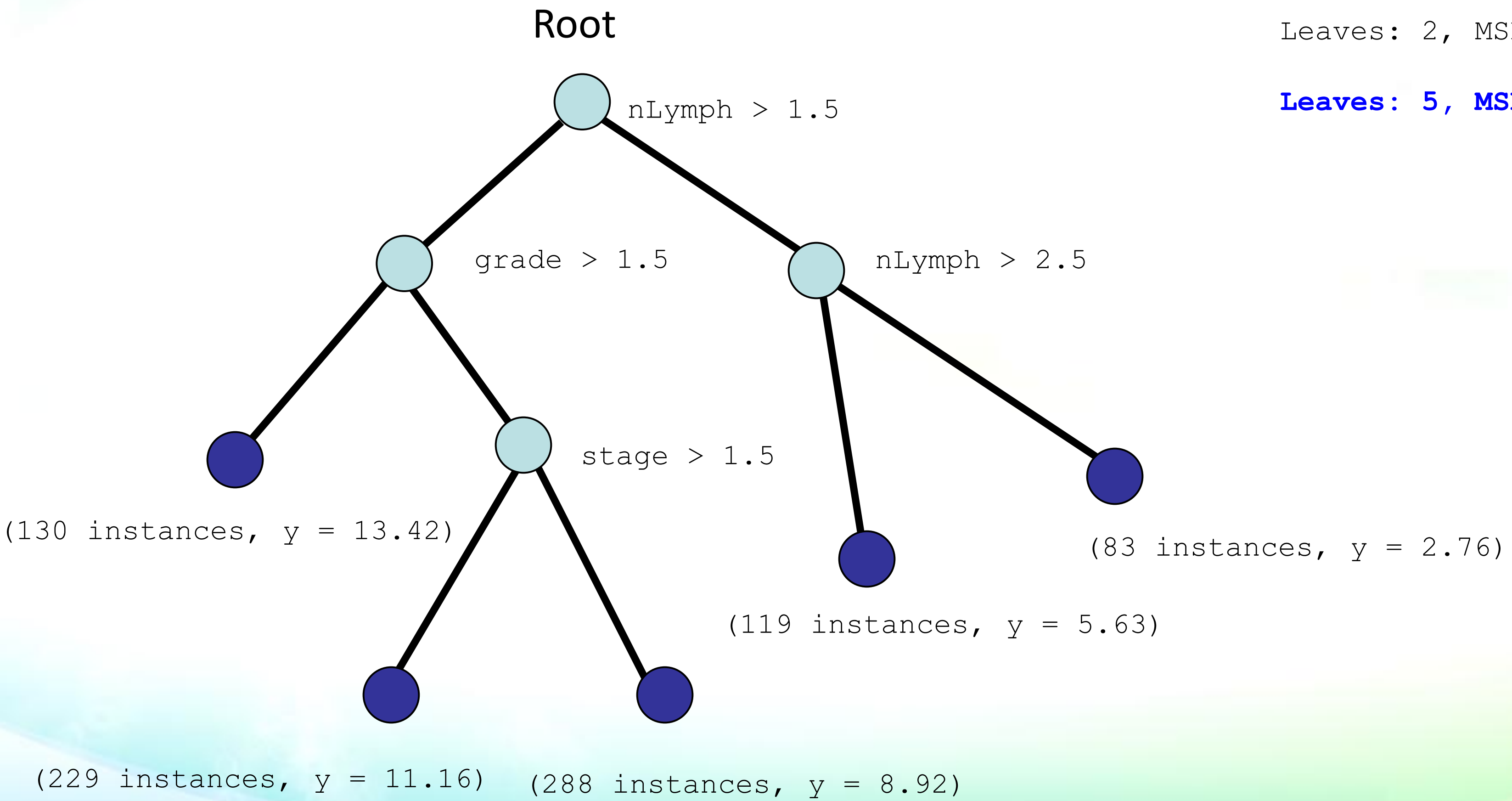


Growing a Regression Tree

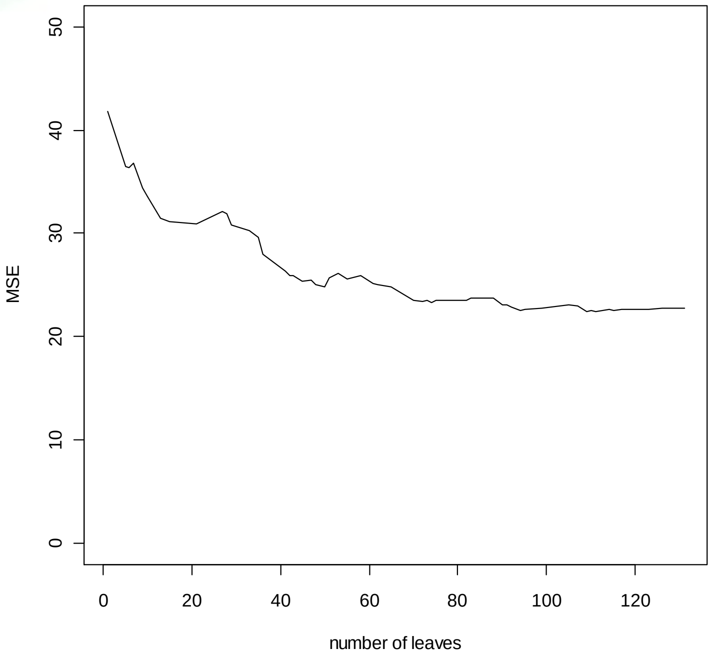
Leaves: 1, MSE = 41.46

Leaves: 2, MSE = 36.32

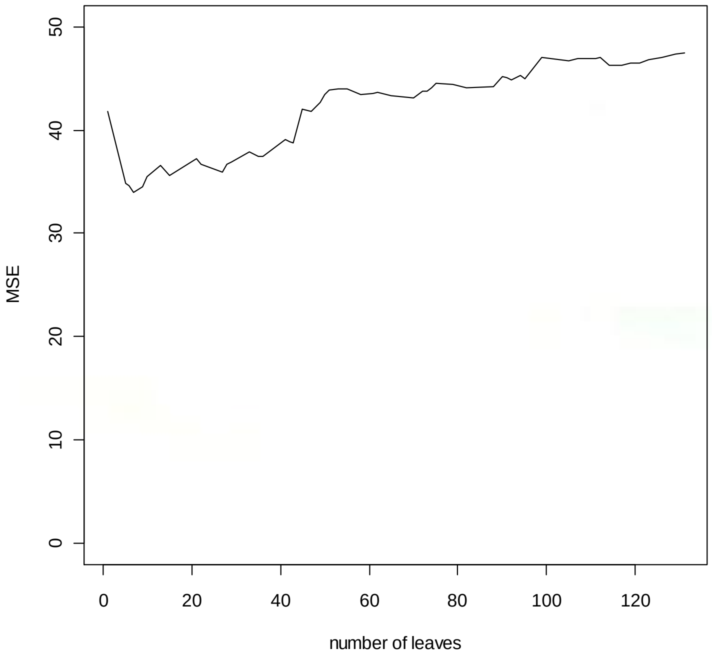
Leaves: 5, MSE = 34.81



Overfitting a Decision Tree



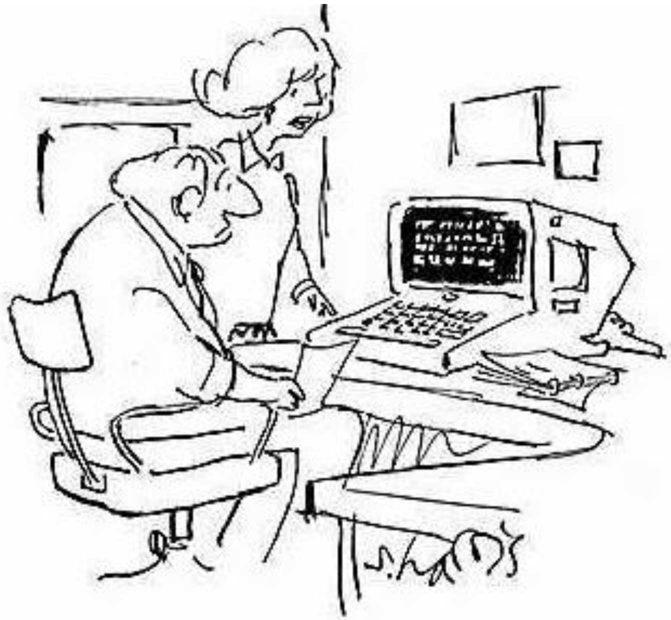
Results on the training data set



Results on the test data set

Further increasing the size of the tree may result in overfitting and a higher error.

Comparing Different Models



"IT FIGURES. IF THERE'S ARTIFICIAL INTELLIGENCE, THERE'S BOUND TO BE SOME ARTIFICIAL STUPIDITY."

Cross-Validation

	DT	NB	ANN
[1,]	**0.729**	0.694	0.553
[2,]	0.753	**0.776**	0.729
[3,]	**0.694**	0.671	0.553
[4,]	**0.800**	0.741	0.694
[5,]	0.671	**0.706**	0.624
[6,]	0.659	**0.671**	0.647
[7,]	**0.812**	0.788	0.788
[8,]	**0.741**	0.694	0.718
[9,]	0.694	**0.718**	0.694
[10,]	**0.750**	**0.750**	0.738
	73%	72%	67%

Up to now, we have used a **Decision Tree** (DT).
Now we add a **Naive Bayes classifier** (NB)
and an **Artificial Neural Network** (ANN).

We randomly split the training data into 10 disjoint subsets of approx. equal size and use **10-fold cross-validation**.

DT and NB have a higher mean accuracy than ANN, but **how significant** are these differences?

Two Models, Single Domain

Is DT better than the ANN?

We use a **t-test** to compare the mean accuracies of the two models and we set α=0.01 (99% **confidence level**).

Our **null hypothesis**: the mean accuracies are equal
Out **alternative hypothesis**: the mean accuracy of DT is greater

We get the following **confidence interval** for the difference in means:
$[0.0207, \infty)$

Our null hypothesis does not fall within the confidence interval. Therefore, the mean accuracy of DT is higher than the mean accuracy of ANN (with 99% confidence).

We repeat the test and come to a similar conclusion for NB and ANN, where the 99% confidence interval is: $[0.0168, \infty)$

Multiple-Comparisons

With t-tests and 10-fold cross-validation, we came to the following conclusions:

DT has higher mean accuracy than ANN (α=0.01, 99% conf. level, p-value = 0.0089)

NB has higher mean accuracy than ANN (α=0.01, 99% conf. level, p-value = 0.0096)

Therefore, DT and NB have higher accuracy than ANN, with 99% confidence!

NO!

Problem: We made **multiple comparisons** (in our case 2) which increases the probability of making the wrong conclusion!

Solution: Use **Bonferroni's correction**. Increase the confidence of individual comparisons by dividing with the number of comparisons (N): $\alpha' = \alpha / N$

If we set α' to 0.005, we can not reject the null hypothesis for neither DT nor NB.
By lowering the confidence to α = 0.05, we can reject the null hypotheses and conclude that DT and NB both have higher accuracy, with 95% confidence.

Several Models, Several Domains

Example:

	[,1]	[,2]	[,3]	[,4]	[,5]	[,6]	[,7]	[,8]	[,9]	[,10]	[,11]
NB	0.574	0.904	0.674	0.557	0.709	0.724	0.205	0.687	0.758	0.633	0.770
DT	0.652	0.872	0.723	0.763	0.449	0.769	0.609	0.829	0.831	0.389	**0.899**
SVM	0.758	0.882	**0.899**	**0.954**	0.693	0.878	0.907	0.827	0.897	**0.900**	0.778
kNN	0.814	0.784	0.879	0.935	0.633	0.791	0.794	0.832	0.824	0.777	0.833
ANN	0.767	0.882	0.821	0.891	**0.786**	0.895	**0.926**	0.841	0.915	0.672	0.862
RF	**0.876**	**0.946**	0.883	0.922	0.785	**0.912**	0.871	**0.891**	**0.941**	0.874	0.824

(scores for 6 different classifiers across 11 data sets)

To account for the differences between the domains, we use the **non-parametric Friedman test**. We observe the models' **ranks** on each domain.

	[,1]	[,2]	[,3]	[,4]	[,5]	[,6]	[,7]	[,8]	[,9]	[,10]	[,11]	mean rank
NB	6	2	6	6	3	6	6	6	6	5	6	5.27
DT	5	5	5	5	6	5	5	4	4	6	1	4.64
SVM	4	4	1	1	4	3	2	5	3	1	5	3.00
kNN	2	6	3	2	5	4	4	3	5	3	3	3.64
ANN	3	4	4	4	1	2	1	2	2	4	2	2.64
RF	1	1	2	3	2	1	3	1	1	2	4	1.91

The resulting p-value = 0.01526. Therefore, the models are not all equally good (95% conf. lvl.)

Post-hoc Comparison

In our previous example, we have rejected the null-hypothesis that the models do not differ in ranks. Now we can perform a post-hoc multiple-comparison test.

Nemenyi test:

$$CD = \underbrace{q_\alpha \sqrt{\frac{k(k+1)}{6N}}}_{\text{Minimum Significant Difference (MSD)}}$$

k – number of samples (=models)
N – number of observations
q – critical value
S_i – rank sum for i-th sample

Differences in mean rank:

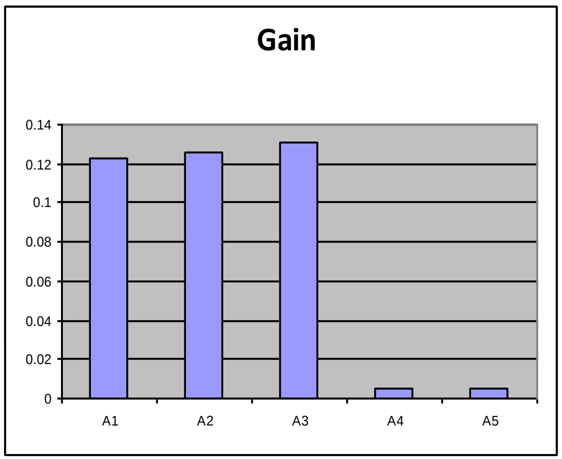
	NB	DT	SVM	kNN	ANN	RF
NB	-	0.64	2.27	1.64	2.64	3.36
DT	-	-	1.64	1.00	2.00	2.73
SVM	-	-	-	0.64	0.36	1.09
kNN	-	-	-	-	1.00	1.73
ANN	-	-	-	-	-	0.73

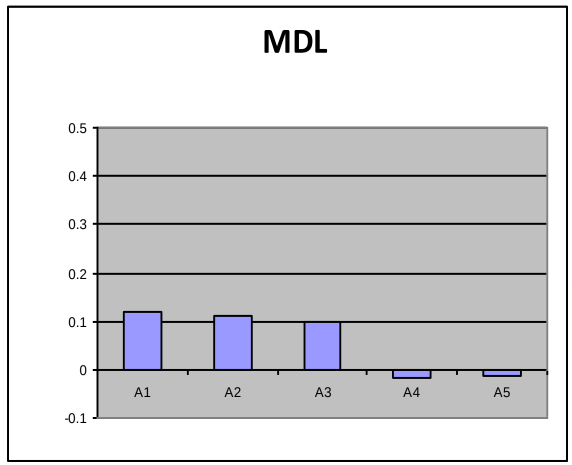
critical value (0.05 conf. level, k = 6) = 2.85
MSD = 2.27

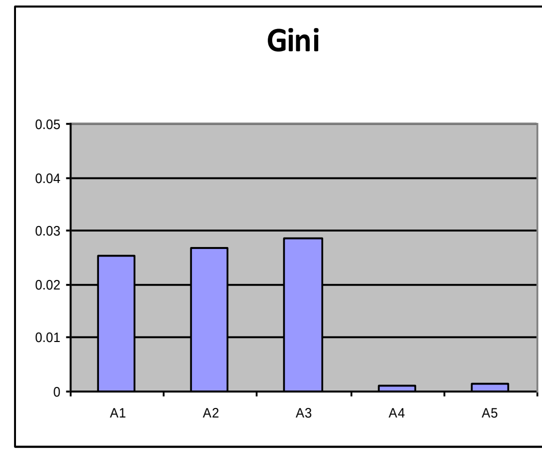
Feature Evaluation

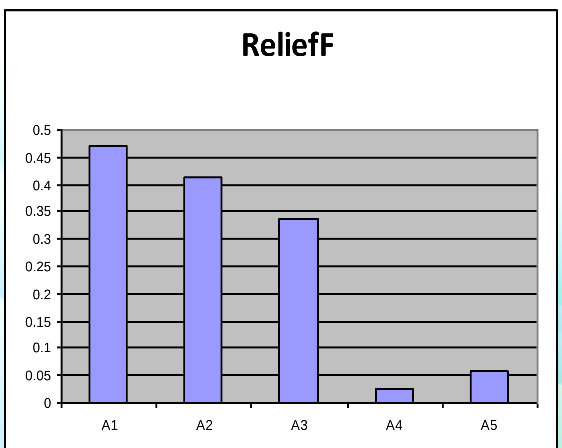
Synthetic Example 1

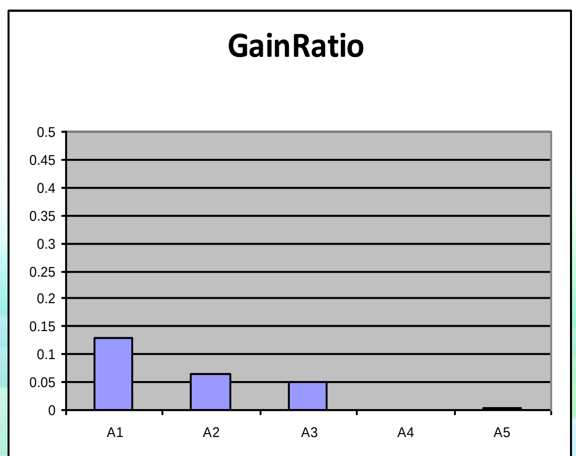
- 1000 instances, 5 attributes, equiprobable attribute values
- A1: 0,1
- A2: 0,1,2,3
- A3: 0,1,2,3,4,5
- A4: 0,1,2,3,4,5
- A5: 0,1,2,3,4,5
- class value = (A1 > 0) OR (A2 > 1) OR (A3 > 2)

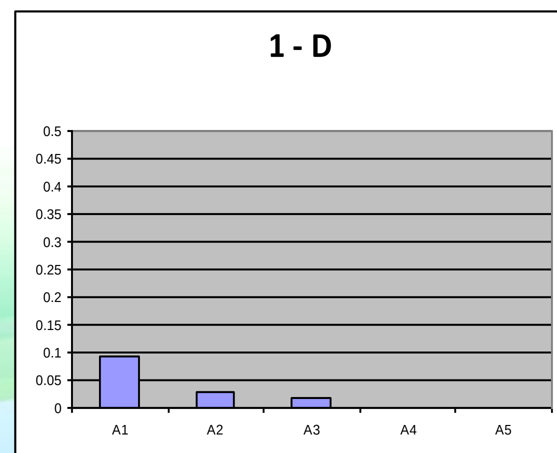






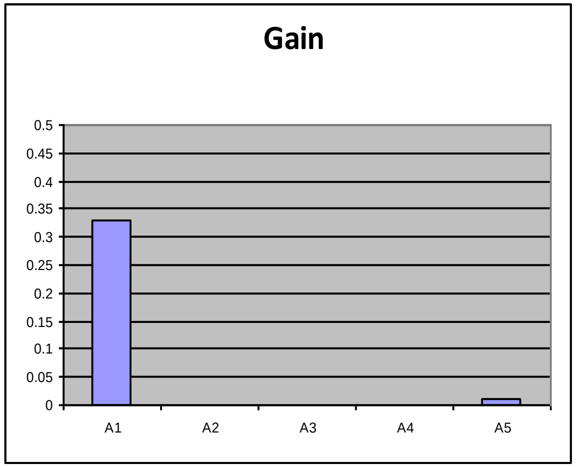


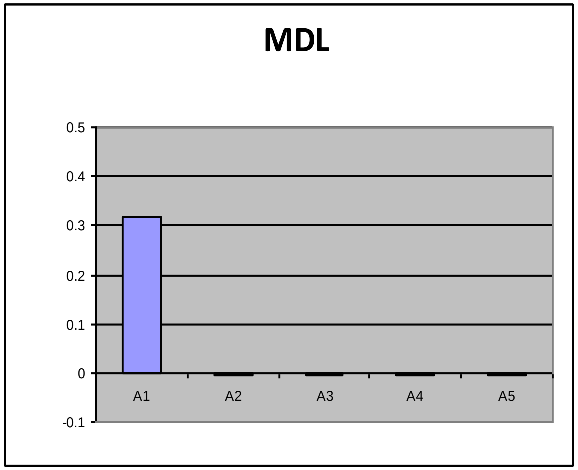


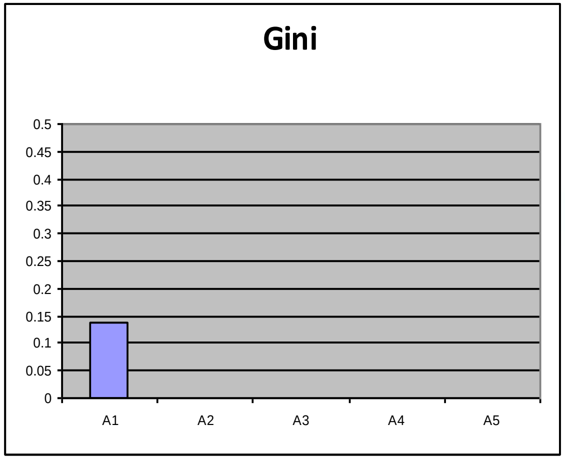


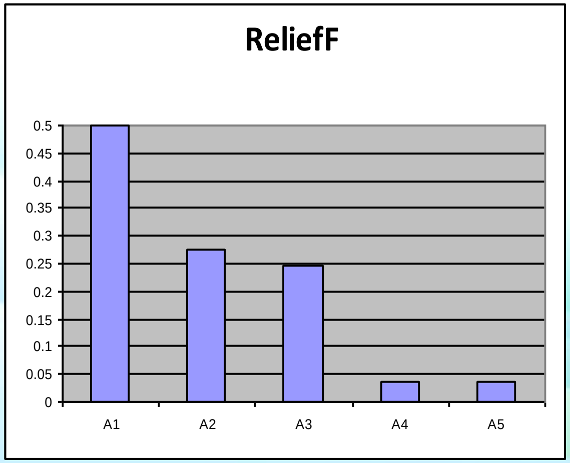
Synthetic Example 2

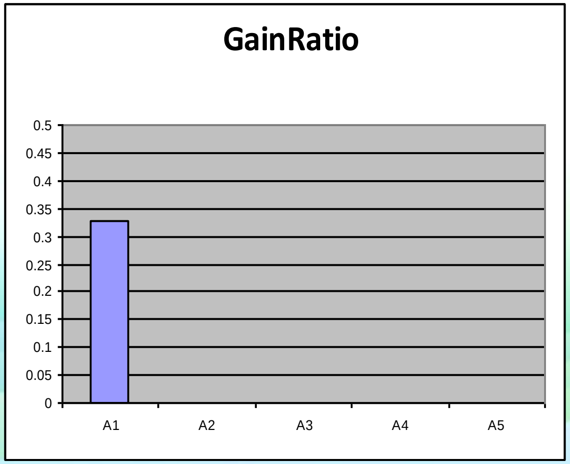
- 1000 instances, 5 attributes, equiprobable attribute values
- A1: 0,1
- A2: 0,1
- A3: 0,1
- A4: 0,1
- A5: 0,1
- class value = A1 OR (A2 XOR A3)

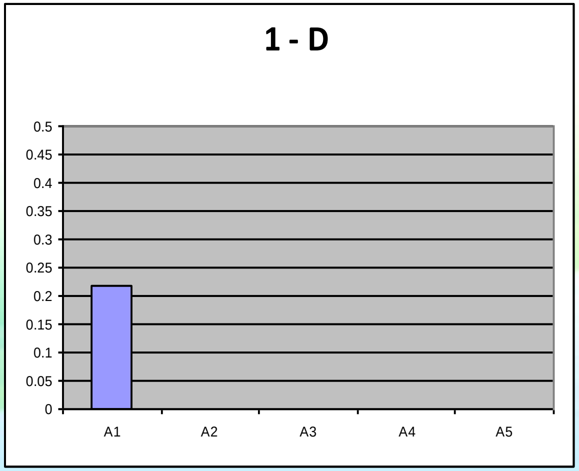






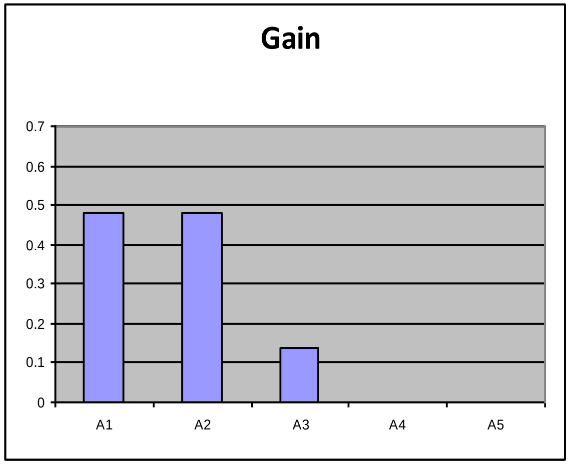


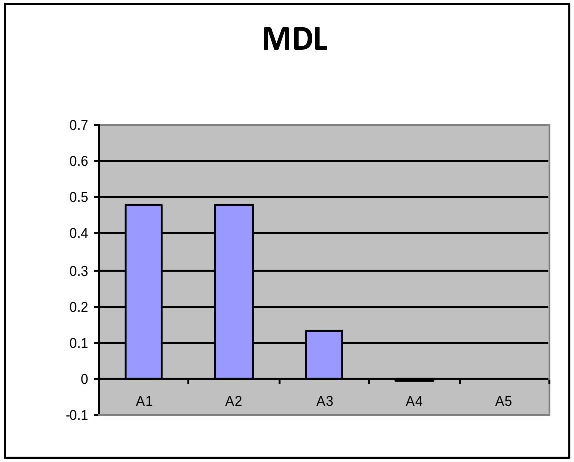


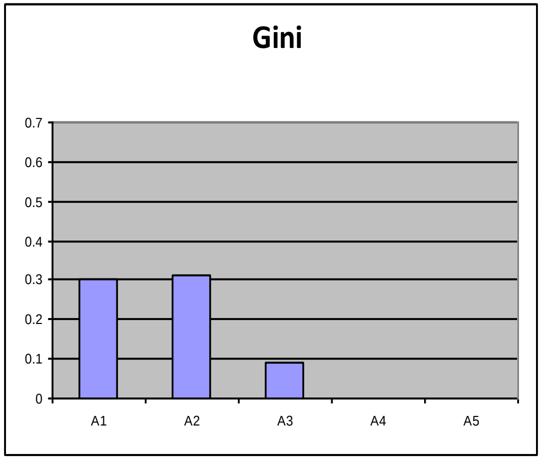


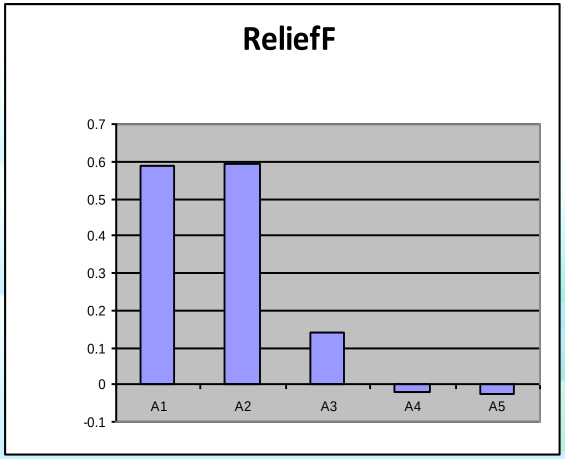
Synthetic Example 3

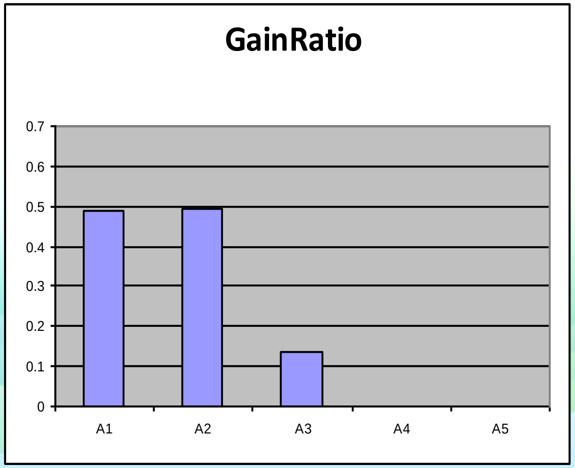
- 1000 instances, 5 attributes, 2 equiprobable class values
- A1: 0,1 (is equal to the class value 90% of the time)
- A2: 0,1 (is equal to the class value 90% of the time)
- A3: 0,1 (is equal to the class value 70% of the time)
- A4: 0,1
- A5: 0,1

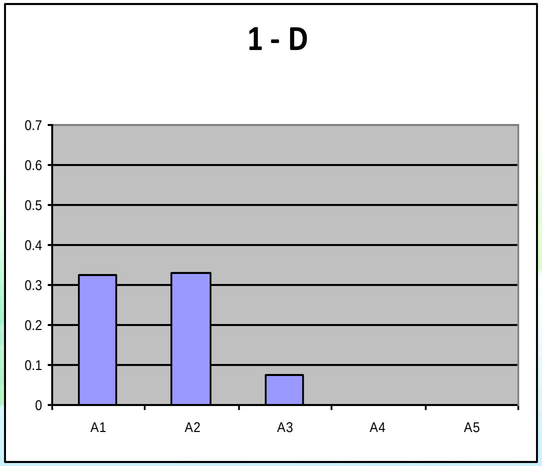




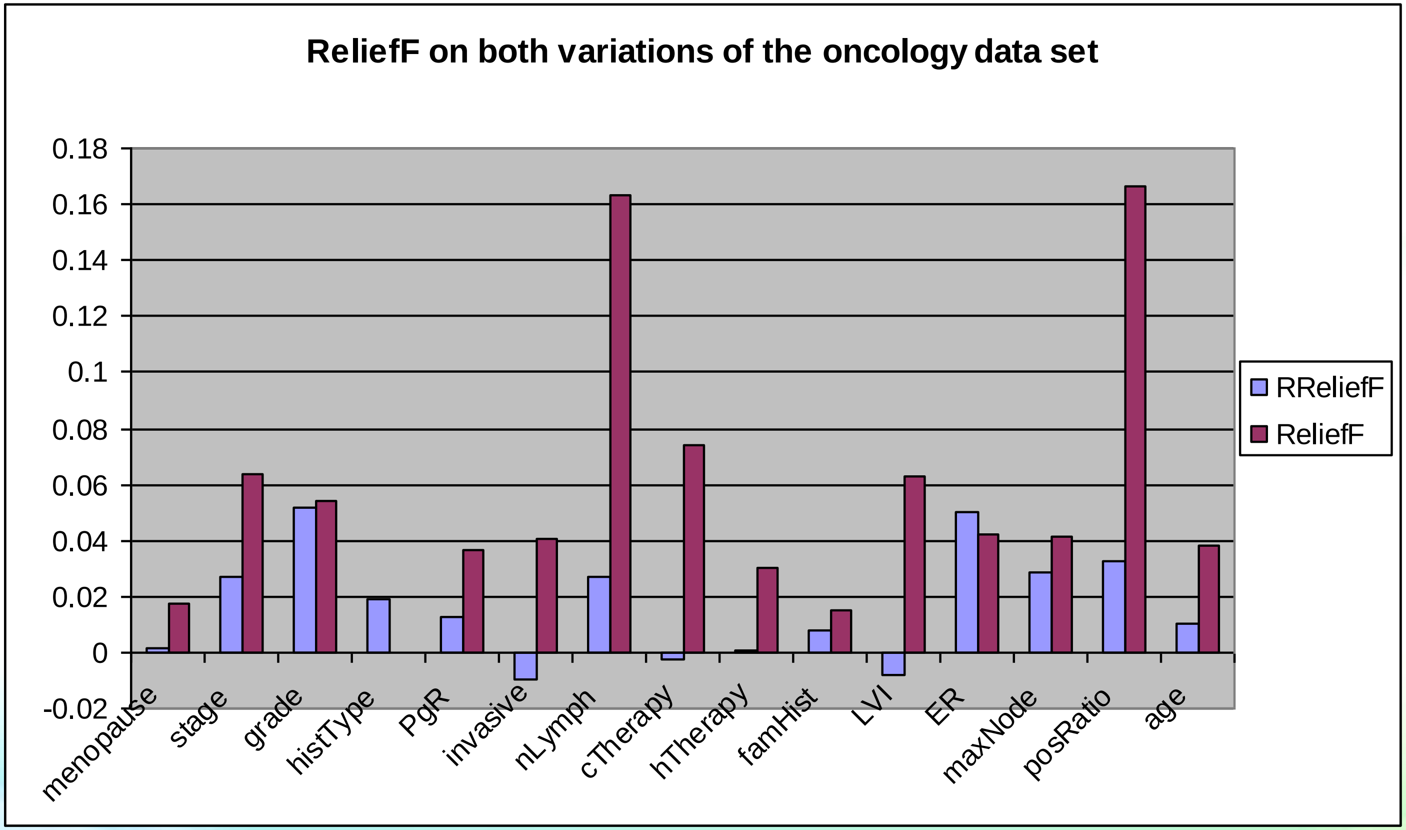








Real-Life Dataset



Data Preprocessing

"It does data processing, word processing and list processing. Get me some data, some words and some lists."

Running Example

Dataset: Ecoli
336 instances, default accuracy is approximately 43% (82% in related work)

7 relevant attributes (all are numeric, two can be treated as binary):

```
1.  mcg: McGeoch's method for signal sequence recognition
2.  gvh: von Heijne's method for signal sequence recognition
3.  lip: von Heijne's Signal Peptidase II consensus sequence score (2 possible values)
4.  chg: presence of charge on N-terminus of predicted lipoproteins (2 possible values)
5.  aac: score of analysis of amino acid content (outer membrane & periplasmic proteins)
6.  alm1: score of the ALOM membrane spanning region prediction program
7.  alm2: score of ALOM program after excluding putative cleavable signal regions
```

The class is the protein localization site. Distribution of class values:

```
cp (cytoplasm)                                         143
im (inner membrane without signal sequence)             77
pp (perisplasm)                                         52
imU (inner membrane, uncleavable signal sequence)       35
om (outer membrane)                                     20
omL (outer membrane lipoprotein)                         5
imL (inner membrane lipoprotein)                         2
imS (inner membrane, cleavable signal sequence)          2
```

Some Models Require Discrete Data

We want to use a naive Bayes classifier. We use leave-one-out cross-validation to evaluate its performance:

mean accuracy = 71.4% is not as good as we would expect,... why?

NB treats each distinct numeric value as a discrete attribute value.

	A1	A2	A3	A4	A5	A6	A7	class
#58	0.40	0.35	0.48	0.5	0.45	0.33	0.42	cp
#272	0.65	0.51	0.48	0.5	0.66	0.54	0.33	om
#130	0.37	0.44	0.48	0.5	0.42	0.39	0.47	cp
#111	0.32	0.33	0.48	0.5	0.60	0.06	0.20	cp
#201	0.58	0.55	0.48	0.5	0.57	0.70	0.74	im
#202	0.36	0.47	0.48	0.5	0.51	0.69	0.72	im
#43	0.40	0.50	0.48	0.5	0.45	0.39	0.47	cp
#98	0.57	0.54	0.48	0.5	0.37	0.28	0.33	cp
#191	0.33	0.37	0.48	0.5	0.46	0.65	0.69	im
#208	0.11	0.50	0.48	0.5	0.58	0.72	0.68	im

We get too many coefficients and very sparse data.

Class Attribute	cp (0.42)	im (0.23)	imL (0.01)	imS (0.01)	imU (0.1)	om (0.06)	omL (0.02)	pp (0.15)
A1								
0	1.0	1.0	0.0	0.0	0.0	0.0	0.0	0.0
0.04	1.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
0.06	0.0	1.0	0.0	0.0	0.0	0.0	0.0	0.0
0.07	1.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
0.1	0.0	1.0	0.0	0.0	0.0	0.0	0.0	0.0
0.11	0.0	1.0	0.0	0.0	0.0	0.0	0.0	0.0
0.12	0.0	2.0	0.0	0.0	0.0	0.0	0.0	0.0
0.16	1.0	1.0	0.0	0.0	0.0	0.0	0.0	0.0
0.17	3.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
0.18	1.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
0.2	1.0	1.0	0.0	0.0	0.0	0.0	0.0	0.0
			. . .					
			. . .					
0.81	0.0	0.0	0.0	0.0	3.0	0.0	0.0	0.0
0.83	0.0	1.0	0.0	0.0	1.0	0.0	0.0	0.0
0.84	0.0	1.0	0.0	0.0	2.0	0.0	0.0	0.0
0.85	0.0	0.0	0.0	1.0	0.0	0.0	0.0	0.0
0.86	0.0	0.0	0.0	0.0	2.0	0.0	0.0	0.0
0.87	0.0	0.0	0.0	0.0	1.0	0.0	0.0	0.0
0.88	0.0	0.0	0.0	0.0	1.0	0.0	0.0	0.0
0.89	0.0	1.0	0.0	0.0	0.0	0.0	0.0	0.0
[total]	143.0	77.0	52.0	35.0	20.0	5.0	2.0	2.0

Discretization

A better solution: Discretization

	A1	A2	A3	A4	A5	A6	A7	class
#58	0.40	0.35	0.48	0.5	0.45	0.33	0.42	cp
#272	0.65	0.51	0.48	0.5	0.66	0.54	0.33	om
#130	0.37	0.44	0.48	0.5	0.42	0.39	0.47	cp
#111	0.32	0.33	0.48	0.5	0.60	0.06	0.20	cp
#201	0.58	0.55	0.48	0.5	0.57	0.70	0.74	im
#202	0.36	0.47	0.48	0.5	0.51	0.69	0.72	im
#43	0.40	0.50	0.48	0.5	0.45	0.39	0.47	cp
#98	0.57	0.54	0.48	0.5	0.37	0.28	0.33	cp
#191	0.33	0.37	0.48	0.5	0.46	0.65	0.69	im
#208	0.11	0.50	0.48	0.5	0.58	0.72	0.68	im

Intervals of equal width.

	A1	A2	A3	A4	A5	A6	A7	class
#58	5	3	1	1	8	3	5	cp
#272	8	6	1	1	12	5	4	om
#130	5	5	1	1	8	4	5	cp
#111	4	3	1	1	11	1	3	cp
#201	7	6	1	1	10	7	8	im
#202	5	5	1	1	9	7	8	im
#43	5	5	1	1	8	4	5	cp
#98	7	6	1	1	7	3	4	cp
#191	4	4	1	1	8	6	7	im
#208	2	5	1	1	10	7	7	im

accuracy improves from 71.4% to **81.8%**

More Discretization

Alternative discretization methods:

	A1	A2	A3	A4	A5	A6	A7	class
#58	0.40	0.35	0.48	0.5	0.45	0.33	0.42	cp
#272	0.65	0.51	0.48	0.5	0.66	0.54	0.33	om
#130	0.37	0.44	0.48	0.5	0.42	0.39	0.47	cp
#111	0.32	0.33	0.48	0.5	0.60	0.06	0.20	cp
#201	0.58	0.55	0.48	0.5	0.57	0.70	0.74	im
#202	0.36	0.47	0.48	0.5	0.51	0.69	0.72	im
#43	0.40	0.50	0.48	0.5	0.45	0.39	0.47	cp
#98	0.57	0.54	0.48	0.5	0.37	0.28	0.33	cp
#191	0.33	0.37	0.48	0.5	0.46	0.65	0.69	im
#208	0.11	0.50	0.48	0.5	0.58	0.72	0.68	im

Equal width (Scott's formula).

	A1	A2	A3	A4	A5	A6	A7	class
#58	5	3	1	1	8	3	5	cp
#272	8	6	1	1	12	5	4	om
#130	5	5	1	1	8	4	5	cp
#111	4	3	1	1	11	1	3	cp
#201	7	6	1	1	10	7	8	im
#202	5	5	1	1	9	7	8	im
#43	5	5	1	1	8	4	5	cp
#98	7	6	1	1	7	3	4	cp
#191	4	4	1	1	8	6	7	im
#208	2	5	1	1	10	7	7	im

accuracy improves from 71.4% to **81.8%**

Binarization.

	A1	A1.1	A2	A3	A4	A5	A5.1	A6	A6.1	A7	class
58	0	0	0	0	1	0	0	0	0	0	cp
272	1	0	0	0	1	1	0	1	0	0	om
130	0	0	0	0	1	0	0	1	0	0	cp
111	0	0	0	0	1	1	0	0	0	0	cp
201	1	0	0	0	1	1	0	1	1	1	im
202	0	0	0	0	1	0	0	1	1	1	im
43	0	0	0	0	1	0	0	1	0	0	cp
98	1	0	0	0	1	0	0	0	0	0	cp
191	0	0	0	0	1	0	0	1	1	1	im
208	0	0	0	0	1	1	0	1	1	1	im

accuracy improves from 71.4% to **85.5%**

Min. entropy (MDL).

	A1	A2	A3	A4	A5	A6	A7	class
58	1	1	1	1	1	1	1	cp
272	2	1	1	1	2	2	1	om
130	1	1	1	1	1	2	1	cp
111	1	1	1	1	2	1	1	cp
201	2	1	1	1	2	3	2	im
202	1	1	1	1	1	3	2	im
43	1	1	1	1	1	2	1	cp
98	2	1	1	1	1	1	1	cp
191	1	1	1	1	1	3	2	im
208	1	1	1	1	2	3	2	im

accuracy improves from 71.4% to **85.4%**

Let's Throw Away Some Information

We replace 300 features values with unknown values or NA's (at random).

	A1	A2	A3	A4	A5	A6	A7	class
#58	5	3	1	1	8	3	5	cp
#272	8	6	1	1	12	5	4	om
#130	5	5	1	1	8	4	5	cp
#111	4	3	1	1	11	1	3	cp
#201	7	6	1	1	10	7	8	im
#202	5	5	1	1	9	7	8	im
#43	5	5	1	1	8	4	5	cp
#98	7	6	1	1	7	3	4	cp
#191	4	4	1	1	8	6	7	im
#208	2	5	1	1	10	7	7	im

→

	A1	A2	A3	A4	A5	A6	A7	class
#58	**NA**	3	1	1	8	3	**NA**	cp
#272	8	6	1	1	12	5	4	om
#130	5	**NA**	1	1	8	**NA**	5	cp
#111	4	3	1	1	**NA**	1	**NA**	cp
#201	7	6	1	1	10	7	8	im
#202	5	5	1	1	9	**NA**	**NA**	im
#43	5	5	1	1	8	4	5	cp
#98	7	6	1	1	7	3	4	cp
#191	4	4	**NA**	1	8	6	7	im
#208	2	5	1	1	10	7	**NA**	im

accuracy decreases from 81.8% to **77.1%**

By omitting instances with at least one NA from the training data, we lose about 200 instances.

Treating Missing Values (1)

	A1	A2	A3	A4	A5	A6	A7	class
#58	**NA**	3	1	1	8	3	**NA**	cp
#272	8	6	1	1	12	5	4	om
#130	5	**NA**	1	1	8	**NA**	5	cp
#111	4	3	1	1	**NA**	1	**NA**	cp
#201	7	6	1	1	10	7	8	im
#202	5	5	1	1	9	**NA**	**NA**	im
#43	5	5	1	1	8	4	5	cp
#98	7	6	1	1	7	3	4	cp
#191	4	4	**NA**	1	8	6	7	im
#208	2	5	1	1	10	7	**NA**	im

Treating NA's as a special value.

	A1	A2	A3	A4	A5	A6	A7	class
#58	NA	3	1	1	8	3	NA	cp
#272	8	6	1	1	12	5	4	om
#130	5	NA	1	1	8	NA	5	cp
#111	4	3	1	1	NA	1	NA	cp
#201	7	6	1	1	10	7	8	im
#202	5	5	1	1	9	NA	NA	im
#43	5	5	1	1	8	4	5	cp
#98	7	6	1	1	7	3	4	cp
#191	4	4	NA	1	8	6	7	im
#208	2	5	1	1	10	7	NA	im

accuracy further decreases from 77.1% to **61.6%**

Replacing with most frequent value

	A1	A2	A3	A4	A5	A6	A7	class
#58	**8**	3	1	1	8	3	**4**	cp
#272	8	6	1	1	12	5	4	om
#130	5	**5**	1	1	8	**3**	5	cp
#111	4	3	1	1	**9**	1	**4**	cp
#201	7	6	1	1	10	7	8	im
#202	5	5	1	1	9	**3**	**4**	im
#43	5	5	1	1	8	4	5	cp
#98	7	6	1	1	7	3	4	cp
#191	4	4	**1**	1	8	6	7	im
#208	2	5	1	1	10	7	**4**	im

accuracy increases from 77.1% to **80.7%**

Treating Missing Values (2)

A test instance contains a missing value:

	A1	A2	A3	A4	A5	A6	A7	class
#98	7	NA	1	1	7	3	4	?

Replace with most common value:

	A1	A2	A3	A4	A5	A6	A7	class
#98	7	5	1	1	7	3	4	?

Predicted class: "cp" (p_{cp} = 0.96)

OR

Use weighted sum of predictions across all possible values:

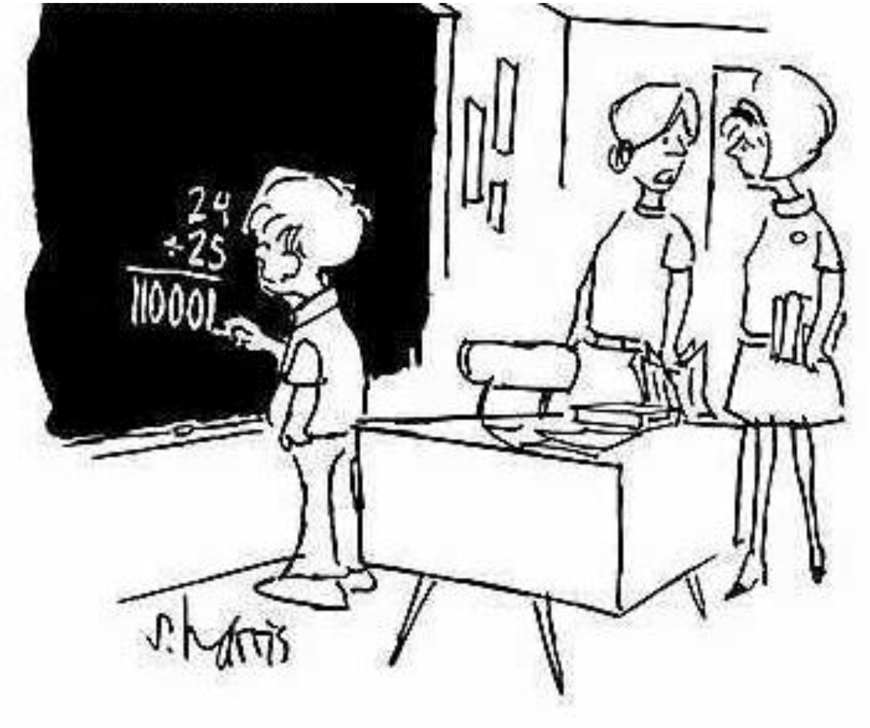
	A1	A2	A3	A4	A5	A6	A7	class	predicted class (prob.)	P(A2 = x)
#98	7	1	1	1	7	3	4	?	"cp" (p_{cp} = 0.83)	0.003
#98	7	2	1	1	7	3	4	?	"cp" (p_{cp} = 0.97)	0.046
#98	7	3	1	1	7	3	4	?	"cp" (p_{cp} = 0.98)	0.102
#98	7	4	1	1	7	3	4	?	"cp" (p_{cp} = 0.98)	0.215
#98	7	5	1	1	7	3	4	?	"cp" (p_{cp} = 0.96)	0.222
#98	7	6	1	1	7	3	4	?	"cp" (p_{cp} = 0.90)	0.182
#98	7	7	1	1	7	3	4	?	"pp" (p_{cp} = 0.20)	0.062
#98	7	8	1	1	7	3	4	?	"pp" (p_{cp} = 0.20)	0.061
#98	7	9	1	1	7	3	4	?	"pp" (p_{cp} = 0.39)	0.043
#98	7	10	1	1	7	3	4	?	"pp" (p_{cp} = 0.15)	0.055
#98	7	11	1	1	7	3	4	?	"cp" (p_{cp} = 0.66)	0.006
#98	7	12	1	1	7	3	4	?	"cp" (p_{cp} = 0.57)	0.003

Predicted class: 0.66 times "cc" OR p'_{cp} = 0.79

More complicated when several values are missing.

The same approach can easily be used to treat missing values on the training set (if the classifier supports weighted examples).

Feature Binarization



"It was bound to happen—they're beginning to think like binary computers."

Regression, Non-Continuous Features

Example dataset
(sensory data, 576 instances, 11 features, continuous class):

```
Occasion {1, 2}
Judges {1, 2, 3, 4, 5, 6}
Interval {1, 2, 3}
Sittings {1, 2, 3, 4}
Position {1, 2, 3, 4}
Squares {1, 2}
Rows {1, 2, 3}
Columns {1, 2, 3, 4}
Halfplot {1, 2}
Trellis {1, 2, 3, 4}
Method {1, 2}
CLASS real
```

One of the most basic approaches is using **linear regression**.
However, we need **numeric** attributes, not **nominal**.

Simple solution = straightforward transformation of feature values to discrete numeric values. Results (using 10-fold CV):

```
Correlation coefficient          0.1229
Mean absolute error              0.6583
Root mean squared error          0.8189
```

```
Linear Regression Model

score =

     -0.0836 * Judges +
     -0.0768 * Rows +
      0.0528 * Trellis +
     15.3891
```

Regression, Binary Features

Alternative solution = transform each value of a nominal feature to a binary feature:

```
Occasion {1, 2}
Judges {1, 2, 3, 4, 5, 6}
Interval {1, 2, 3}
Sittings {1, 2, 3, 4}
Position {1, 2, 3, 4}
Squares {1, 2}
Rows {1, 2, 3}
Columns {1, 2, 3, 4}
Halfplot {1, 2}
Trellis {1, 2, 3, 4}
Method {1, 2}
score numeric
```

```
Occasion numeric
Judges=1 numeric
Judges=2 numeric
Judges=3 numeric
Judges=4 numeric
Judges=5 numeric
Judges=6 numeric
Interval=1 numeric
Interval=2 numeric
Interval=3 numeric
Sittings=1 numeric
Sittings=2 numeric
Sittings=3 numeric
Sittings=4 numeric
Position=1 numeric
Position=2 numeric
Position=3 numeric
Position=4 numeric
Squares numeric
Rows=1 numeric
Rows=2 numeric
Rows=3 numeric
Columns=1 numeric
Columns=2 numeric
Columns=3 numeric
Columns=4 numeric
Halfplot numeric
Trellis=1 numeric
Trellis=2 numeric
Trellis=3 numeric
Trellis=4 numeric
Method numeric
score numeric
```

Results:

```
Linear Regression Model

score =

      0.2656 * Judges=2 +
      0.1719 * Judges=3 +
     -0.224  * Judges=4 +
     -0.2969 * Judges=6 +
      0.1068 * Interval=2 +
     -0.1644 * Position=2 +
      0.4167 * Rows=2 +
     -0.1536 * Rows=3 +
     -0.2778 * Trellis=2 +
      0.1875 * Trellis=3 +
     15.0289
```

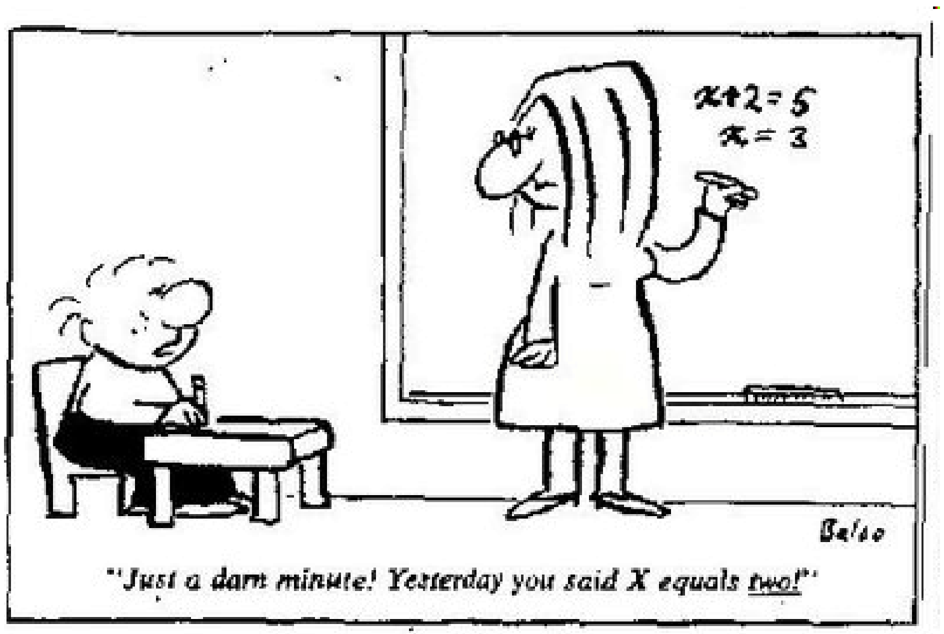
A more complex model.

```
Correlation coefficient                  0.1229
Mean absolute error                      0.6583
Root mean squared error                  0.8189
```

Better results.

```
Correlation coefficient                  0.3822
Mean absolute error                      0.6088
Root mean squared error                  0.7618
```

Incremental Learning



An Example From Sports Betting

Odds offered by a known online bookmaker for a UEFA Championship League quarter-final match between Inter and Manchester (home, draw, and away):

Inter Milan	2.30	3.10	2.90	**Manchester United**

Odds tell us what the payout for an individual outcome is. For example, betting 1€ on Inter will pay 2.3€. If they win, of course.

Odds also imply what the teams' chances of winning are. The odds above suggest that Inter is a slight favourite with a 43% chance of winning (1 / 2.30). Manchester, on the other hand, has a 34% chance.

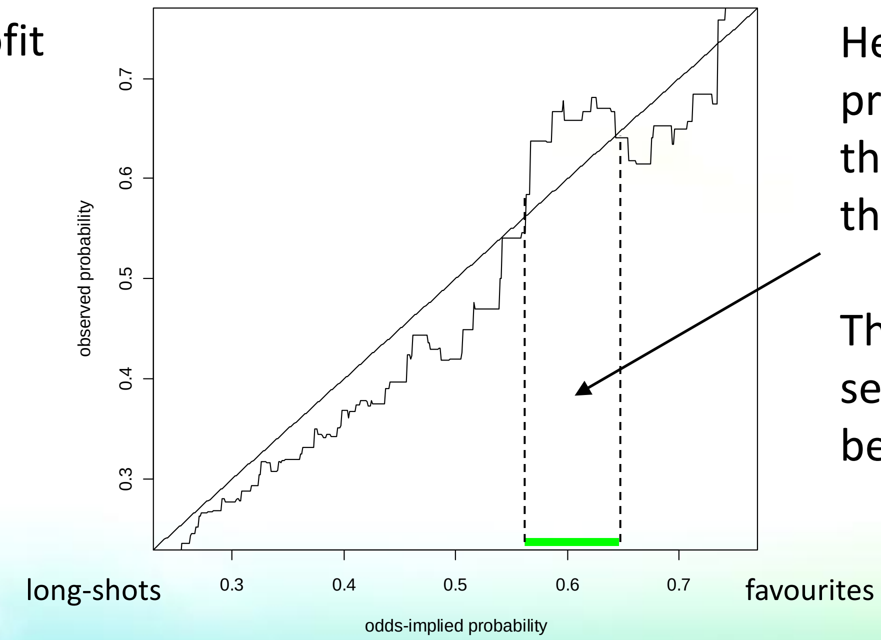
Task:
- A bookmaker offers us betting odds for soccer matches
- Bookmakers can make mistakes and publish favourable odds
- Can we beat the bookmaker and make money by betting on a particular odds band?

We Can Learn from Past Examples

A nearest neighbor approach is used to estimate the probability of the home team winning:

Using the odds and outcomes of 1000 past matches, we get:

The diagonal line indicates zero-profit opportunities.



Here the observed probability is larger than the probability implied by the odds.

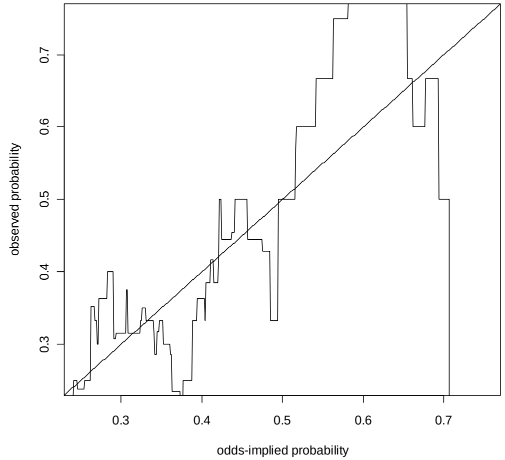
Therefore, this odds band seems to be a profitable betting opportunity.

Bet on the home team whenever the offered odds are between 1.5 and 1.8 !!!

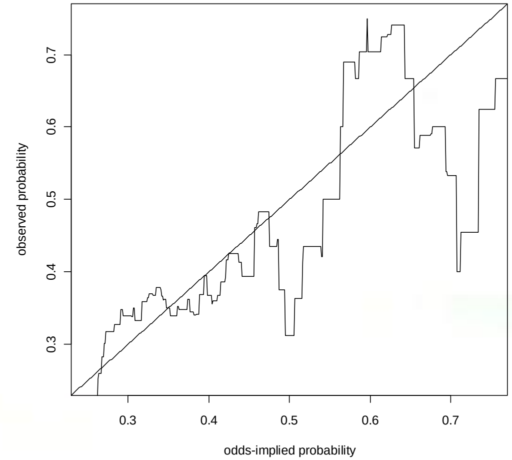
Our Estimation Improves over Time

However,...

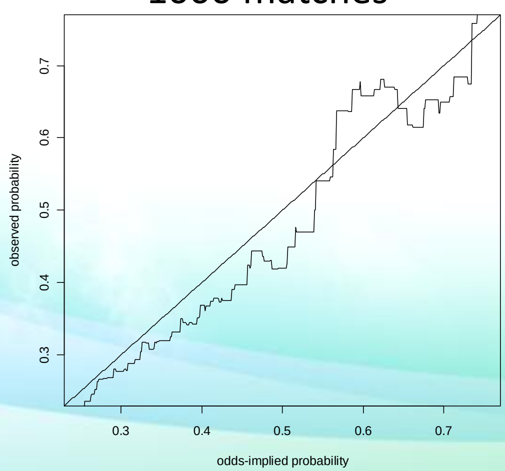
100 matches



500 matches

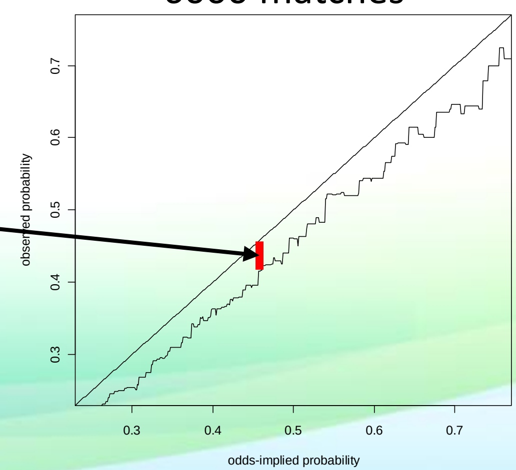


1000 matches



The bookmaker's profit margin.

6000 matches



Bootstrapping

An Illustrative Example

A simple and straightforward way of estimating standard errors, confidence intervals, etc...

An example:

From a large basket full of blue and yellow marbles we draw, at random, 100 marbles:

, etc ...

Let's say we drew 70 x  and 30 x 

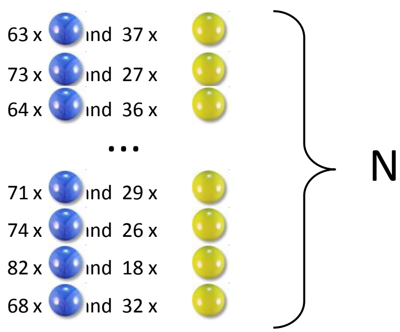
The estimated mean ratio of blue marbles in the basket is 0.70.
But how "good" is this estimation?

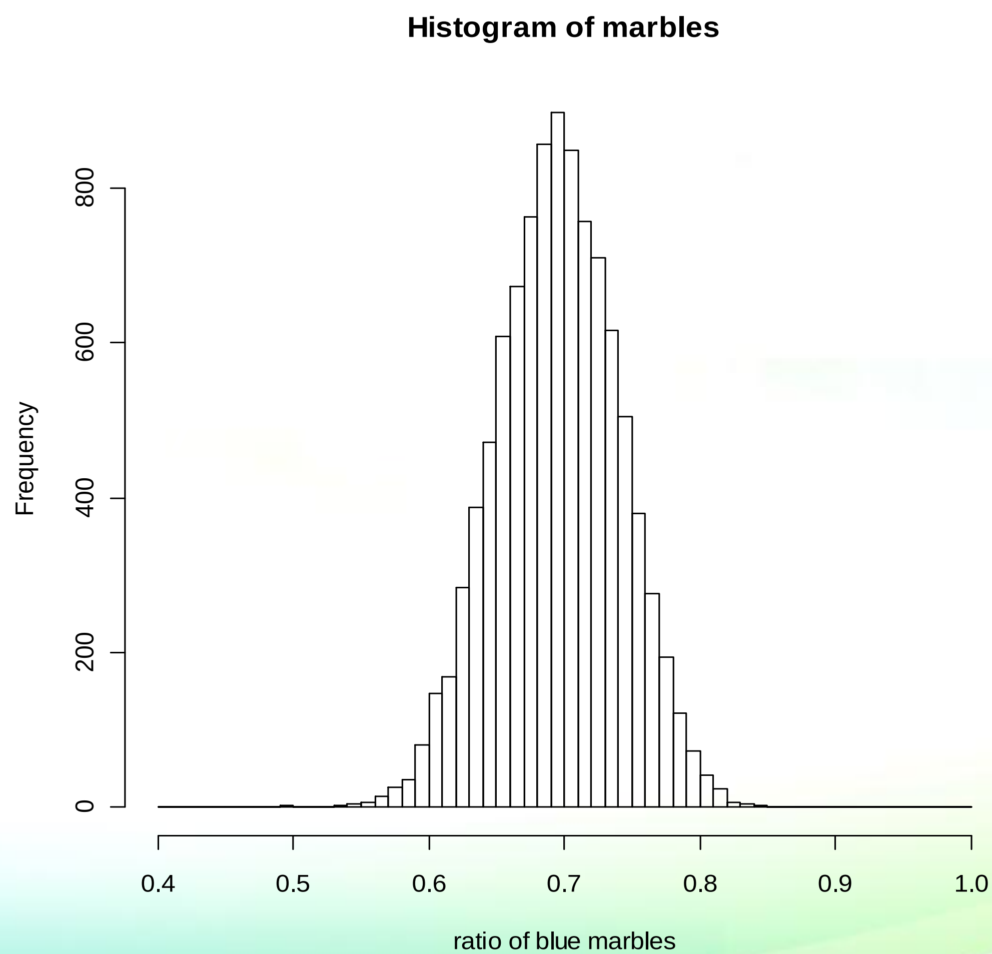
Resampling and C.I. Estimation

We put the selected 100 marbles in a bag,
draw 100 (with replacement!).

(approx. 37% repetitions per sample)

We repeat this process N times:
(recommended N >= 10.000)



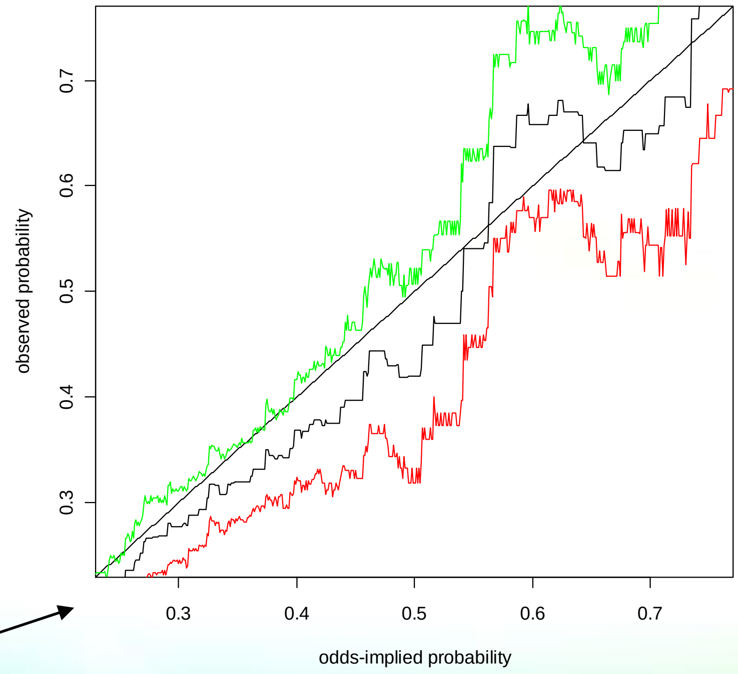


From the distribution of the sample means, we can estimate, for example, the 90% confidence interval for the estimated mean.

In our case:
between 0.63 and 0.77

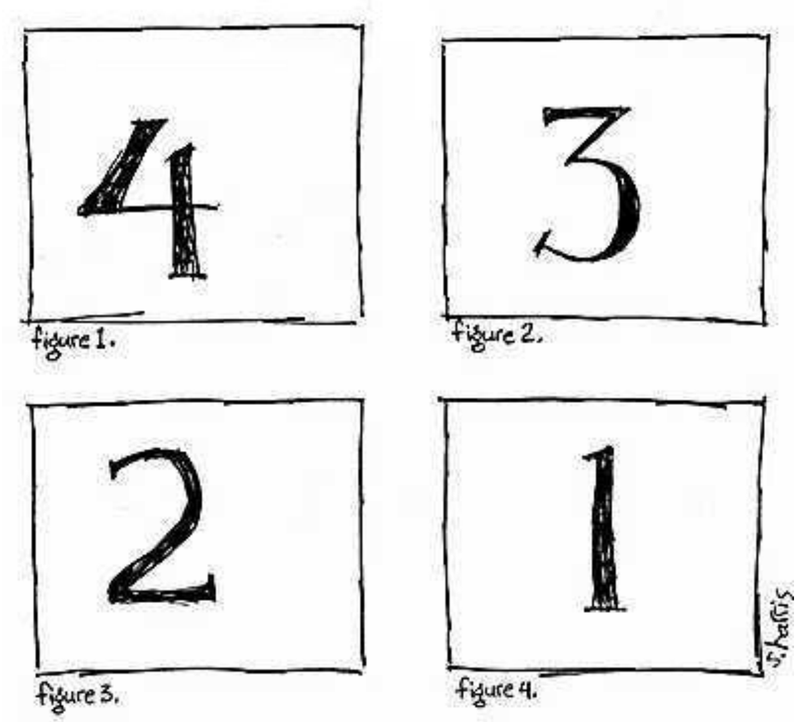
A Practical Application

One of the advantages of bootstrapping is that there is no need to assume that the data follows a normal distribution or any other statistical distribution.

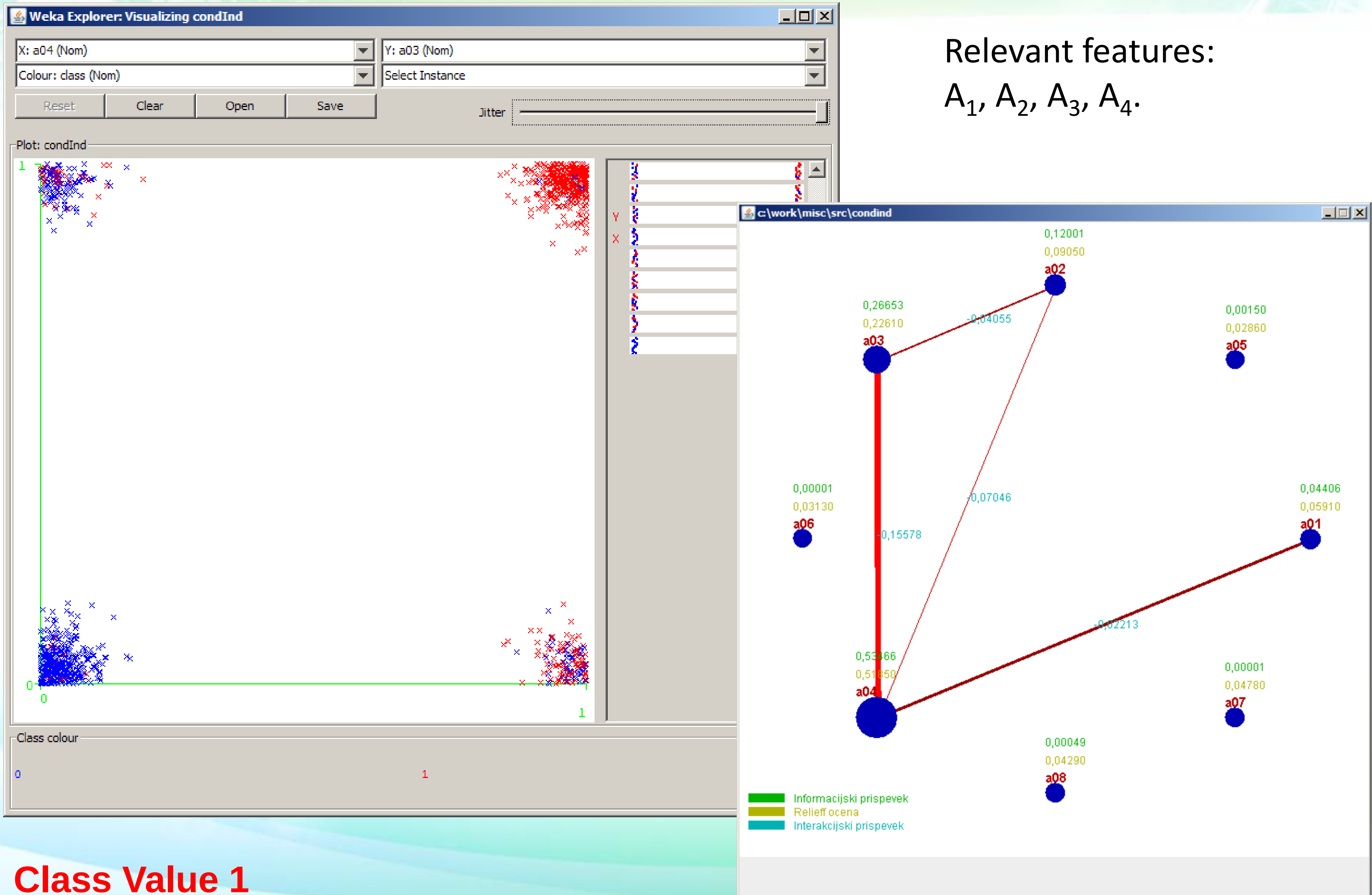


After observing the 90% confidence intervals (obtained by bootstrapping), we can no longer be certain that the (0.55, 0.65) odds band is profitable.

Feature Visualization



CondInd Dataset



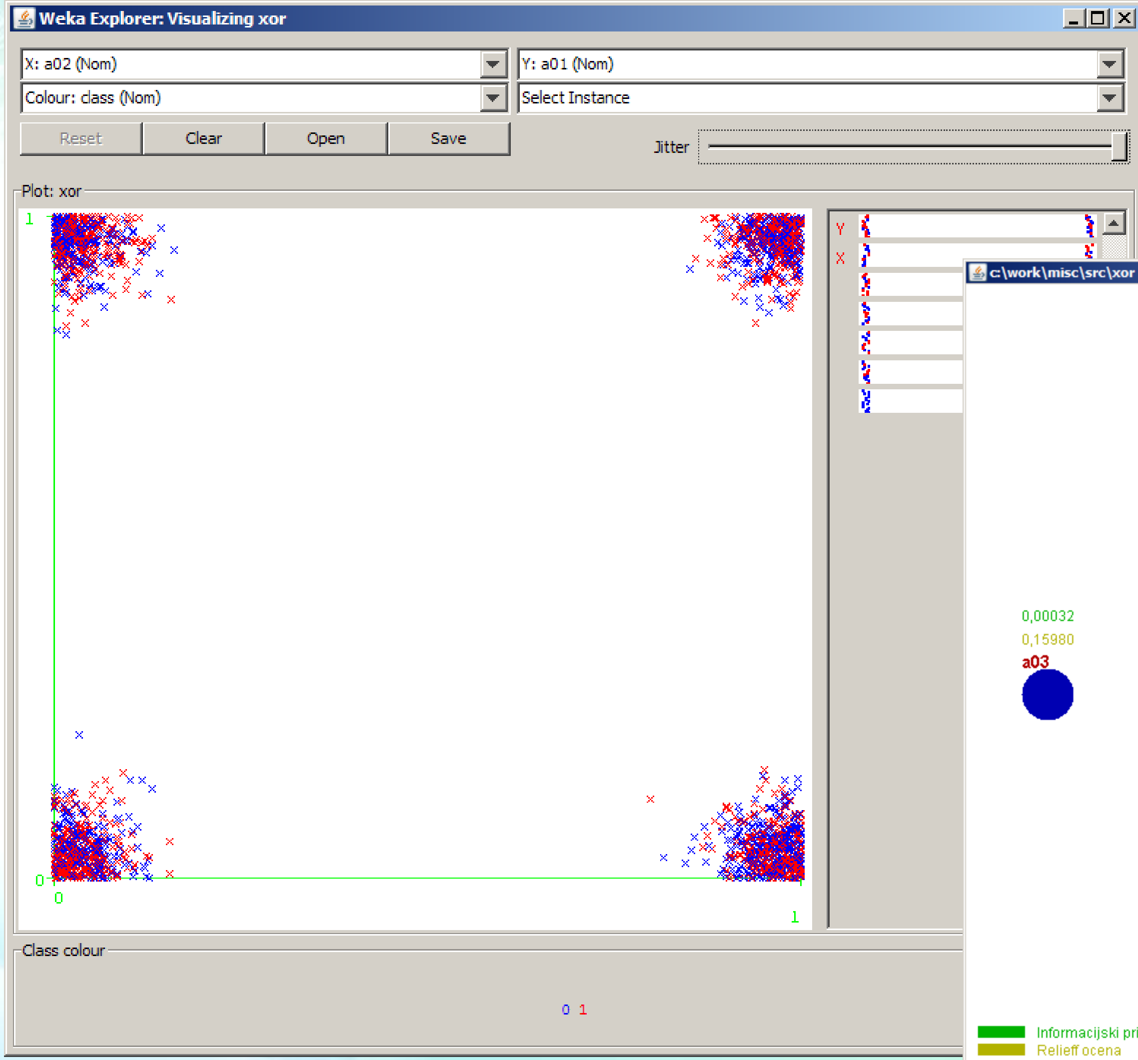
Relevant features:

A_1, A_2, A_3, A_4.

Class Value 1

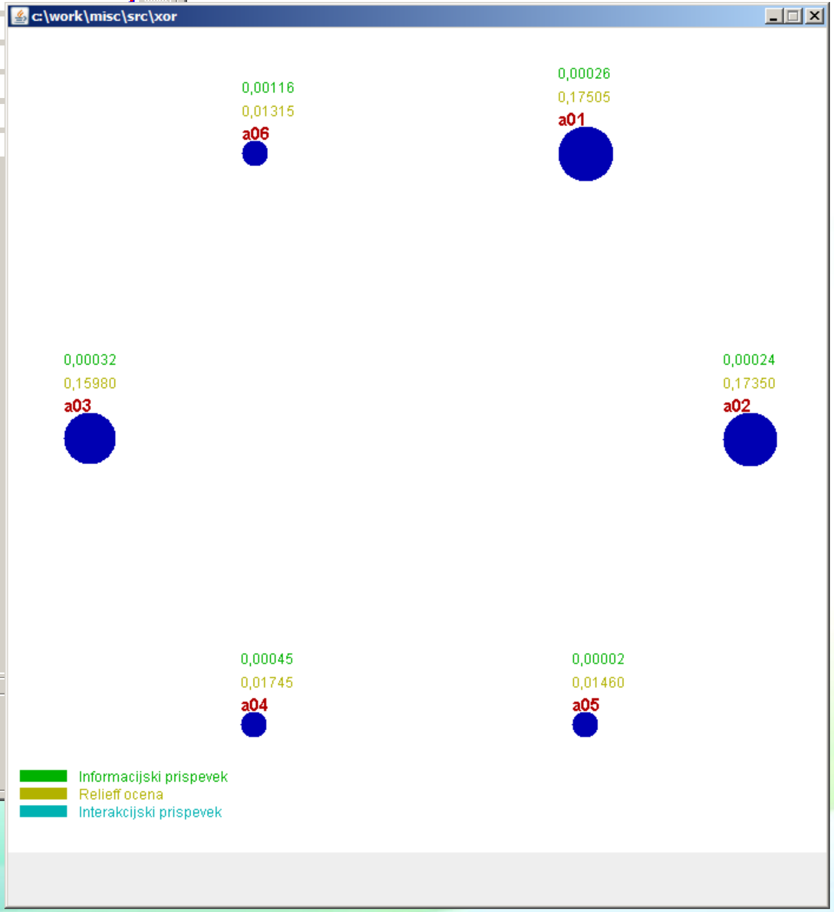
Class Value 0

Xor Dataset



Relevant features:

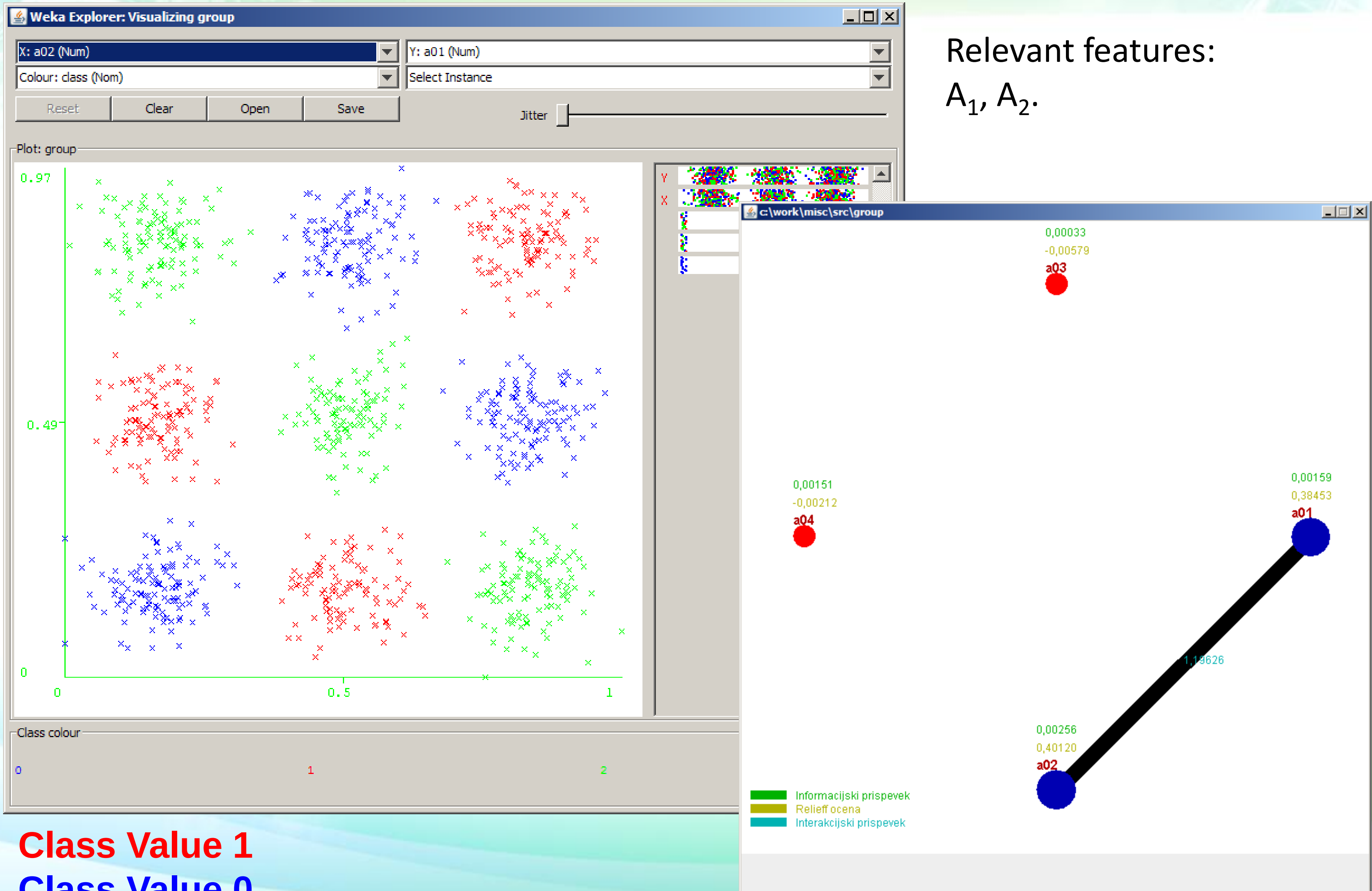
A_1, A_2, A_3.



Class Value 1

Class Value 0

Groups Dataset



Relevant features:

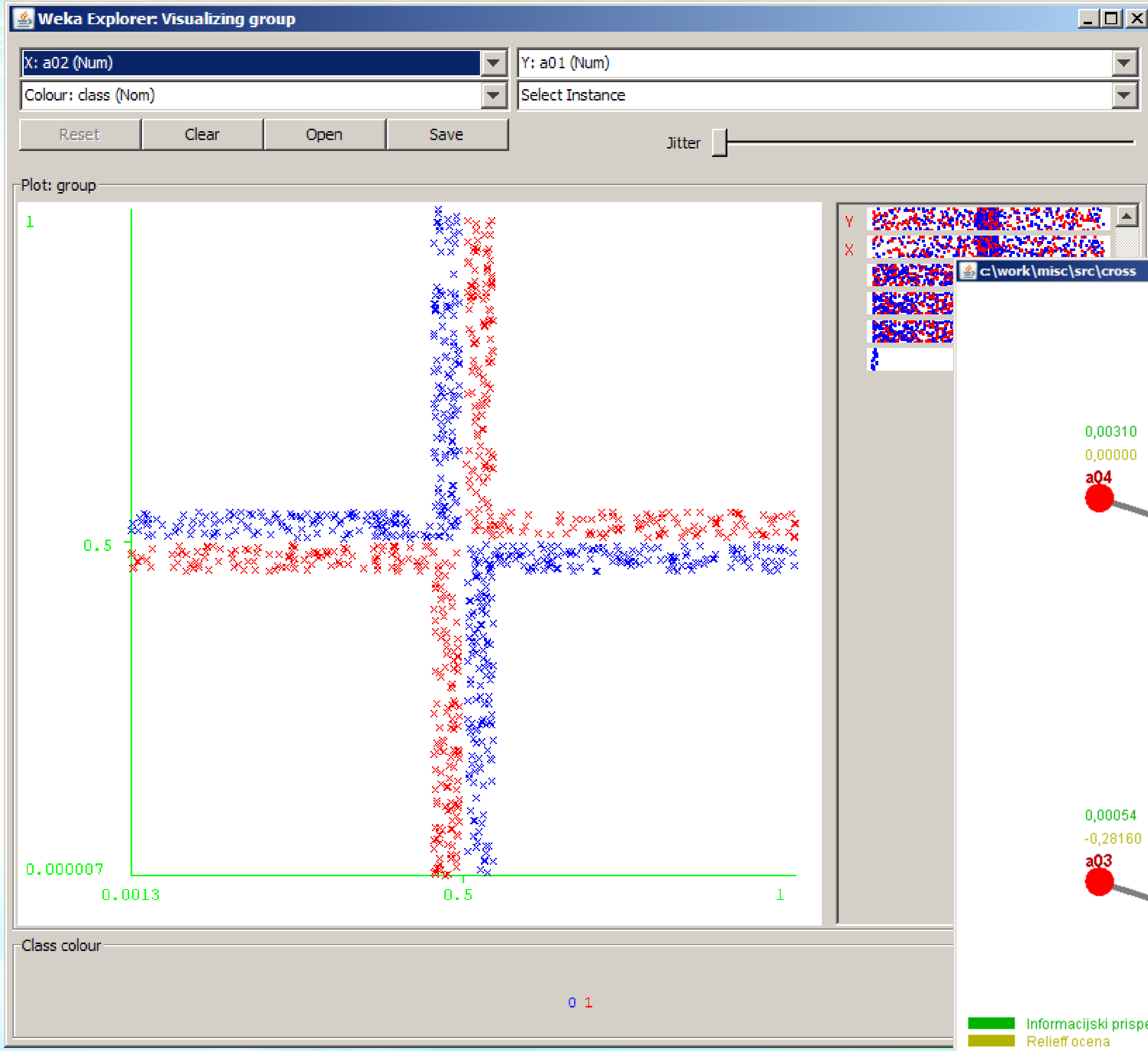
A_1, A_2.

Class Value 1

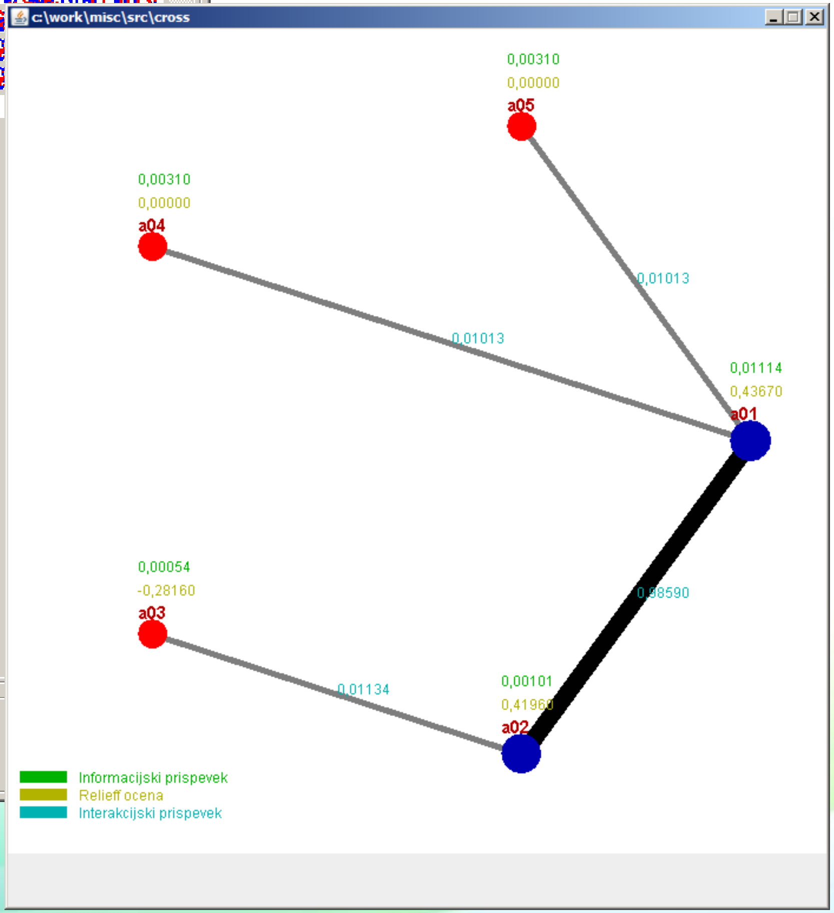
Class Value 0

Class Value 2

Cross Dataset

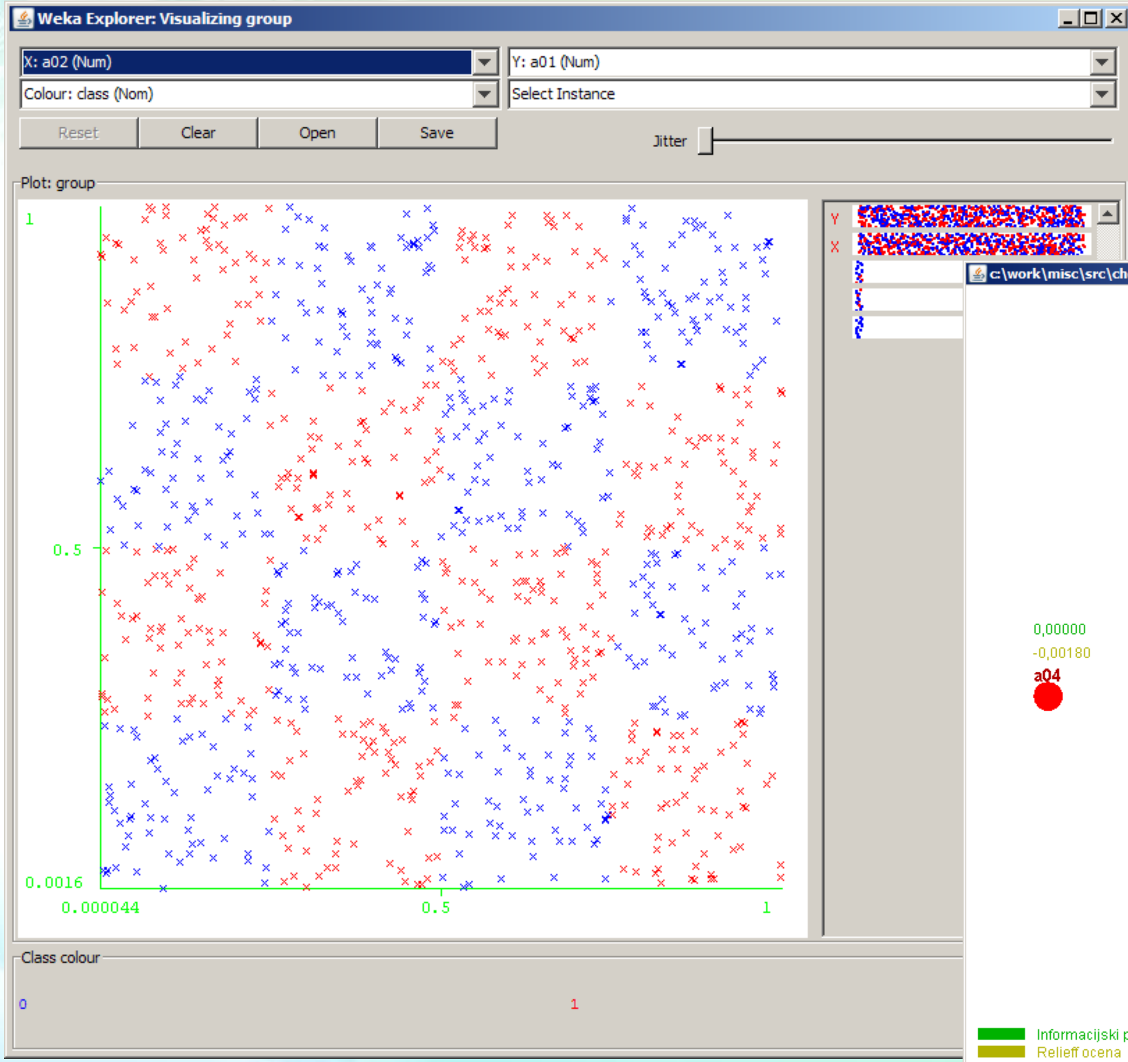


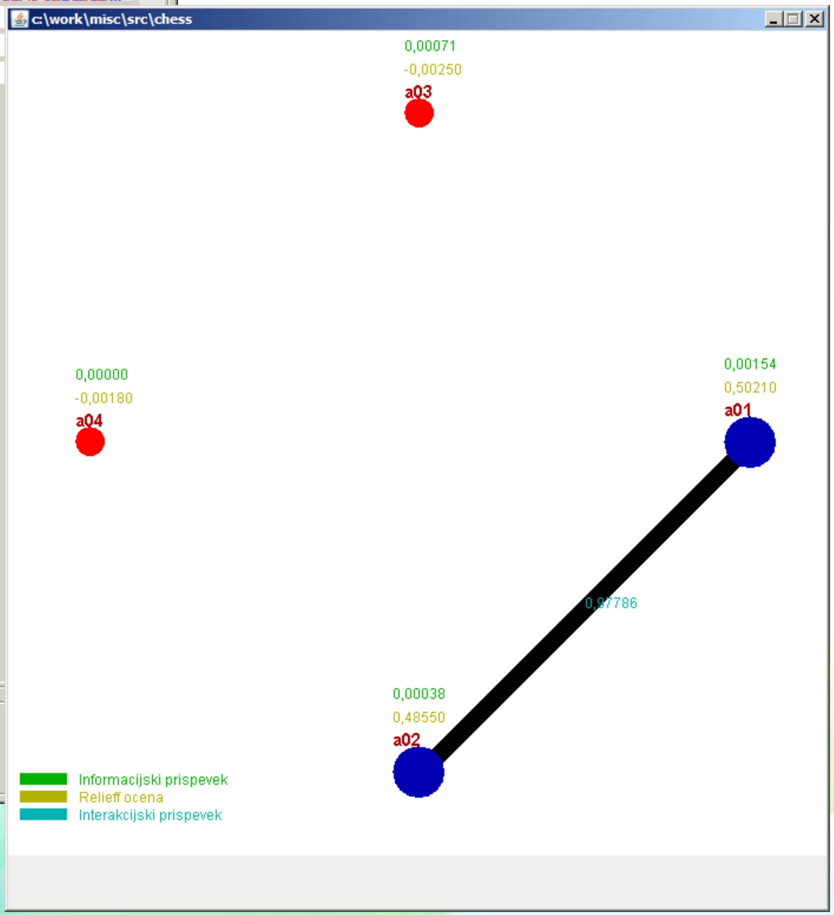
Relevant features:
A_1, A_2.



Class Value 1
Class Value 0

Cross Dataset





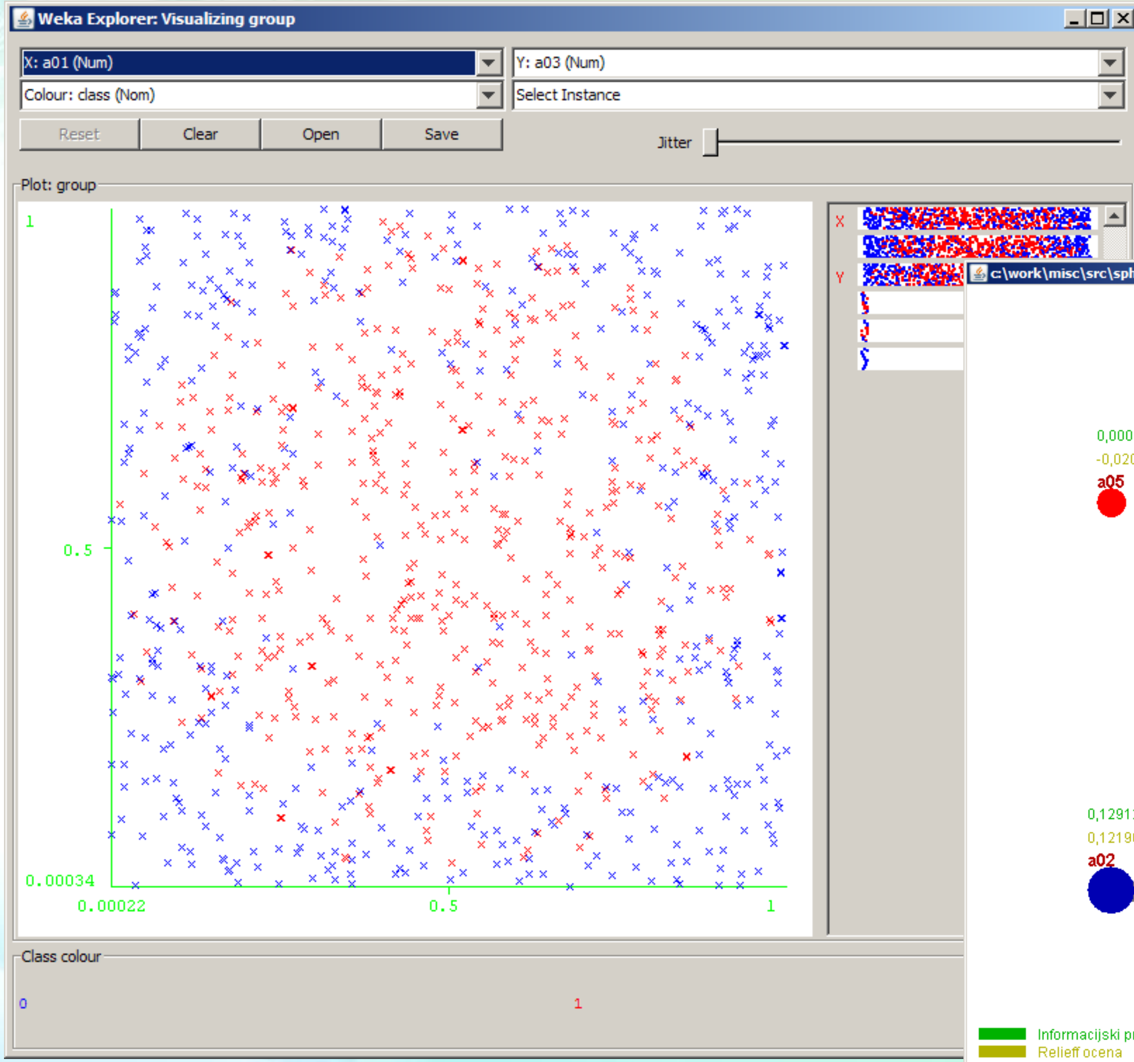
Relevant features:

A_1, A_2.

Class Value 1

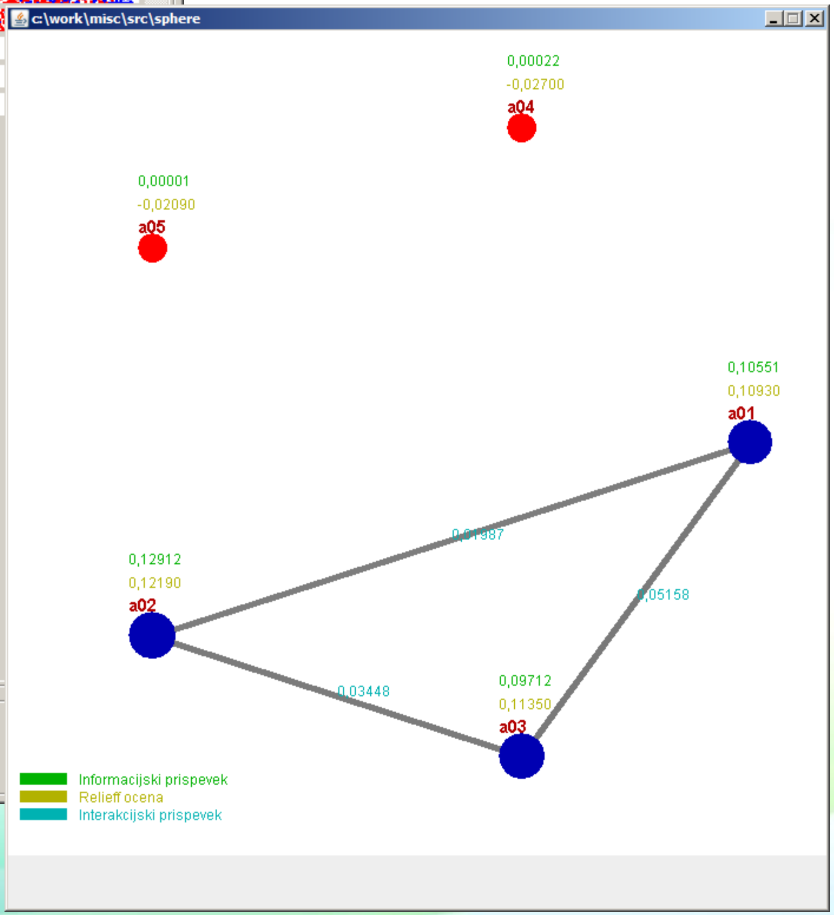
Class Value 0

Sphere Dataset



Relevant features:

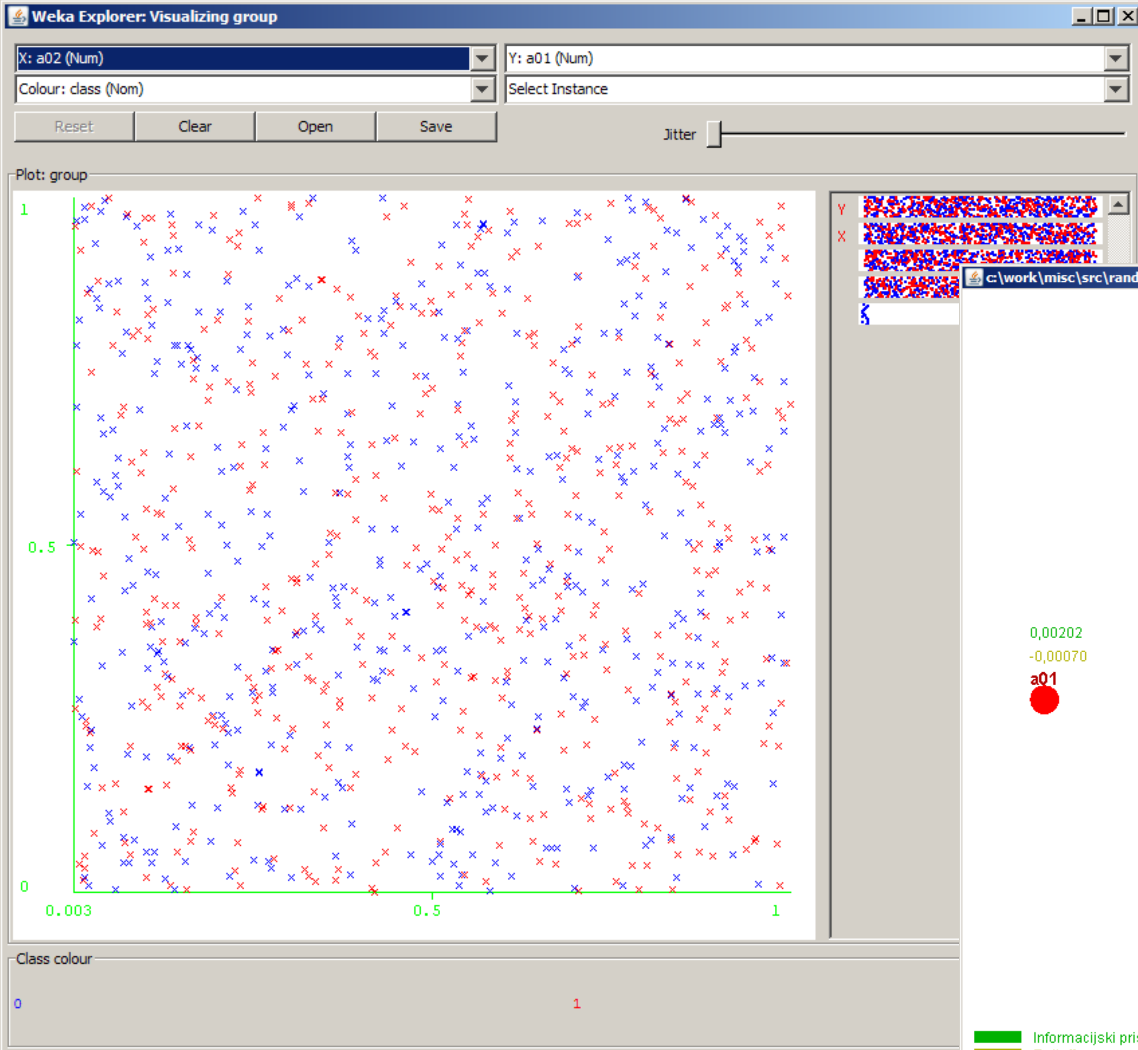
A_1, A_2, A_3.

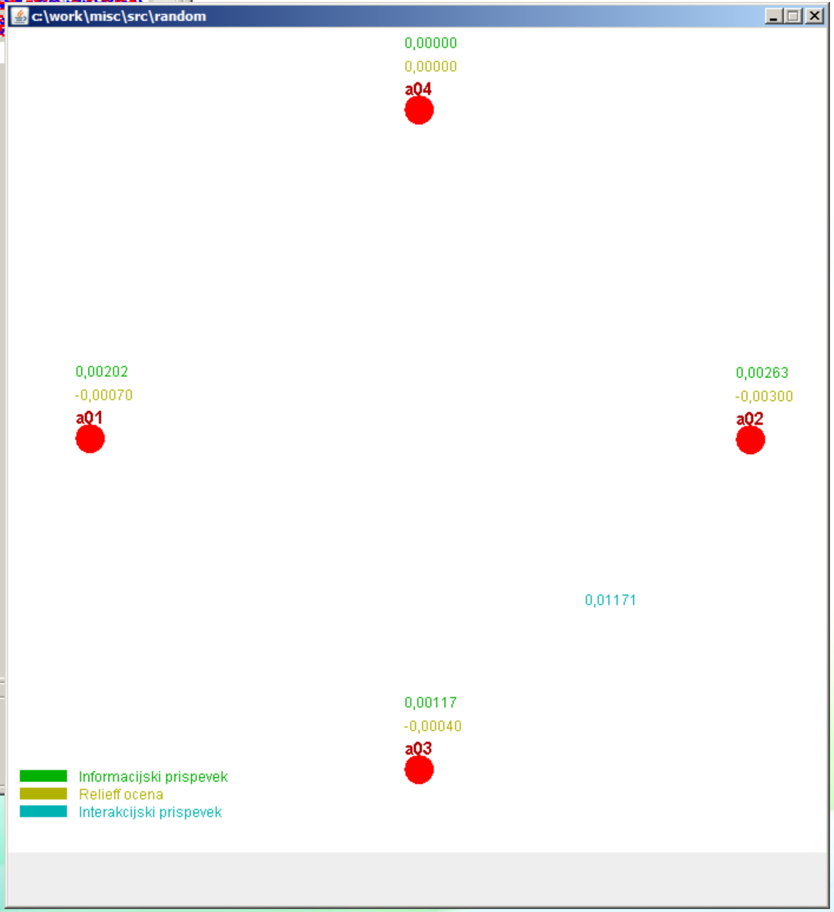


Class Value 1

Class Value 0

Random Dataset





Relevant features:

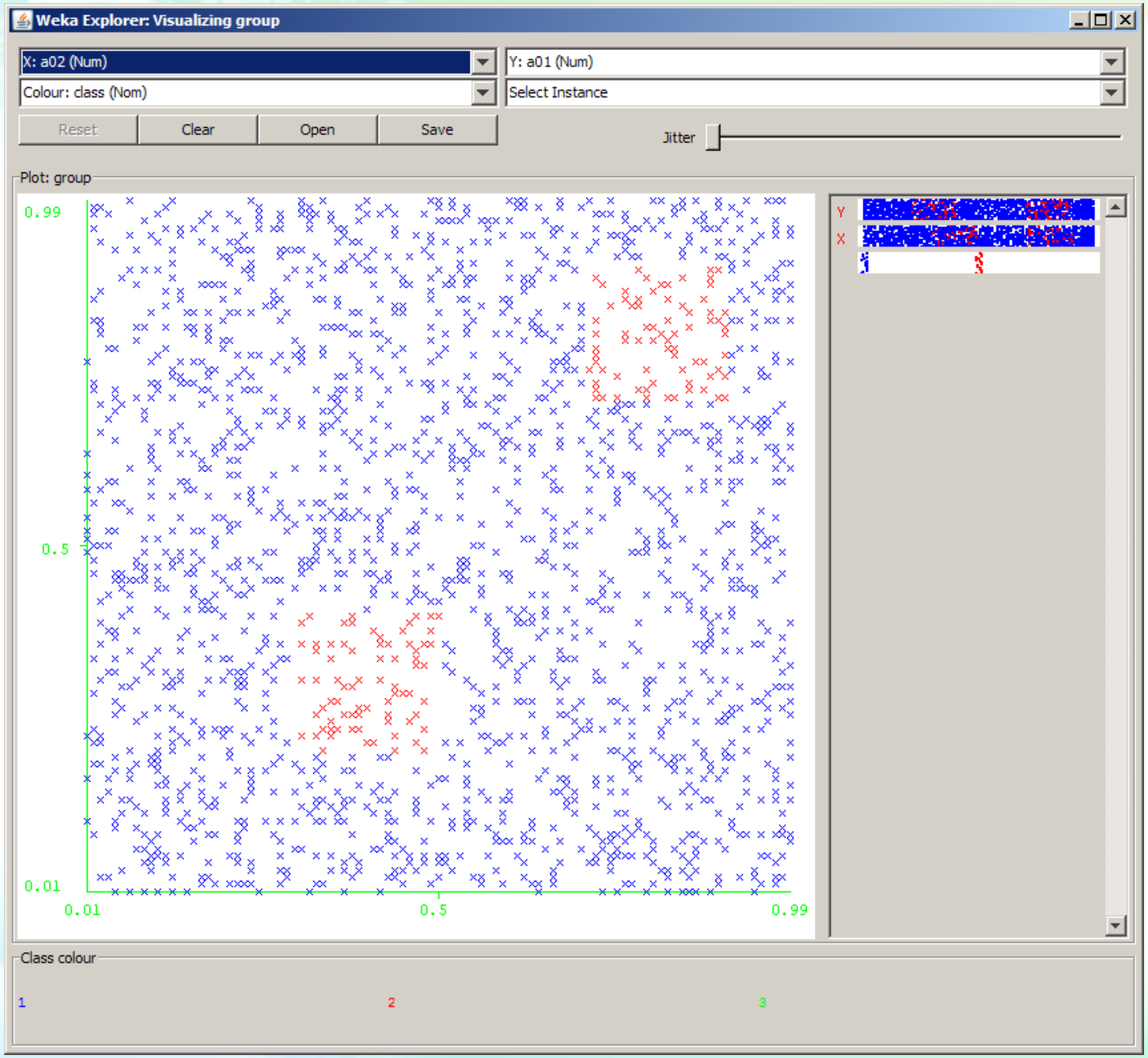
-

Class Value 1

Class Value 0

Decision Rules and Trees

An Illustrative Example



Relevant features:

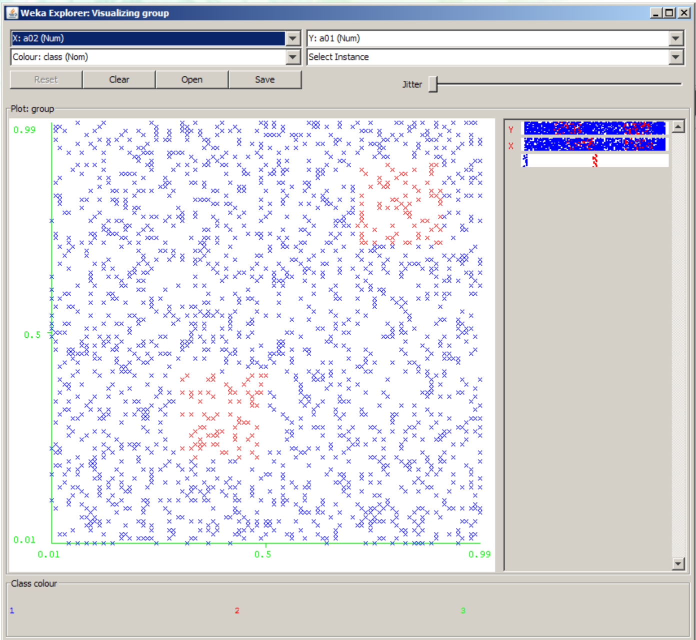
A_2, A_1.

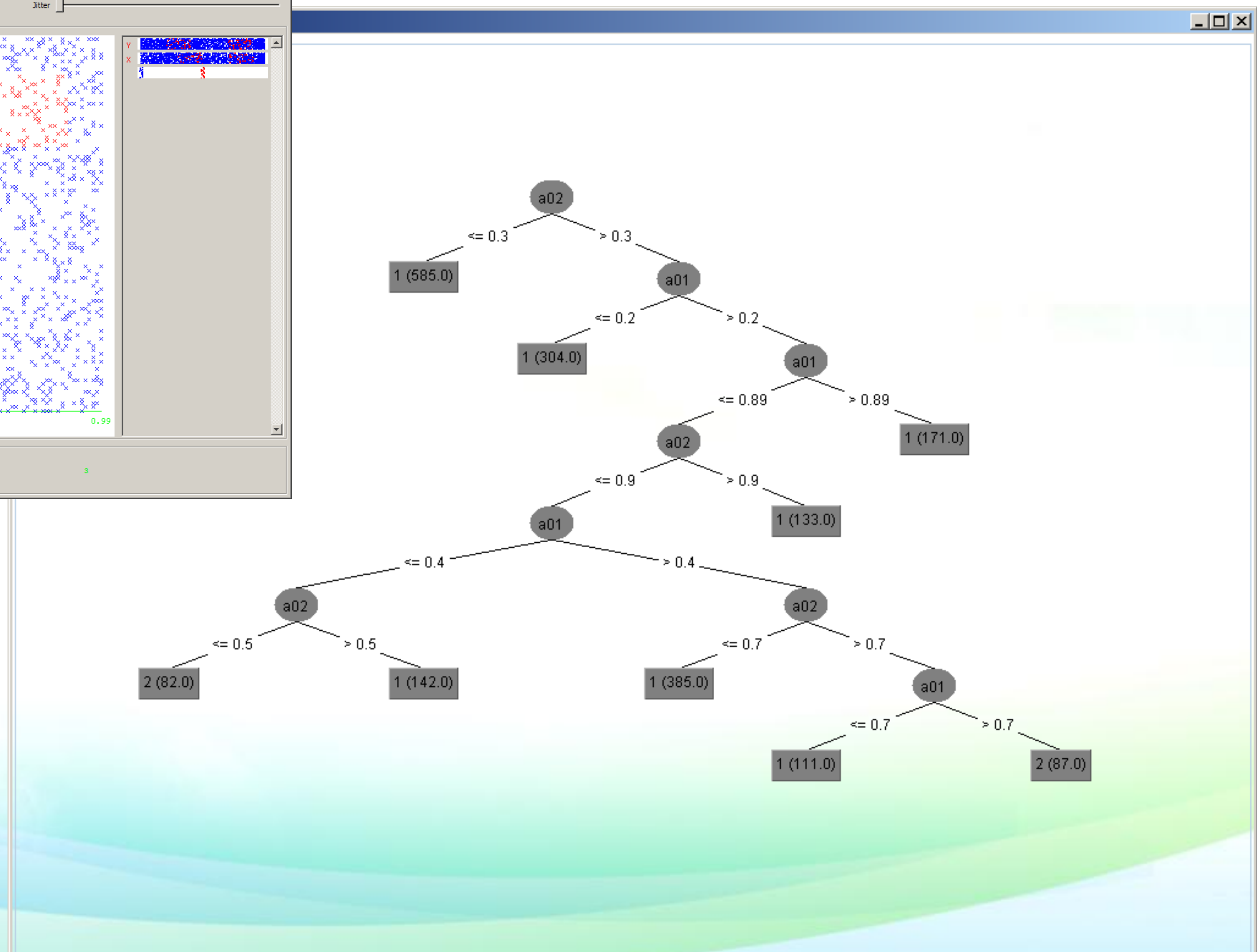
Class Value 2

Class Value 1

A Decision Tree

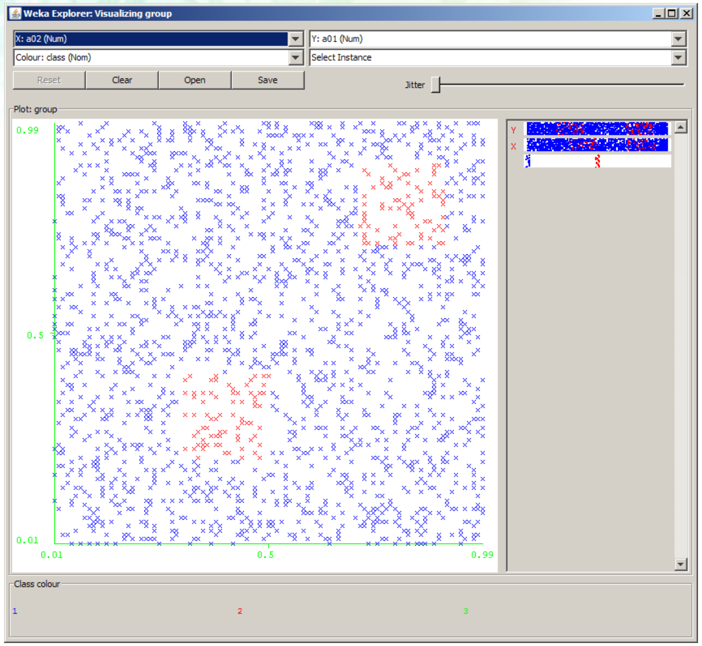
This decision tree achieves 99.95% accuracy.





A Set of Decision Rules

These decision rules achieve 99.85% accuracy.

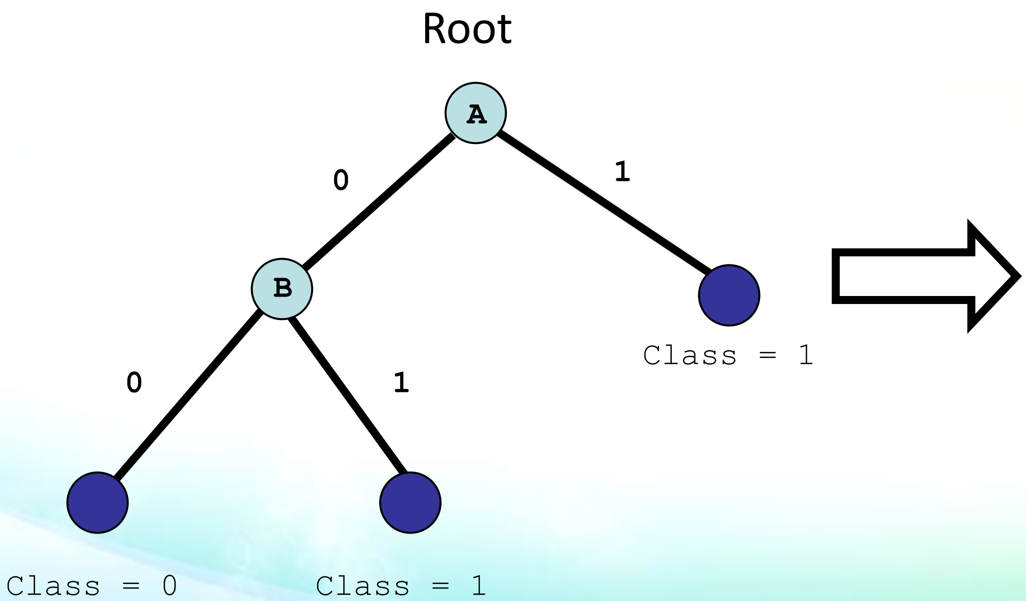



```
class = 1  (2000.0/169.0)

Except (a02 > 0.305) and (a02 <= 0.495) and (a01 <= 0.405) and (a01 > 0.205)
   => class = 2  (58.0/0.0) [23.0/0.0]

Except (a02 > 0.715) and (a01 > 0.705) and (a01 <= 0.895) and (a02 <= 0.905)
   => class = 2  (58.0/0.0) [28.0/0.0]
```

Decision Tree -> Set of Rules

Straightforward:
- Each terminal node corresponds to one rule
- The antecendent is composed of rules along the path from the root to the terminal node
- The consequent is the class value assigned in the terminal node

Problem: Can lead to an overly complex set of rules!

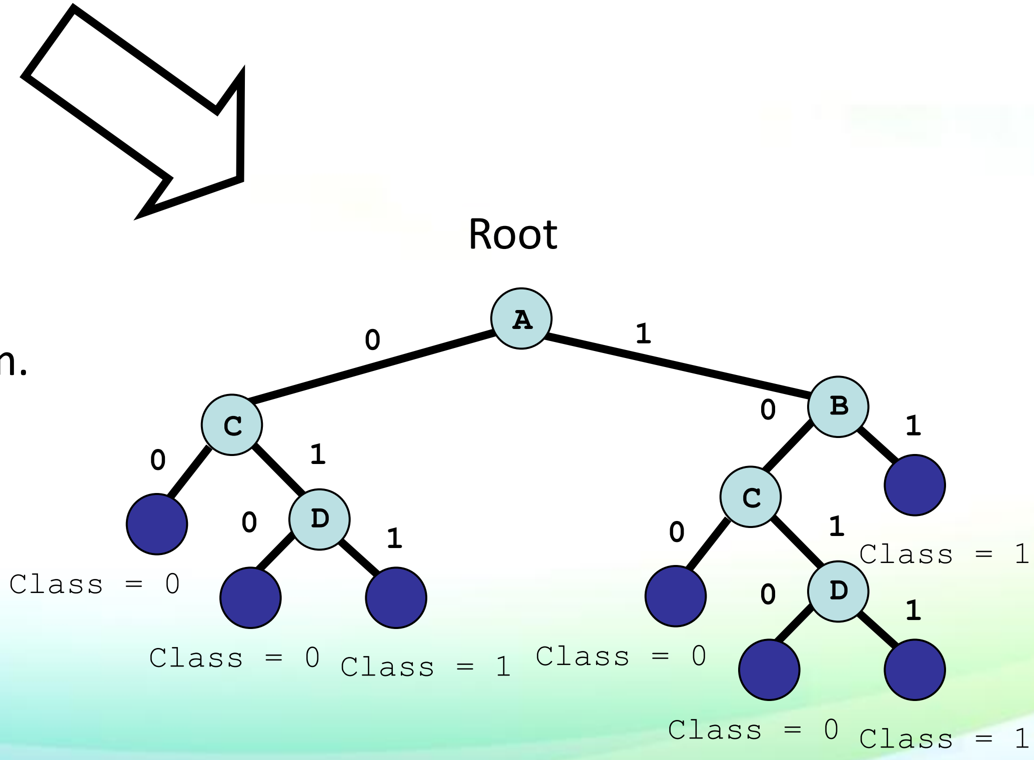



```
if (A = 1) then class = 1

if (A = 0 and B = 0) then class = 0

if (A = 0 and B = 1) then class = 1
```

Set of Rules -> Decision Tree

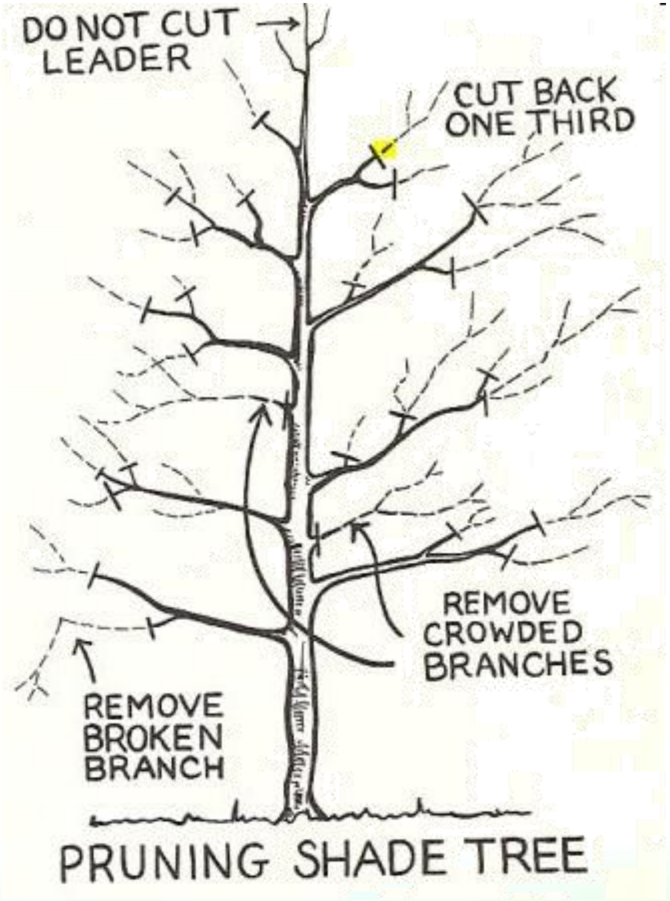
Usually more difficult!

```
if (A = 1 and B = 1) then class = 1

if (C = 1 and D = 1) then class = 1
```



"Repeated subtree" problem.

Decision Tree Pruning

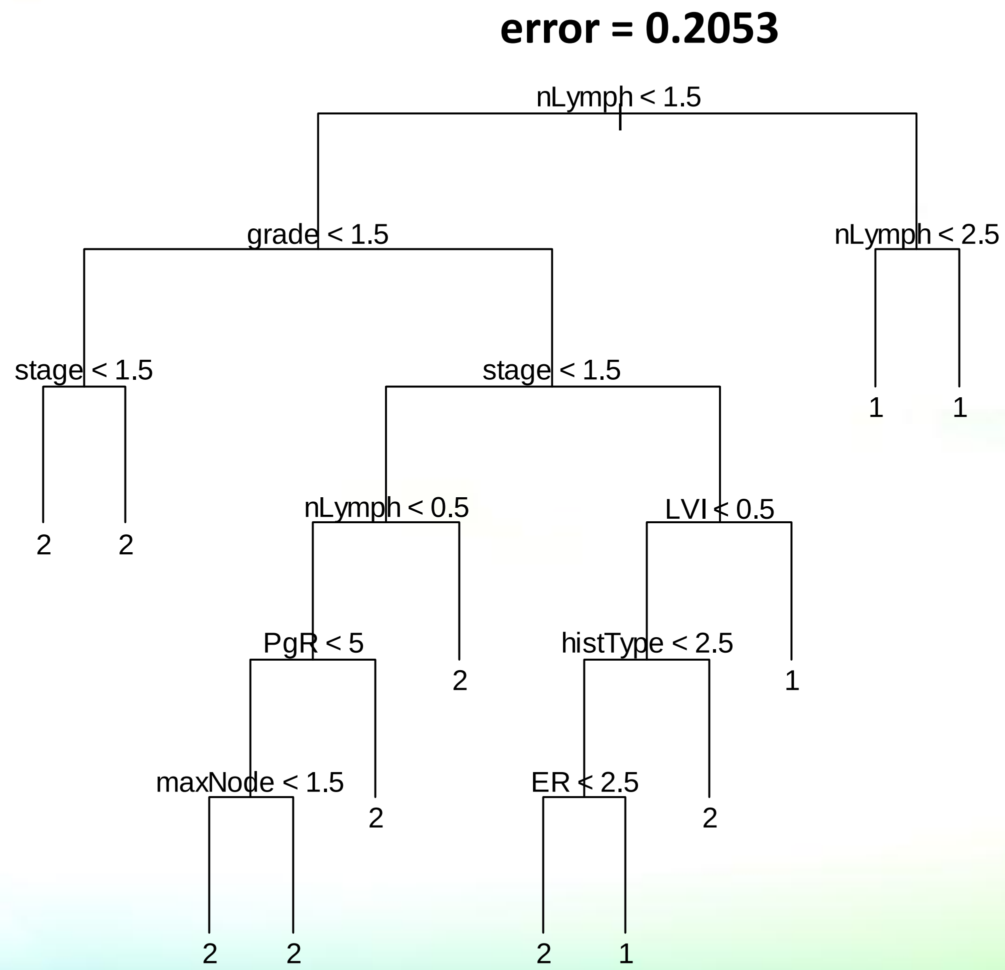


Part 1: Classification Tree

An Unpruned Classification Tree

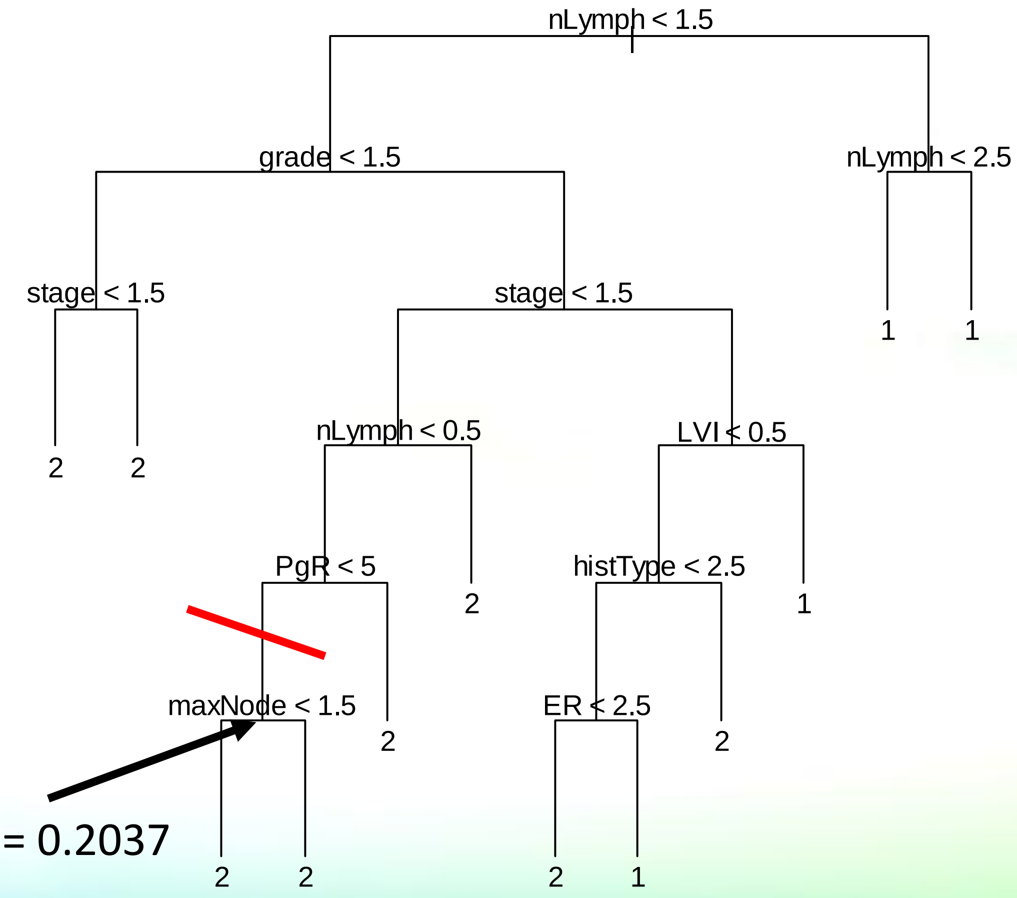
Oncology Training Dataset
+
Independent Pruning Dataset

We use the Brier score to measure the error.



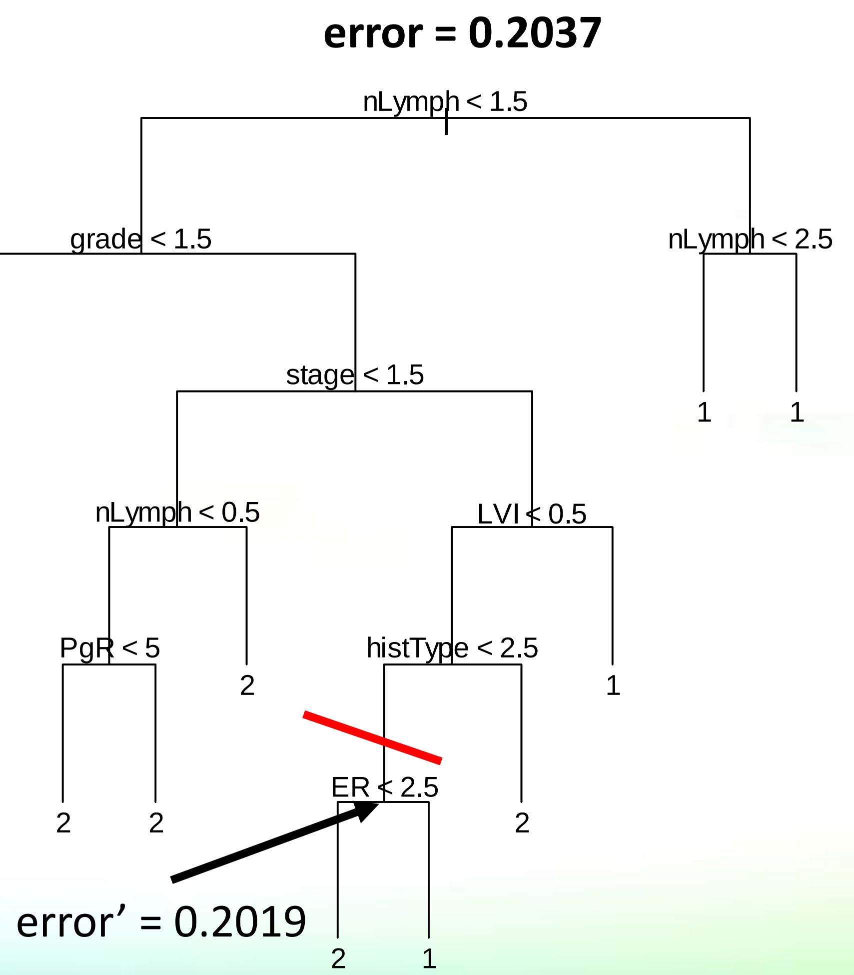
Snip,...

error = 0.2053



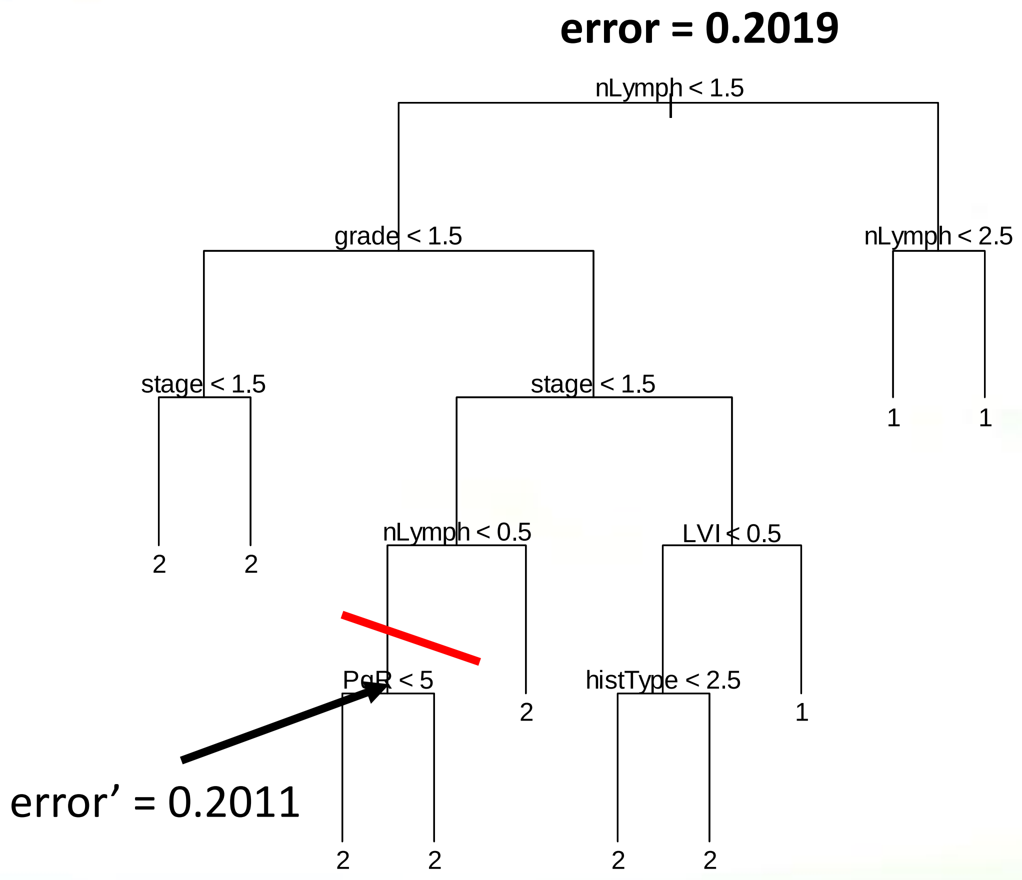
error' = 0.2037

Snip,...

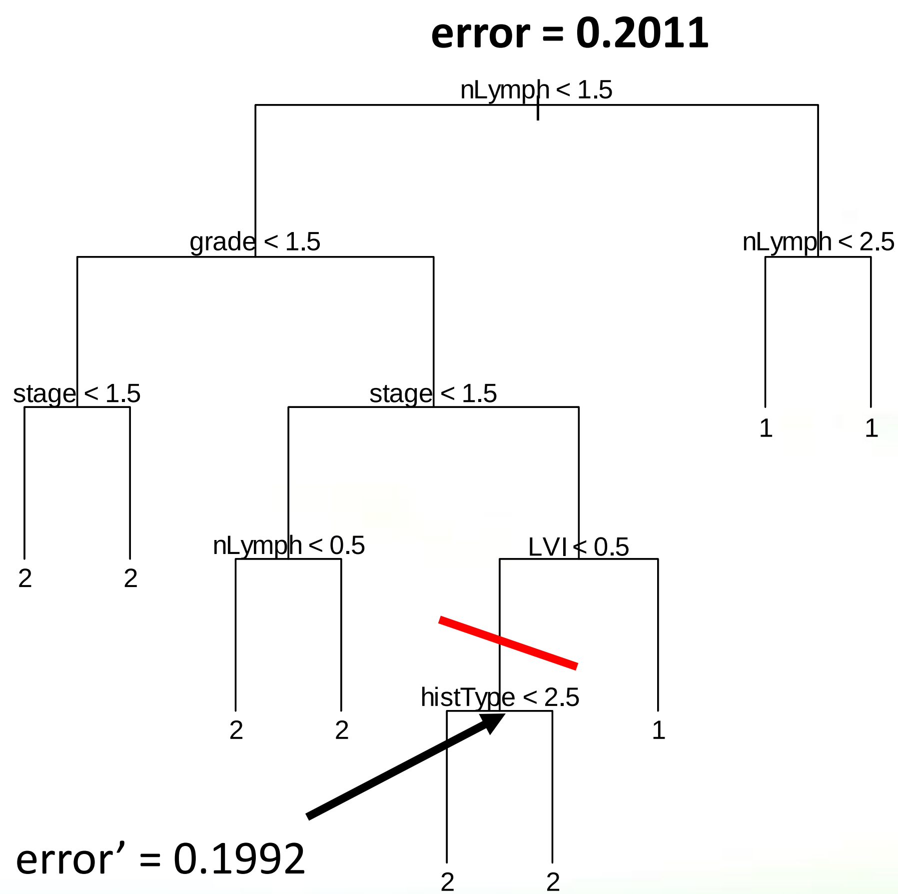


error = 0.2037
nLymph < 1.5
grade < 1.5
nLymph < 2.5
stage < 1.5
stage < 1.5
1
1
2
2
nLymph < 0.5
LVI < 0.5
PgR < 5
2
histType < 2.5
1
ER < 2.5
2
2
2
error’ = 0.2019
2
1

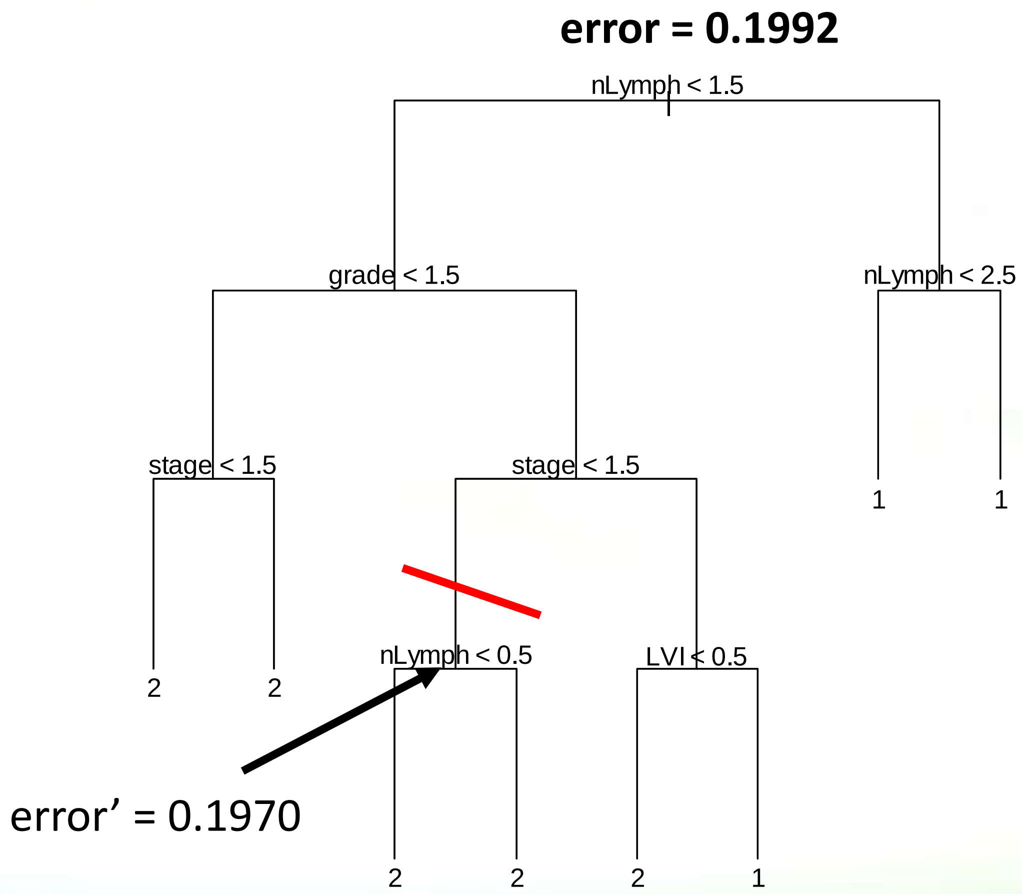
Snip,...


error = 0.2019
nLymph < 1.5
grade < 1.5
nLymph < 2.5
stage < 1.5
stage < 1.5
1
1
2
2
nLymph < 0.5
LVI < 0.5
PgR < 5
2
histType < 2.5
1
error' = 0.2011
2
2
2
2

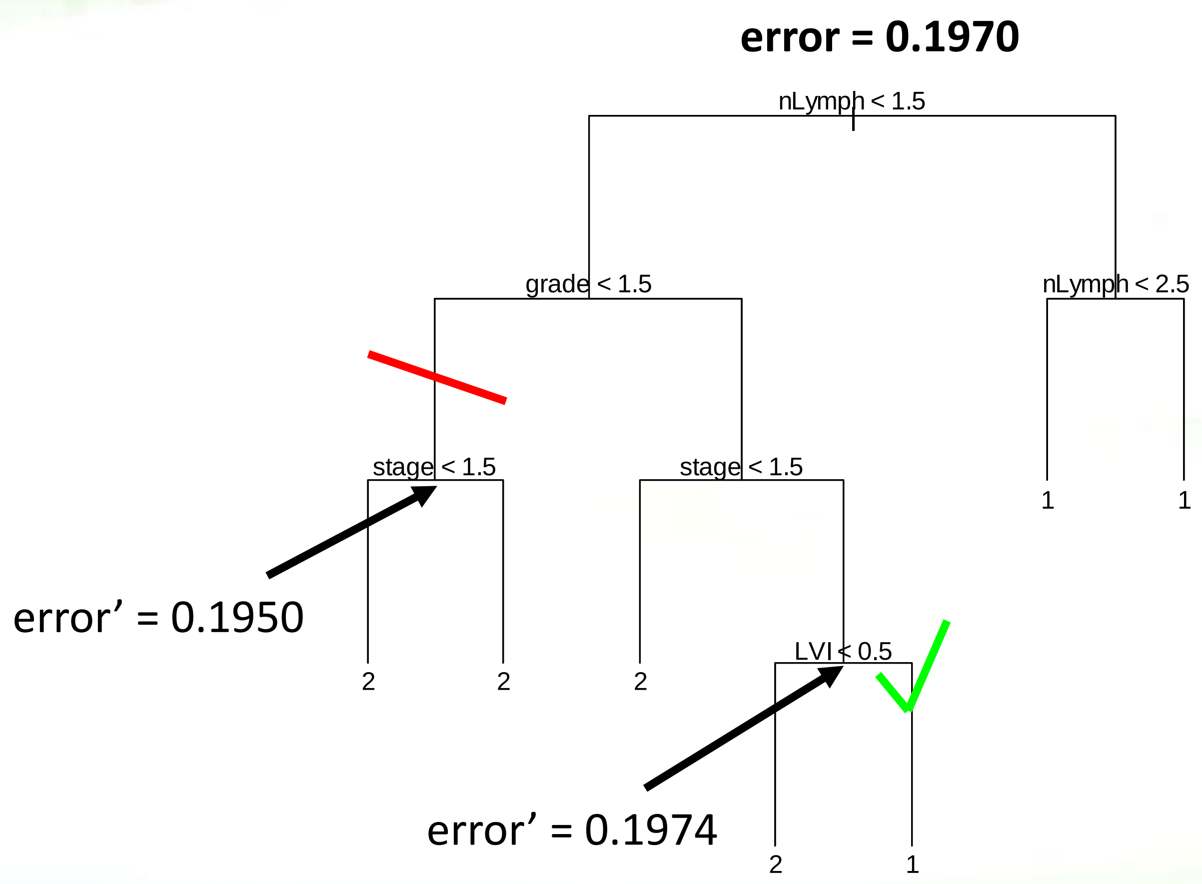
Snip,...


error = 0.2011
nLymph < 1.5
grade < 1.5
nLymph < 2.5
stage < 1.5
stage < 1.5
1
1
2
2
nLymph < 0.5
LVI < 0.5
histType < 2.5
2
2
1
error' = 0.1992
2
2

Snip,...

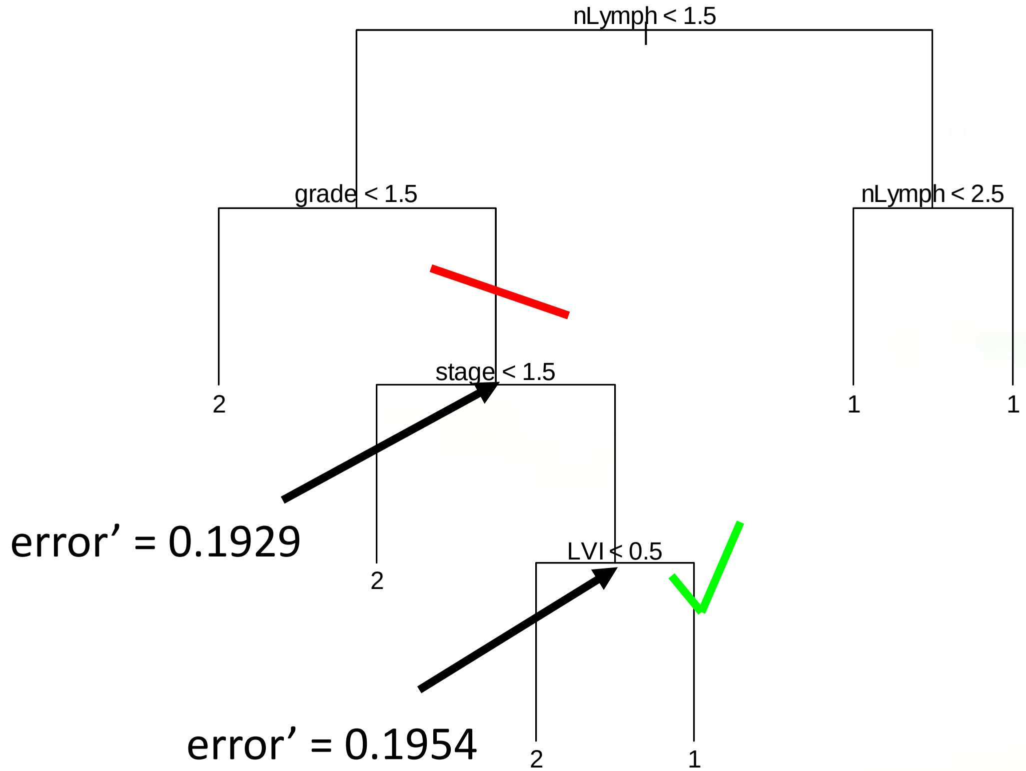

error = 0.1992
nLymph < 1.5
grade < 1.5
nLymph < 2.5
stage < 1.5
stage < 1.5
1
1
nLymph < 0.5
LVI < 0.5
2
2
error’ = 0.1970
2
2
2
1

Snip,...

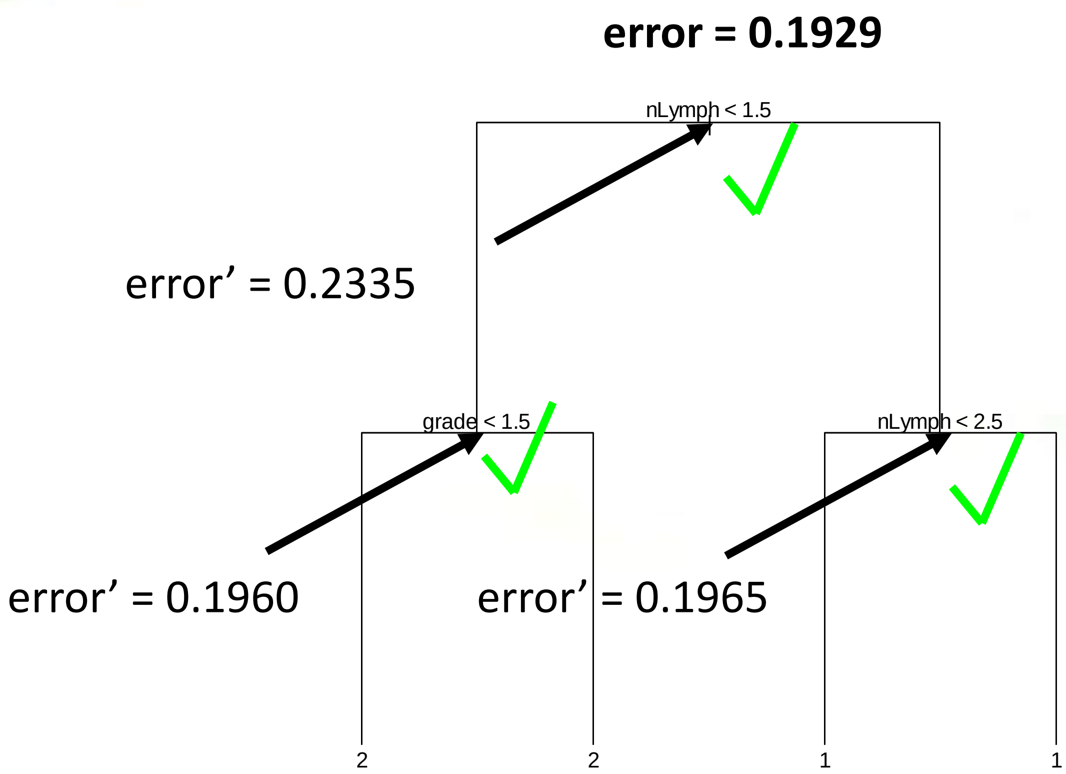

error = 0.1970
nLymph < 1.5
grade < 1.5
nLymph < 2.5
stage < 1.5
stage < 1.5
1
1
error' = 0.1950
LVI < 0.5
2
2
2
error' = 0.1974
2
1

Snip,...

error = 0.1950



Pruned Tree



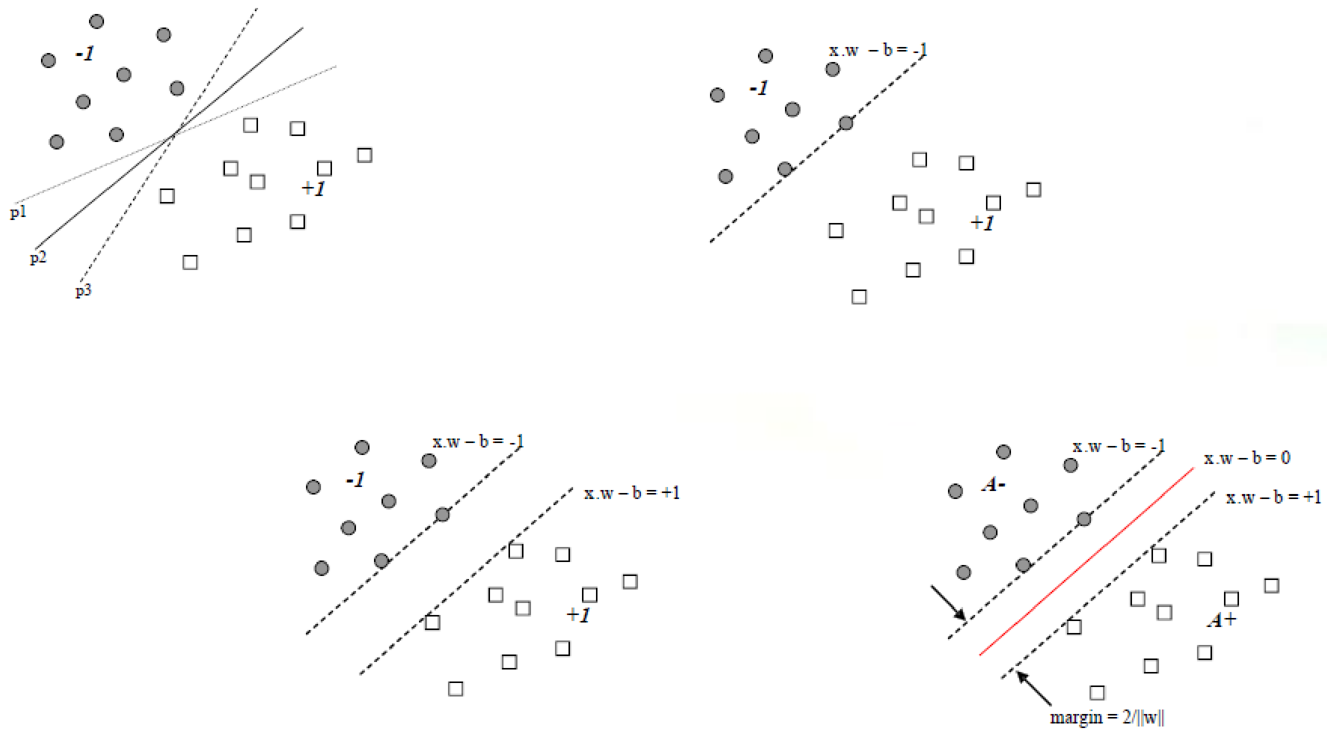
None of the remaining nodes justifies pruning.

Numerical Methods

Linear Regression

Java applet

SVM

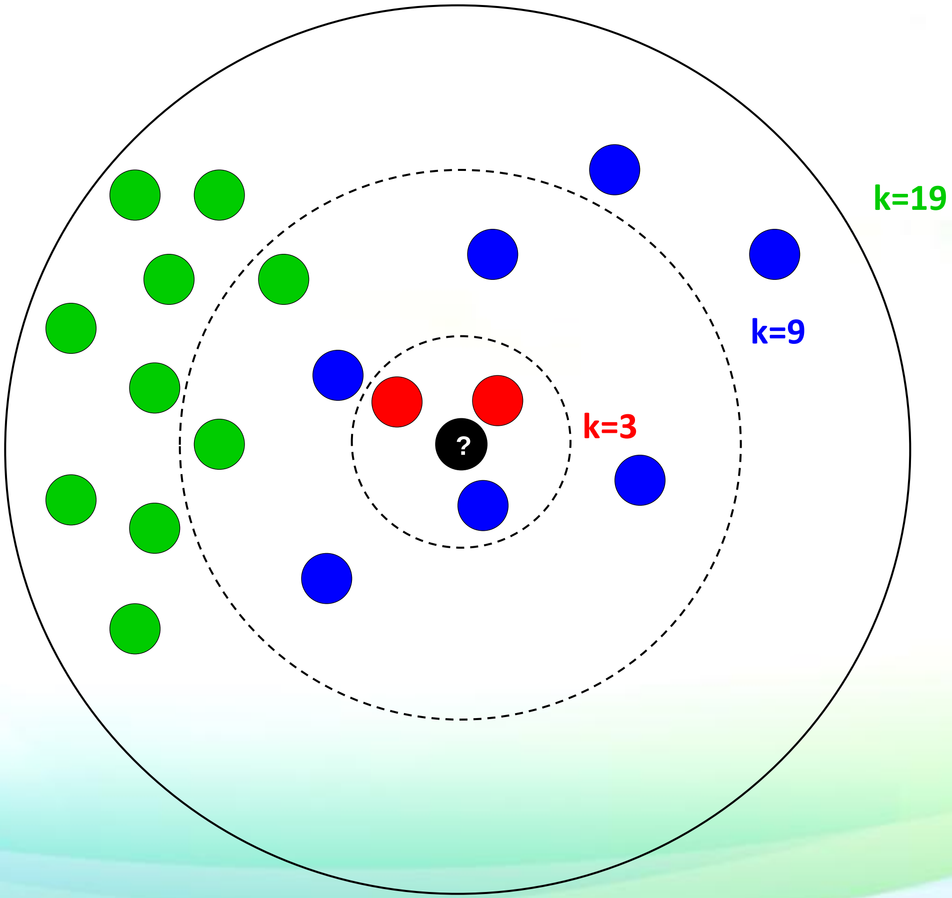


SVM

SVM video

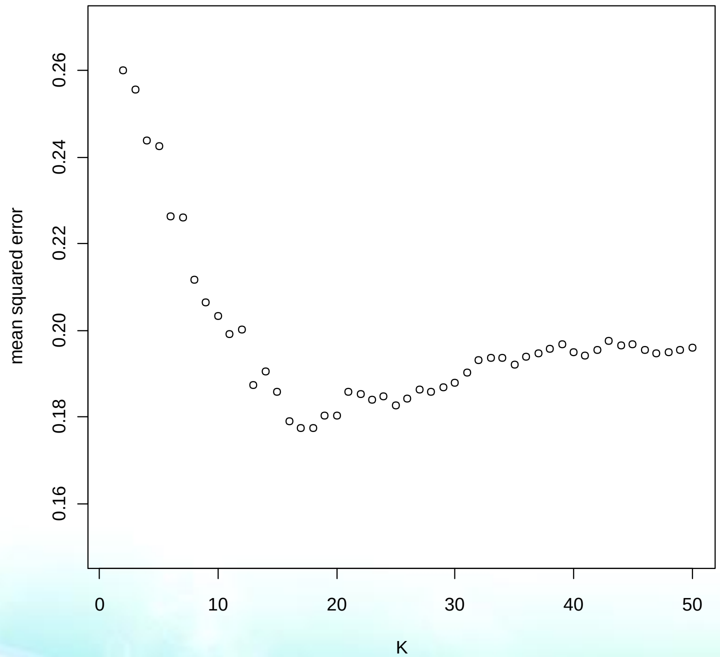
k-Nearest Neighbors

Different values of *k* can produce very different results:



k-Nearest Neighbors

Example:
- Soccer data, 100 test instances, 200 training instances
- Predicting home win from nearest neighboring past matches
- Similarity measure: absolute difference in home win probabilities (from bookmaker)



lower *k* => more variance in predictions (noise)

Tradeoff

higher *k* => a larger bias (less distinct boundaries)

Milan vs Atalanta
(Milan win odds: 1.5)

Arsenal vs Blackburn
(Arsenal win odds: 1.4))

Similar

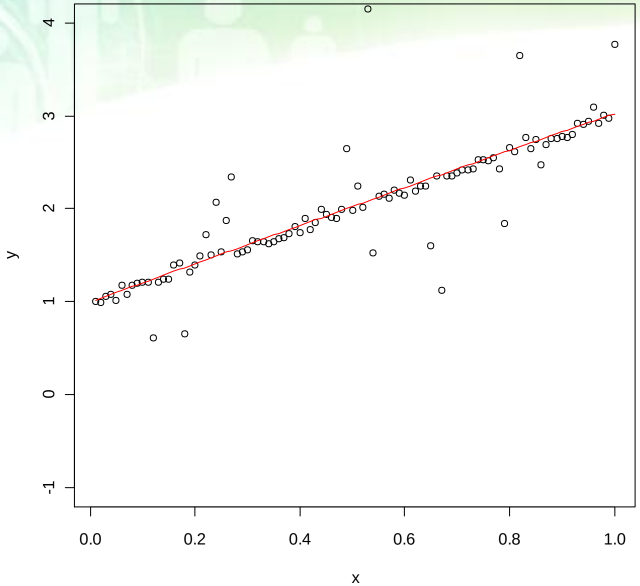
Barcelona vs Almeria
(Barcelona win odds: 1.35))

Less similar

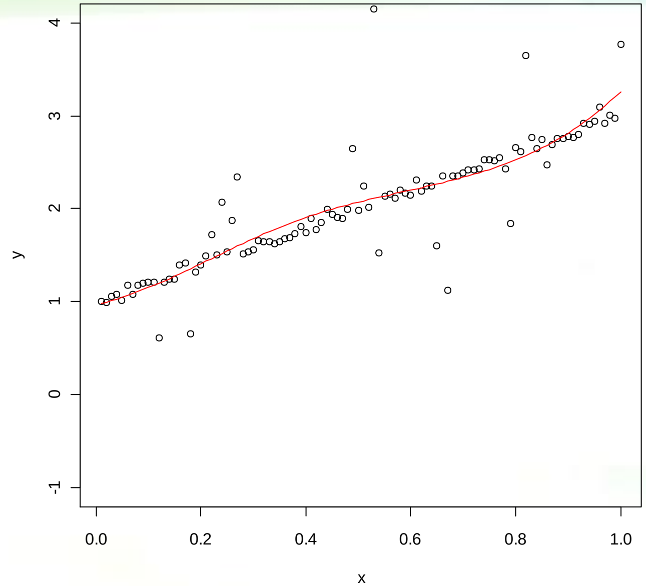
Wolfsburg vs Schalke 04
(Wolfsburg win odds: 2.0))

ANN Overfitting

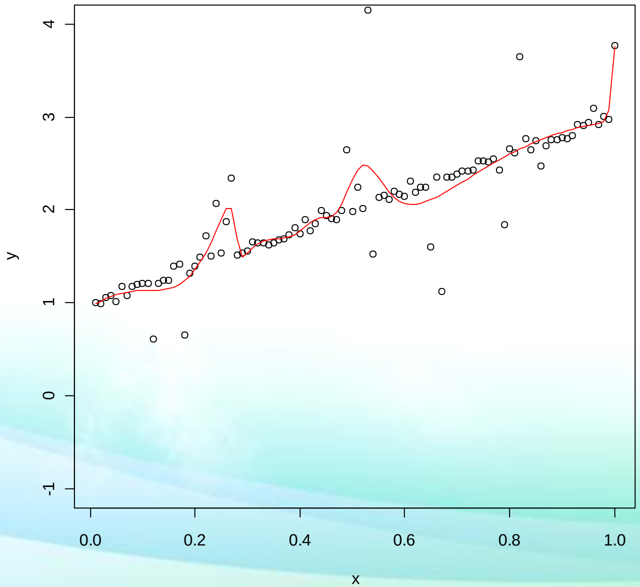
(1)



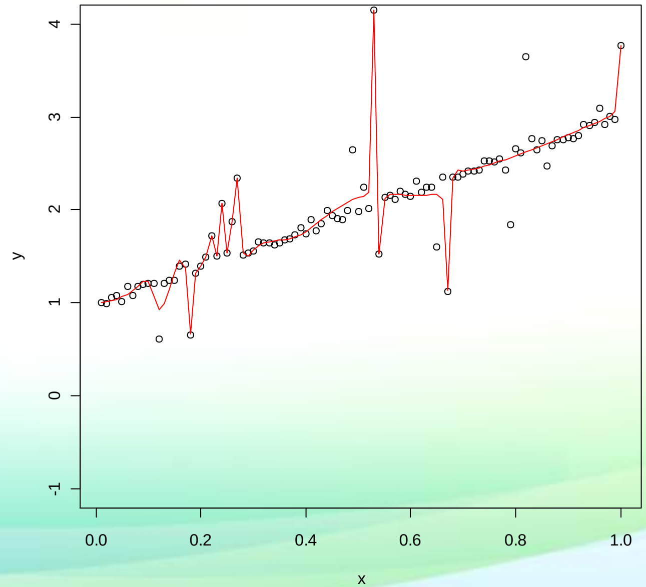
(3)



(15)

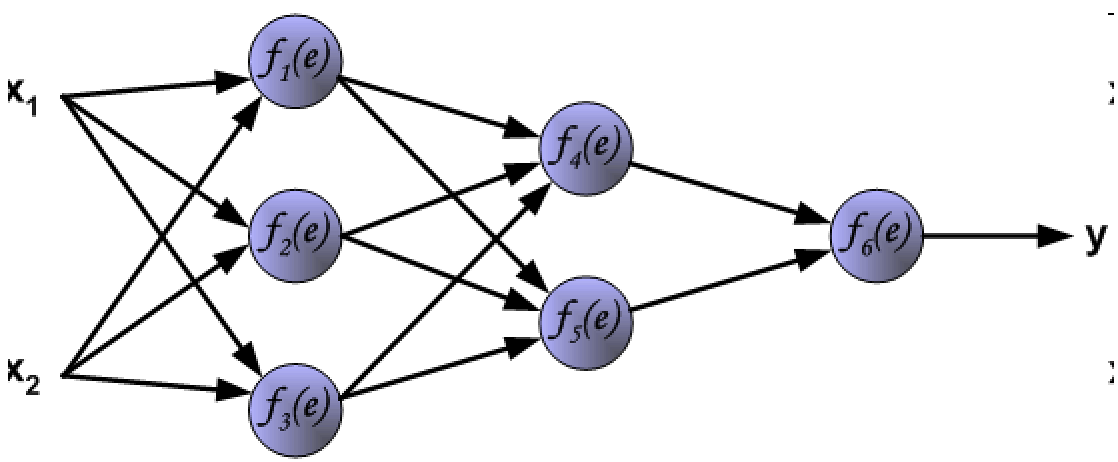


(50)



(# of neurons in hidden layer)

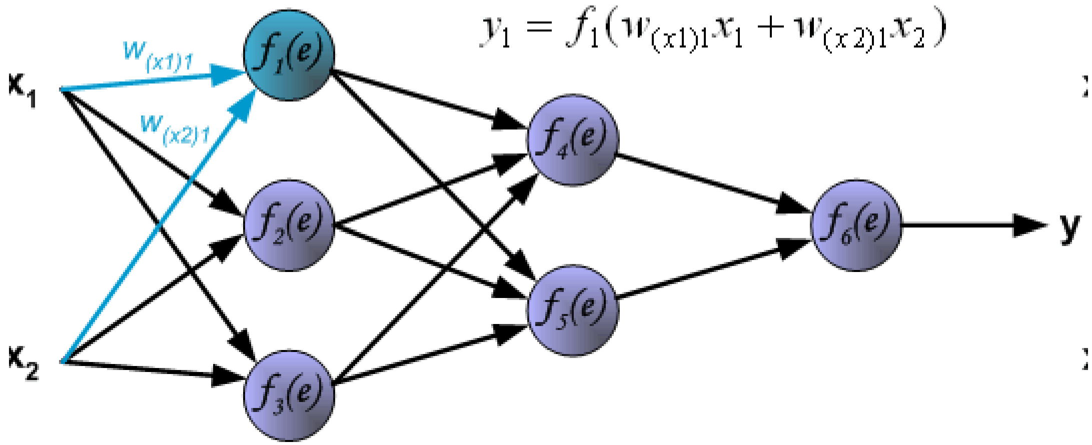
ANN Backpropagation



http://home.agh.edu.pl/~vlsi/AI/backp_t_en/backprop.html

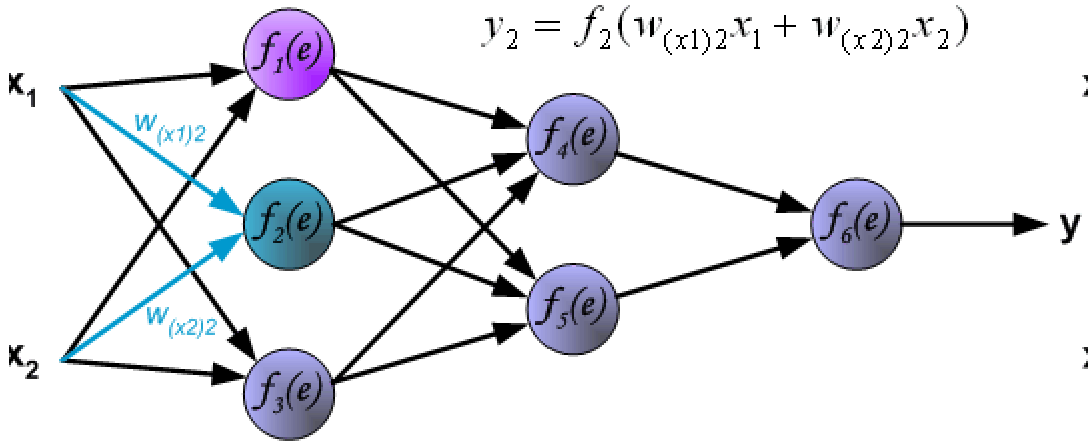
ANN Backpropagation

$$y_1 = f_1(w_{(x1)1} x_1 + w_{(x2)1} x_2)$$

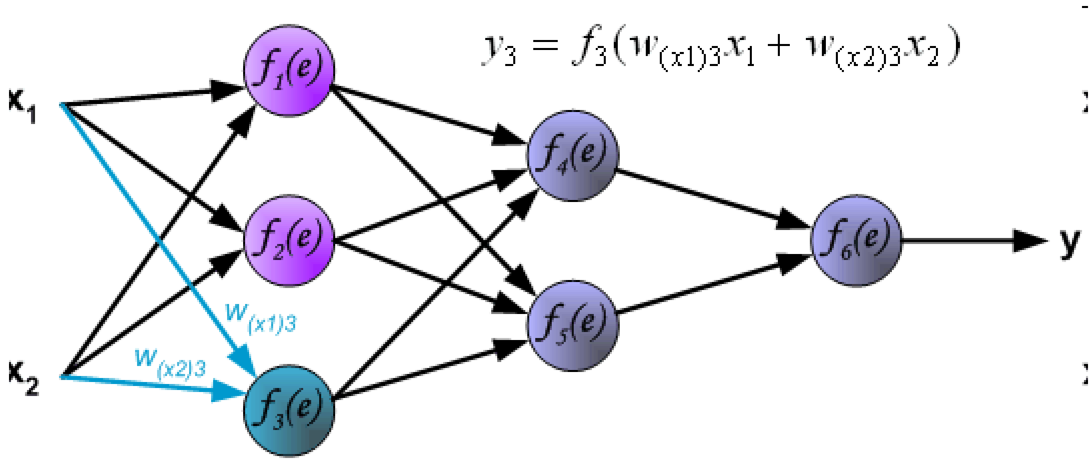


ANN Backpropagation

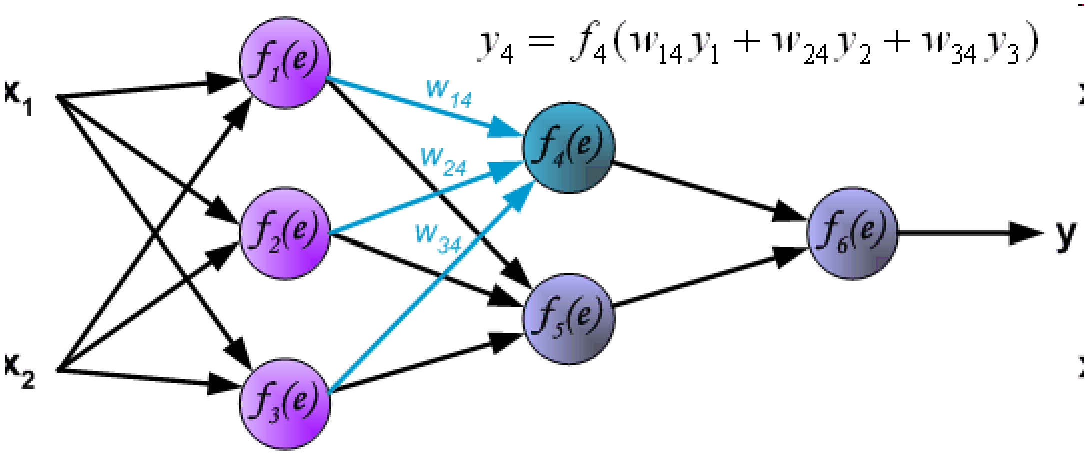
$$y_2 = f_2(w_{(x1)2}x_1 + w_{(x2)2}x_2)$$



ANN Backpropagation

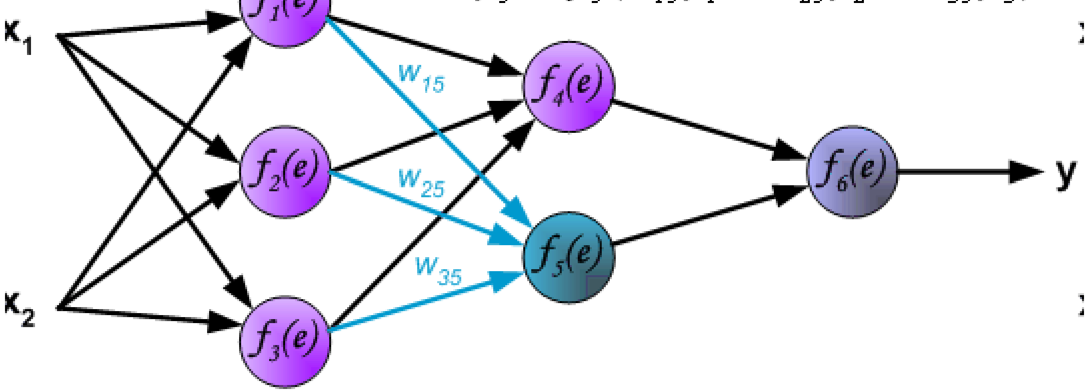


ANN Backpropagation

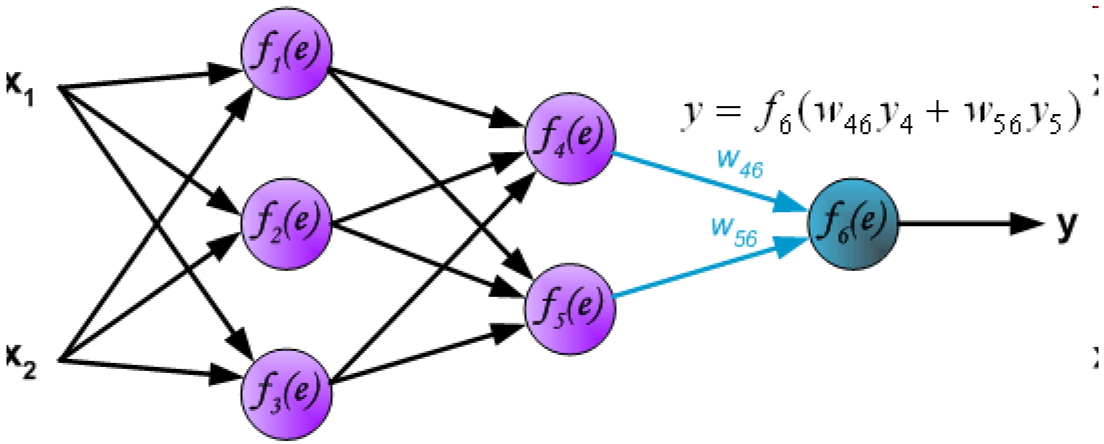


ANN Backpropagation

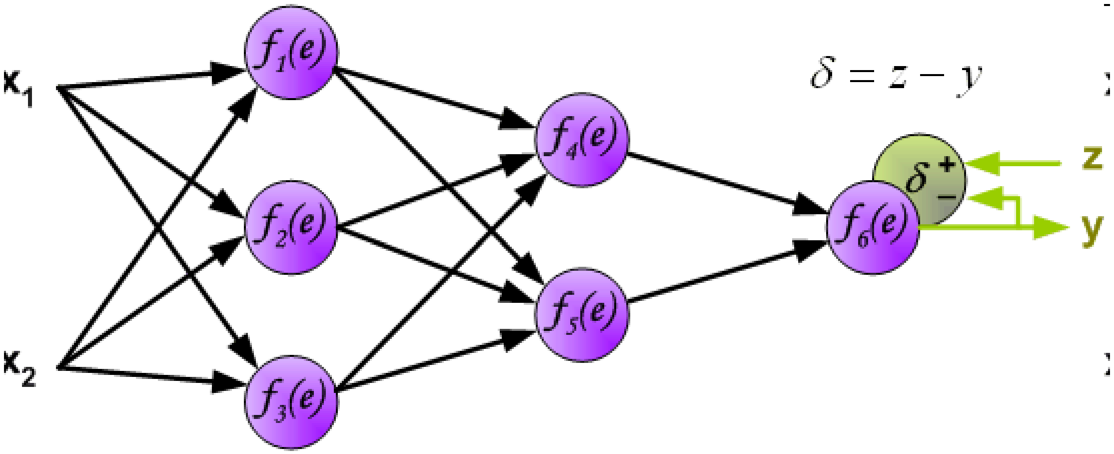
$$y_5 = f_5(w_{15}y_1 + w_{25}y_2 + w_{35}y_3)$$



ANN Backpropagation

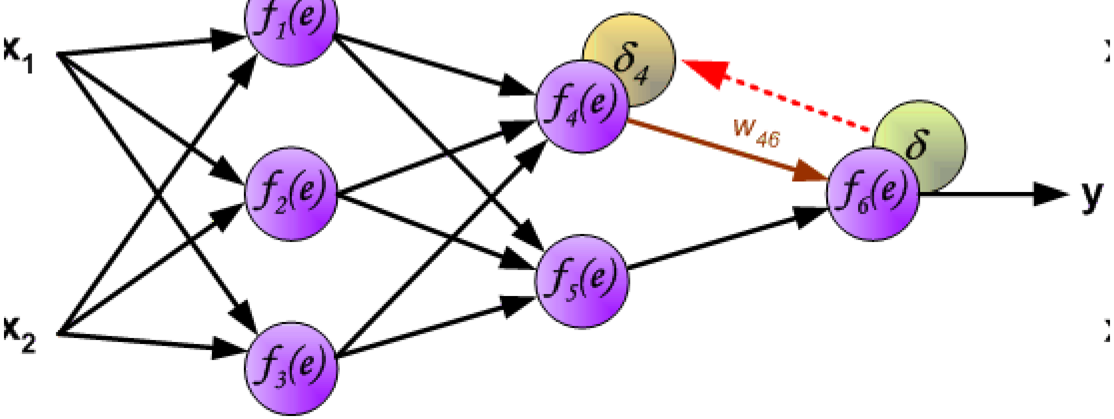


ANN Backpropagation

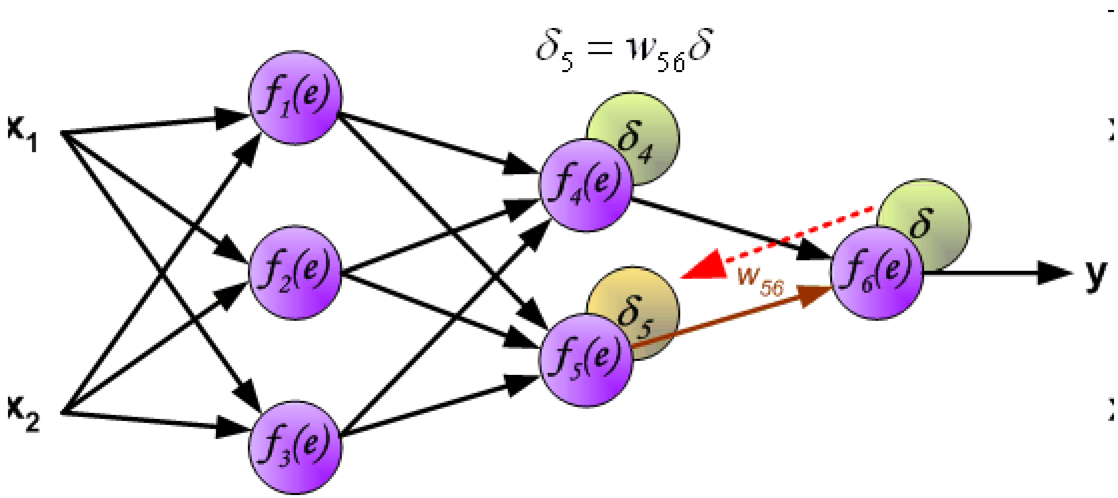


ANN Backpropagation

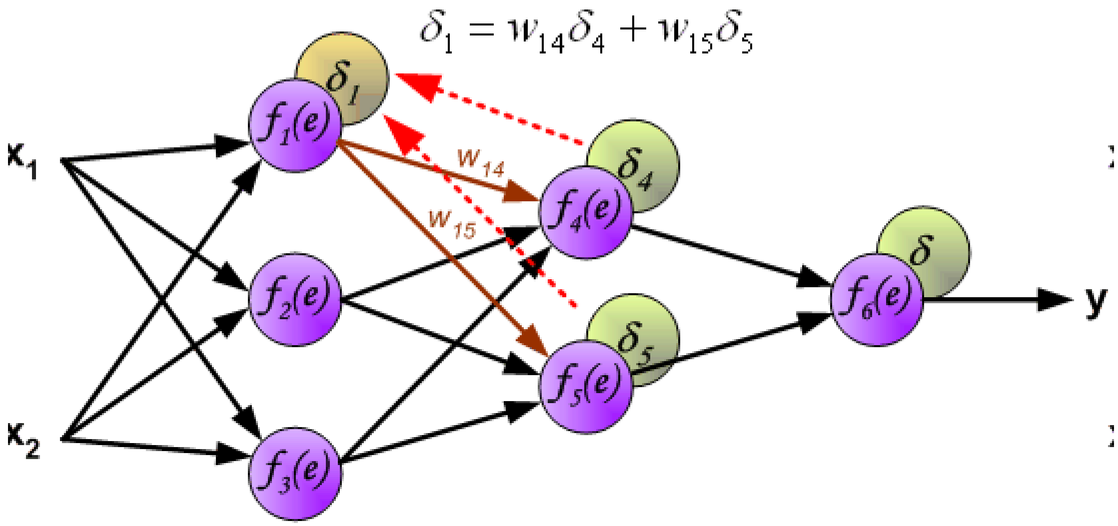
$$\delta_4 = w_{46}\delta$$



ANN Backpropagation

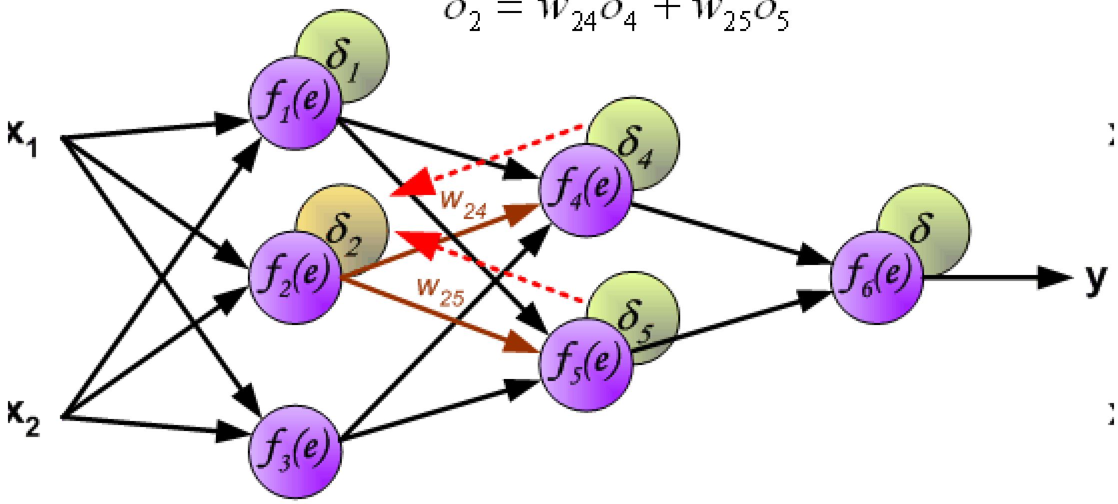


ANN Backpropagation



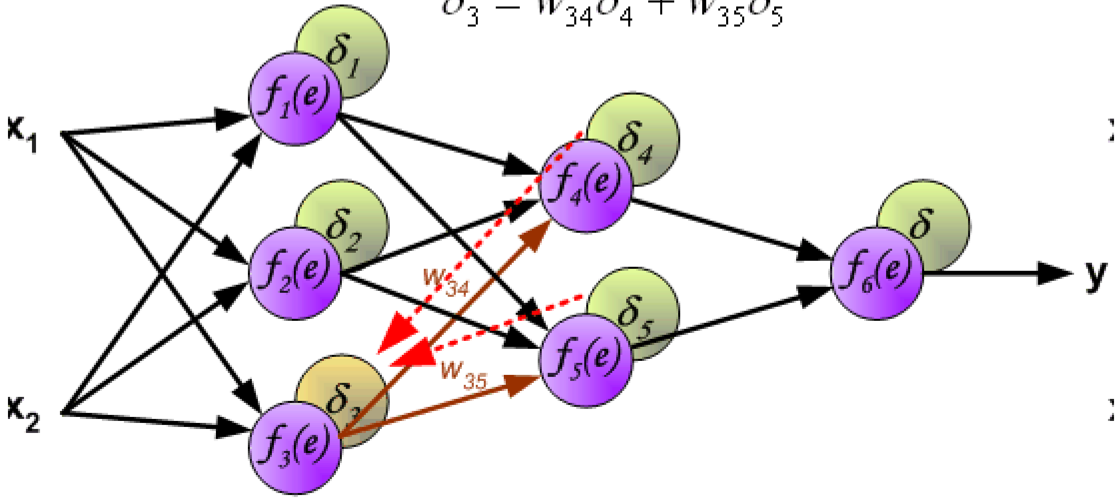
ANN Backpropagation

$$\delta_2 = w_{24}\delta_4 + w_{25}\delta_5$$



ANN Backpropagation

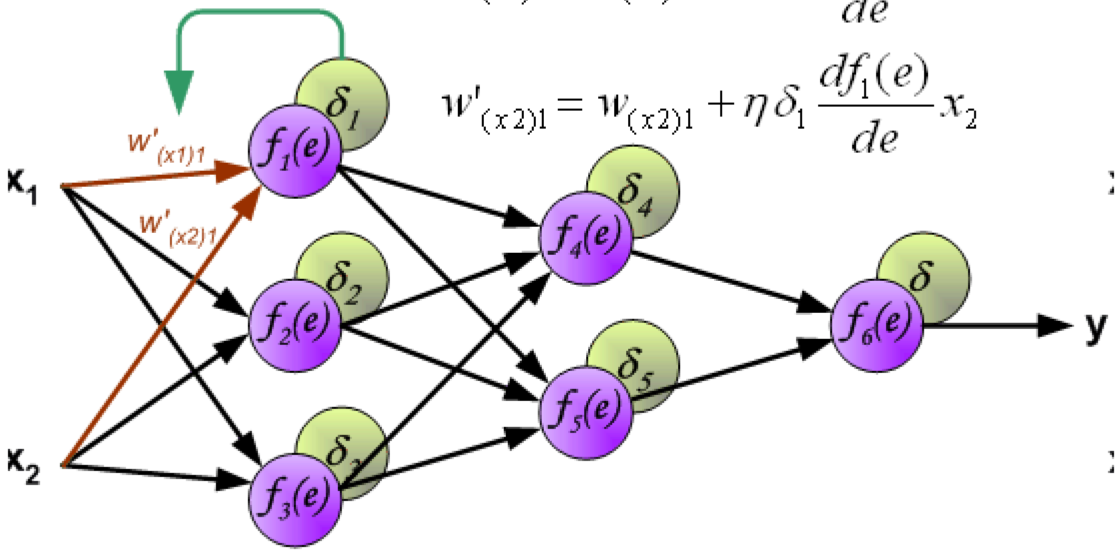
$$\delta_3 = w_{34}\delta_4 + w_{35}\delta_5$$



ANN Backpropagation

$$w'_{(x1)1} = w_{(x1)1} + \eta \delta_1 \frac{df_1(e)}{de} x_1$$

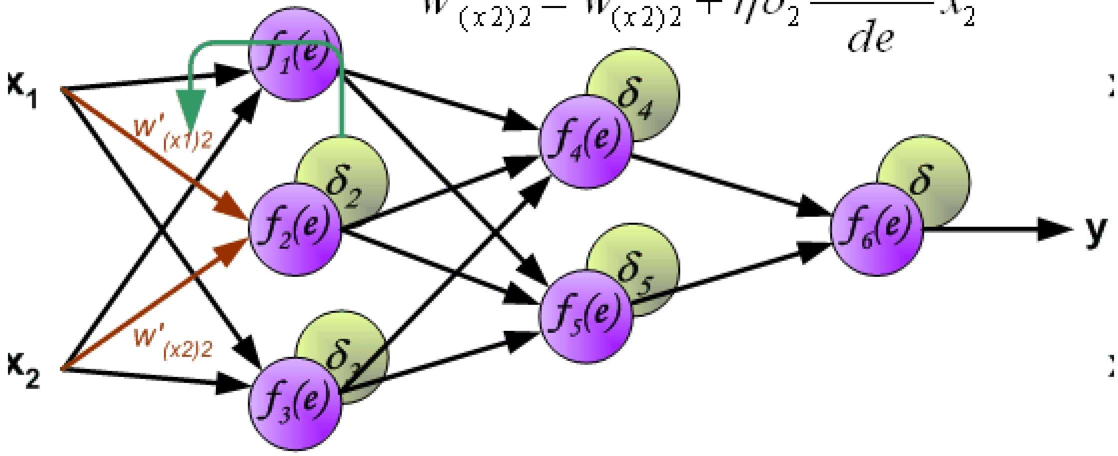
$$w'_{(x2)1} = w_{(x2)1} + \eta \delta_1 \frac{df_1(e)}{de} x_2$$



ANN Backpropagation

$$w'_{(x1)2} = w_{(x1)2} + \eta \delta_2 \frac{df_2(e)}{de} x_1$$

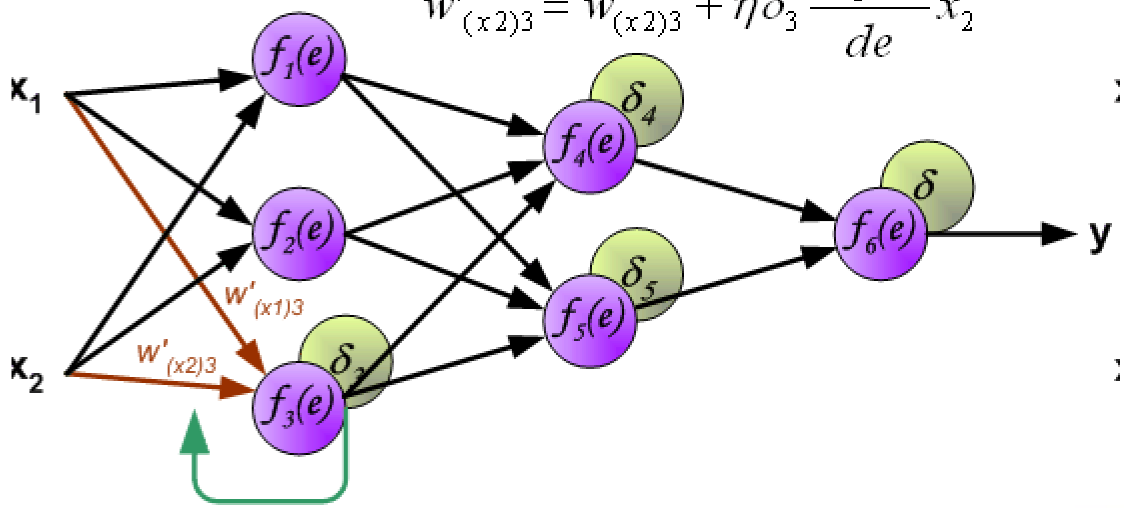
$$w'_{(x2)2} = w_{(x2)2} + \eta \delta_2 \frac{df_2(e)}{de} x_2$$



ANN Backpropagation

$$w'_{(x1)3} = w_{(x1)3} + \eta \delta_3 \frac{df_3(e)}{de} x_1$$

$$w'_{(x2)3} = w_{(x2)3} + \eta \delta_3 \frac{df_3(e)}{de} x_2$$

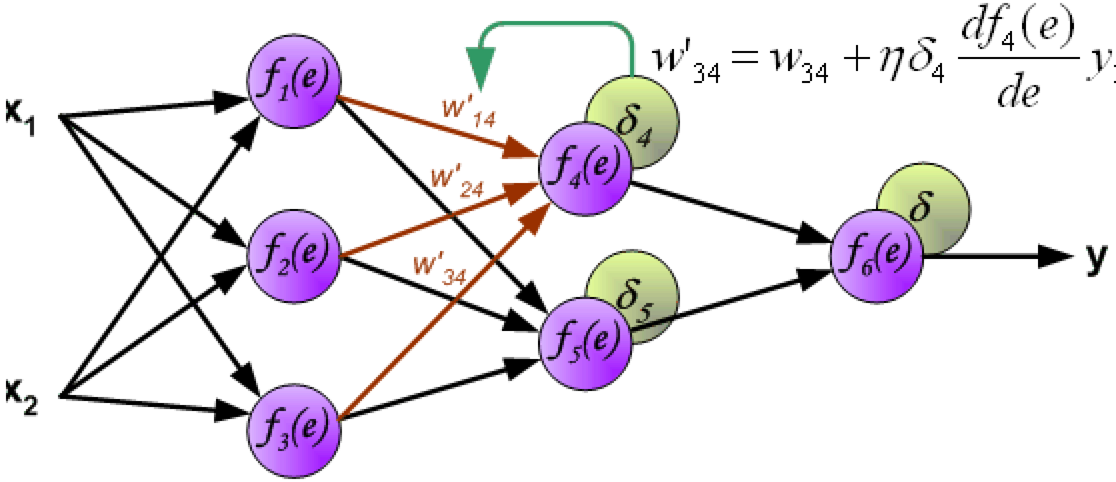


ANN Backpropagation

$$w'_{14} = w_{14} + \eta \delta_4 \frac{df_4(e)}{de} y_1$$

$$w'_{24} = w_{24} + \eta \delta_4 \frac{df_4(e)}{de} y_2$$

$$w'_{34} = w_{34} + \eta \delta_4 \frac{df_4(e)}{de} y_3$$

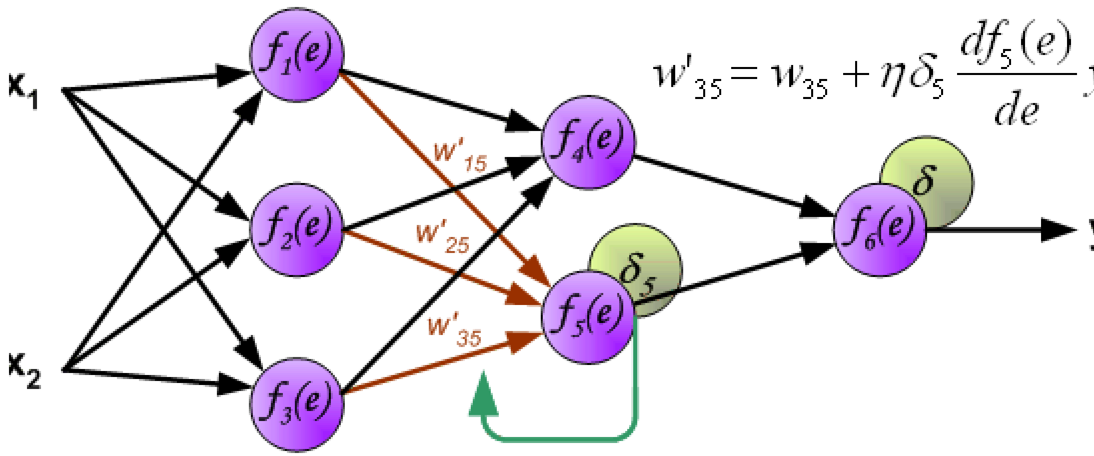


ANN Backpropagation

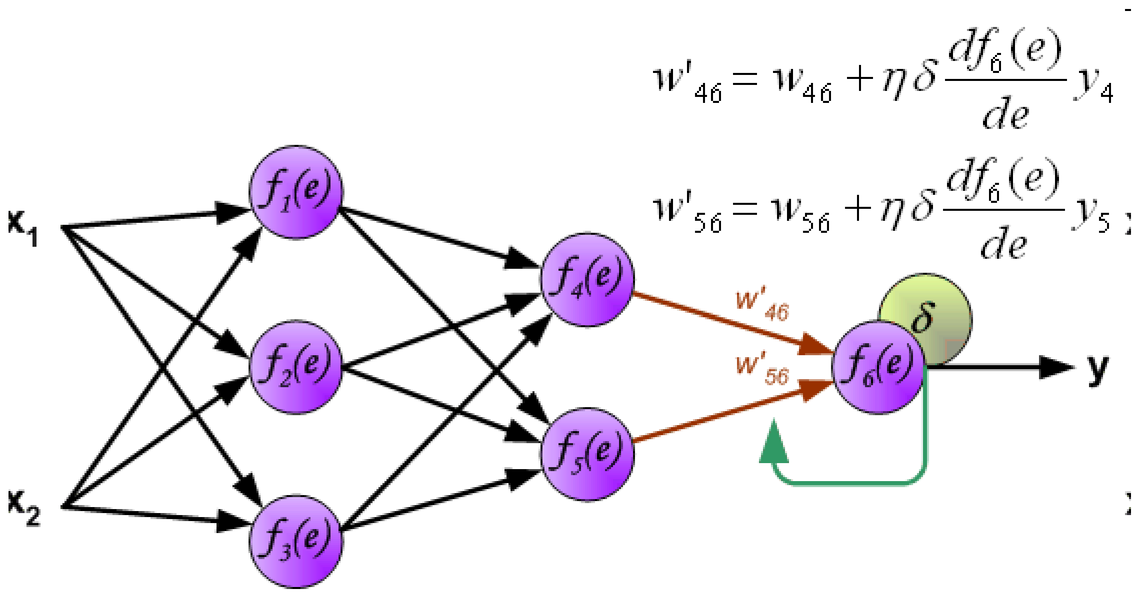
$$w'_{15} = w_{15} + \eta \delta_5 \frac{df_5(e)}{de} y_1$$

$$w'_{25} = w_{25} + \eta \delta_5 \frac{df_5(e)}{de} y_2$$

$$w'_{35} = w_{35} + \eta \delta_5 \frac{df_5(e)}{de} y_3$$



ANN Backpropagation



Comparative Studies (1)

Seongwook Youn and Dennis McLeod : A Comparative Study for Email Classification

Data Size	NN	SVM	Naïve Bayesian	J48
1000	93.50%	92.70%	97.20%	95.80%
2000	97.15%	95.00%	98.15%	98.25%
3000	94.17%	92.40%	97.83%	97.27%
4000	89.60%	91.93%	97.75%	97.63%
4500	93.40%	90.87%	96.47%	97.56%

With 55 features

Fig. 1. Classification result based on data size

Comparative Studies (2)

A Comparison of Prediction Accuracy, Complexity, and Training Time of Thirty-three Old and New Classification Algorithms

Table 2. Hardware and software platform for each algorithm. The workstations are DEC 3000 Alpha Model 300 (DEC), Sun SPARCstation 20 Model 61 (SS20), and Sun SPARCstation 5 (SS5).

Algorithm		Platform	Algorithm		Platform
Tree & Rules			ST1	Splus tree, 1-SE	DEC/S
QU0	QUEST, univariate 0-SE	DEC/F90	LMT	LMDT, linear	DEC/C
QU1	QUEST, univariate 1-SE	DEC/F90	CAL	CAL5	SS5/C++
QL0	QUEST, linear 0-SE	DEC/F90	T1	T1, single split	DEC/C
QL1	QUEST, linear 1-SE	DEC/F90			
FTU	FACT, univariate	DEC/F77	Statistical		
FTL	FACT, linear	DEC/F77	LDA	Linear discriminant anal.	DEC/SAS
C4T	C4.5 trees	DEC/C	QDA	Quadratic discriminant anal.	DEC/SAS
C4R	C4.5 rules	DEC/C	NN	Nearest-neighbor	DEC/SAS
IB	IND bayes style	SS5/C	LOG	Linear logistic regression	DEC/F90
IBO	IND bayes opt style	SS5/C	FM1	FDA, degree 1	SS20/S
IM	IND mml style	SS5/C	FM2	FDA, degree 2	SS20/S
IMO	IND mml opt style	SS5/C	PDA	Penalized LDA	SS20/S
IC0	IND cart, 0-SE	SS5/C	MDA	Mixture discriminant anal.	SS20/S
IC1	IND cart, 1-SE	SS5/C	POL	POLYCLASS	SS20/S
OCU	OC1, univariate	SS5/C			
OCL	OC1, linear	SS5/C	Neural Network		
OCM	OC1, mixed	SS5/C	LVQ	Learning vector quantization	SS20/S
ST0	Splus tree, 0-SE	DEC/S	RBF	Radial basis function network	DEC/SAS

Comparative Studies (2)

Table 4. Minimum, maximum, and 'naive' plurality rule error rates for each dataset. A '√'-mark indicates that the algorithm has an error rate within one standard error of the minimum for the dataset. A 'X'-mark indicates that the algorithm has the worst error rate for the dataset. The mean error rate for each algorithm is given in the second row.

	Decision trees and rules																						Statistical algorithms									Nets		Error rates		
	QUO	QU1	QLO	QL1	FTU	FTL	C4T	C4R	IB	IBO	IM	IMO	ICO	IC1	OCU	OCL	OCM	STO	ST1	LMT	CAL	T1	LDA	QDA	NN	LOG	FM1	FM2	PDA	MDA	POL	LVQ	RBF	Min	Max	Naive
Mean	.221	.226	.207	.211	.238	.234	.220	.220	.229	.219	.220	.219	.215	.227	.227	.260	.230	.232	.233	.220	.270	.354	.208	.301	.281	.204	.242	.217	.213	.207	.195	.269	.257			
#√	8	8	10	9	4	12	7	8	3	8	5	6	4	5	9	4	4	5	5	1	4	4	10	4	4	13	12	12	10	9	15	4	8			
#X	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	3	1	0	1	0	0	11	0	3	4	0	2	1	0	1	0	4	0			
bcw			√	√						√						X										√						√	√	0.028	0.085	.350
bcw+			√	√																		X				√	√	√		√		√	√	0.029	0.076	
cmc	√	√																							X						√			0.434	0.601	.573
cmc+	√	√																							X						√			0.432	0.577	
dna						√																X					√	√			√			0.047	0.379	.492
dna+						√																X						√			√			0.044	0.379	
hea			√	√		√																	√			√			√			X		0.141	0.341	.444
hea+				√		√																	√						√			X		0.148	0.311	
bos							√	√				√					√								√		√				√	X	√	0.221	0.314	.657
bos+							√								√												√	X			√			0.225	0.422	
led			√			√	√	√		√	√	√			√	√						X	√	√		√	√	√	√	√	√			0.268	0.816	.890
led+						√									√							X	√			√	√	√	√	√	√			0.265	0.813	
bld								√								√	√					X					√	√			√			0.279	0.432	.419
bld+																						X									√			0.286	0.441	
pid	√	√	√	√		√		√			√		√		√	X		√			√		√	√		√	√		√	√	√		√	0.221	0.310	.333
pid+	√	√	√	√		√		√					√			X							√			√	√	√	√	√	√			0.217	0.318	
sat																						X										√		0.098	0.400	.765
sat+																												√				X		0.116	0.410	
seg										√															√		X							0.022	0.515	.857
seg+										√																	X							0.026	0.574	
smo	√	√	√	√	√	√	√				√	√		√	√	√	√	√	√		√	√	√	X		√	√	√	√	√	√		√	0.304	0.454	.305
smo+	√	√	√	√	√	√					√		√	√	√	√	√	√	√	√	√	√	√		X	√		√	√	√	√		√	0.305	0.445	
thy							√	√	√				√	√	√			√	√					X										0.006	0.890	.073
thy+							√	√	√		√	√												X										0.005	0.875	
veh																						X		√										0.145	0.487	.739
veh+																						X		√										0.155	0.487	
vot	√	√	√		√					√		√		√	√				√			√						√		X				0.036	0.062	.386
vot+	√	√	√	√	√	√	√	√		√		√		√	√		X	√	√		√	√	√			√	√	√	√		√		√	0.041	0.066	
wav																						X				√							√	0.151	0.477	.667
wav+						√																			X	√	√			√		√		0.16	0.446	
tae						X				√															√									0.325	0.693	.656
tae+									√	√									X				√		√	√			√	√				0.445	0.696	

Comparative Studies (2)

Table 5. Ordering of algorithms by mean error rate and mean rank of error rate

Mean error rate	POL	LOG	MDA	QL0	LDA	QL1	PDA	IC0	FM2	IB0	IM0
	.195	.204	.207	.207	.208	.211	.213	.215	.218	.219	.219
	C4R	IM	LMT	C4T	QU0	QU1	OCU	IC1	IB	OCM	ST0
	.220	.220	.220	.220	.221	.226	.227	.227	.229	.230	.232
	ST1	FTL	FTU	FM1	RBF	OCL	LVQ	CAL	NN	QDA	T1
	.233	.234	.238	.242	.257	.260	.269	.270	.281	.301	.354
Mean rank of error rate	POL	FM1	LOG	FM2	QL0	LDA	QU0	C4R	IM0	MDA	PDA
	8.3	12.2	12.2	12.2	12.4	13.7	13.9	14.0	14.0	14.3	14.5
	C4T	QL1	IB0	IM	IC0	FTL	QU1	OCU	IC1	ST0	ST1
	14.5	14.6	14.7	14.9	15.0	15.4	16.6	16.6	16.9	17.0	17.7
	LMT	OCM	IB	RBF	FTU	QDA	LVQ	OCL	CAL	NN	T1
	18.5	18.9	19.0	19.1	20.7	22.8	24.0	24.3	25.1	25.5	27.5

Table 7. Ordering of algorithms by median training time

C4T	FTU	FTL	LDA	QDA	C4R	NN	IB	IM	T1	OCU
5s	7s	8s	10s	15s	20s	20s	34s	34s	36s	46s
IC1	IC0	PDA	LVQ	MDA	QU1	QU0	LOG	LMT	QL1	QL0
47s	52s	56s	1.1m	3m	3.2m	3.2m	4m	5.7m	5.9m	5.9m
OCM	ST1	OCL	ST0	FM1	IB0	IM0	CAL	POL	FM2	RBF
13.7m	14.4m	14.9m	15.1m	15.6m	27.5m	33.9m	1.3h	3.2h	3.8h	11.3h

Comparative Studies (3)

An Empirical Comparison of Supervised Learning Algorithms

Table 2. Normalized scores for each learning algorithm by metric (average over eleven problems)

MODEL	CAL	ACC	FSC	LFT	ROC	APR	BEP	RMS	MXE	MEAN	OPT-SEL
BST-DT	PLT	.843*	.779	**.939**	**.963**	**.938**	.929*	**.880**	**.896**	**.896**	**.917**
RF	PLT	.872*	.805	.934*	.957	.931	**.930**	.851	.858	.892	.898
BAG-DT	–	.846	.781	.938*	.962*	.937*	.918	.845	.872	.887*	.899
BST-DT	ISO	.826*	.860*	.929*	.952	.921	.925*	.854	.815	.885	.917*
RF	–	**.872**	.790	.934*	.957	.931	**.930**	.829	.830	.884	.890
BAG-DT	PLT	.841	.774	.938*	.962*	.937*	.918	.836	.852	.882	.895
RF	ISO	.861*	**.861**	.923	.946	.910	.925	.836	.776	.880	.895
BAG-DT	ISO	.826	.843*	.933*	.954	.921	.915	.832	.791	.877	.894
SVM	PLT	.824	.760	.895	.938	.898	.913	.831	.836	.862	.880
ANN	–	.803	.762	.910	.936	.892	.899	.811	.821	.854	.885
SVM	ISO	.813	.836*	.892	.925	.882	.911	.814	.744	.852	.882
ANN	PLT	.815	.748	.910	.936	.892	.899	.783	.785	.846	.875
ANN	ISO	.803	.836	.908	.924	.876	.891	.777	.718	.842	.884
BST-DT	–	.834*	.816	**.939**	**.963**	**.938**	.929*	.598	.605	.828	.851
KNN	PLT	.757	.707	.889	.918	.872	.872	.742	.764	.815	.837
KNN	–	.756	.728	.889	.918	.872	.872	.729	.718	.810	.830
KNN	ISO	.755	.758	.882	.907	.854	.869	.738	.706	.809	.844
BST-STMP	PLT	.724	.651	.876	.908	.853	.845	.716	.754	.791	.808
SVM	–	.817	.804	.895	.938	.899	.913	.514	.467	.781	.810
BST-STMP	ISO	.709	.744	.873	.899	.835	.840	.695	.646	.780	.810
BST-STMP	–	.741	.684	.876	.908	.853	.845	.394	.382	.710	.726
DT	ISO	.648	.654	.818	.838	.756	.778	.590	.589	.709	.774
DT	–	.647	.639	.824	.843	.762	.777	.562	.607	.708	.763
DT	PLT	.651	.618	.824	.843	.762	.777	.575	.594	.706	.761
LR	–	.636	.545	.823	.852	.743	.734	.620	.645	.700	.710
LR	ISO	.627	.567	.818	.847	.735	.742	.608	.589	.692	.703
LR	PLT	.630	.500	.823	.852	.743	.734	.593	.604	.685	.695
NB	ISO	.579	.468	.779	.820	.727	.733	.572	.555	.654	.661
NB	PLT	.576	.448	.780	.824	.738	.735	.537	.559	.650	.654
NB	–	.496	.562	.781	.825	.738	.735	.347	-.633	.481	.489