# Data Mining With Weka

Hands-on Tutorial on the open-source java-based Data Mining Toolbox Weka

Alban@ihr.mrc.ac.uk

# This Tutorial:

- Practical experience with Weka
  - Download software
  - Load datasets
  - Run algorithms
  - Understand underlying models
- Explanation of basic principles of Machine Learning
  - Supervised/unsupervised learning
  - A bit of theory
  - Evaluation metrics
- Interfacing Matlab with Weka
  - Use Java objects and methods from Matlab

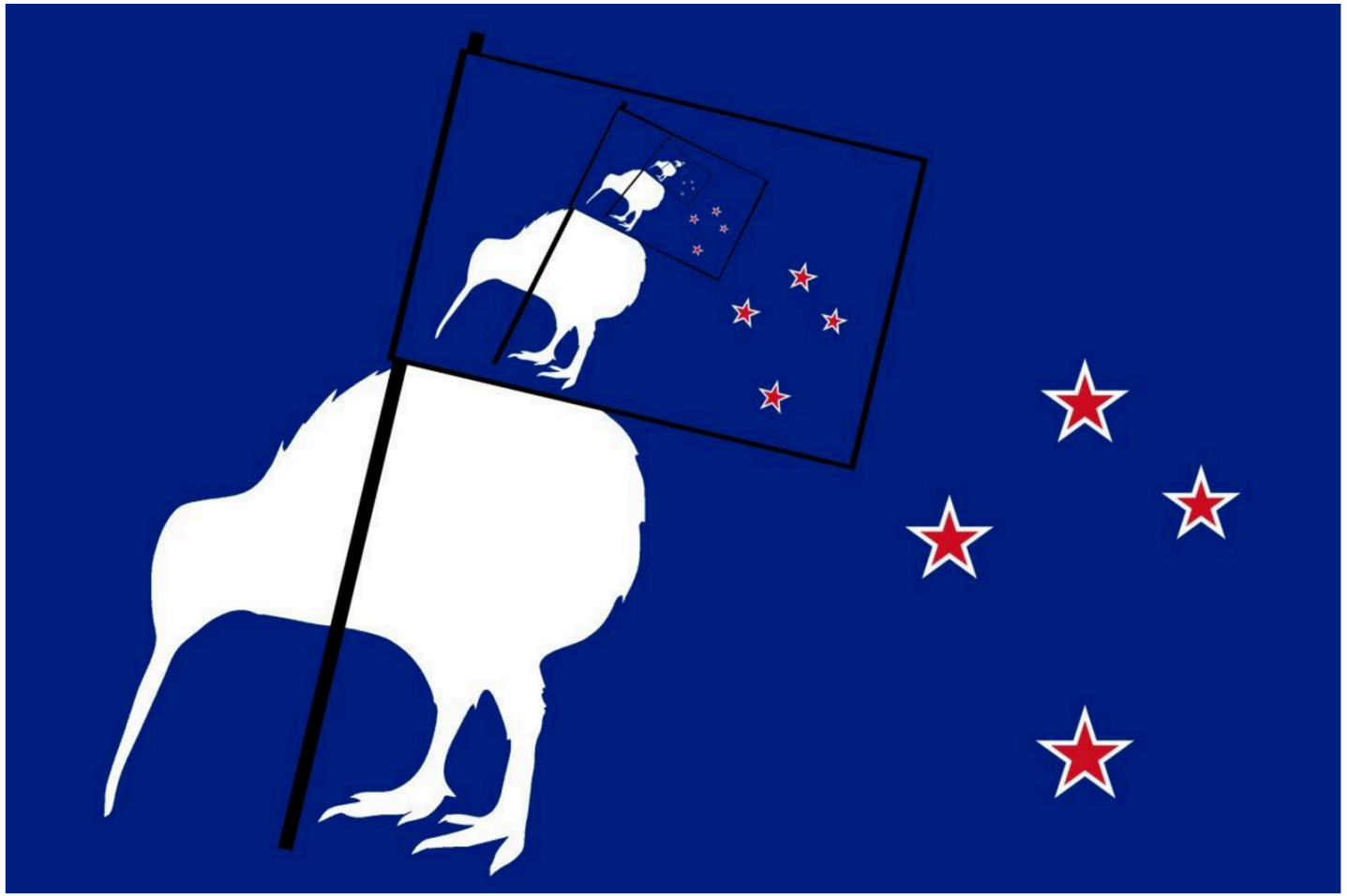
NO MATHS TODAY

# New Zealand

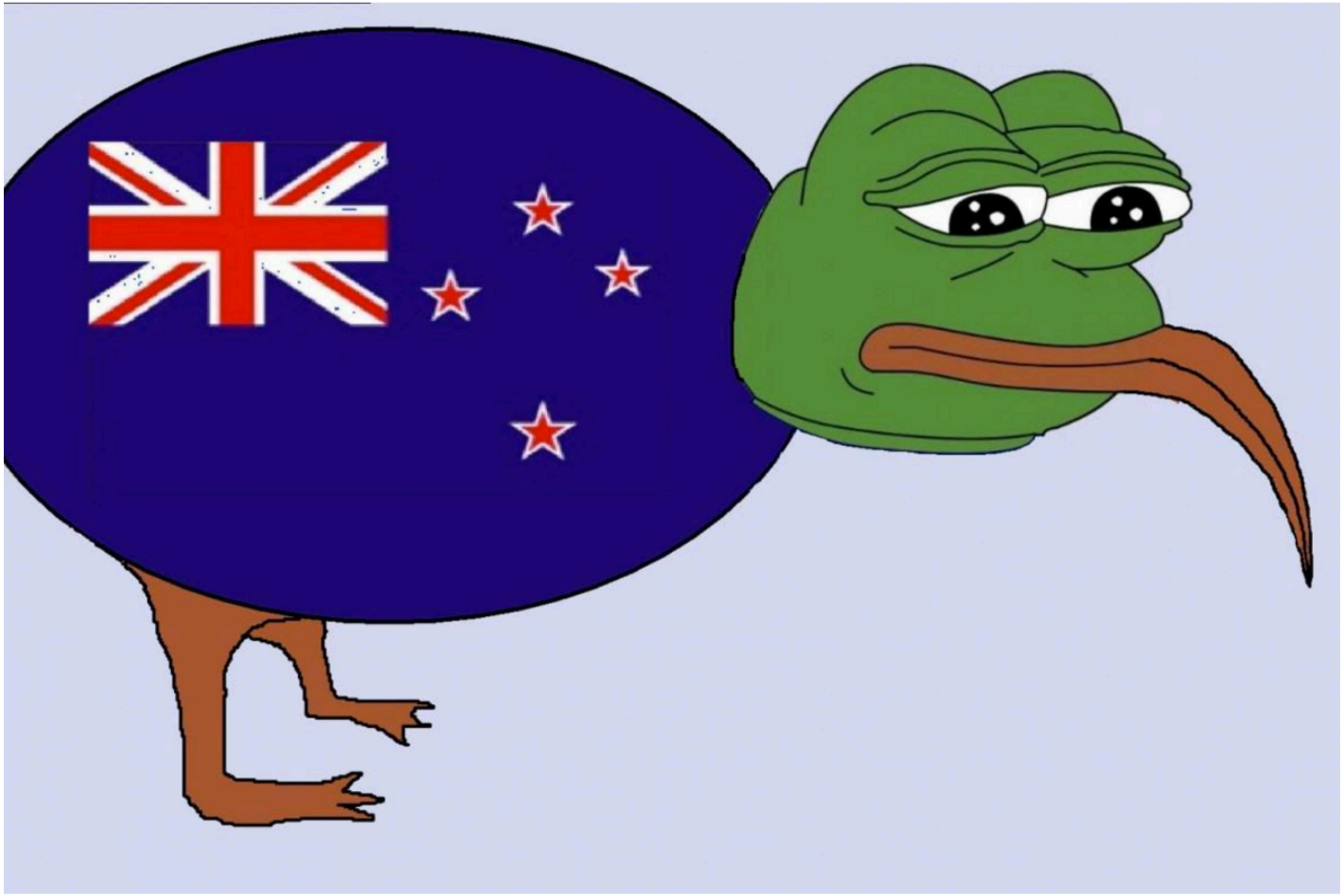
The current New Zealand flag (L) the referendum winning blue and black Kyle Lockwood designed flag (C) and the second placed red and blue flag (R)

- http://www.independent.co.uk/news/world/politics/new-zealands-potential-new-flag-design-says-goodbye-to-the-union-jack-and-welcomes-the-silver-fern-a6773576.html#

# Other propositions were



# Other propositions were



# Motivation

- Data is everywhere – and that's only the beginning
- Necessity to be able to handle data, learn from it, predict it
- Many tools out there, find the ones that suit your needs

# Data Mining

- Data Mining is the analysis of data and the use of software techniques for finding patterns and regularities in datasets

- Interdisciplinary field involving
  - Databases
  - Statistics
  - Machine Learning
  - High-Performance Computing
  - Visualization
  - Mathematics

- In a nutshell: do whatever it takes to extract knowledge from data.

# What is Weka?


- **W**aikato **E**nvironment for **K**nowledge **A**nalysis

- Data mining toolkit: popular suite of machine learning software and visualization tools

-> Use for data analysis and predictive modeling (regression, classification, attribute selection, clustering)

- Continuous efforts since 1994

# Why use Weka?

- Free availability (GNU license)
- Doesn't require Data Mining expertise
- Portability (java is nº1 must-know language for programmers)
- Extensibility
- Comprehensive collection of data preprocessing and modeling techniques
- Nice user-interface for exploration
- Active community
- One of the most widely used data mining systems
- Pedagogical value - also with their book 'Data Mining: Practical Machine Learning Tools and Techniques'

# Hands-on with Weka

- Download software (3min)
- Load datasets (arff format, 3min, next slide)
- Explore Dataset (Preprocess & Visualize panels, 5min)
- Run algorithms (Classify & Cluster panels, 2min)
- Understand underlying models (after ML explanations)

# ARFF File Format

- Text file describing a set of instances sharing a set of attributes

http://www.cs.waikato.ac.nz/~ml/weka/arff.html

% The first lines with '%' are usually comments describing the dataset
@**RELATION** <dataset-name>
@**ATTRIBUTE** <attribute1-name> <datatype>
@**ATTRIBUTE** <attribute2-name> <datatype>
@**ATTRIBUTE** class {Iris-setosa, Iris-versicolor, Iris-virginica}
@**DATA**
5.1, 3.5, Iris-setosa
4.9, ? , Iris-versicolor
…

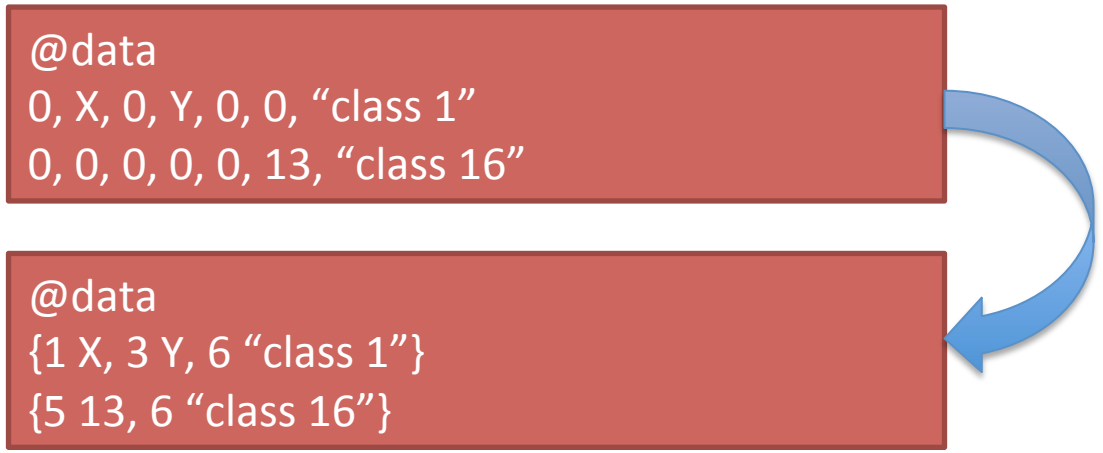
- The datatype can be of any of (version 3.2.1):

numeric, <nominal-specification-in-curly-brackets>, string, date [<date-format>]

- Missing values are represented by `?'

# Sparse ARFF Files

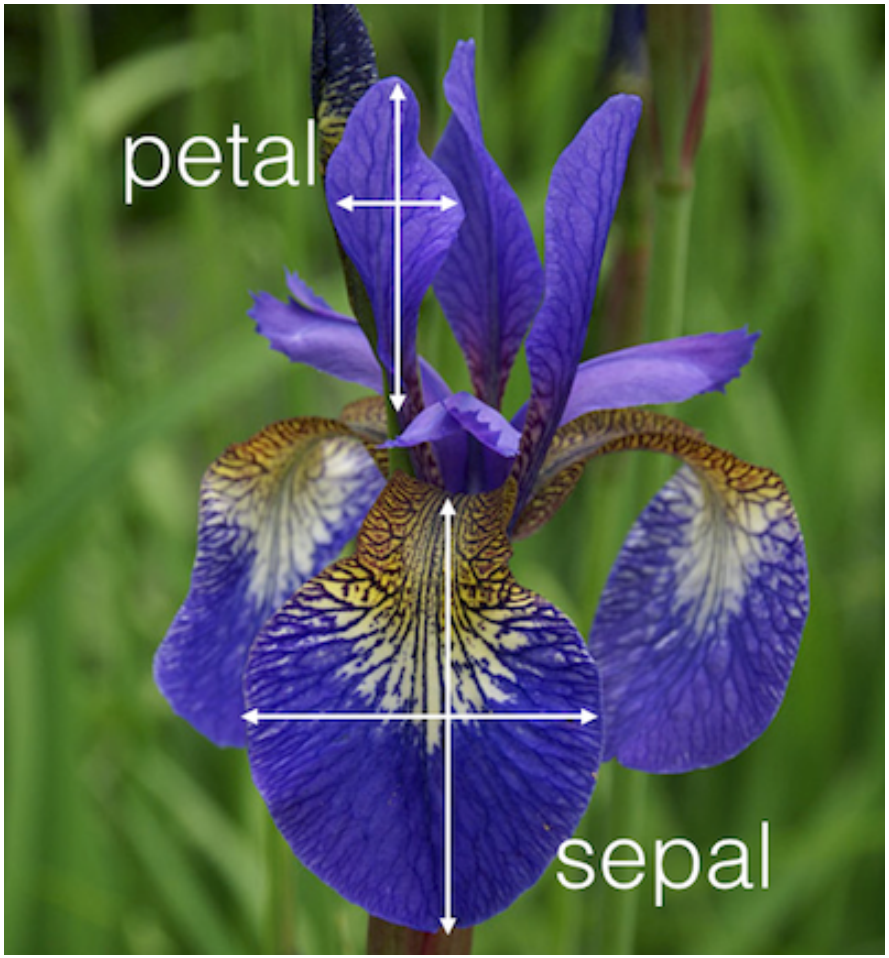
- Similar to ARFF files, except that 0s not represented

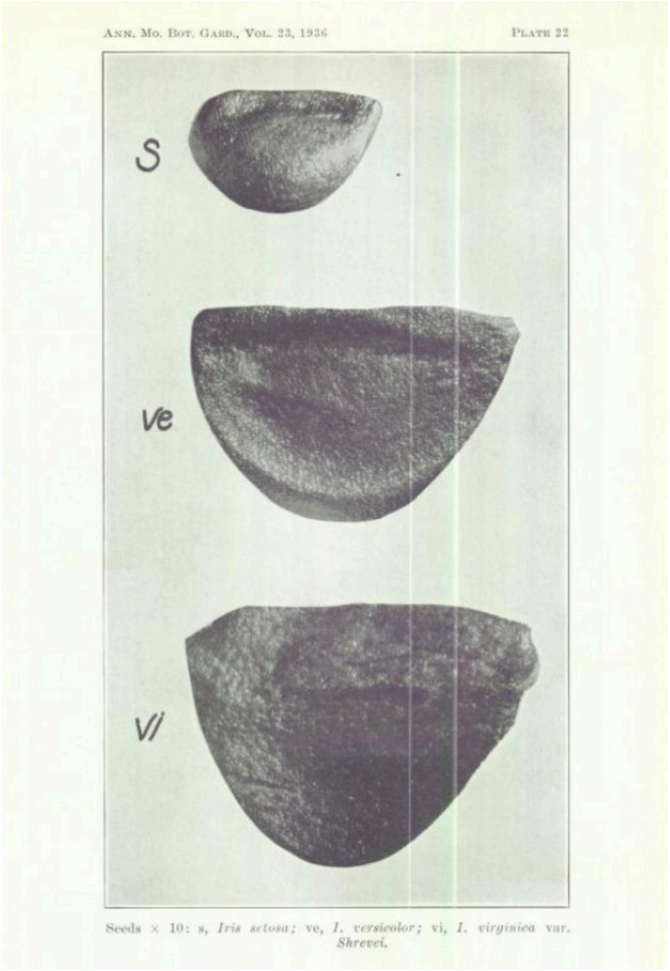



Java’s starting index: 0

# Iris dataset

150 instances, 3 classes (type of iris), 4 features (sepal/petal length/width)
Collected in 1936 by Edgar Anderson ‘The species problem in Iris’ (right: seeds)



# Iris dataset

Virginica

Versicolor

Setosa

# Machine Learning

- Herbert Simon: “Learning is any process by which a system improves performance from experience.”
- Tasks:
  - Classification (labeling)
  - Regression (guess real value)
  - Clustering (regrouping similar objects)
  - Ranking (order a collection, recommendation)
  - Problem solving (games, calculus, driving)
  - Attribute selection
  - Association rules (?)

# Machine Learning

- Examples of everyday ML

Google's ranking algorithm

Spam filters

Face detection

Self-driving cars

Credit card fraud detection

Speech recognition

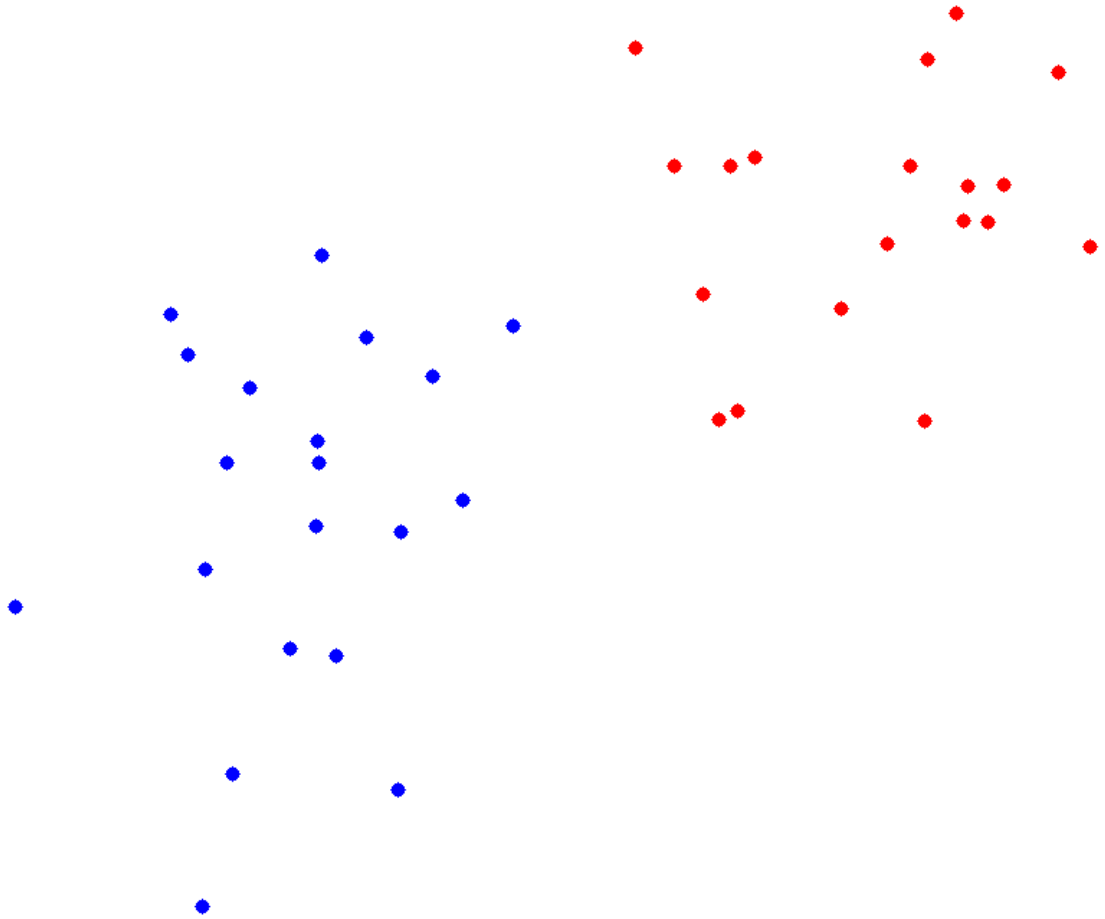
Digit recognition

Medical diagnosis

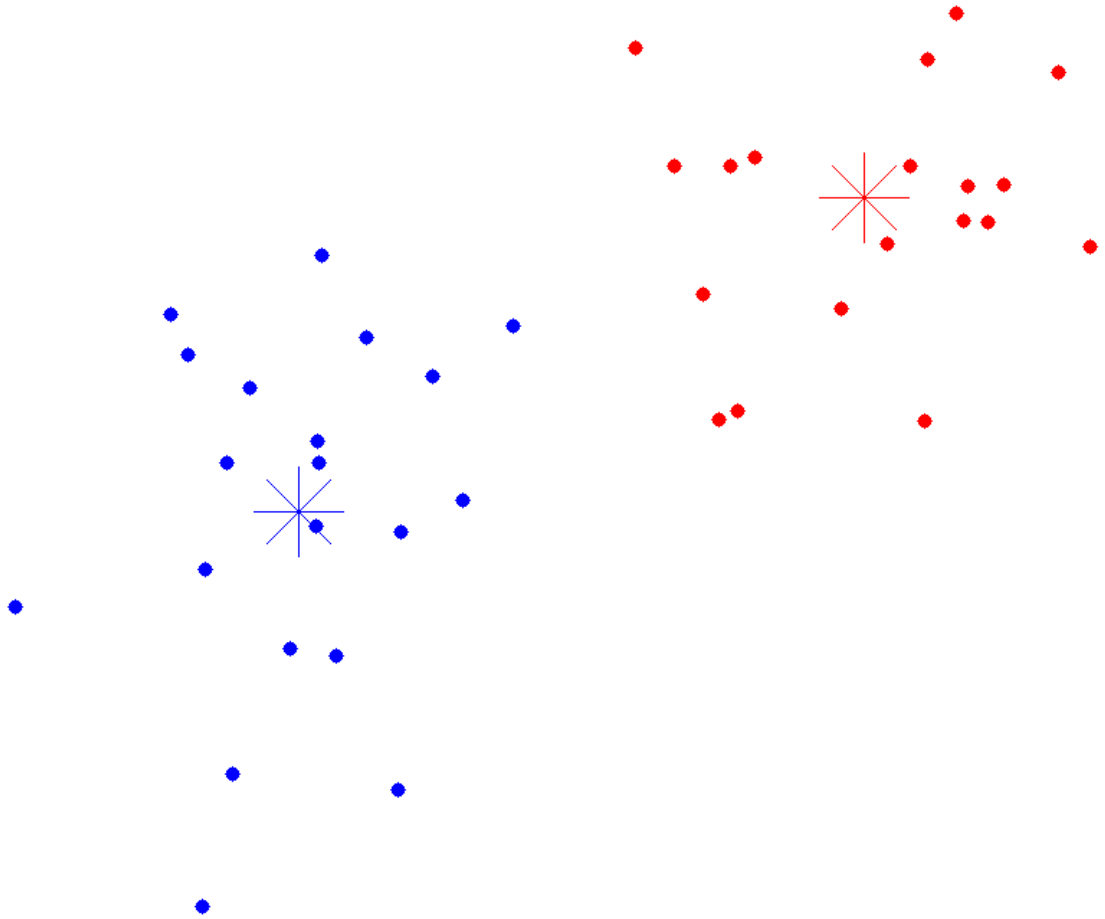
Customer segmentation

…

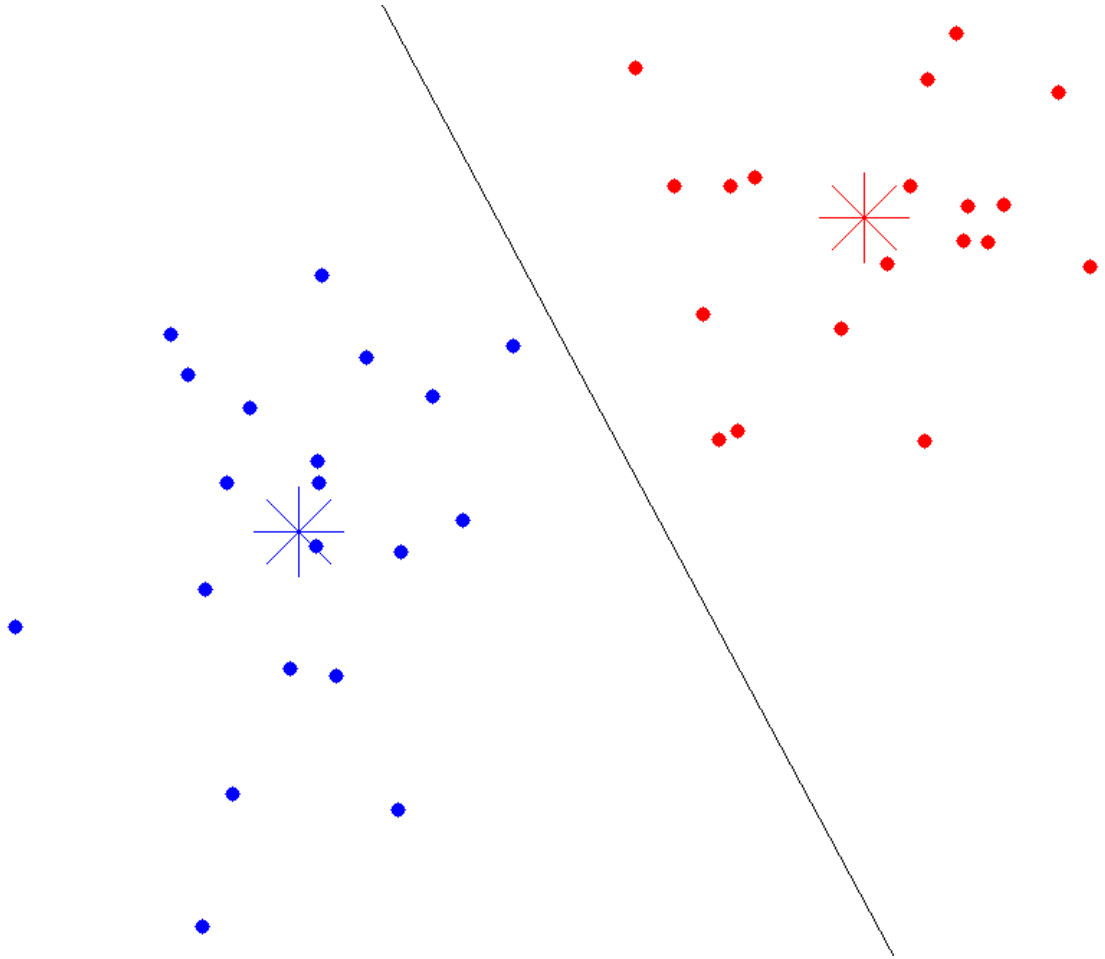
# Classification



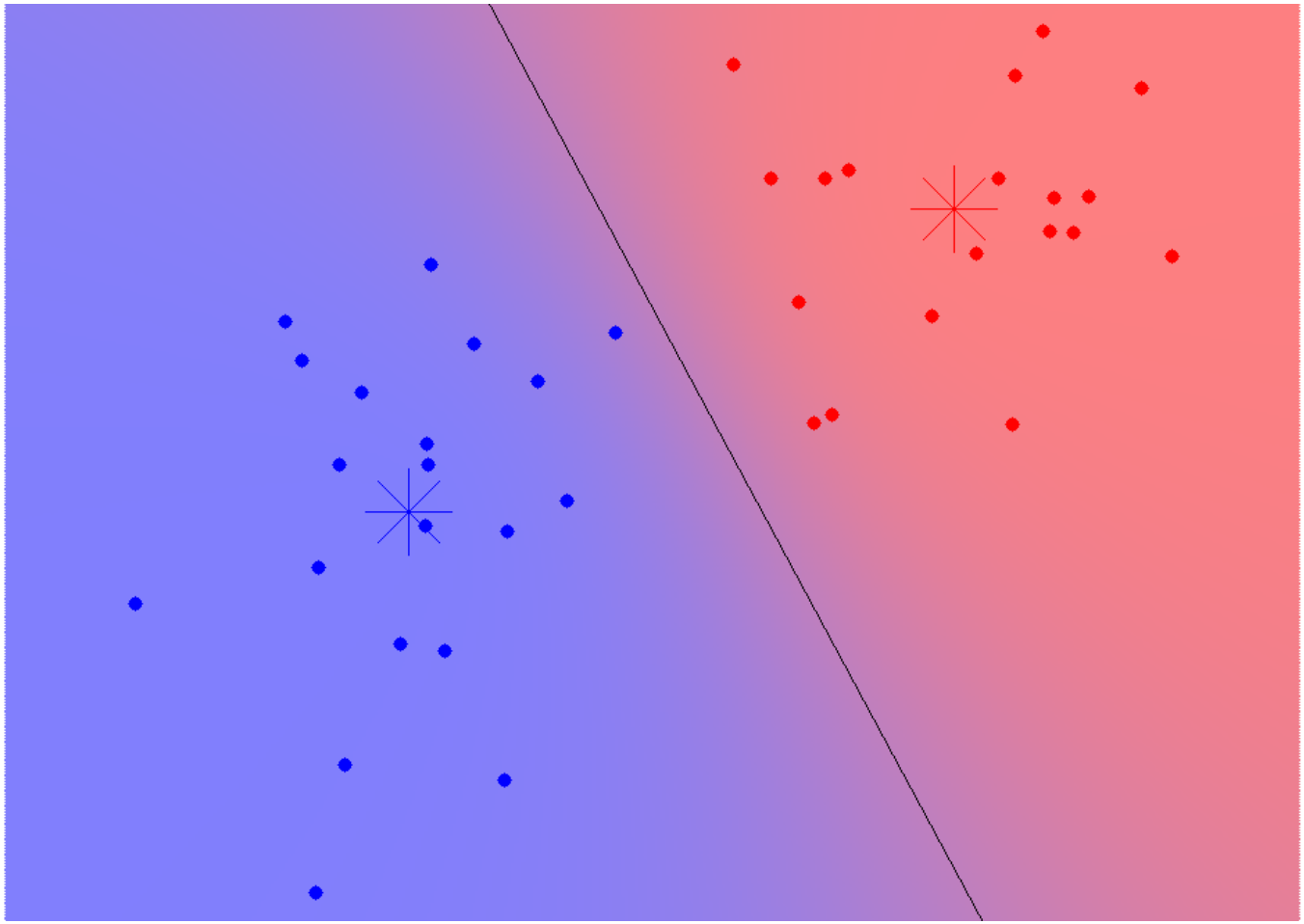
# Classification



# Classification



# Classification



# Classification

- Many strategies / algorithms
  - Linear classifiers
  - Support Vector Machines (well-founded theory)
  - k-nearest neighbours (decision from neighbours)
  - Boosting (meta-algorithm)
  - Neural networks
- Many measures of performance
  - Simplest: percentage of correct classification
- Let's become a classifier

# Decision Trees

- Decision support that uses a tree-like graph of model of decisions
- Weka's tree.UserClassifier allows the user to manually build a tree, using shapes (e.g. rectangles) on scatter plots
- Weka's tree.J48 builds a decision tree on the features' values

# J48 – Decision tree

- Top-down induction of decision trees:
  - Uses information entropy to select, at each node, which attribute will most effectively split the data
  - Splitting criterion: Kullback-Leibler divergence (information gain)
  - Recursively applied in sublists
  - Stop if all instances have same class

# Ibk – k-nearest neighbour

- No model – lazy learning
- Chooses majority class of the K-nearest neighbours

+++ Very general, only requires a notion of distance

+++ Useful for theoretical studies

--- Computationally costly

--- Doesn’t bring any insight of the data

# Problem of Overfitting

- Overfitting happens when a model is excessively complex, leading to poor predictive power
- Typically, too many parameters relative to the number of observations
- Evaluating a model (e.g. J48's decision tree) should be done using unseen data:

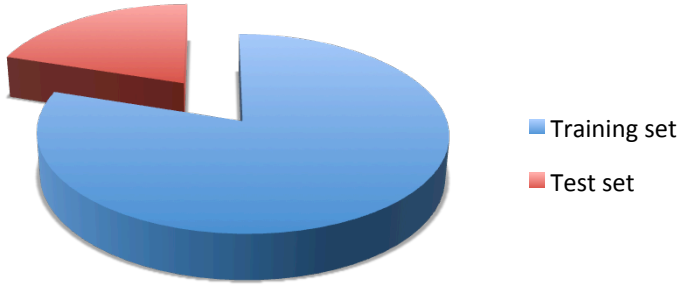
TRAINING DATA ≠ TEST DATA

# Fairness: training/testing framework

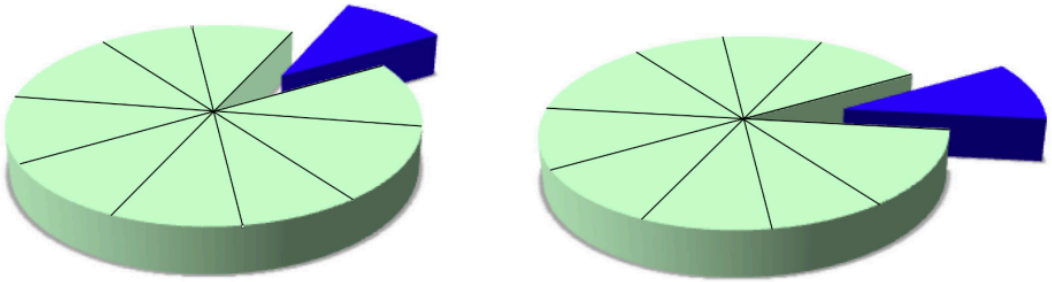
- ~~Use training set~~

- Supplied test set
(data sets in 2 different files)

- Percentage Split
(ex: hold out 20%)




- Cross-Validation
(ex: 10-fold CV)



(repeat 10 times)

# Weka's Cross-Validation

- Weka performs by default a stratified 10-fold cross-validation, with randomness pre-saved in seeds for reproducibility (instances of all classes are distributed in all folds)
- Weka would build 11 models, the first being built on the entire dataset (not used)

# CV better than holdout

CV: Validation protocol widely used (reduces variance of the estimate)

Sample mean $\bar{x} = \frac{\sum x_i}{n}$

Variance $\sigma^2 = \frac{\sum (x_i - \bar{x})^2}{n-1}$

Standard deviation $\sigma$

| holdout (10%) | cross-validation (10-fold) |
|---|---|
| 75.3 | 73.8 |
| 77.9 | 75.0 |
| 80.5 | 75.5 |
| 74.0 | 75.5 |
| 71.4 | 74.4 |
| 70.1 | 75.6 |
| 79.2 | 73.6 |
| 71.4 | 74.0 |
| 80.5 | 74.5 |
| 67.5 | 73.0 |
| $\bar{x} = 74.8$ | $\bar{x} = 74.5$ |
| $\sigma = 4.6$ | $\sigma = 0.9$ |

Rule of thumb:

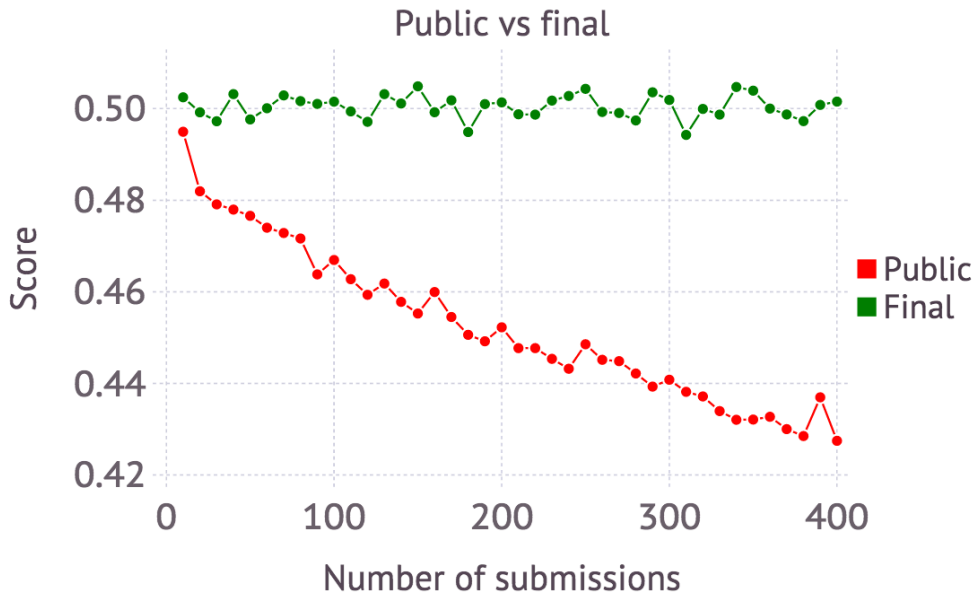
- Lots of data? Use percentage split
- Else, stratified 10-fold Cross-validation

# Example of overfitting: Kaggle

**Algorithm** (Wacky Boosting):

1. Choose $y_1, \ldots, y_k \in \{0,1\}^N$ uniformly at random.
2. Let $I = \{i \in [k] : s_H(y_i) < 0.5\}$.
3. Output $\hat{y} = \text{majority}\{y_i : i \in I\}$, where the majority is component-wise.

Lo and behold, this is what happens:




http://blog.mrtz.org/2015/03/09/competition.html

# Measures of performance

$$C_1 = \begin{pmatrix} 10 & 0 & 0 \\ 0 & 10 & 0 \\ 0 & 0 & 10 \end{pmatrix} \qquad C_2 = \begin{pmatrix} 0 & 0 & 10 \\ 10 & 0 & 0 \\ 0 & 10 & 0 \end{pmatrix} \qquad C_3 = \begin{pmatrix} 3 & 3 & 3 \\ 3 & 3 & 3 \\ 3 & 3 & 3 \end{pmatrix}$$

$$\begin{aligned} P_{CG}(C_1) &= 1 \\ \kappa(C_1) &= 1 \\ MI(C_1) &= 1.585 \end{aligned} \qquad \begin{aligned} P_{CG}(C_2) &= 0 \\ \kappa(C_2) &= -0.5 \\ MI(C_2) &= 1.585 \end{aligned} \qquad \begin{aligned} P_{CG}(C_3) &= 0.333 \\ \kappa(C_3) &= 0 \\ MI(C_3) &= 0 \end{aligned}$$

# Measures of performance

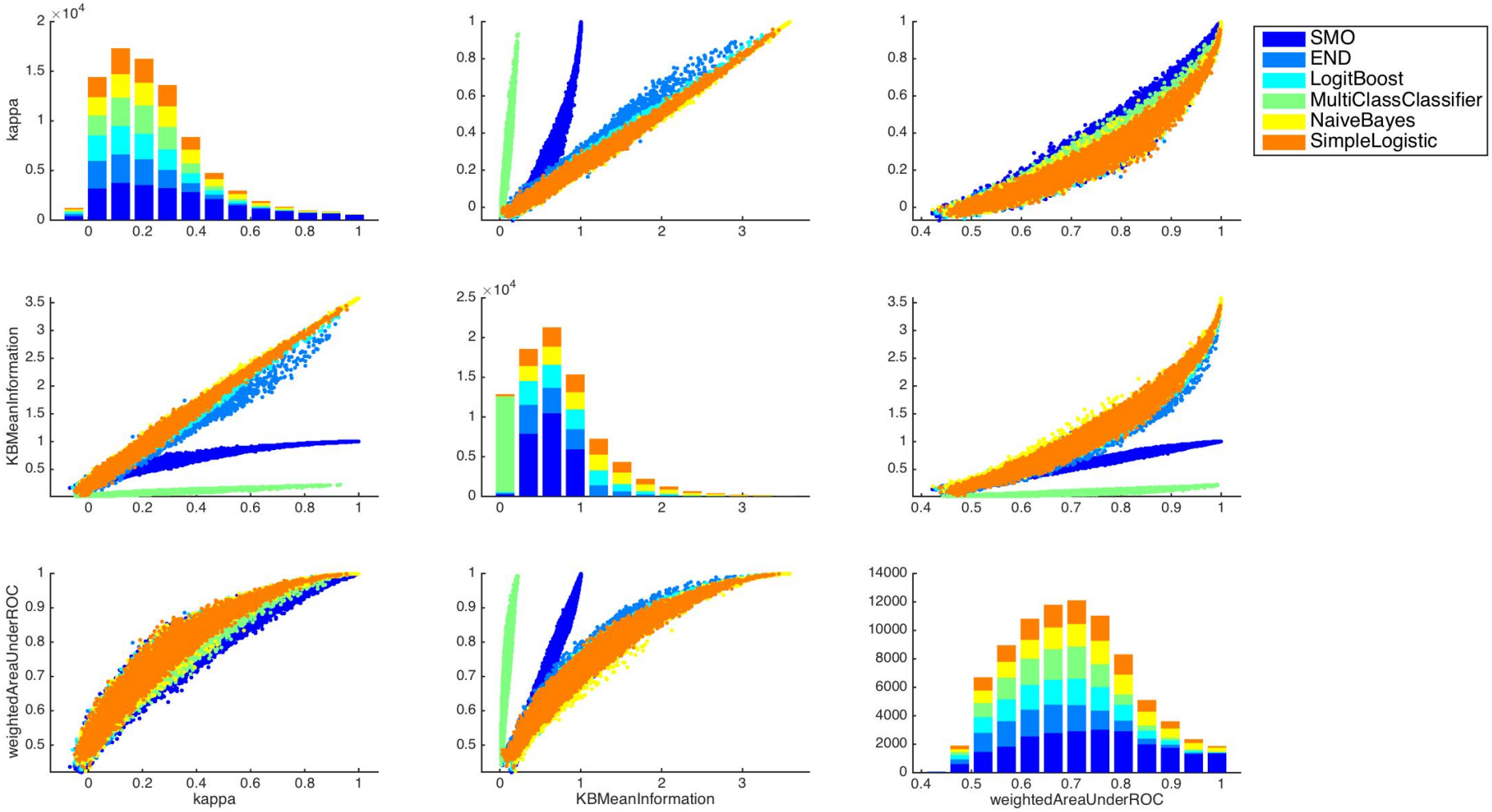
$$C_1 = \begin{pmatrix} 10 & 0 & 0 \\ 0 & 10 & 0 \\ 0 & 0 & 10 \end{pmatrix} \qquad C_2 = \begin{pmatrix} 0 & 0 & 10 \\ 10 & 0 & 0 \\ 0 & 10 & 0 \end{pmatrix} \qquad C_3 = \begin{pmatrix} 3 & 3 & 3 \\ 3 & 3 & 3 \\ 3 & 3 & 3 \end{pmatrix}$$

$$\begin{aligned} P_{CG}(C_1) &= 1 \\ \kappa(C_1) &= 1 \\ MI(C_1) &= 1.585 \end{aligned} \qquad \begin{aligned} P_{CG}(C_2) &= 0 \\ \kappa(C_2) &= -0.5 \\ MI(C_2) &= 1.585 \end{aligned} \qquad \begin{aligned} P_{CG}(C_3) &= 0.333 \\ \kappa(C_3) &= 0 \\ MI(C_3) &= 0 \end{aligned}$$

We can think of those measures as markers who have different strategies when marking an MCQ test:

- Marker 1 would simply give points when the student gives the right answer (Pct),
- Marker 2 counts the number of right answers, but wants the students who chose randomly to have 0 in the end (kappa),
- Marker 3 gives points if the student shows in any way that he has understood the problem, even if he gives only wrong answers - imagine a stubborn student that decided to always give the solution on the right of the correct one (MI).

# Measures of performance



# Data Mining with Weka

- How can we learn something about the data using a classifier?
  - rules.ZeroR (baseline)
  - rules.OneR (How far can go w/ 1 attribute)
  - trees.J48 (Efficient decision tree)
  - lazy.Ib1 (~ lazy.Ibk -K 1)
  - meta.ClassificationViaClustering (is clustering enough?)
  - Functions.SimpleLogistic (linear function of attributes)
  - functions.SMO (SVM)

# Weka by command-line

- Reference: Weka Primer

- Set environment variable CLASSPATH

setenv CLASSPATH path/to/weka/weka.jar

- Classification:

java weka.classifiers.<package>.<class> [options]

- Important generic options:

| | |
|---|---|
| -t <training_file> | To specify training file |
| -T <test_file> | If none, 10-fold CV by default |
| -x <number_of_folds> | |
| -d <output_file_save_model> | To save the model built |
| -l <input_file_saved_model> | To use a saved model on new data (lower case L) |

java weka.classifiers.trees.J48 -t /Users/pmaal/weka_data/iris.arff

java -Xmx1024m  weka.classifiers.bayes.NaiveBayes -t \
/Users/pmaal/weka_data/segment-challenge.arff -T /Users/pmaal/weka_data/segment-test.arff -p 0

- Options of a given classifier: simply run

java weka.classifiers.bayes.NaiveBayes

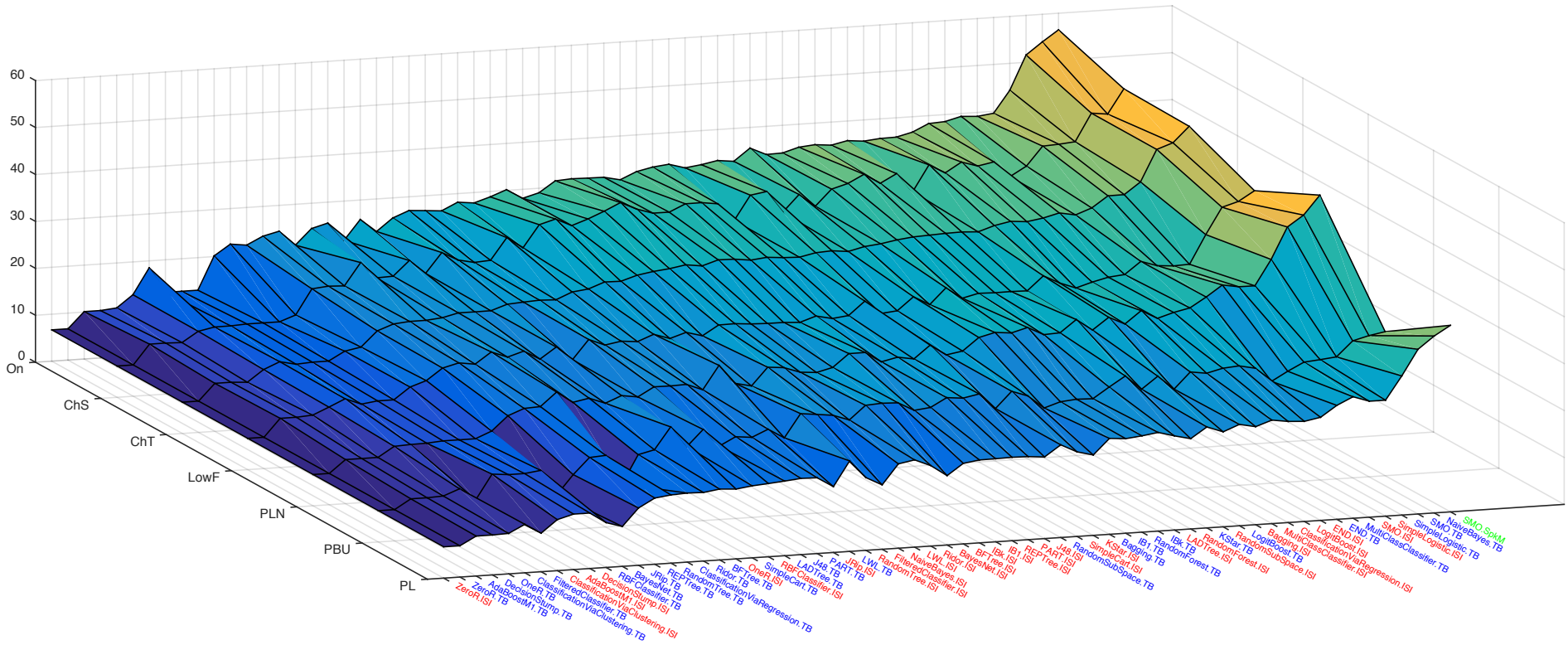
- Some statistics on a dataset

java weka.core.Instances /Users/pmaal/weka_data/iris.arff

# Dataset: Importance of preprocessing

- Importance of features: good data beats complex classifier
- There is no single best classifier (but still…)
- 3 Options (depending on what you do/your skills):
  - Preprocess before calling Weka (e.g. with Matlab)
  - Write your own preprocessing as a Weka package
  - Use Weka's filters

# Dataset: Importance of preprocessing



# Using Filters

- Edit data files
  - Ex: delete first and second attribute

java weka.filters.AttributeFilter –R 1,2 –I iris.arff –o iris_new.arff

- Create new features
  - Ex: Principal component. Clustering result
  - Ex: Unsupervised.Attribute.AddCluster with 3 clusters
- Attribute selection
- Change datatype
  - Ex: DiscretizeFilter discretizes a range of numeric attributes in the dataset into nominal attributes
  - Ex: NominalToBinaryFilter converts nominal attributes into binary ones
- Can be applied at run time, using meta.FilteredClassifier
- Others
  - classAssigner to change default position of attribute to classify

# Package Manager

- GUI > Tools > Package Manager
  - To install 3rd party tools (try some)

# Matlab with Weka

- Make an .arff file
- Addjavapath
- Import packages
- Create java object
- Manage options
- Run classification

# Meka: Multi-label extension to Weka

- http://meka.sourceforge.net/
- To predict multiple output variables for each instance.
- Literally built on top of Weka:

```
java meka.classifiers.multilabel.BR –t music.arff \
     –W weka.classifiers.bayes.NaiveBayes
```

- Run: . ~/Downloads/meka-release-1.9.0/run.sh

# References

- Data Mining With Weka (video series)

http://www.cs.waikato.ac.nz/ml/weka/mooc/dataminingwithweka/

- Andrew Ng's course Machine Learning
- Java, must-know language:

http://www.devtopics.com/most-popular-programming-languages/

- Weka tutos:
  - http://www.cs.ccsu.edu/~markov/weka-tutorial.pdf