Weka machine learning algorithms in Stata

Alexander Zlotnik, PhD
Technical University of Madrid (Universidad Politécnica de Madrid)

Stata & Weka

- Descriptive statistics

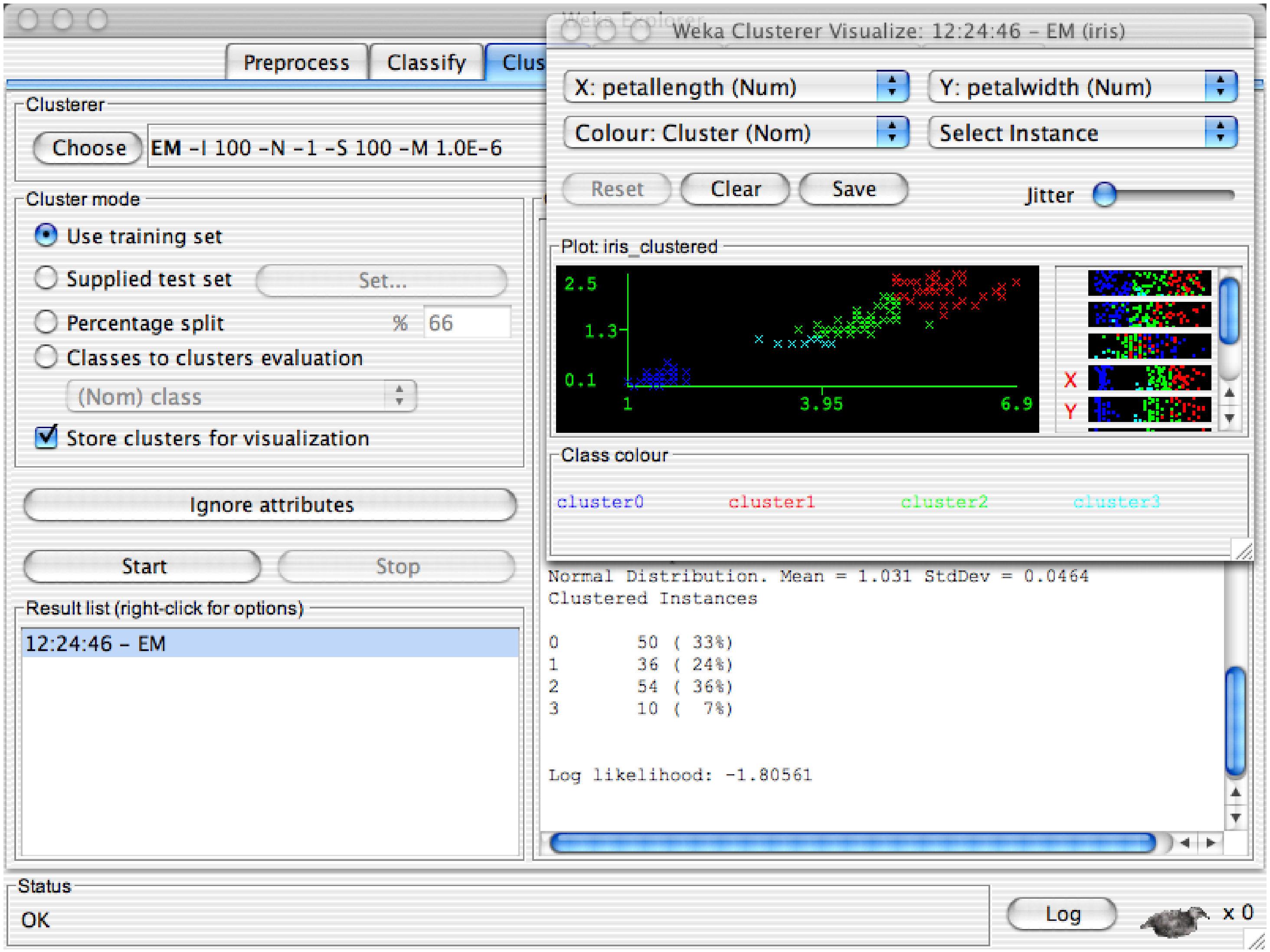
Stata

- Inferential statistics
  - Frequentist approach
  - Bayesian approach (Stata v14+)

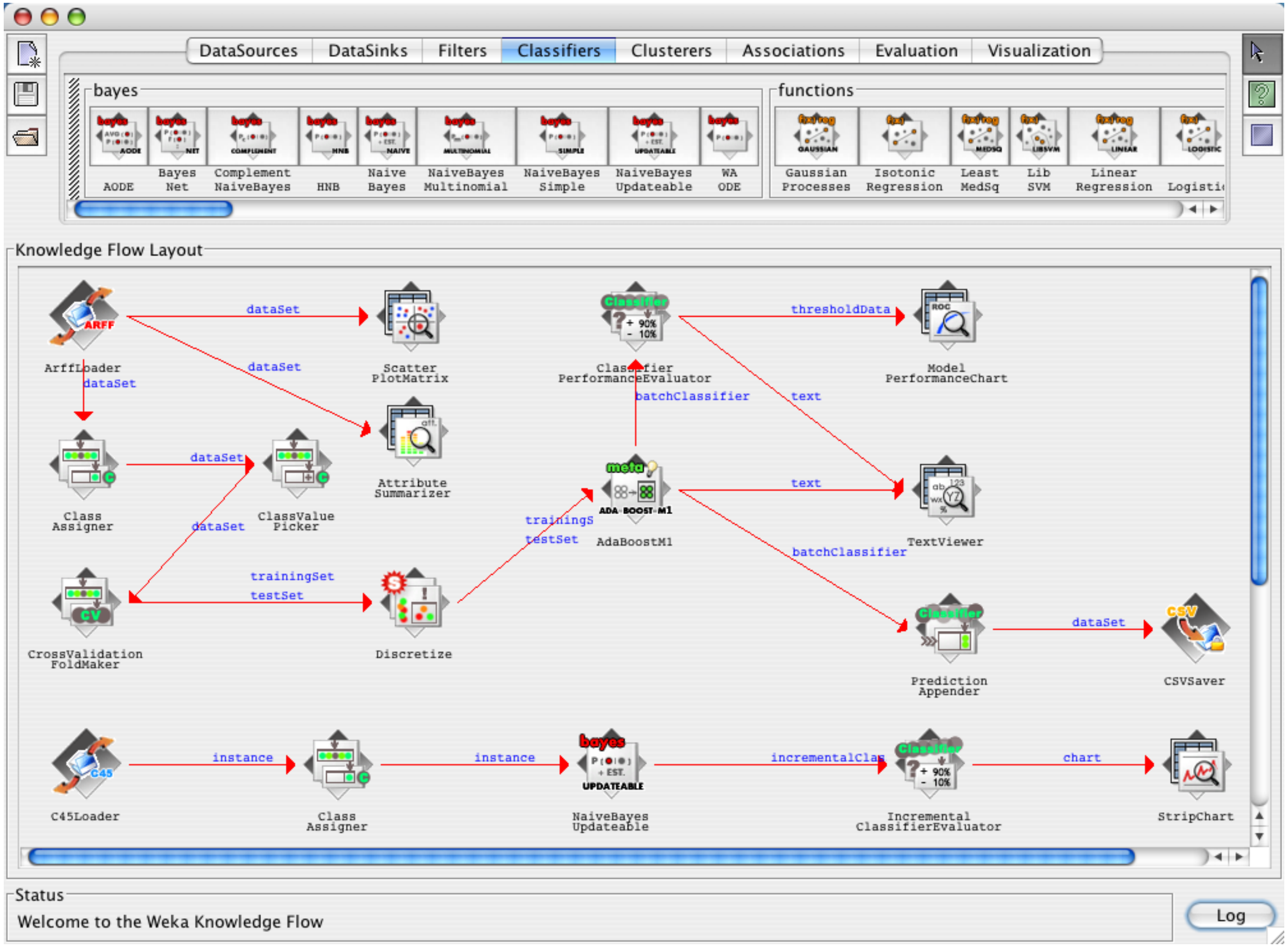
- Predictive statistics
  - Classical algorithms
  - Statistical learning / machine learning algorithms (modern artificial intelligence techniques)

Weka

Weka


Weka Explorer
Weka Clusterer Visualize: 12:24:46 - EM (iris)
Preprocess
Classify
Clus
Clusterer
Choose
EM -I 100 -N -1 -S 100 -M 1.0E-6
Cluster mode
Use training set
Supplied test set
Set...
Percentage split
% 66
Classes to clusters evaluation
(Nom) class
Store clusters for visualization
Ignore attributes
Start
Stop
Result list (right-click for options)
12:24:46 - EM
X: petallength (Num)
Y: petalwidth (Num)
Colour: Cluster (Nom)
Select Instance
Reset
Clear
Save
Jitter
Plot: iris_clustered
2.5
1.3
0.1
1
3.95
6.9
X
Y
Class colour
cluster0
cluster1
cluster2
cluster3
Normal Distribution. Mean = 1.031 StdDev = 0.0464
Clustered Instances
0 50 (33%)
1 36 (24%)
2 54 (36%)
3 10 (7%)
Log likelihood: -1.80561
Status
OK
Log
x 0

Weka



DataSources
DataSinks
Filters
Classifiers
Clusterers
Associations
Evaluation
Visualization
bayes
functions
AODE
Bayes Net
Complement NaiveBayes
HNB
Naive Bayes
NaiveBayes Multinomial
NaiveBayes Simple
NaiveBayes Updateable
WA ODE
Gaussian Processes
Isotonic Regression
Least MedSq
Lib SVM
Linear Regression
Knowledge Flow Layout
ArffLoader
dataSet
Scatter PlotMatrix
Classifier PerformanceEvaluator
thresholdData
Model PerformanceChart
batchClassifier
text
Attribute Summarizer
Class Assigner
ClassValue Picker
AdaBoostM1
TextViewer
trainingSet
testSet
CrossValidation FoldMaker
Discretize
Prediction Appender
CSVSaver
C45Loader
instance
NaiveBayes Updateable
incrementalClas
Incremental ClassifierEvaluator
chart
StripChart
Status
Welcome to the Weka Knowledge Flow
Log

Why?

Traditional predictive problems

Examples:

- Loan = {yes / no}
- Surgery = {yes / no}
- Survival time ≥ 5 years = {yes / no}

search engine / e-commerce predictive problems

- If user X searched for terms {“royal”, “palace”, “Madrid”}, how to we prioritize the results based on his previous search history?

- If customer X bought items {“color pencils”, “watercolor paint”}, what else can we sell to this same customer?

search engine / e-commerce predictive problems

… this could be also described as “software customized for each user”

a.k.a. “intelligent software”

JAMA | Original Investigation | INNOVATIONS IN HEALTH CARE DELIVERY

Development and Validation of a Deep Learning Algorithm for Detection of Diabetic Retinopathy in Retinal Fundus Photographs

Varun Gulshan, PhD; Lily Peng, MD, PhD; Marc Coram, PhD; Martin C. Stumpe, PhD; Derek Wu, BS; Arunachalam Narayanaswamy, PhD; Subhashini Venugopalan, MS; Kasumi Widner, MS; Tom Madams, MEng; Jorge Cuadros, OD, PhD; Ramasamy Kim, OD, DNB; Rajiv Raman, MS, DNB; Philip C. Nelson, BS; Jessica L. Mega, MD, MPH; Dale R. Webster, PhD

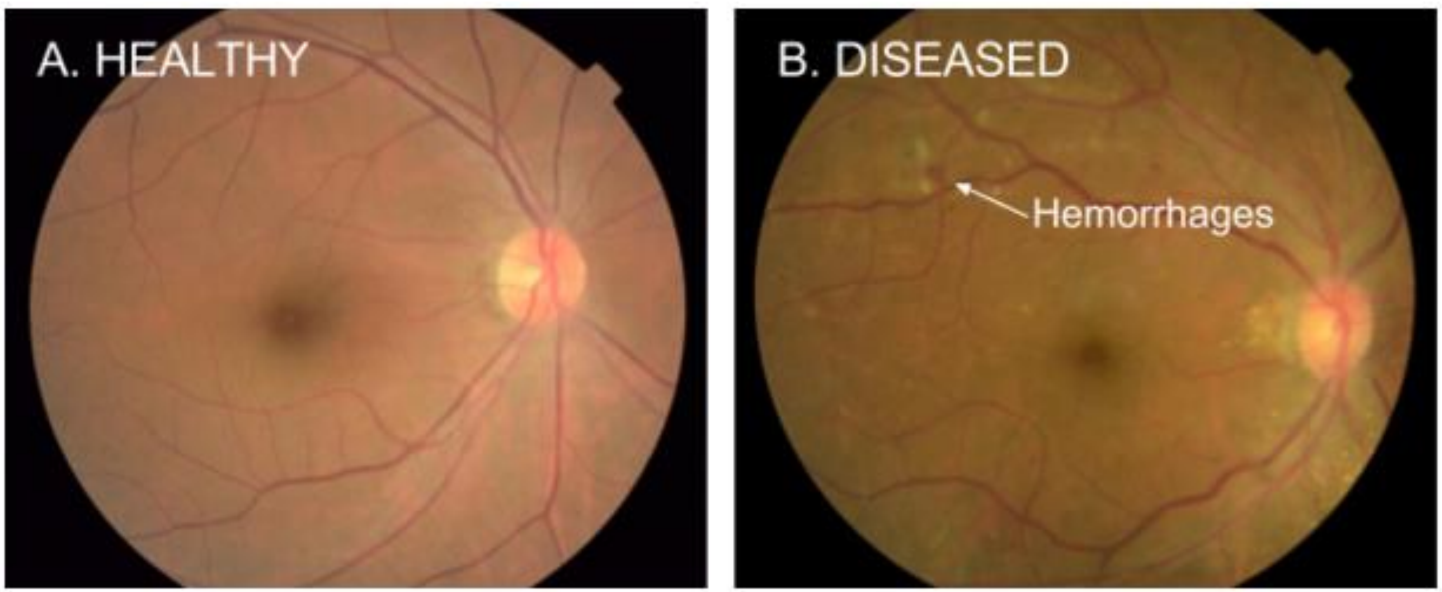


Figure 1. Examples of retinal fundus photographs that are taken to screen for DR. The image on the left is of a healthy retina (A), whereas the image on the right is a retina with referable diabetic retinopathy (B) due a number of hemorrhages (red spots) present.

Index

- **The (purely) predictive approach = machine learning**

- Common issues & solutions for AI problems

- Stata-Weka interface

(purely) predictive approach

=

machine learning

=

statistical learning

(purely) predictive approach

1. Define dependents variables

2. Set optimization objective
   (examples:
   - area under the ROC curve,
   - Homser Lemeshow calibration metrics,
   - RMSE …)

3. Choose relevant independent variables

4. Iterate through different algorithms and independent variable combinations until an <u>adequate solution</u> is found

(purely) predictive approach

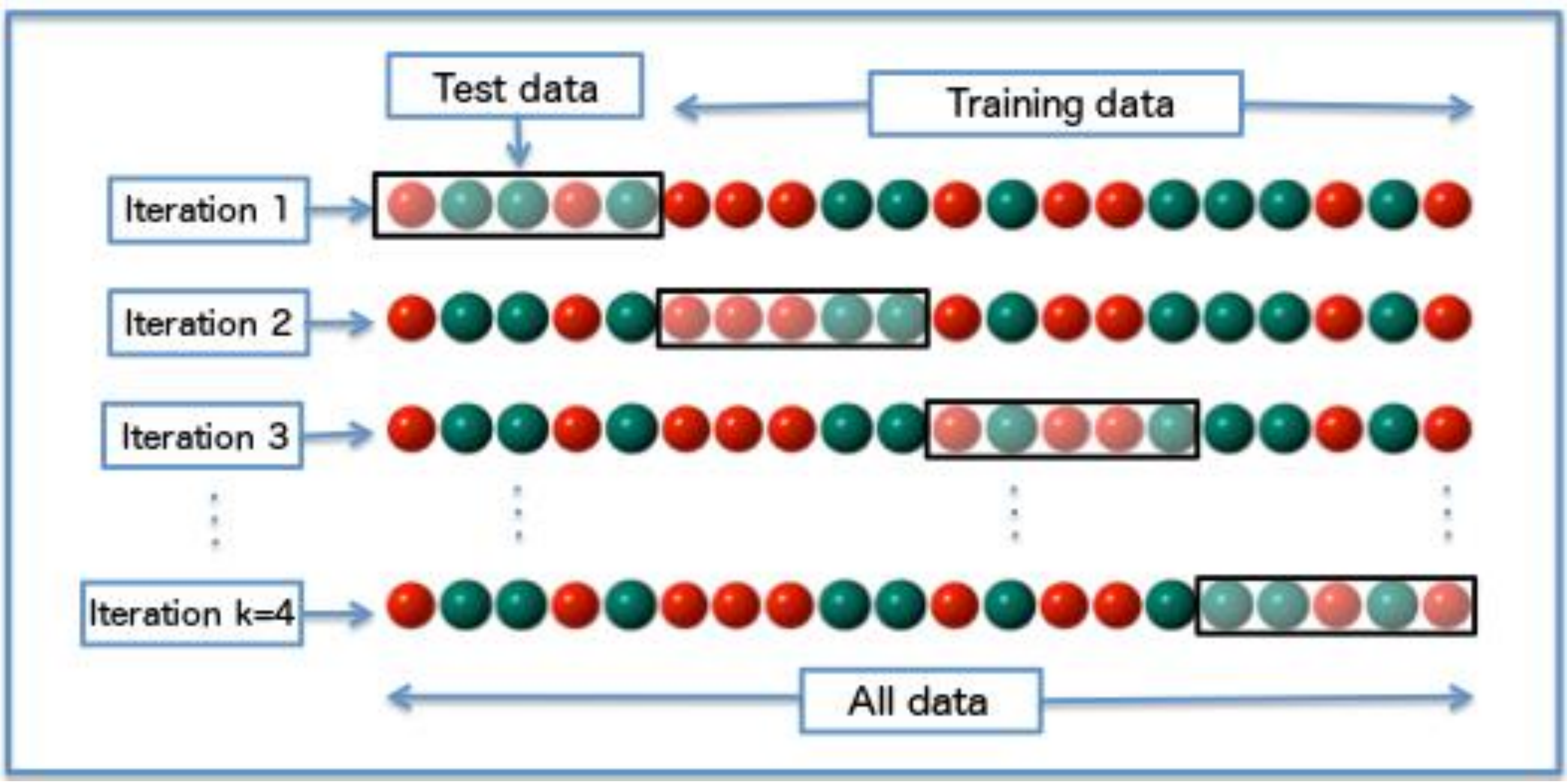
- Possible algorithms:
  - Classical statistics
    - Linear regression
    - Logistic regression
    - GLM
    - (…)
  - Machine learning
    - Decision trees (CART; C4.5; etc…)
    - Bayesian networks
    - Artificial neural networks
    - (…)

(purely) predictive approach

- Data is separated in *at least* 3 groups:
  - Train dataset
    - Used to choose an algorithm
      (example: ordinary regression, SVM, or ANN)
  - Validation dataset
    - Choose algorithm parameters => generate a “model”
      (example: kernel type and kernel parameters in SVM)
  - Test dataset
    - Evaluate results of different “models” on the test dataset

(purely) predictive approach

- Often, K-fold cross-validation is used:



Index

- The (purely) predictive approach
  = machine learning
- **Common issues & solutions for AI problems**
- Stata-Weka interface

What is an adequate solution in machine learning problems?

- Well-tested (i.e. stable results on several relevant test datasets)

- Reasonably fast (i.e. adequate response time)

- Production-ready (i.e. can be deployed)

… which is hard to achieve:

All possible variable combinations

+

Lots of data

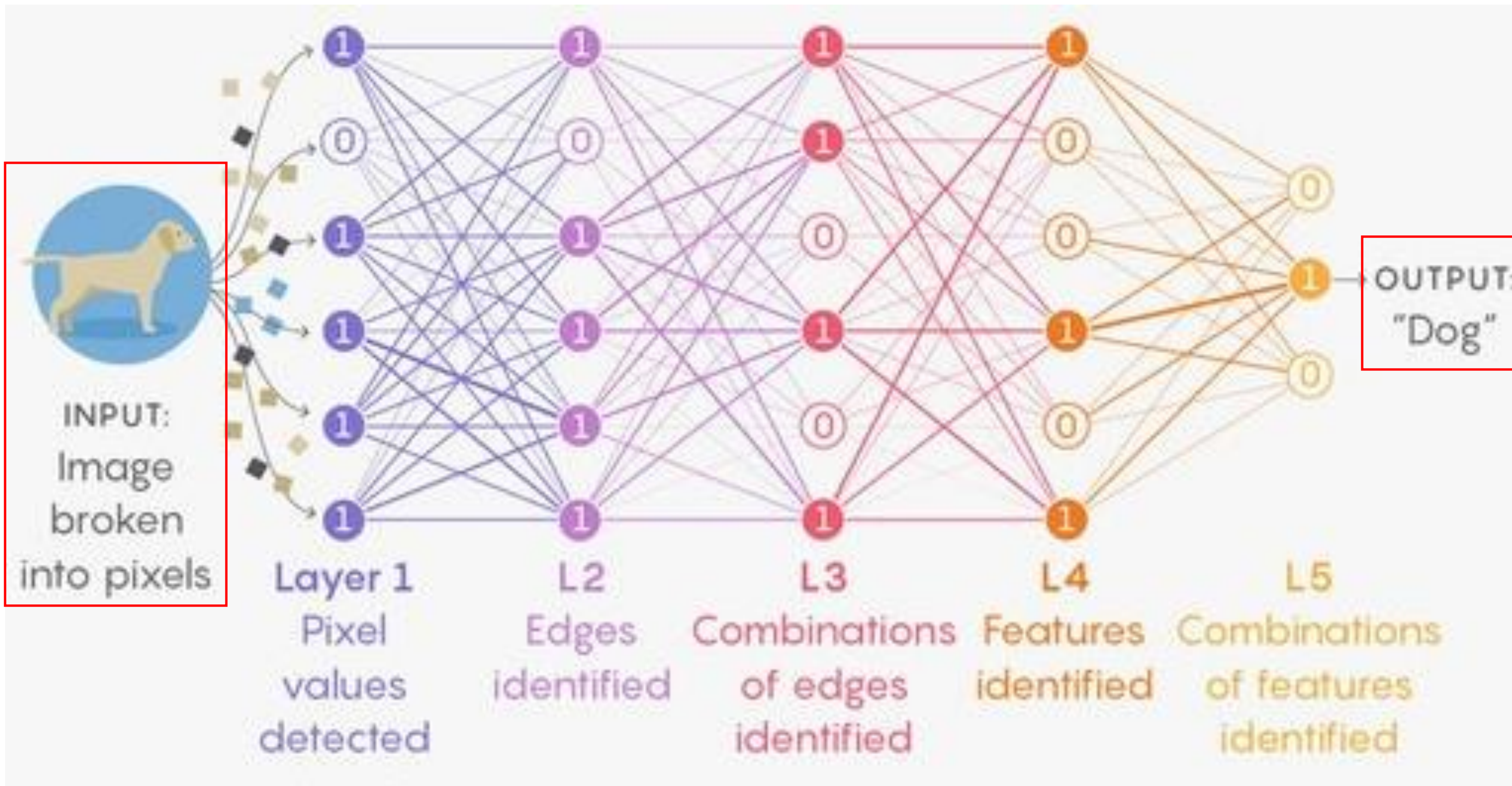
+

All possible models
(algorithm + algorithm parameters)

=

Too much computational time !!!

Why can there be many variables?



Source:
https://macnzmark.files.wordpress.com/2017/10/graph-il.jpg

x **1000** columns

1000 rows

x **16** bits (color encoding)

Common issues

- M samples

  where M >> 10^6 (a.k.a. “big data”)

- N variables

  where N >> 10^3

- Sometimes N variables > M samples

Solutions

- Dimensionality reduction techniques (that reduce computational time) such as:
  - PCA (principal component analysis)
  - SVD (singular-value decomposition)

- Automatic variable selection methods such as:
  - Forward / backward / mixed variable selection
  - LASSO (least absolute shrinkage and selection operator)

Solutions

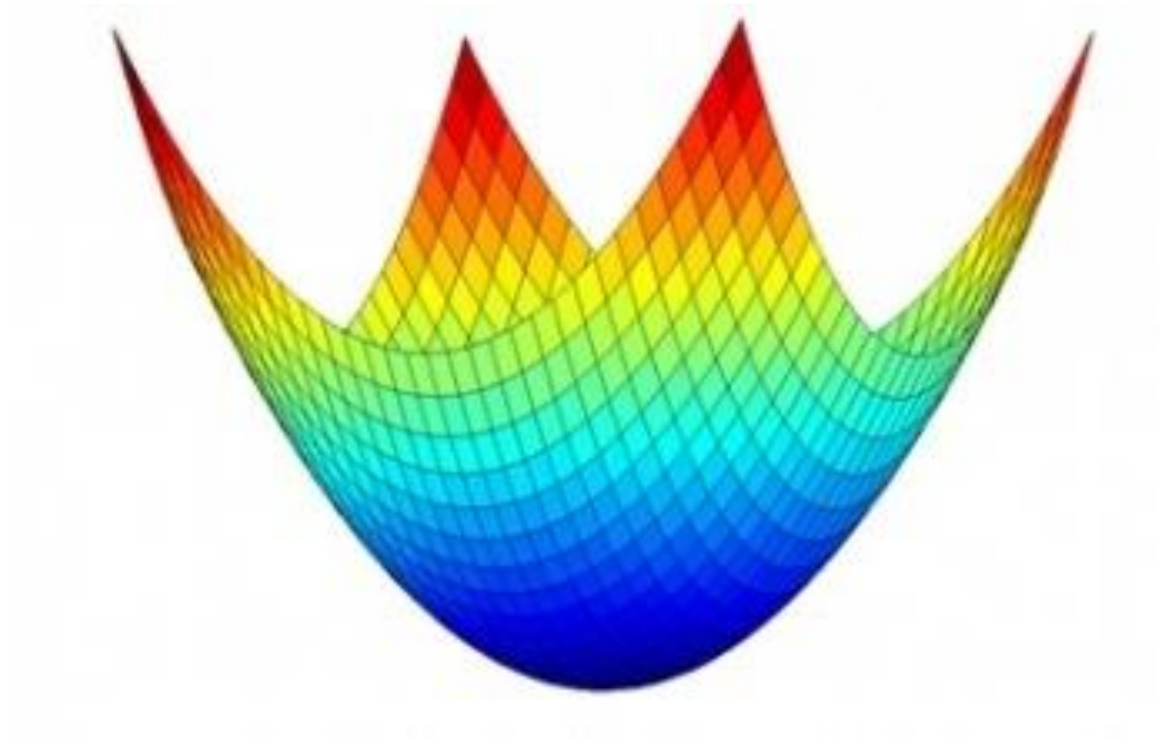
- Modern machine learning algorithms (highly *resistant to overfitting*) such as:
  - Penalized logistic regression
  - Ensemble methods
    (examples: LogitBoost / AdaBoost)
  - Support vector machines
  - Deep learning artificial neural networks

  … and, generally, some knowledge about mathematical optimization can help.

What is optimization?

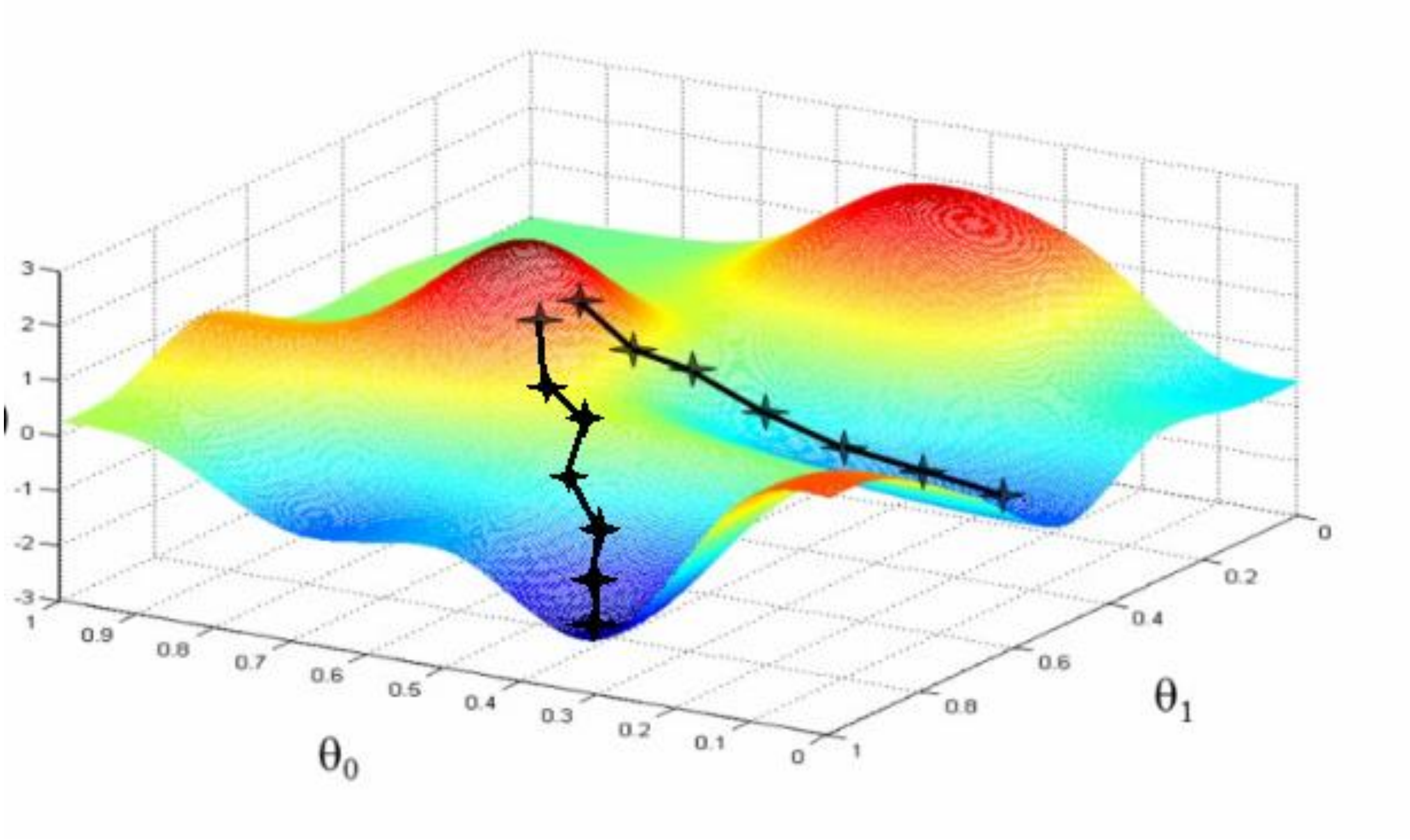
- Find a minimum = optimum.
- Optimization problems have constraints that make it solvable.
- Mathematical optimization includes several sub-topics (vector spaces, derivation, stability, computational complexity, *et cetera*).

Convex optimization



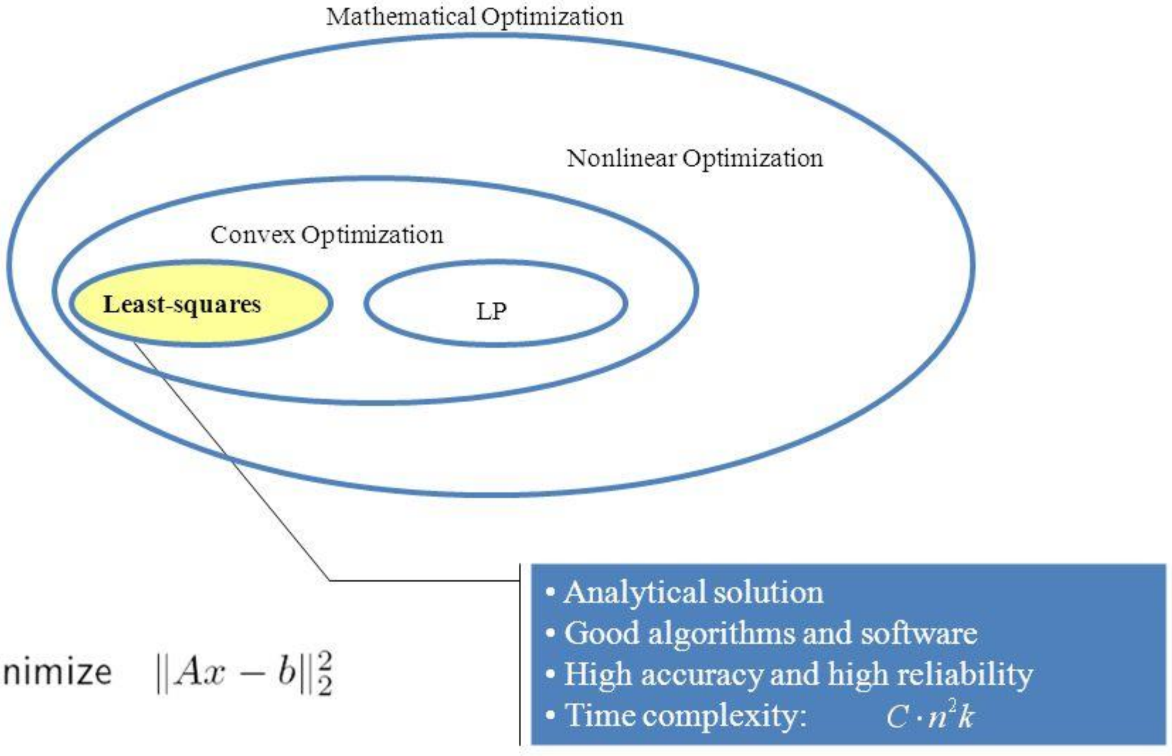
Examples:
- linear regression
- logistic regression
- linear programming / “linear optimization” => Leonid Kantorovich, 1941
- support vector machines (SVMs) => Vladimir Vapnik, 1960s

Nonlinear optimization



Examples:
- multilayer perceptron artificial neural networks
- deep learning artificial neural networks

Optimization problems



minimize $\|Ax - b\|_2^2$

A mature technology!

Source: Anjela Govan, North Carolina State University

Index

- The (purely) predictive approach = machine learning

- Common issues & solutions for AI problems

- **Stata-Weka interface**

Why Stata?

- More familiar than other languages to many Statisticians.

- Highly optimized (fast) mathematical optimization libraries for traditional statistical methods (such as linear or logistic regressions).

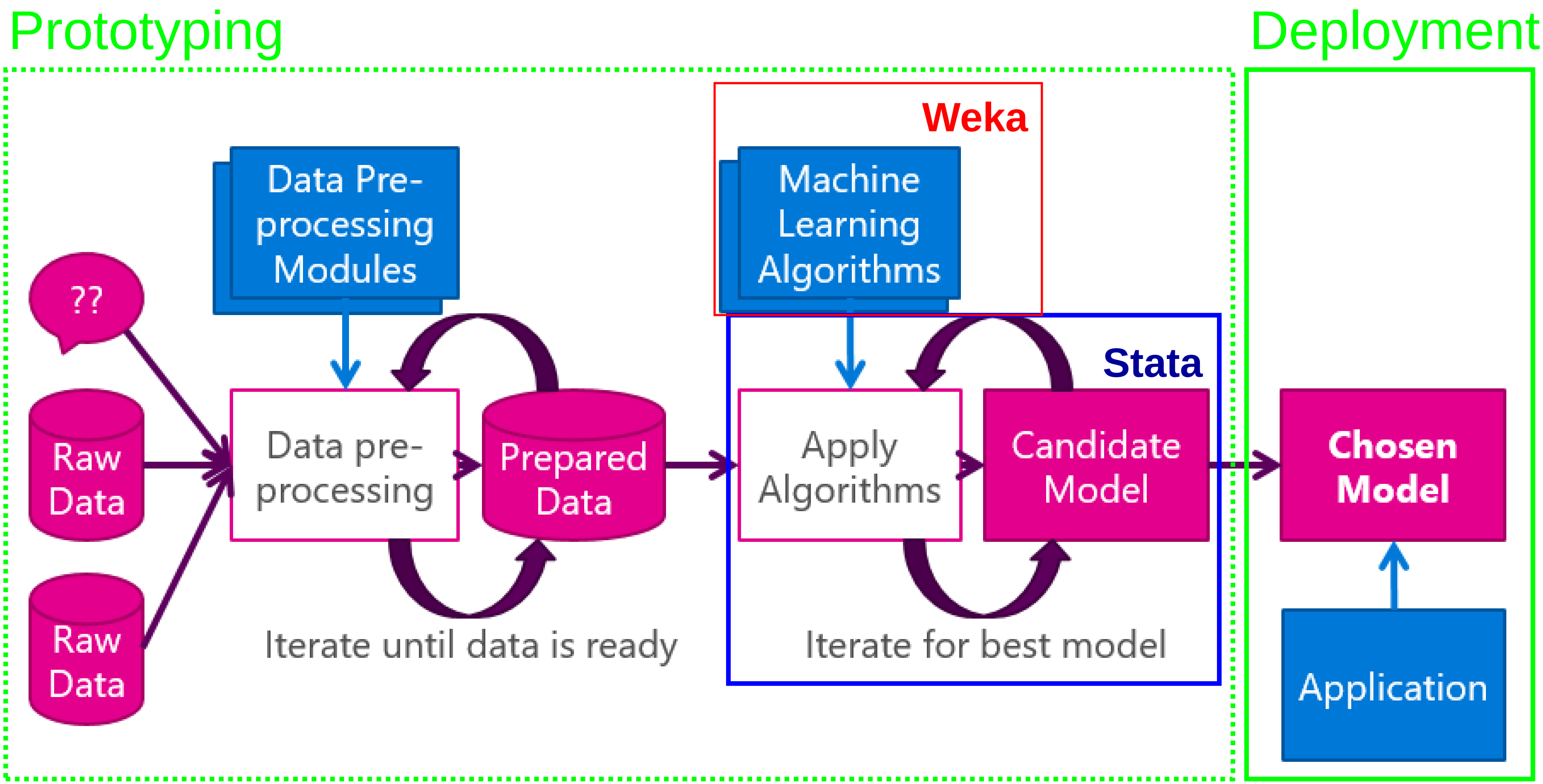
Why Stata?

- We may try different models
  in *other* software packages …

- … and then choose the best in Stata

  (Stata has many command for comparing results of predictive experiments f.ex. **-rocreg-**).

Intelligent software lifecycle



Source: https://blogs.msdn.microsoft.com/martinkearn/2016/03/01/machine-learning-is-for-muggles-too/

Why Weka?

- Open source => Code can be modified
- Good documentation
- Easy to use
- Has most modern machine-learning algorithms (including ensemble classifiers)
- Time series
  (generalized regression machine-learning models; *usually better than S ARIMA X or VAR models*)

Stata-Weka interface

Modify Weka API

Then

- Load data in Stata
- Call Weka from Stata
- Calculate results in Weka
- Return results from Weka to Stata
- Process results in Stata

Stata-Weka interface

- Modified version of Weka API in Java (`StataWekaCMD`)

```
import weka.core.Instances;
import weka.experiment.InstanceQuery;
...
InstanceQuery query = new InstanceQuery();
query.setUsername("nobody");
query.setPassword("");
query.setQuery("select * from whatsoever");
// You can declare that your data set is sparse
// query.setSparseData(true);
Instances data = query.retrieveInstances();
```

```
// create new instance of scheme
weka.classifiers.functions.SMO scheme = new weka.classifiers.functions.SMO();
// set options
scheme.setOptions(weka.core.Utils.splitOptions("-C 1.0 -L 0.0010 -P 1.0E-12 -N 0
```

```
for (int i = 0; i < test.numInstances(); i++) {
  double pred = fc.classifyInstance(test.instance(i));
  System.out.print("ID: " + test.instance(i).value(0));
  System.out.print(", actual: " + test.classAttribute().value((int) test.instance
  System.out.println(", predicted: " + test.classAttribute().value((int) pred));
}
```

Stata-Weka interface

- Stata:
  - Export to Weka-readable CSV file
  - Call Java program from Stata:

```
!java -jar "C:\TEMP\StataWekaCMD.jar"
`param1' ... `paramN'
```

Stata-Weka interface

- Java program (StataWekaCMD.jar):
  - Call *modified* instance of Weka & produce output
  - Adapt Weka output to Stata-readable CSV & export it

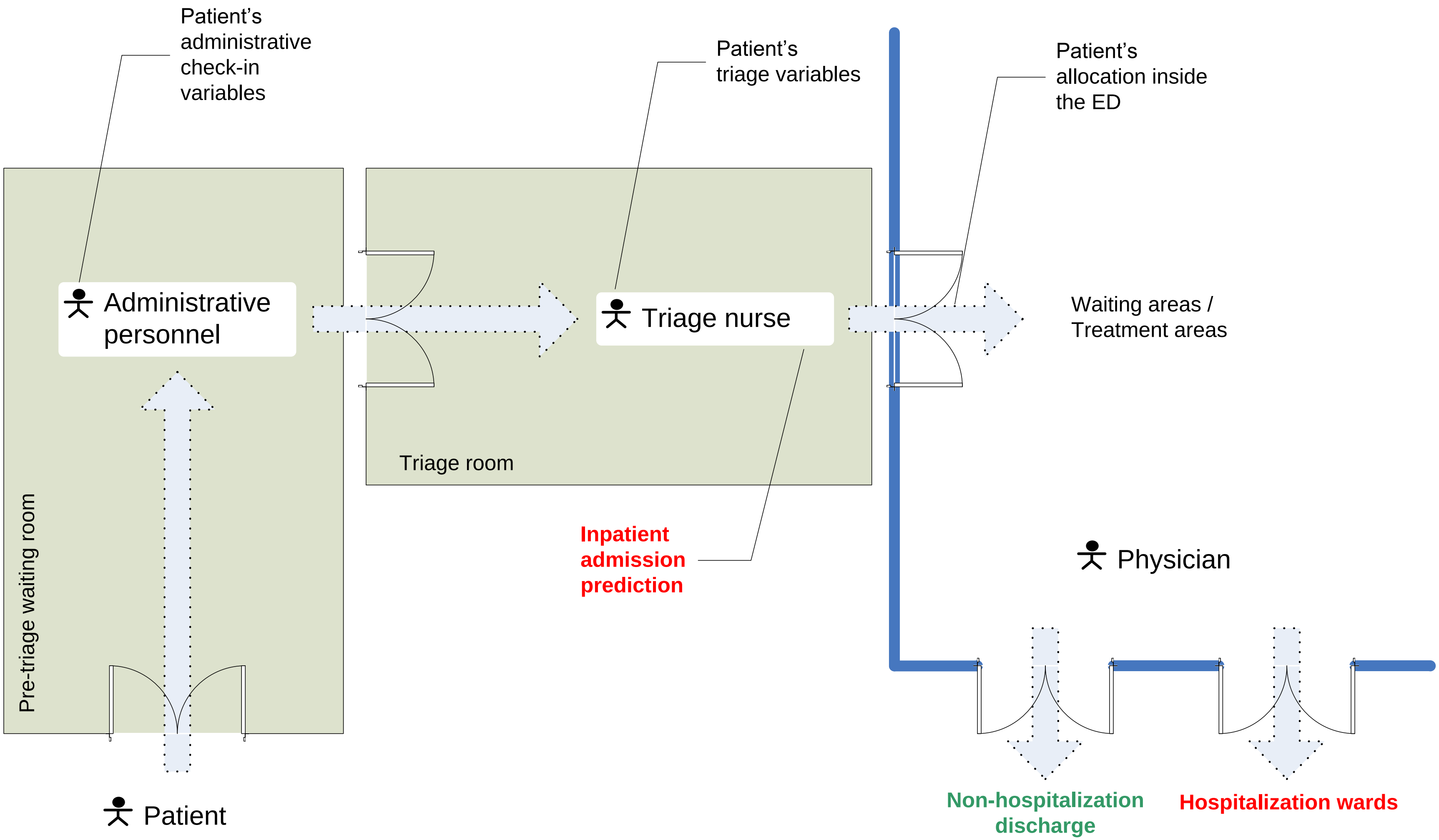
Stata-Weka interface

- Stata:
  - Process classification result file:

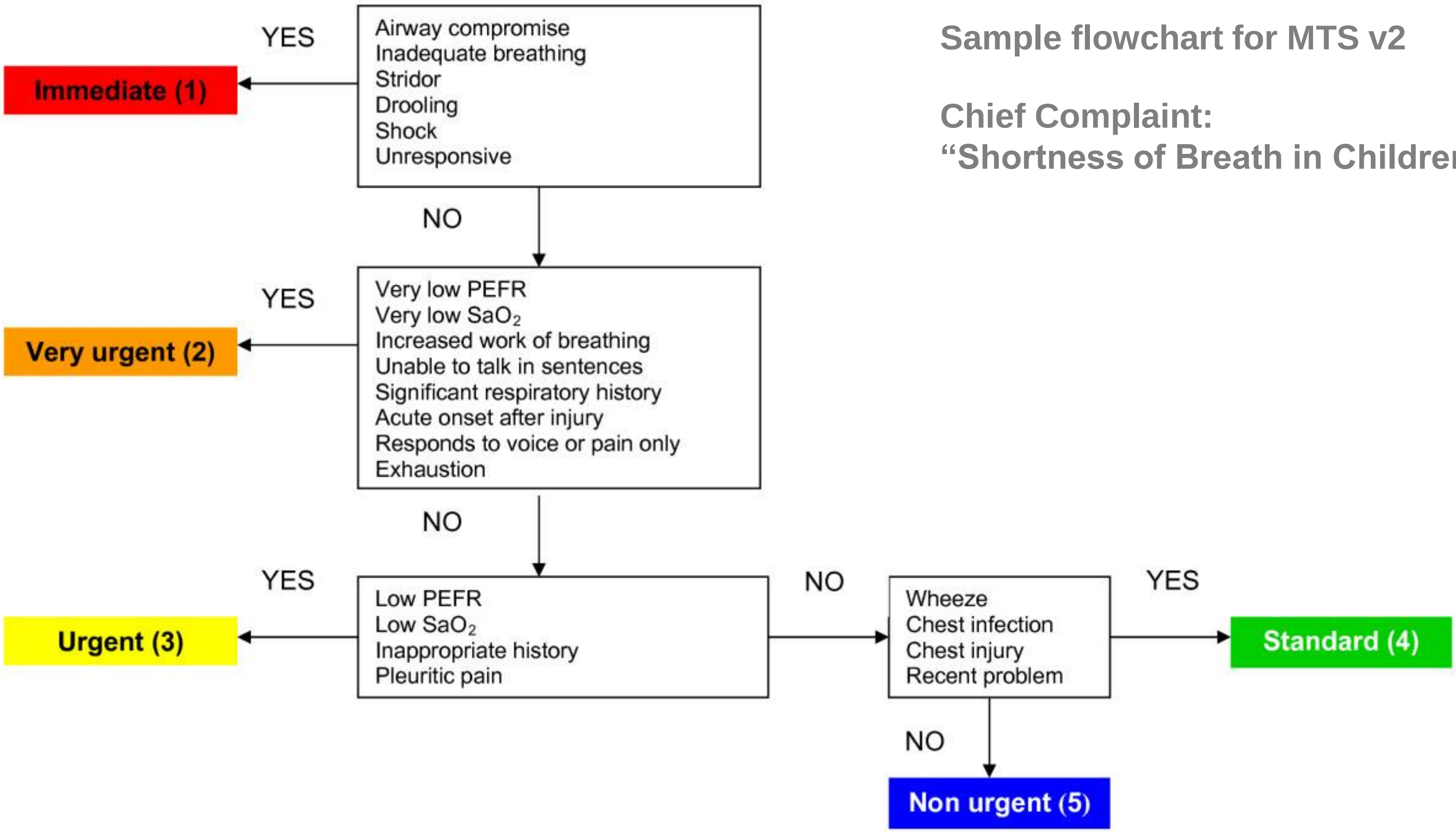
```
preserve
insheet weka_output.csv
save weka_output.dta, replace
restore
merge 1:1 PK using weka_output.dta
```

Let’s see an example

Inpatient admission prediction from the Emergency Department



Manchester Triage System (MTS)



Sample flowchart for MTS v2

Chief Complaint:
“Shortness of Breath in Children”

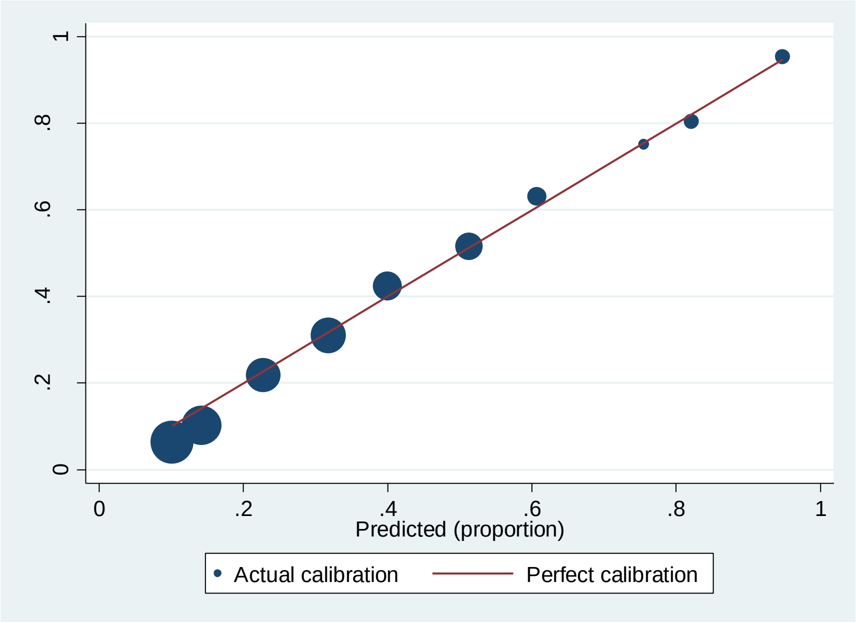
However…

- Priority of care ≠ Clinical severity
- Example:
  - Patient with terminal stage 4 cancer with a chief complaint “mild fever”:
    - Priority of care = Low (MTS level = 5)
    - Clinical severity = High => Likely admission

Objectives

- Design a system that can predict the **probability of inpatient admission (yes / no)** from the ED *right after triage*.

- With *adequate* discrimination (AUROC > 0.85) and calibration (H-L $\chi^2 < 15.5$ => H-L p-value > 0.05).



Algorithms

- Logistic regression (LR)
- Artificial neural network (ANN)
- Custom algorithm

Custom algorithm definition

1. Compute M1 = base logistic regression for the whole dataset
2. FOR EACH CC = Chief complaint

    Compute $M2_{CC}$ = LogitBoost submodel for this Chief complaint

    IF ((H-L DF $|_{M2CC}$ >= H-L DF $|_{M1}$) AND (H-L χ^2 $|_{M2CC}$ <= H-L χ^2 $|_{M1}$))

        Use $M2_{CC}$ for this chief complaint

    ELSE

        Use M1 for this chief complaint

    END IF

   END FOR

3. Output the predictions of the ensemble

Hybrid Stata-Weka application

Custom algorithm definition

1. Compute M1 = base logistic regression for the whole dataset

2. FOR EACH CC = Chief complaint

    Compute $M2_{CC}$ = LogitBoost submodel for this Chief complaint

    IF ((H-L DF $|_{M2CC}$ >= H-L DF $|_{M1}$) AND (H-L χ^2 $|_{M2CC}$ <= H-L χ^2 $|_{M1}$))

        Use $M2_{CC}$ for this chief complaint

    ELSE

        Use M1 for this chief complaint

    END IF

  END FOR

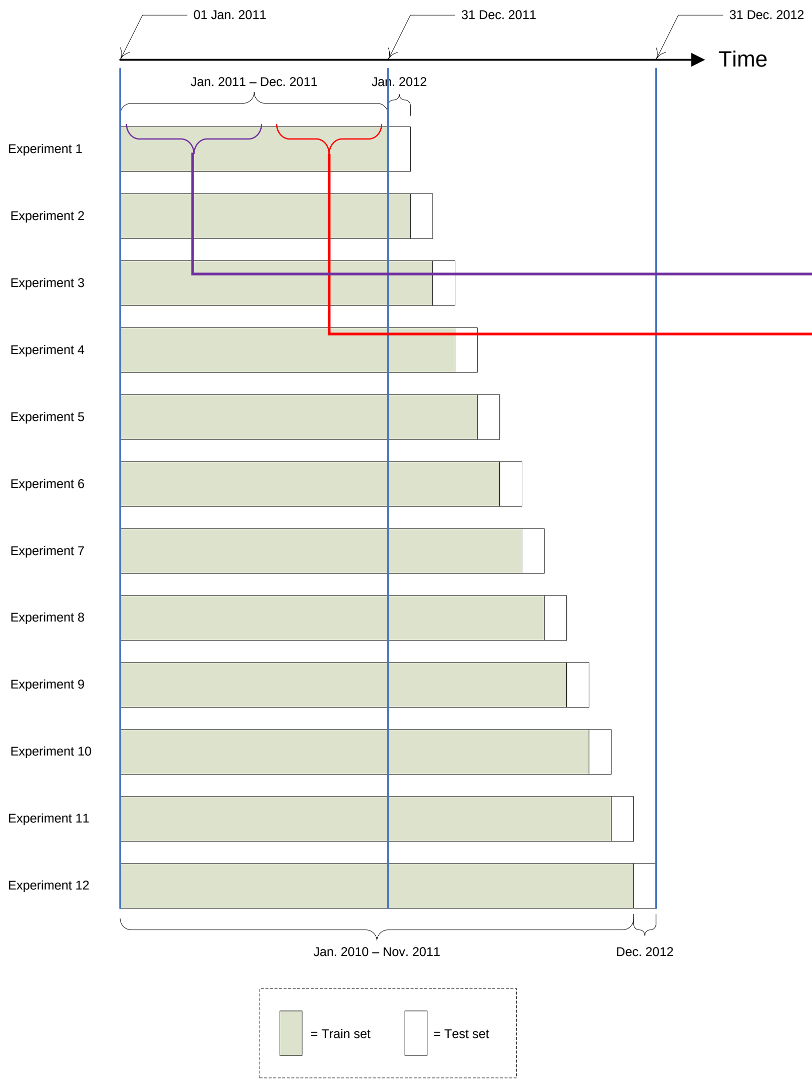
3. Output the predictions of the ensemble

Stata

Weka

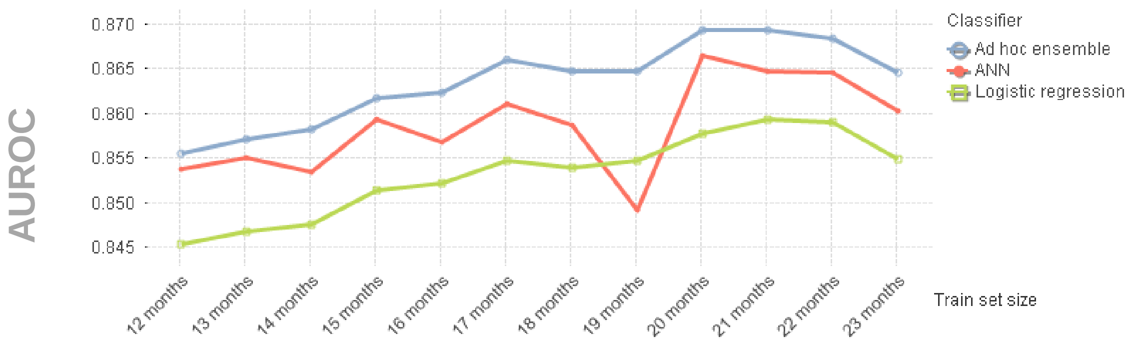
Stata

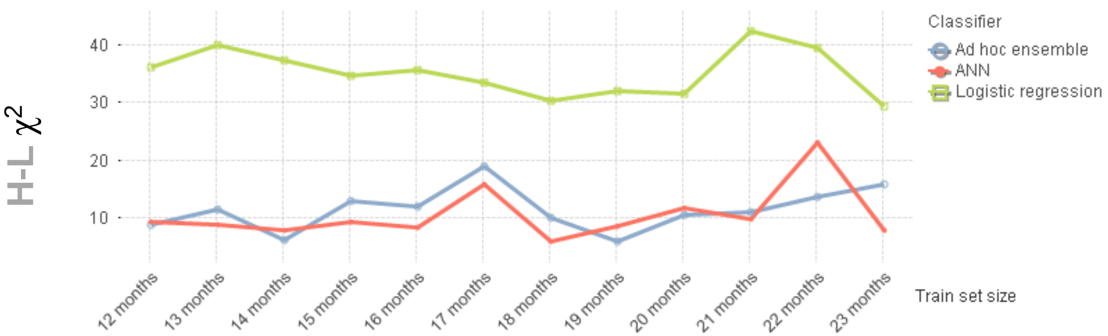
Model evaluation



- **Within each iteration:**
  - *Ordered* split in:
    - 2/3 data = train
    - 1/3 data= validation
  - Repeat grouping of MTS CC on
    - 2/3 data = train
  - Repeat ANN parameterization
    - 1/3 data = validation
  - *Next* month = test

Model evaluation





- **Logistic regression**
  - AUROC = 0.8531
    95% CI (0.8501, 0.8561)
  - H-L χ^2 = 35.15
    95% CI (32.57, 37.73)
- **ANN**
  - AUROC = 0.8568
    95% CI (0.8531, 0.8606)
  - H-L χ^2 = 10.47
    95% CI (7.78, 13.17)
- **Custom algorithm**
  - AUROC = 0.8635
    95% CI (0.8605, 0.8665)
  - H-L χ^2 = 11.4
    95% CI (9.10, 13.75)

Thank you !