Cirrhosis Prediction with ML methods

Alexander BAUMANN, Naomi Mona CHMIELEWSKI

March 30, 2022

Outline

Motivation

- Cirrhosis is scarring (fibrosis) of the liver caused by long-term liver damage
- Cirrhosis can eventually lead to liver failure, which can be life threatening
- Treatment may be able to stop cirrhosis from getting worse

$\Rightarrow$ Desire to predict cirrhosis

The Data [1, 2, 3]

- data collected from 1974 to 1984 in a Mayo Clinic trial in order to study the effects of a drug
- Total of 418 patients
  - 312 patients with full data
  - 106 patients with only some basic measurements
- 18 predictive features
- four liver stages: healthy, fatty, fibrosis, cirrhosis
- our goal: classify between cirrhosis (1) and no cirrhosis (0)

Sample from data set

	ID	N_Days	Status	Drug	Age	Sex	Ascites	Hepatomegaly	Spiders	Edema	Bilirubin
0	1	400	D	D-penicillamine	21464	F	Y	Y	Y	Y	14.5
1	2	4500	C	D-penicillamine	20617	F	N	Y	Y	N	1.1
2	3	1012	D	D-penicillamine	25594	M	N	N	N	S	1.4
3	4	1925	D	D-penicillamine	19994	F	N	Y	Y	S	1.8
4	5	1504	CL	Placebo	13918	F	N	Y	Y	N	3.4

	Cholesterol	Albumin	Copper	Alk_Phos	SGOT	Tryglicerides	Platelets	Prothrombin	Stage
0	261.0	2.60	156.0	1718.0	137.95	172.0	190.0	12.2	4.0
1	302.0	4.14	54.0	7394.8	113.52	88.0	221.0	10.6	3.0
2	176.0	3.48	210.0	516.0	96.10	55.0	151.0	12.0	4.0
3	244.0	2.54	64.0	6121.8	60.63	92.0	183.0	10.3	4.0
4	279.0	3.53	143.0	671.0	113.15	72.0	136.0	10.9	3.0

Table: Sample from raw Cirrhosis data set

Preprocessing

- drop the columns of "ID", "N_Days", "Hepatomegaly" and "Status"
- replace missing values of numerical features by the median
- replace missing values of categorical features by the most frequent or most reasonable class
- label-encoding of categorical variables
- binarize target value

First Model

- we used 5-fold validation on all models and compared average accuracies as well as computing time of the 5-fold validation
- we used gridsearch on all models
- first simple model on preprocessed data: support vector classifier
- average test accuracy over 5 folds: 0.68
- computation time 0.28 seconds per fold

Dimensionality Reduction

- PCA with 3 components

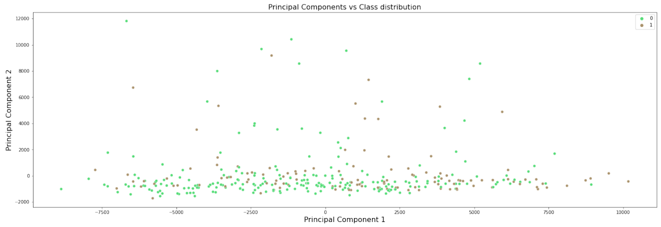


Figure: First Principal Components of PCA

- similar results for Kernel PCA - see notebook. We could not find any parameter combination that resulted in seperable data
- implementing SVM on top of PCA and KPCA yielded expectedly mediocre results: test accuracy 0.68 for both, computation time 0.06s and 0.08s respectively
- LDA and QDA performed better (0.71 both), but the data was still not seperable

Boosting

- AdaBoost: accuracy 0.72, computation time 0.12 seconds
- XGBoost: accuracy 0.76, computation time 0.24 seconds

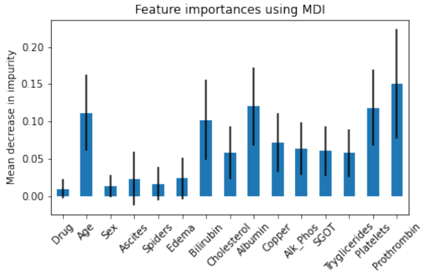
Other Methods

Model	Accuracy	Time in seconds
KNN	0.62	0.07
MLP	0.65	0.23
Gaussian Naive Bayes	0.73	0.01
Random Forest	0.74	1.77

Table: Summary of other methods

Feature Selection

- Feature importances from Random Forest:



- Combining feature selection by thresholding these importances and XGBoost achieved an accuracy of 0.76

Feature Selection

- LASSO + Logistic Regression, LASSO + LinearSVC and Elastic Net achieved maximal accuracy of 0.73
- most frequently selected features: Age, Cholesterol, Copper, Alk-Phos, Platelets

Improvement Propositions

- for this section, we split the full data set into 80% train and 20% test, performing 5-fold validation on the train set and testing the improved method on the test set
- we trained five different XGBoost classifiers using the 5-fold validation, saving each classifier seperately
- we used the best performing parameters identified previously through gridsearch
- our final model consisted in predicting cirrhosis (1) if 3 or more classifiers predicted cirrhosis, and 0 otherwise
- we were able to achieve the best accuracy yet: 0.77
- sklearn has an implementation of weighted majority vote. Giving the best performing classifier (during training) the highest weight, we achieved 80% accuracy

Conclusion - Data

- very few healthy samples (circa 5%), as the data come from a study to combat cirrhosis
  $\Rightarrow$ Imbalanced classes make multi-class classification practically impossible
- too many missing values in an already small data set
- still interesting to see the feature importances

Comparison of models

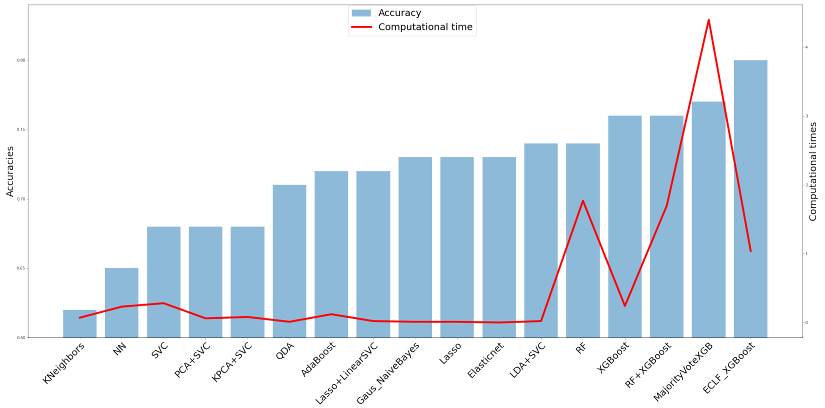


Figure: Accuracies and computational times of all models

Conclusion - Models

- interpretable models like logistic regression + penalisation did not perform very well, but are very fast
- best models were ensemble models (random forest, boosting, ensemble vote) which take longer to train and are not at all interpretable
- other papers indicated that much better accuracies can be achieved on different, more complete data sets (see [1], Indian Liver Patient Dataset, only detect 'Liver Disease' or 'No Liver Disease'); generally use methods like decision tree, random forest and SVM, where accuracies of around 90% are possible
- with a larger, more complete data set, ensemble methods might approach a similar accuracy to [1] for liver cirrhosis prediction. Note that ensemble methods are slow, which might have an impact on very large data sets.

References

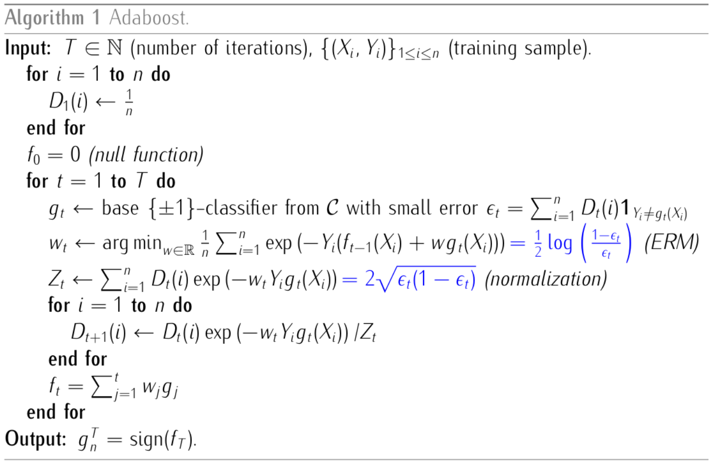
[1] P M Dattatreya et al. “Machine Learning Techniques in Analysis and Prediction of Liver Disease”. In: *IJIRT* 8.2 (2021).

[2] E Rolland Dickson et al. “Prognosis in primary biliary cirrhosis: model for decision making”. In: *Hepatology* 10.1 (1989), pp. 1–7.

[3] fedesoriano. *Cirrhosis Prediction Dataset*. 2020. URL: https://www.kaggle.com/fedesoriano/cirrhosis-prediction-dataset.

Appendix

AdaBoost

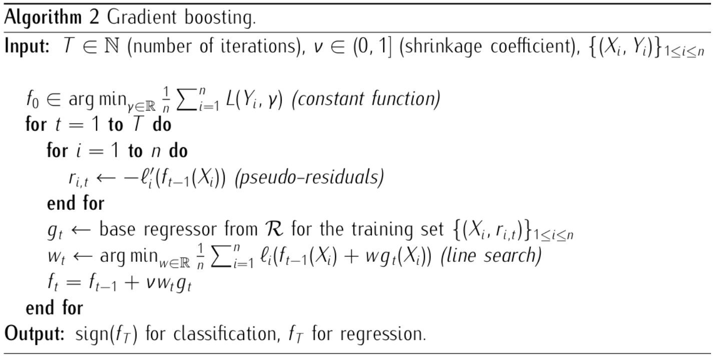


Algorithm 1 Adaboost.

```
Input: $T \in \mathbb{N}$ (number of iterations), $\{(X_i, Y_i)\}_{1 \leq i \leq n}$ (training sample).
  for $i = 1$ to $n$ do
    $D_1(i) \leftarrow \frac{1}{n}$
  end for
  $f_0 = 0$ (null function)
  for $t = 1$ to $T$ do
    $g_t \leftarrow$ base $\{\pm 1\}$-classifier from $\mathcal{C}$ with small error $\epsilon_t = \sum_{i=1}^n D_t(i) \mathbf{1}_{Y_i \neq g_t(X_i)}$
    $w_t \leftarrow \arg\min_{w \in \mathbb{R}} \frac{1}{n} \sum_{i=1}^n \exp\left(-Y_i(f_{t-1}(X_i) + w g_t(X_i))\right) = \frac{1}{2} \log\left(\frac{1-\epsilon_t}{\epsilon_t}\right)$ (ERM)
    $Z_t \leftarrow \sum_{i=1}^n D_t(i) \exp\left(-w_t Y_i g_t(X_i)\right) = 2\sqrt{\epsilon_t(1-\epsilon_t)}$ (normalization)
    for $i = 1$ to $n$ do
      $D_{t+1}(i) \leftarrow D_t(i) \exp\left(-w_t Y_i g_t(X_i)\right) / Z_t$
    end for
    $f_t = \sum_{j=1}^t w_j g_j$
  end for
Output: $g_n^T = \mathrm{sign}(f_T)$.
```

Figure: Pseudocode of AdaBoost (from M Sangnier - Introduction to Machine Learning, 2021)

Gradient Boosting



Algorithm 2 Gradient boosting.

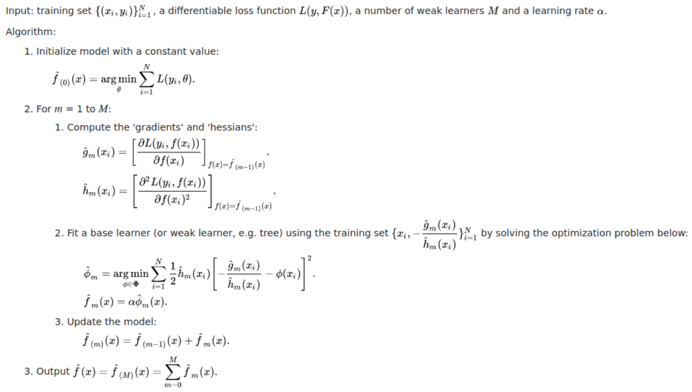
Input: $T \in \mathbb{N}$ (number of iterations), $\nu \in (0,1]$ (shrinkage coefficient), $\{(X_i, Y_i)\}_{1 \le i \le n}$

```
$f_0 \in \arg\min_{\gamma \in \mathbb{R}} \frac{1}{n} \sum_{i=1}^{n} L(Y_i, \gamma)$ *(constant function)*
for $t = 1$ to $T$ do
    for $i = 1$ to $n$ do
        $r_{i,t} \leftarrow -\ell_i'(f_{t-1}(X_i))$ *(pseudo-residuals)*
    end for
    $g_t \leftarrow$ base regressor from $\mathcal{R}$ for the training set $\{(X_i, r_{i,t})\}_{1 \le i \le n}$
    $w_t \leftarrow \arg\min_{w \in \mathbb{R}} \frac{1}{n} \sum_{i=1}^{n} \ell_i(f_{t-1}(X_i) + w g_t(X_i))$ *(line search)*
    $f_t = f_{t-1} + \nu w_t g_t$
end for
```

Output: $\operatorname{sign}(f_T)$ for classification, f_T for regression.

Figure: Pseudocode of Gradient Boosting (from M Sangnier - Introduction to Machine Learning, 2021)

XGBoost



```
Input: training set $\{(x_i, y_i)\}_{i=1}^{N}$, a differentiable loss function $L(y, F(x))$, a number of weak learners $M$ and a learning rate $\alpha$.
Algorithm:
  1. Initialize model with a constant value:
       $\hat{f}_{(0)}(x) = \arg\min_{\theta} \sum_{i=1}^{N} L(y_i, \theta).$
  2. For $m$ = 1 to $M$:
       1. Compute the 'gradients' and 'hessians':
            $\hat{g}_m(x_i) = \left[\frac{\partial L(y_i, f(x_i))}{\partial f(x_i)}\right]_{f(x)=\hat{f}_{(m-1)}(x)}.$
            $\hat{h}_m(x_i) = \left[\frac{\partial^2 L(y_i, f(x_i))}{\partial f(x_i)^2}\right]_{f(x)=\hat{f}_{(m-1)}(x)}.$
       2. Fit a base learner (or weak learner, e.g. tree) using the training set $\{x_i, -\frac{\hat{g}_m(x_i)}{\hat{h}_m(x_i)}\}_{i=1}^{N}$ by solving the optimization problem below:
            $\hat{\phi}_m = \arg\min_{\phi \in \Phi} \sum_{i=1}^{N} \frac{1}{2} \hat{h}_m(x_i) \left[-\frac{\hat{g}_m(x_i)}{\hat{h}_m(x_i)} - \phi(x_i)\right]^2.$
            $\hat{f}_m(x) = \alpha \hat{\phi}_m(x).$
       3. Update the model:
            $\hat{f}_{(m)}(x) = \hat{f}_{(m-1)}(x) + \hat{f}_m(x).$
  3. Output $\hat{f}(x) = \hat{f}_{(M)}(x) = \sum_{m=0}^{M} \hat{f}_m(x).$
```

Figure: Pseudocode of XGBoost (from https://en.wikipedia.org/wiki/XGBoost)