MUNI
ECON

# Artificial Intelligence in Finance

## Supervised learning - discrete outcomes part B

**Štefan Lyócsa**

Department of Finance, Faculty of Economics and Administration

November 21, 2024

# Outline for Section 1

# Introduction

The logistic regression is a **popular benchmark**:

- It is simple to interpret.
- It is fast to estimate - no tuning required.
- It leads to **nonlinearities** along and across features.
- You can still attempt to **enhance** the model via:
  - Feature transformations (although difficult to interpret see Mood (2010, [3]).
  - Feature interactions.
  - Bagging.

There are some **limitations** if many features are included:

- Estimation uncertainty - too many parameters.
- **Over-fitting** is likely.

# Introduction

Following the regularization approach for regression one can **adjust the logistic regression** as well:

- LASSO logistic regression (LLR).
- RIDGE logistic regression (RLR).
- Elastic Net logistic regression (ENLR).
- Complete subset logistic regression.

# Outline for Section 2

# LASSO logistic regression

Recall that the **log-likelihood to maximize** for logistic regression is given by:

$$\max_{\boldsymbol{\beta}} \rightarrow LL(\boldsymbol{\beta}) = \sum_{i=1}^{n} Y_i \left( \sum_{j=1}^{k} \beta_j X_{i,j} \right) - log \left( 1 + e^{\sum_{j=1}^{k} \beta_j X_{i,j}} \right) \quad (1)$$

After adding the **penalty term**, the expression becomes:

$$\begin{aligned} \max_{\boldsymbol{\beta}} \rightarrow LL(\boldsymbol{\beta}) \quad = \quad & \left[ \sum_{i=1}^{n} Y_i \left( \sum_{j=1}^{k} \beta_j X_{i,j} \right) - log \left( 1 + e^{\sum_{j=1}^{k} \beta_j X_{i,j}} \right) \right] \\ & -\lambda \sum_{j=1}^{k} |\beta_j| \end{aligned} \quad (2)$$

# LASSO logistic regression

As before the $\lambda$ parameter needs to be estimated $\rightarrow$ **cross-validation**:

$$\max_{\beta} \rightarrow LL(\beta) = \left[\sum_{i=1}^{n} Y_i \left(\sum_{j=1}^{k} \beta_j X_{i,j}\right) - log\left(1 + e^{\sum_{j=1}^{k} \beta_j X_{i,j}}\right)\right] - \lambda \sum_{j=1}^{k} |\beta_j| \quad (3)$$

**What** classification **accuracy** measure **to use**?

# LASSO logistic regression

**What** classification **accuracy** measure **to use**?

- **Deviance** (cross-entropy):

$$D_i = -2\left[log(\hat{p}_i)Y_i + log(1-\hat{p}_i)(1-Y_i)\right] \tag{4}$$

- AUC - does not require explicit threshold.
- **Custom based**:
  - Profit, revenue.
  - Balanced accuracy.
  - Precision (depends)....

# RIDGE logistic regression

Using the **penalty term** from RIDGE leads to:

$$\max_{\beta} \rightarrow LL(\beta) = \left[ \sum_{i=1}^{n} Y_i \left( \sum_{j=1}^{k} \beta_j X_{i,j} \right) - log \left( 1 + e^{\sum_{j=1}^{k} \beta_j X_{i,j}} \right) \right] - \lambda \sum_{j=1}^{k} \beta_j^2 \quad (5)$$

# Elastic Net logistic regression

Combining the LASSO and RIDGE **penalty terms** leads to:

$$\max_{\boldsymbol{\beta}} \rightarrow LL(\boldsymbol{\beta}) = \left[\sum_{i=1}^{n} Y_i \left(\sum_{j=1}^{k} \beta_j X_{i,j}\right) - log\left(1 + e^{\sum_{j=1}^{k} \beta_j X_{i,j}}\right)\right] - \lambda\left[\frac{1-\alpha}{2}\sum_{j=1}^{k}\beta_j^2 + \alpha\sum_{j=1}^{k}|\beta_j|\right] \quad (6)$$

Apart from $\lambda$ we need to estimate or assume $\alpha \in (0, 1)$ as well.

# Titanic dataset

How are regularization methods doing in our datasets?
**Titanic** dataset:

- Predicting the survival of a passenger.
- No imbalance procedures.
- Threshold set to 0.427 the proportion of survived passengers.
- 1046 obs. originally, 836 in mildly unbalanced training and 210 in testing dataset with 5 **features**.

| Models | Sensitivity | Specificity | Balanced Acc |
|---|---|---|---|
| LR | 0.828 | 0.743 | 0.786 |
| LLR | 0.829 | 0.743 | 0.786 |
| RLR | 0.814 | 0.750 | 0.782 |
| ENLR | 0.829 | 0.743 | 0.786 |

# P2P Zopa dataset

How are regularization methods doing in our datasets?
**Zopa** dataset:

- Predicting the default of a loan.
- 8% bad loans $\rightarrow$ unbalanced data!
- We have enough 20000 observations $\rightarrow$ under-sample the majority.
- 2609 in training and 653 in testing data with 160 **features**.

| Models | Sensitivity | Specificity | Balanced Acc |
|---|---|---|---|
| LR | 0.73 | 0.75 | 0.74 |
| LLR | 0.68 | 0.82 | 0.75 |
| RLR | 0.70 | 0.76 | 0.73 |
| ENLR | 0.68 | 0.82 | 0.75 |

# Firm defaults

How are regularization methods doing in our datasets?
**Firm defaults** dataset:

- Predicting the default of a firm in next period.
- Random under-sampling of majority and random over-sampling of minority.
- 6819 obs. originally, 1400 in balanced training and 1364 in testing dataset and 91 **features**.

| Models | Sensitivity | Specificity | Balanced Acc |
|---|---|---|---|
| LR | 0.73 | 0.87 | 0.80 |
| LLR | 0.82 | 0.86 | 0.84 |
| RLR | 0.82 | 0.85 | 0.84 |
| ENLR | 0.82 | 0.86 | 0.84 |

# Outline for Section 3

# Bagging revisited

**Bagging** is based on estimating a model on a **bootstrapped** sample. That is, we create a *new* dataset by randomly selecting (with replacement) observations from the original dataset. Repeating the bootstrapping (sampling) $\rightarrow$ estimation $\rightarrow$ prediction. Sampling $B$ times, leads to a **distribution** of predictions for every testing observation $i$. Specifically, i.e. $\hat{p}_{i,b}, i = 1, 2, ....; b = 1, 2, ..., B$.

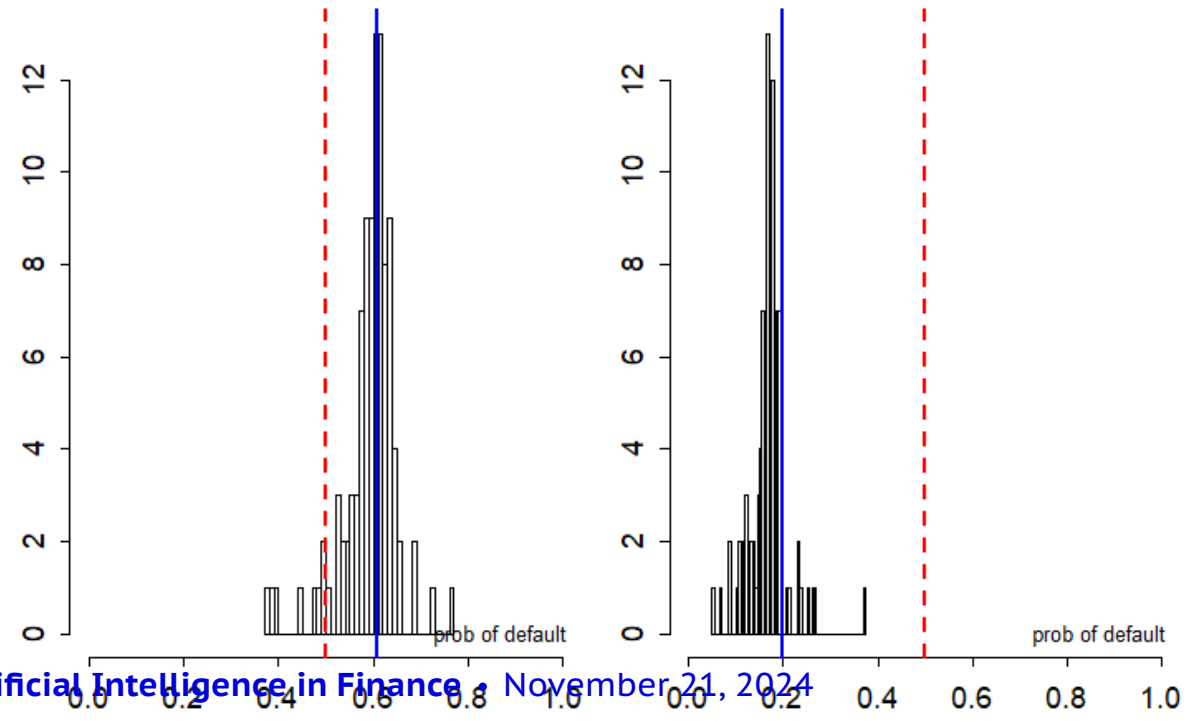
Why is a distribution of the predicted probability in **classification** tasks useful?

# Bagging revisited

Why is a distribution of the predicted probability in **classification** tasks useful?

- Bagging can improve prediction accuracy - **averaging many over-fitted models**.
- It gives us an estimate of **confidence in our predictions**.

The P2P loan default:



# P2P loan

As an investor, you invest if $\hat{Y}_i = 0$, which happens on a balanced sample if $\hat{p}_i < 0.5$: Using LASSO you face:

| | Observed $Y_i = 0$ | Observed $Y_i = 1$ |
|---|---|---|
| Predicted $\hat{Y}_i = 0$ | 262 | 105 |
| Predicted $\hat{Y}_i = 1$ | 59 | 227 |

This leads to a 71.39% success across 367 loans. Let's be more conservative and:

- Invest **only** into loans, where the $95^{th}$ quantile of the predicted probability is below the threshold, i.e. $\sum_{b=1}^{B} I(\hat{p}_{i,b} > 0.5) \leq 0.95$.

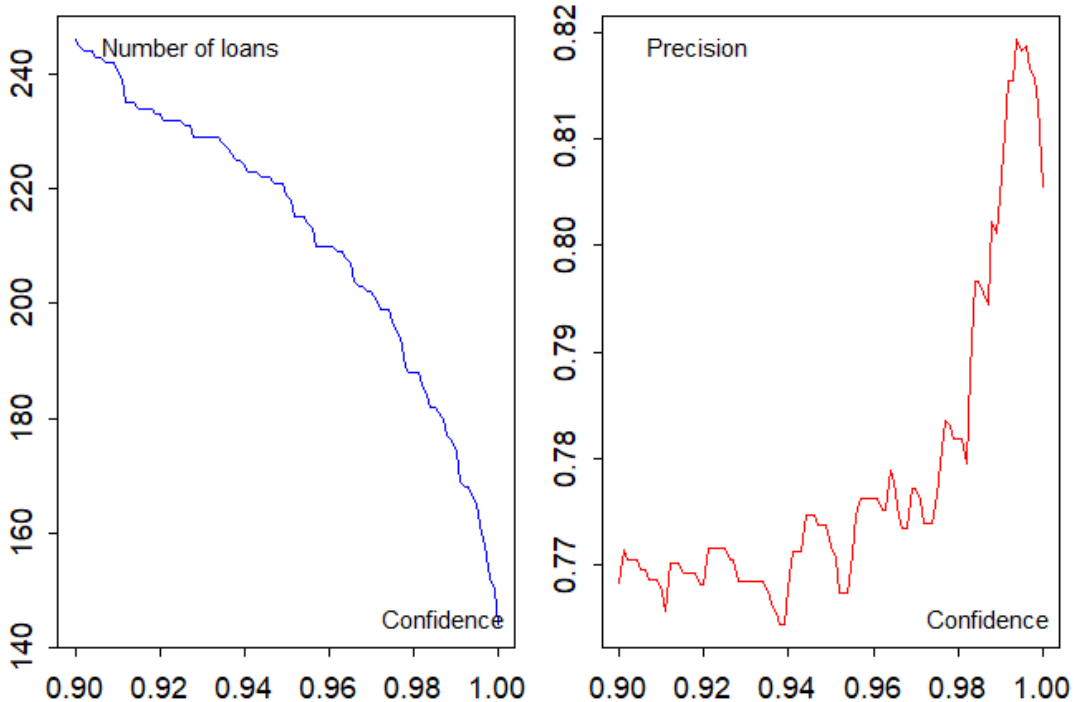
The confusion matrix investor faces is:

| | Observed $Y_i = 0$ | Observed $Y_i = 1$ |
|---|---|---|
| Predicted $\hat{Y}_i = 0$ | 169 | 50 |
| Predicted $\hat{Y}_i = 1$ | 0 | 0 |

The success goas to 77.16% but **only** across 219 loans.

# P2P loan

The higher the confidence the higher the expected precision (identify a default correctly) - **there is a price we pay**.




Does that mean that loan market is **inefficient** see Lyócsa and Výrost (2018, [2])? Or that we are going to have **higher profits**?

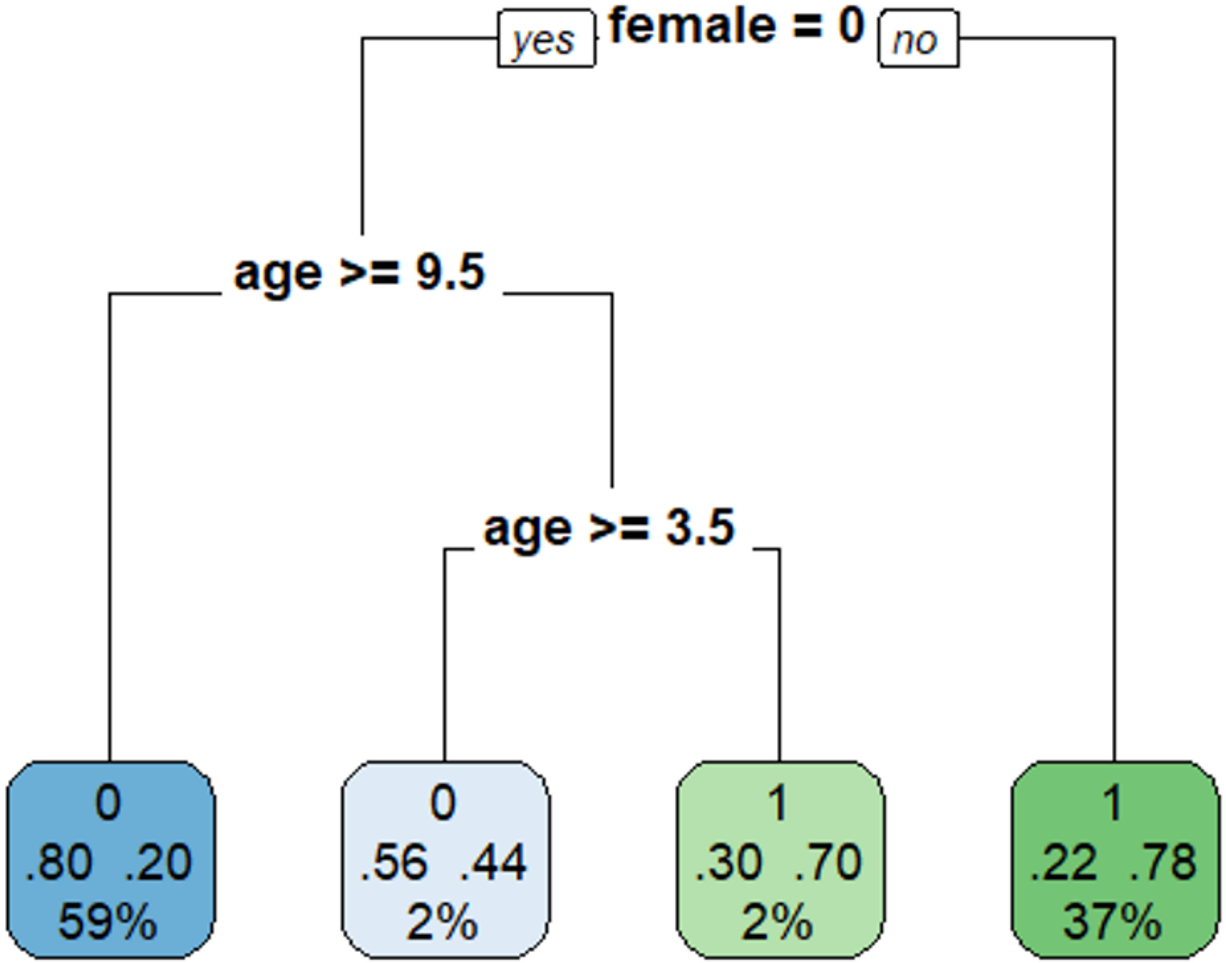
# Outline for Section 4

# Introduction

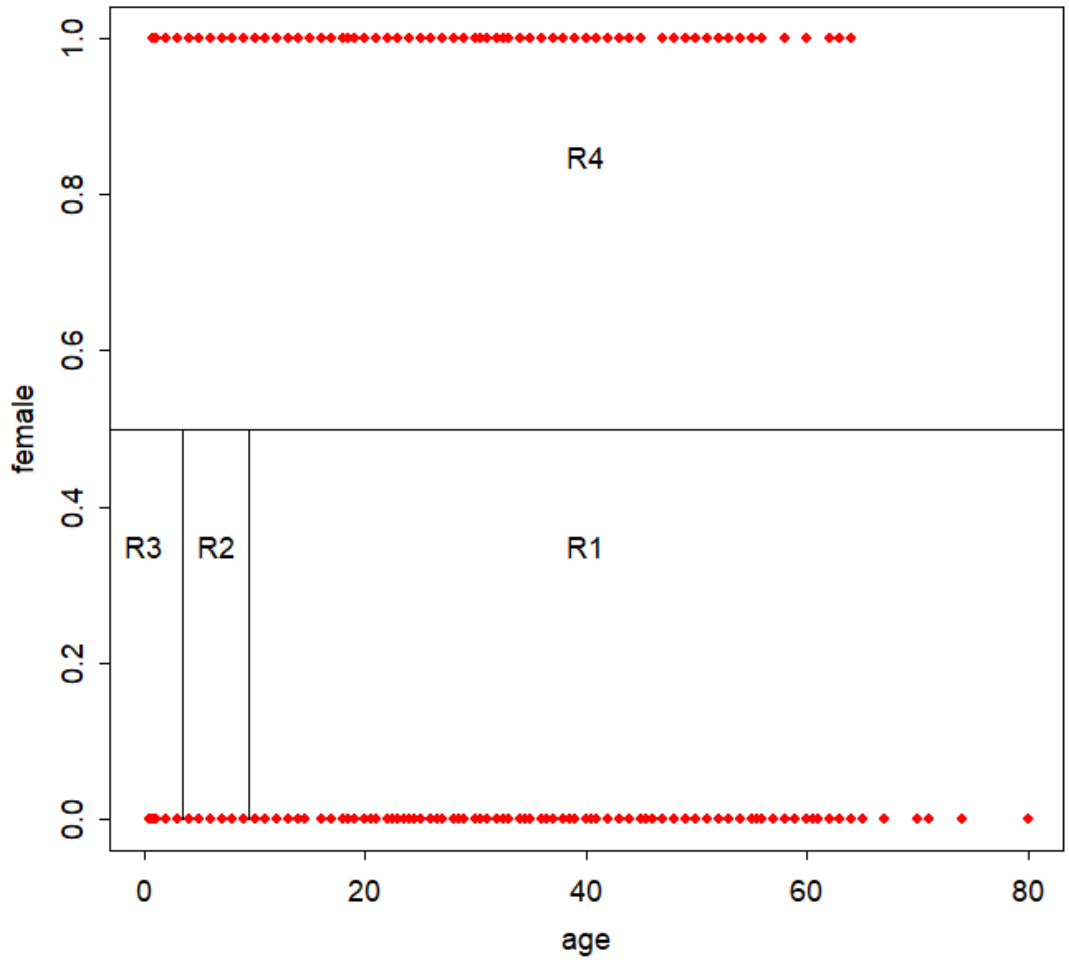
- Involve **stratifying a feature space** into simpler regions - subsets of data.
- Prediction for a specific observation from the testing sample is usually the **most occurring class** in the **terminal region**.
- Simple decision trees can be improved via:
  - pre-prunning.
  - post-prunning.
  - bagging.
  - bagging and randomization - random forest.
  - boosting.

# Example

Shallow tree

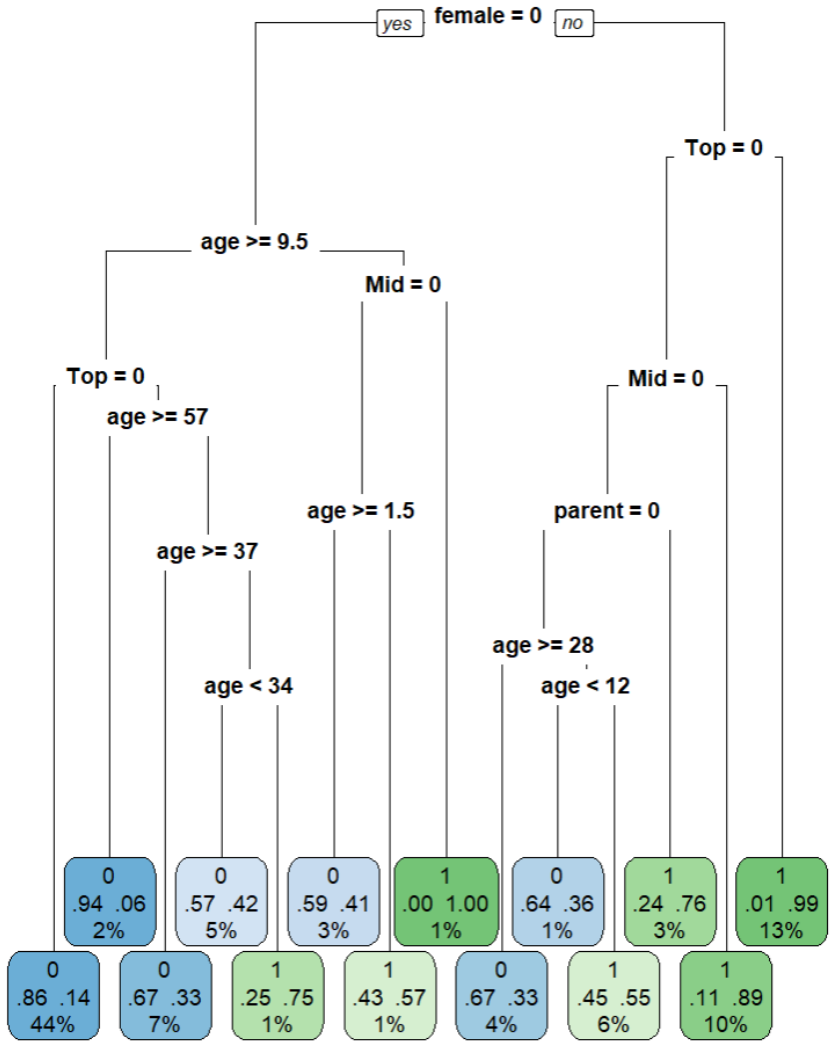


# Example



# Example

Deeper tree



# Splitting the nodes

How do we **find splitting points**?

To split a node ($t$), measures employed are based on **degree of impurity** of a node(s). Highest impurity is (0.5, 0.5), lowest is (0.0, 1.0) or (1.0, 0.0), i.e. the smaller the degree of impurity, the more **skewed** the **class distribution** (Tan et al., 2016, [4]).

Let $p(c|t)$ denote the proportion of observations of class $c$ in node $t$:

- **Classification error** for given node ($t$):

$$I_{CE} = 1 - \max_c p(c|t) \tag{7}$$

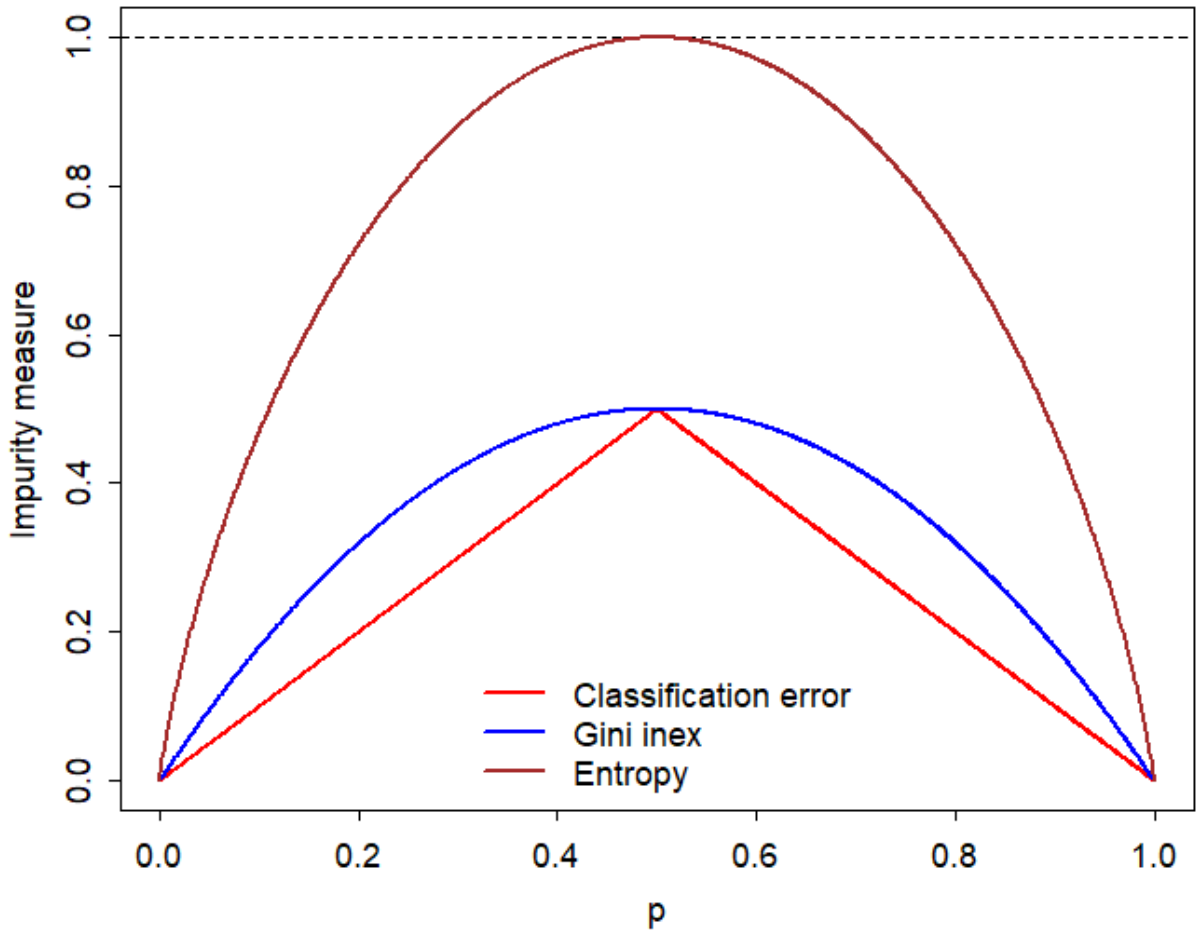
- **Gini index** (we will use) for node ($t$):

$$I_{CE} = 1 - \sum_c p(c|t)^2 \tag{8}$$

- **Entropy** for node ($t$):

$$I_{CE} = -\sum_c p(c|t) log_2 p(c|t) \tag{9}$$

# Splitting the nodes



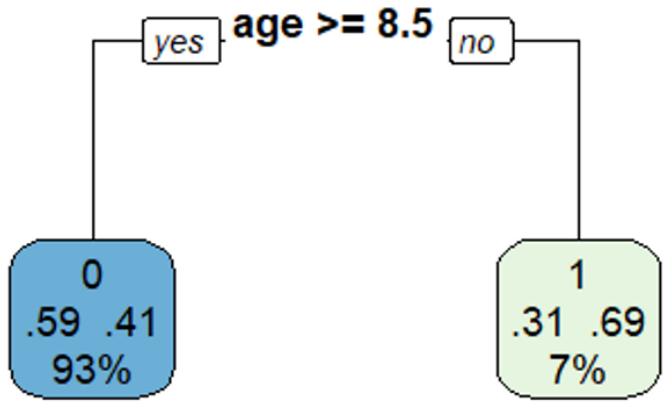
# Splitting the nodes

Similarly as for regression trees, in **decission trees** the ultimate goal is to find terminal regions - $R_1, R_2, ..., R_J$ that minimize some loss functions (James et al., 2013, [1]). As before, a greedy approach is used, the **recursive binary splitting** algorithm.

A split is decided by **comparing the degree of impurity of the parent node with child nodes**. Let $k$ be number of classes (2 for **binary splits**), $N_j$ number of observations in child node $j$ and $I(t_j)$ the impurity measure of child note $j$. The goal is to find a split that maximizes:

$$\Delta = I(t) - \sum_{j=1}^{k} I(t_j) \frac{N_j}{N} \tag{10}$$

# Splitting the nodes - Example




$$\Delta = I(t) - \sum_{j=1}^{k} I(t_j)\frac{N_j}{N}$$

$$0.489 - \frac{0.429 \times 61}{836} - \frac{0.482 \times 775}{836} = 0.0107$$

# Prunning

The **pre-pruning** approach (early stopping rules):

- Limit the **maximum depth** of the tree.
- Set a **minimum number** needed to consider a **split**.
- Set a **minimum number** of observations in a terminal **region** (bucket size).

The **post-pruning** approach (bottom-up from a deep tree):

- Introducing penalization for too complex trees.

# Outline for Section 5

# Bagging for decision trees

Recall that bagging is based on the idea that averaging unbiased but potentially over-fitted model's predictions will reduce the out-of-sample error.

Let's have data denoted as $Z$ with $i = 1, 2, \ldots, N$ observations. In a non-parametric bootstrap:

1. Each observation in $Z$ has the same probability of being selected.
2. **Randomly** select $N$ observations from $Z$, **with replacement** and create a new dataset $Z^*$.
3. Estimate a given model/statistics using the dataset $Z^*$.
4. Repeat step 2 and 3 until we have $B$ models/statistics.

# Bagging: Introduction

Using data from the training sample, for each bootstrap sample you estimate a complete (deep) tree $T^{*b}$ and generate a corresponding forecast for observation $i$ that belongs to the testing dataset, $\hat{p}_i^{*,b} \in (0, 1)$. The prediction using **bagging** is given by a simple average:

$$\hat{p}_i = B^{-1} \sum_{b=1}^{B} \hat{p}_i^{*,b} \tag{11}$$

This approach should work well for deep trees - why? Predictions from such trees have **low bias but high variance**.

# Titanic dataset

How are regularization methods doing in our datasets?
**Titanic** dataset:

- Predicting the survival of a passenger.
- No imbalance procedures.
- Threshold set to 0.427 the proportion of survived passengers.
- 1046 obs. originally, 836 in mildly unbalanced training and 210 in testing dataset with 5 **features**.

| Models | Sensitivity | Specificity | Balanced Acc |
|---|---|---|---|
| LR | 0.828 | 0.743 | 0.786 |
| LLR | 0.829 | 0.743 | 0.786 |
| RLR | 0.814 | 0.750 | 0.782 |
| ENLR | 0.829 | 0.743 | 0.786 |
| DC-BAG | 0.757 | 0.821 | 0.789 |

# P2P Zopa dataset

How are regularization methods doing in our datasets?
**Zopa** dataset:

- Predicting the default of a loan.
- 8% bad loans $\rightarrow$ unbalanced data!
- We have enough 20000 observations $\rightarrow$ under-sample the majority.
- 2609 in training and 653 in testing data with 160 **features**.

| Models | Sensitivity | Specificity | Balanced Acc |
|---|---|---|---|
| LR | 0.73 | 0.75 | 0.74 |
| LLR | 0.68 | 0.82 | 0.75 |
| RLR | 0.70 | 0.76 | 0.73 |
| ENLR | 0.68 | 0.82 | 0.75 |
| DC-BAG | 0.80 | 0.75 | 0.78 |

# Firm defaults

How are regularization methods doing in our datasets?
**Firm defaults** dataset:

- Predicting the default of a firm in next period.
- Random under-sampling of majority and random over-sampling of minority.
- 6819 obs. originally, 1400 in balanced training and 1364 in testing dataset and 91 **features**.

| Models | Sensitivity | Specificity | Balanced Acc |
|---|---|---|---|
| LR | 0.73 | 0.87 | 0.80 |
| LLR | 0.82 | 0.86 | 0.84 |
| RLR | 0.82 | 0.85 | 0.84 |
| ENLR | 0.82 | 0.86 | 0.84 |
| DC-BAG | 0.82 | 0.90 | 0.86 |

# Outline for Section 6

# Random classification forest

Similarly as with random forest for regressions, random forest combines bagging with a random selection of features to consider at each split $\rightarrow$ **decorrelated trees**. Key parameters to hyper-tune:

- Depth of the trees (should be deep).
- Number of random features selected at each split.
- Number of trees.

Other pre-prunning parameters can be tuned as well.

# Titanic dataset

How are regularization methods doing in our datasets?
**Titanic** dataset:

- Predicting the survival of a passenger.
- No imbalance procedures.
- Threshold set to 0.427 the proportion of survived passengers.
- 1046 obs. originally, 836 in mildly unbalanced training and 210 in testing dataset with 5 **features**.

| Models | Sensitivity | Specificity | Balanced Acc |
|---|---|---|---|
| LR | 0.828 | 0.743 | 0.786 |
| LLR | 0.829 | 0.743 | 0.786 |
| RLR | 0.814 | 0.750 | 0.782 |
| ENLR | 0.829 | 0.743 | 0.786 |
| DC-BAG | 0.757 | 0.821 | 0.789 |
| RF | 0.814 | 0.778 | 0.796 |

# P2P Zopa dataset

How are regularization methods doing in our datasets?
**Zopa** dataset:

- Predicting the default of a loan.
- 8% bad loans $\rightarrow$ unbalanced data!
- We have enough 20000 observations $\rightarrow$ under-sample the majority.
- 2609 in training and 653 in testing data with 160 **features**.

| Models | Sensitivity | Specificity | Balanced Acc |
|---|---|---|---|
| LR | 0.73 | 0.75 | 0.74 |
| LLR | 0.68 | 0.82 | 0.75 |
| RLR | 0.70 | 0.76 | 0.73 |
| ENLR | 0.68 | 0.82 | 0.75 |
| DC-BAG | 0.80 | 0.75 | 0.78 |
| RF | 0.80 | 0.72 | 0.76 |

# Firm defaults

How are regularization methods doing in our datasets?
**Firm defaults** dataset:

- Predicting the default of a firm in next period.
- Random under-sampling of majority and random over-sampling of minority.
- 6819 obs. originally, 1400 in balanced training and 1364 in testing dataset and 91 **features**.

| Models | Sensitivity | Specificity | Balanced Acc |
|---|---|---|---|
| LR | 0.73 | 0.87 | 0.80 |
| LLR | 0.82 | 0.86 | 0.84 |
| RLR | 0.82 | 0.85 | 0.84 |
| ENLR | 0.82 | 0.86 | 0.84 |
| RF | 0.78 | 0.93 | 0.85 |

[1] Gareth James et al. *An introduction to statistical learning*. Springer, 2013.

[2] Štefan Lyócsa and Tomáš Výrost. “To bet or not to bet: a reality check for tennis betting market efficiency”. In: *Applied Economics* 50.20 (2018), pp. 2251–2272.

[3] Carina Mood. “Logistic regression: Why we cannot do what we think we can do, and what we can do about it”. In: *European sociological review* 26.1 (2010), pp. 67–82.

[4] Pang-Ning Tan, Michael Steinbach, and Vipin Kumar. *Introduction to data mining*. Pearson Education India, 2016.

MUNI
ECON

# Artificial Intelligence in Finance

## Supervised learning - discrete outcomes part B

**Štefan Lyócsa**

Department of Finance, Faculty of Economics and Administration

November 21, 2024

MASARYK
UNIVERSITY