MUNI
ECON

# Artificial Intelligence in Finance

## Supervised learning - continuous outcome part C

**Štefan Lyócsa**

Department of Finance, Faculty of Economics and Administration

October 23, 2024

# Outline for Section 1

# Introduction

There are good reasons why we still cover Linear Regression models. They are often reasonable to start with:

- It is **simple** to interpret.
- It is **fast** to estimate.
- It can be enhanced by **nonlinear transformations** of features:
  - You can add **interaction terms**.
  - You can add **dummy variables**.
  - You can use **Box-Cox** transformations.

It appears that other class of models tends to be more successful (kaggle competitions):

**Tree-Based Methods**

# Introduction

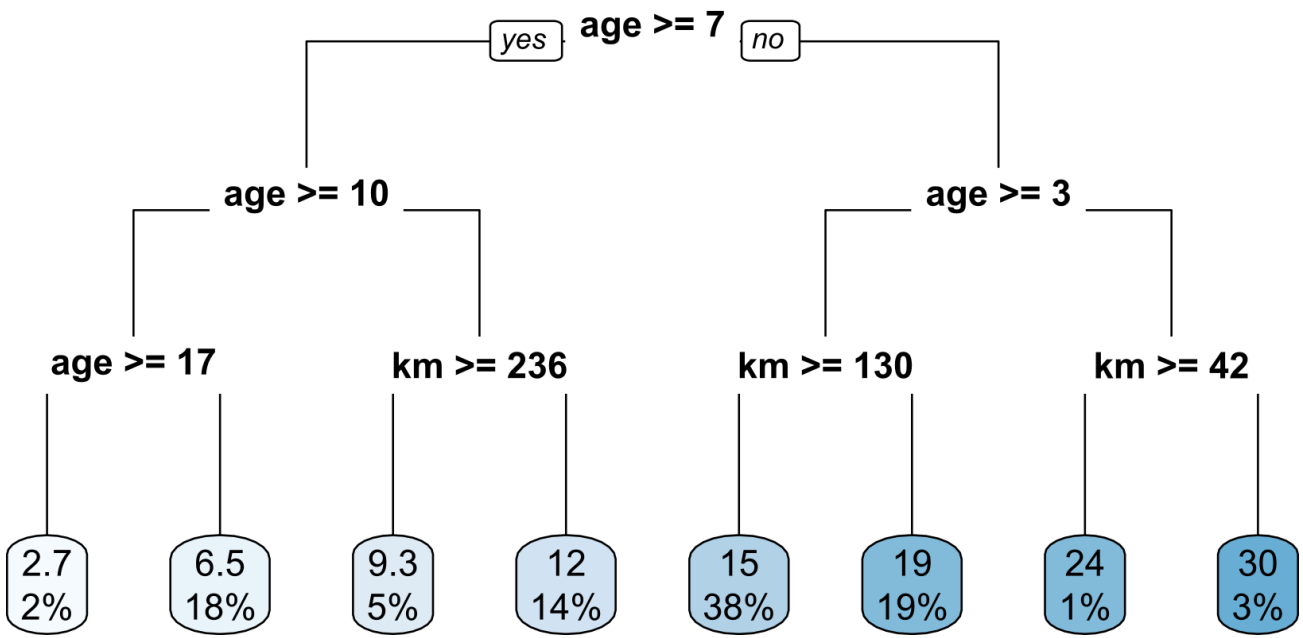
**Tree-Based Methods**:

- Involve **stratifying a feature space** into simpler regions - subsets of data.
- Based on the features, an observation from a testing sample is assigned to one of the regions. The **prediction** is than equal to the average value of training observations in that region.
- Simple decision trees can be improved via:
  - pre-prunning.
  - post-prunning.
  - bagging.
  - bagging and randomization - random forest.
  - boosting.

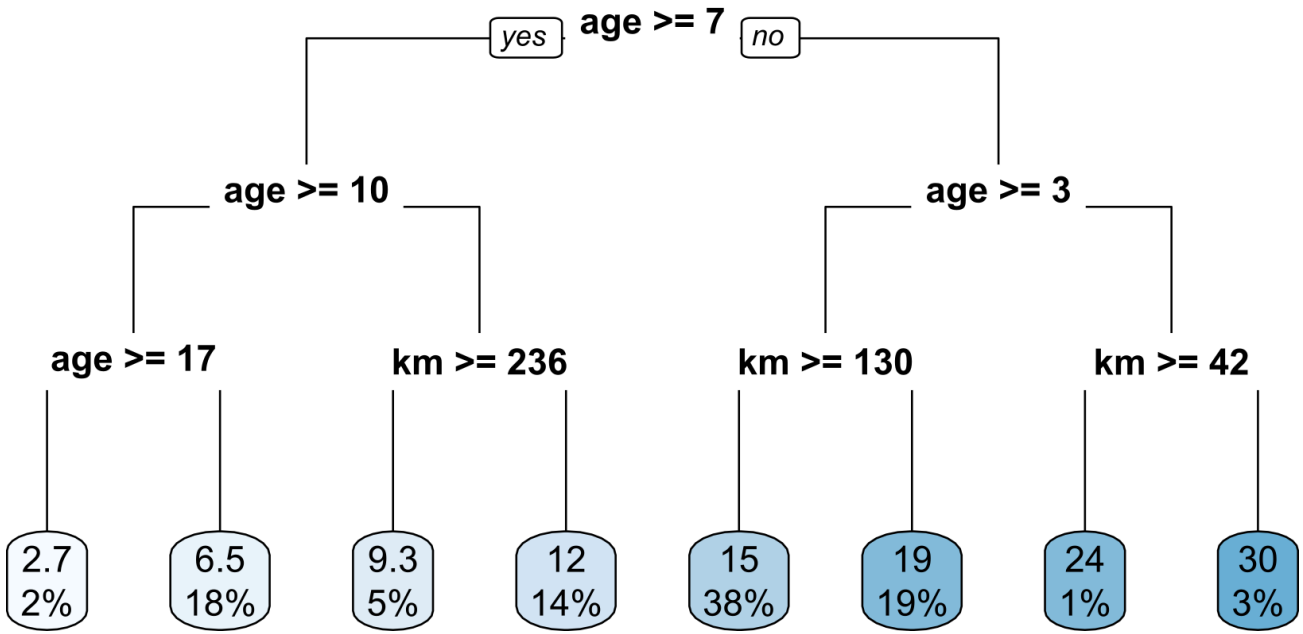
# Outline for Section 2

# Decision trees: Example

Let's have a sample of 900 used cars with the target variable being the **price** $Y_i, i = 1, 2, \ldots$. Training sample consists of 720 and testing of 180 cars. Consider two features, **age** (in terms of years) and **km** (which is the milage). A simplified decision-tree might look like:

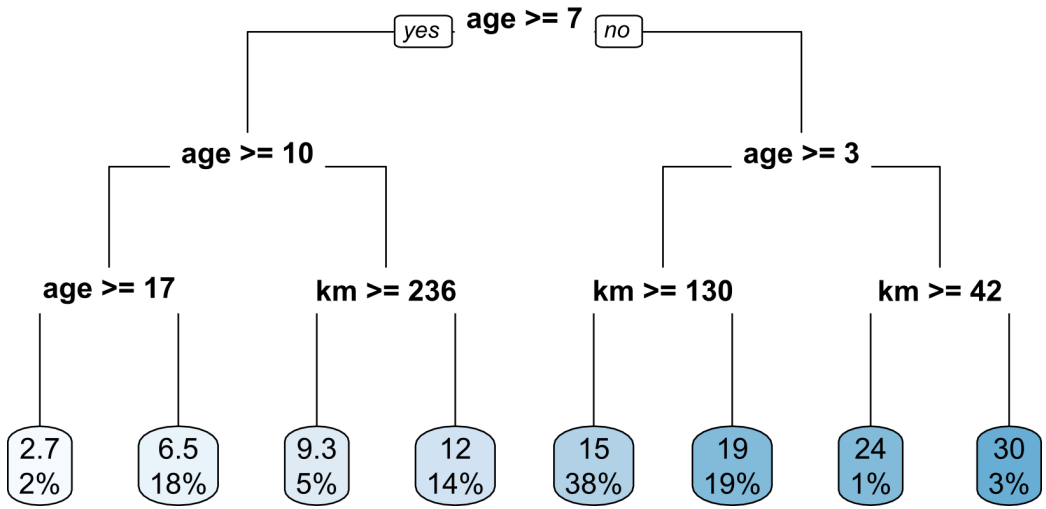


# Decision trees: Terminology




- Decision tree, e.g. tree - a series of splitting rules.
- Terminal node, e.g. leave, terminal region.
- Internal node.
- Branch.

# Decision trees: Terminology

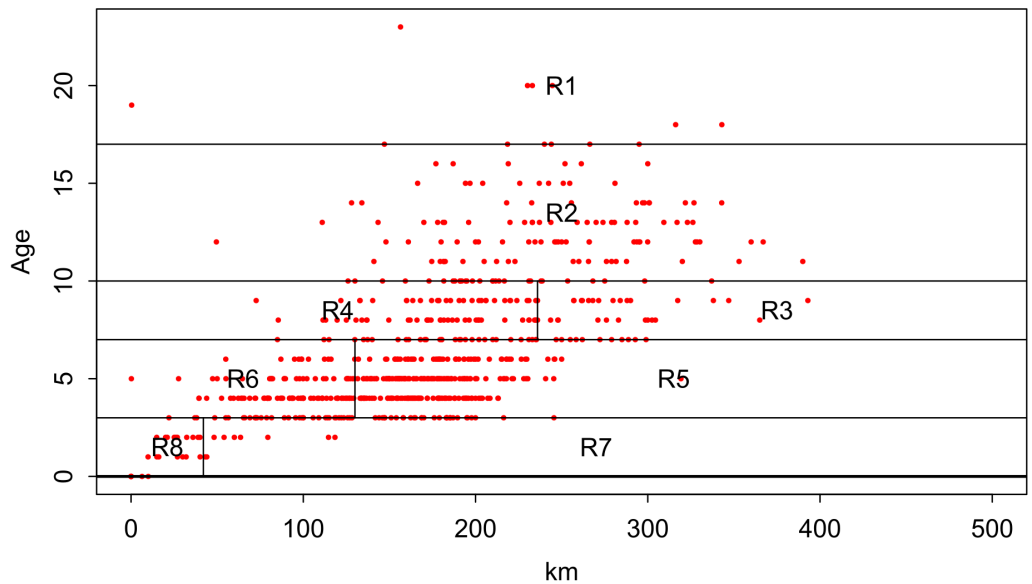



The tree creates **terminal regions** of the **feature space**:

- $R_1 = \{X | \text{age} \geq 17\}$
- $R_2 = \{X | \text{age} < 17 \land \text{age} \geq 10\}$
- $R_3 = \{X | \text{age} < 10 \land \text{age} \geq 7 \land \text{km} \geq 236\}$
- $R_4 = \{X | \text{age} < 10 \land \text{age} \geq 7 \land \text{km} < 236\}$
- $R_5 = \{X | \text{age} \geq 3 \land \text{age} < 7 \land \text{km} \geq 130\}$
- $R_6 = \{X | \text{age} \geq 3 \land \text{age} < 7 \land \text{km} < 130\}$
- $R_7 = \{X | \text{age} < 3 \land \text{km} \geq 42\}$
- $R_8 = \{X | \text{age} < 3 \land \text{km} < 42\}$

# Decision trees: Terminology

Segmentation of the feature space:




**Prediction**:
Observations that fall into a given region have the same prediction, equal to the mean of the target values from the training sample.

# Decision trees: Reverse binary splitting

How do we **find splitting points**? In case of a **regression tree**, the ultimate goal is to find terminal nodes - regions - $R_1, R_2, \ldots, R_J$ that minimize RSS (James et al., 2013, [2]):

$$\sum_{j=1}^{J} \sum_{i \in R_j} (Y_i - \bar{Y}_{R_j})^2 \tag{1}$$

Considering all possible values $\rightarrow$ computationally infeasible approach. Instead, the **recursive binary splitting** is a greedy algorithm that only looks at the current best split at a time.

# Decision trees: Reverse binary splitting

Let $X_{i,k}$ be the $k^{th}, k = 1, 2, ..., p$ feature of the $i^{th}, i = 1, 2, ..., N$ observation and $X_{s,k}$ a cut-point, that splits the feature space into $\{X_k | X_{i,k} < X_{s,k}\}$ and $\{X_k | X_{i,k} \geq X_{s,k}\}$. Given a cut-point, we can divide a feature space to:

$$R_1(s,k) = \{\boldsymbol{X} | X_k < X_{s,k}\} \text{ and } R_2(s,k) = \{\boldsymbol{X} | X_k \geq X_{s,k}\} \tag{2}$$

# Decision trees: Reverse binary splitting

1. Start at the top of the tree (no split).
2. We seek a feature $k$ and a cut-point $s$ that minimizes:

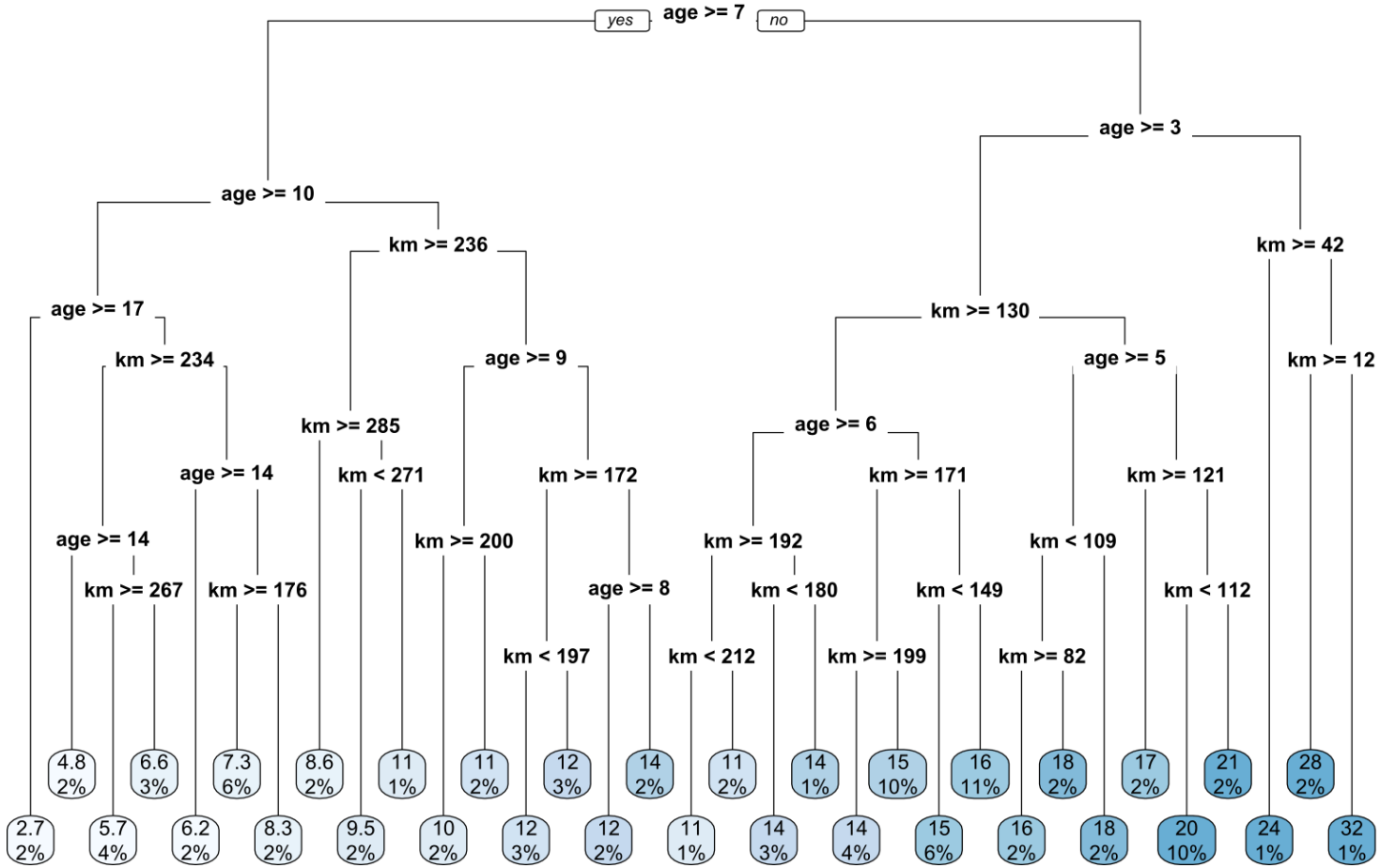
$$\sum_{i:\boldsymbol{X}_i \in R_1(j,s)}^{p} (Y_i - \bar{Y}_{R_1})^2 + \sum_{i:\boldsymbol{X}_i \in R_2(j,s)}^{p} (Y_i - \bar{Y}_{R_2})^2 \tag{3}$$

3. Repeat Step 2 for each sub-tree until a **stopping criterion** is reached. This means that we have a set of terminal regions $R_1, R_2, \ldots, R_J$

We predict the target variable using the mean of the training observations from the terminal region where that test observation belongs.

# Decision trees: Example

Deeper tree - max depth of 6. Prediction on the testing sample led to: MSE = 6.99, MAE = 1.95.



# Outline for Section 3

# Pre-prunning: Introduction

Decision trees are **nice to interpret**, but tend to over-fit the data and do not perform very well in an out-of-sample context. Several strategies attempt to address the over-fitting issue.

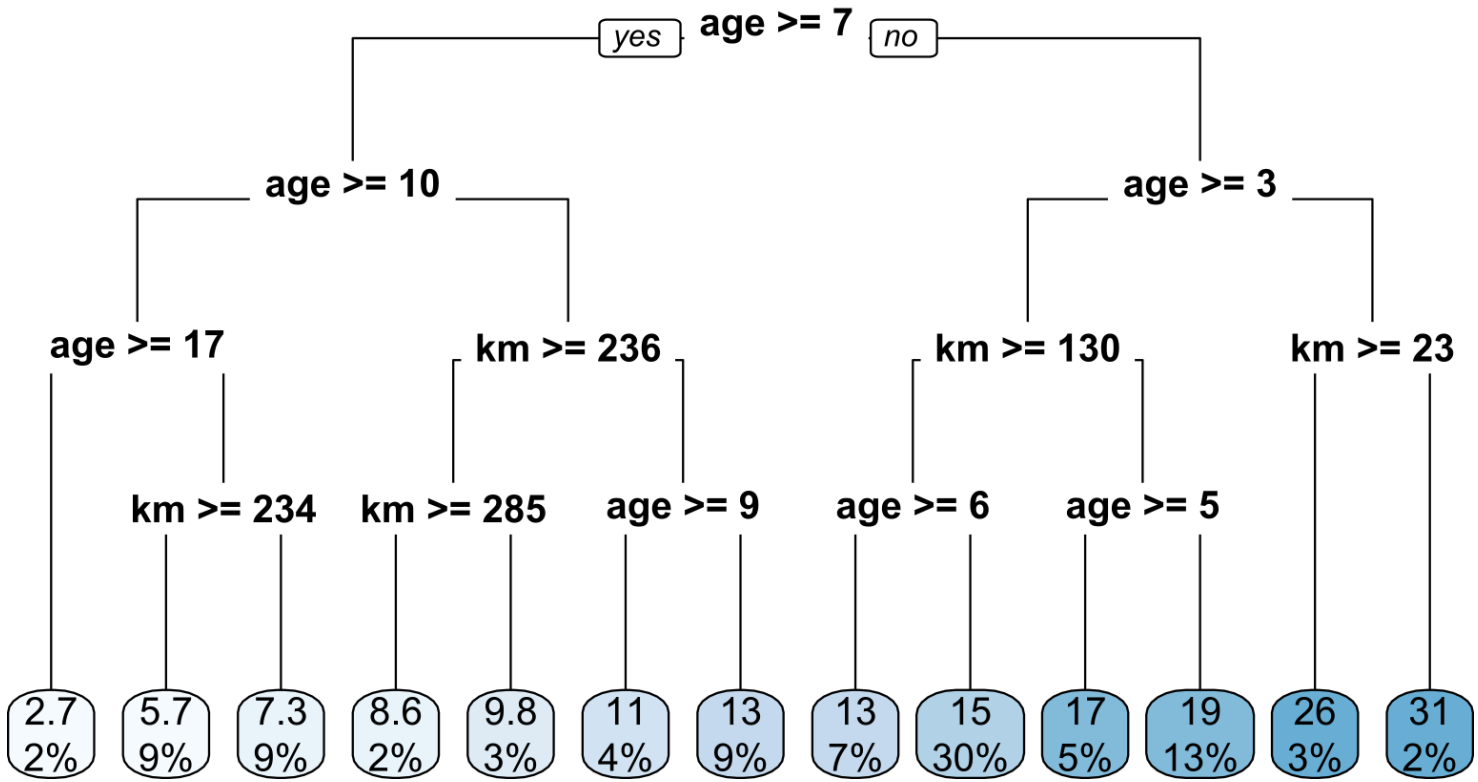
The **pre-pruning** approach:

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

How can we estimate these parameters?

# Pre-prunning: Example

Shallow tree - max depth of 4, minimum split size at 25 and minimum bucket size at 12. Prediction on the testing sample led to: MSE = 7.04, MAE = 2.02.



# Outline for Section 4

# Pruning: Post-pruning

Let denote a **complete tree** (no constraints on the construction of the tree, e.g. maximum depth) as $T_0$ (e.g. $T_\infty$ is a tree without a single split). For each $\alpha \geq 0$, there exists a tree $T \subset T_0$ that **minimizes**:

$$\sum_{j=1}^{|J|} \sum_{i:\boldsymbol{X}_i \in R_j} (Y_i - \bar{Y}_{R_j})^2 + \alpha |J| \tag{4}$$

, where $|J|$ denotes the number of terminal nodes of the tree $T$.
The tuning parameter $\alpha$ presents a hyperparameter that introduces a penalization for tree's complexity.

- Note, when $\alpha = 0$ we have a complete tree.
- If you increase $\alpha$ a lower value of the expression tends to be achieved after **removing a split**, i.e. **pruning**.

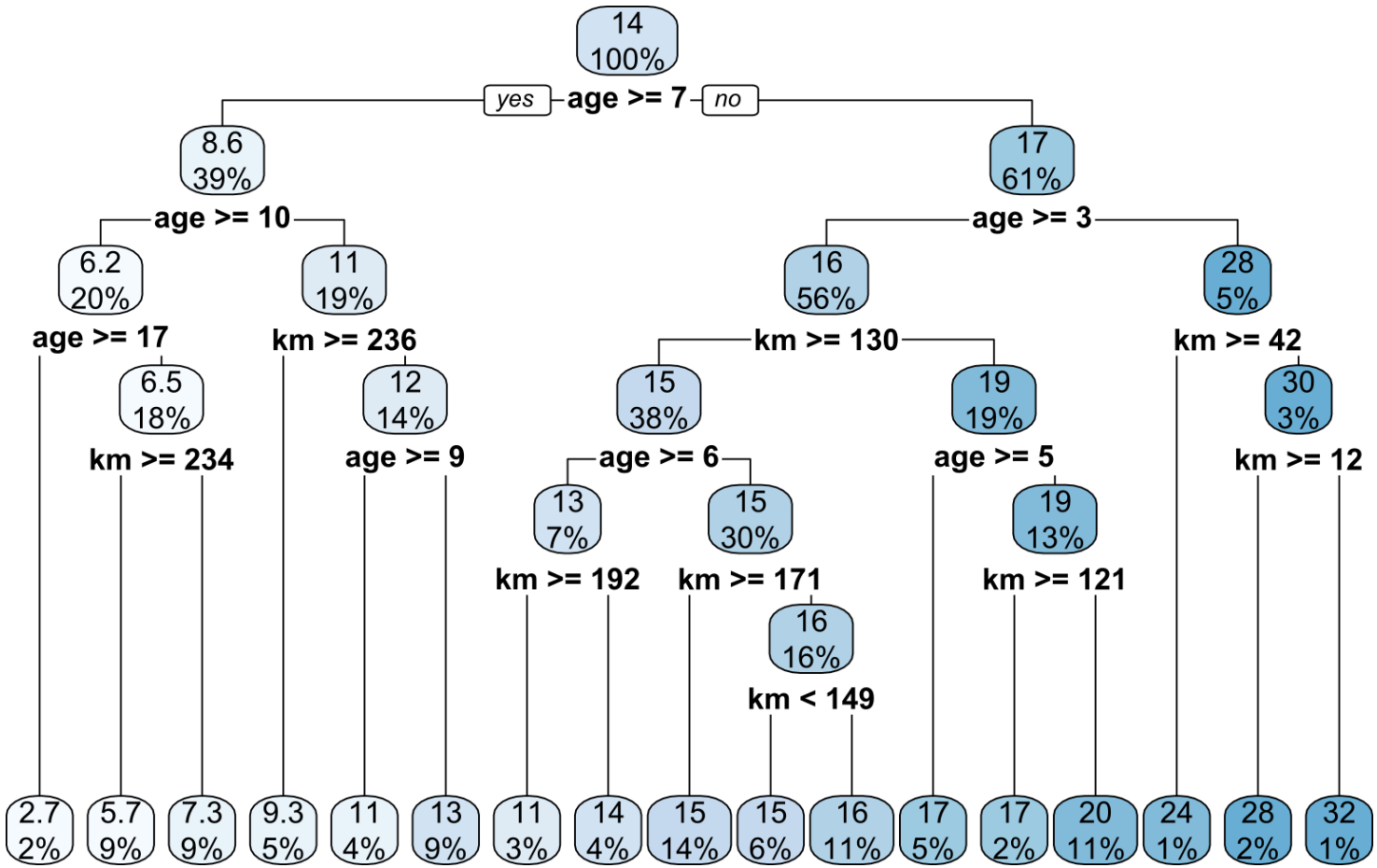
# Pruning: Post-pruning

**Cost complexity pruning** algorithm:

1. Grow a complete tree via recursive binary splitting (or other methods).
2. Find a sequence of best sub-trees for different values of $\alpha$.
3. Use k-fold cross-validation to find the optimum $\alpha$.
4. Estimate a sub-tree on the full training sample using the optimum $\alpha$.

# Pruning: Example

A 10-fold CV led to a pruned-tree with $\alpha = 0.001602364$. Prediction on the testing sample led to: MSE = 6.59, MAE = 1.96.



# Outline for Section 5

# Comparison

A **linear regression** assumes a model of a form:

$$\hat{Y}_i = f(\boldsymbol{X}_i) = \beta_0 + \sum_{k=1}^{p} \beta_k X_{i,k} \tag{5}$$

A **regression tree** assumes a model of the form:

$$\hat{Y}_i = f(\boldsymbol{X}_i) = \sum_{j=1}^{J} E[Y|\boldsymbol{X}_i \in R_j] I(\boldsymbol{X}_i \in R_j) \tag{6}$$

# Example

We also fit a linear regression with estimated coefficients:

$$price_i = 25.6 - 0.033 \times km_i - 0.927 \times age_i + \hat{u}_i \tag{7}$$

The $R^2 = 77.56\%$ and the prediction on the testing sample led to: MSE = 6.96, MAE = 2.12.

Several improvements to decision trees exists: bagging, random forest and boosting.

- bagging,
- random forest,
- boosting.

# Outline for Section 6

# Bagging: Introduction

**Re-sampling** - repeatedly drawing samples from a (training) dataset.

- Cross-validation (you already know about).
- **Bootstrapping** (Efron 1979, [1]) involves a random re-sample from the original dataset in order to create a new dataset. Depending on the type of data (cross-section, time-series, spatial,...).

# Bagging: Introduction

Let's have a dataset $Z$ with $N$ 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.

For time-series you need to bootstrap in a different way:

- **fixed block length** bootstrap (e.g. Politis et al., 1989, [4]),
- **stationary** bootstrap of Politis and Romano (1994, [3]).

# Bagging: Introduction

**Bagging** is a **general-purpose** bootstrap aggregation **technique** that exploits the fact that **averaging reduces variance**.

Using data from the training sample, **for each** bootstrap sample you estimate a complete (deep) tree $T^{*b}$ and generate a corresponding forecast $\hat{f}^{*,b}$. The prediction using **bagging** is given by a (some) average across trees, i.e.:

$$\hat{f}_{bag} = B^{-1} \sum_{b=1}^{B} \hat{f}^{*b} \tag{8}$$

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

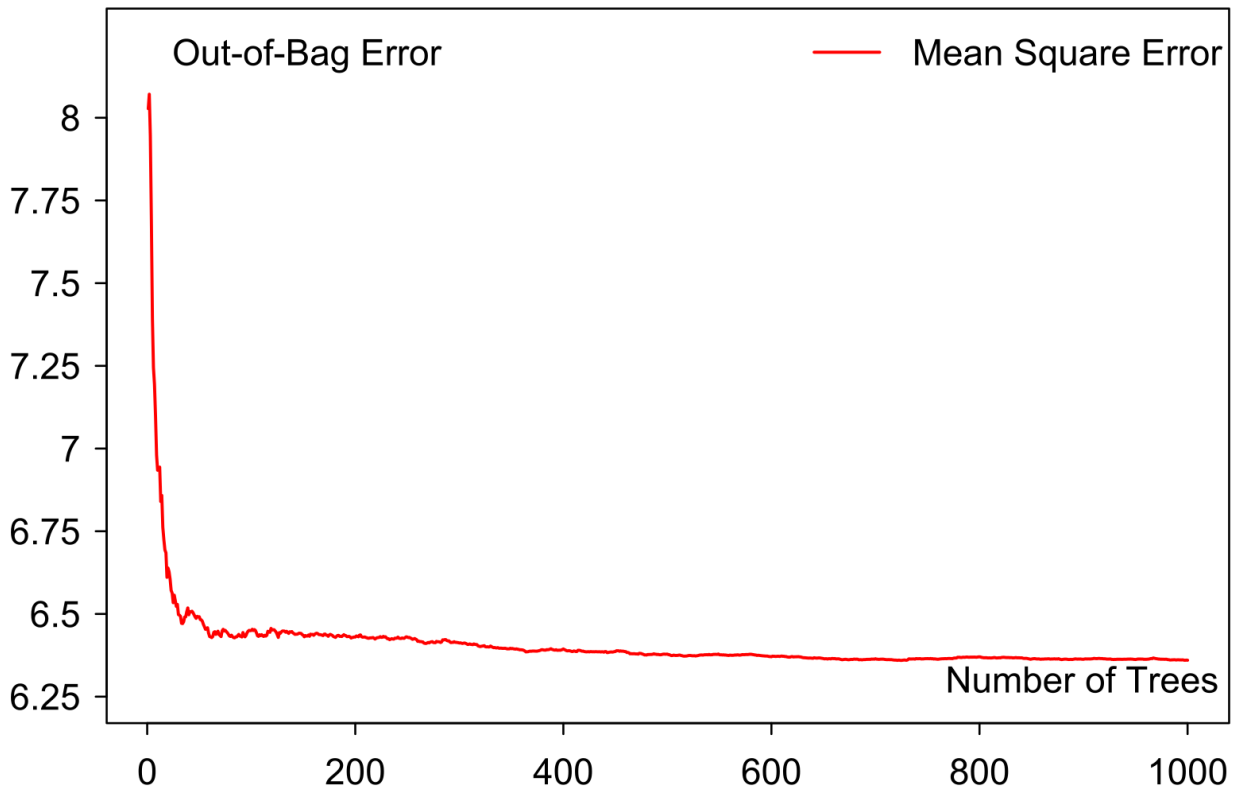
# Bagging: Out-of-Bag error

**How many** bootstrap samples (B) should we create?

- In each bootstrap sample, trees use $2N/3$ of all training observations on average.
- Remaining observations are referred to as **out-of-bag** (OOB) observations.
- OOB can be predicted in an out-of-sample context (just like in CV).
- With increasing B, the **OOB error** should decrease until it reaches a plateau.

# Bagging: Example

With $B = 1000$ the prediction on the testing sample led to an impressive improvement in MSE = 6.16 and MAE = 1.89.



# Outline for Section 7

# Random Forest: Introduction

An **issue** with bagging:

- With trees, bagging leads to (positively) **correlated trees**, i.e. predictions are similar, because trees are similar.
- Same set of **important predictors** is chosen to split the trees.
- Averaging works best for uncorrelated predictions (recall the portfolio theory...).

**Random forest** uses bagging as well, but with a **twist**.

- At each split we are not searching from all features the one that gives us the best split, instead **random set** of $m$ features is selected ($m \approx \sqrt{p}$) first.
- This leads to **decorrelated trees**.

# Random Forest: Introduction

Random forest tends to work quite well, but one needs to tune several hyper-parameters:

- **Number of trees**, $B$, in the bagging procedure.
- Number of **randomly chosen features** at each split, $m$.
- **Depth** of a tree.
- Other pre-pruning parameters:
  - Minimum number of observations to split.
  - Minimum number of observations in the terminal node.

Tuning is driven via a k-fold CV.

# Random Forest: Example

Continuing with the previous example we cross-validated among following parameters.

- Number of samples $B = 100, 250, 500, 1000$.
- Number of randomly chosen features at a split 1.
- Depth of the tree $3, 6, 9, 30$ and deep tree.
- 10-fold CV.

Optimum parameters are 250 trees, depth 6. The prediction error on the test sample is further improved to MSE = 5.76 and MAE = 1.88.

# Outline for Section 8

# Boosting: Introduction

**Boosting** is a **general-purpose technique** in statistical learning.

- In bagging, trees are created in **parallel**. One tree is independent from the previous one.
- In boosting, trees are created **sequentially**. We are using information from the previous tree to improve our prediction in the next one.

# Boosting: Introduction

The algorithm:

1. Estimate a model (tree) using a matrix of features $\boldsymbol{X}$ and target variable $\boldsymbol{Y}$ from the training dataset, i.e. $T^0 = T(\boldsymbol{X}, \boldsymbol{Y})$.
2. Given the model (tree) $T^0$ predict observations $\hat{\boldsymbol{Y}}_{T^0}$ and calculate residuals $\boldsymbol{R}^0 = \boldsymbol{Y} - \hat{\boldsymbol{Y}}_{T^0}$.
3. Iterate over $b = 1, 2, ...., B$
   - 3.1 Estimate tree $T^b = T(\boldsymbol{X}, \boldsymbol{R}^{b-1})$ (note that instead of $\boldsymbol{Y}$ we have $\boldsymbol{R}^{b-1}$ here.
   - 3.2 Predict residuals $\hat{\boldsymbol{R}}_{T^b}$ and update the prediction $\hat{\boldsymbol{Y}}_{T^b} = \hat{\boldsymbol{Y}}_{T^{b-1}} + \lambda \hat{\boldsymbol{R}}_{T^b}$.
   - 3.3 Update residuals $\boldsymbol{R}^b = \boldsymbol{Y} - \hat{\boldsymbol{Y}}_{T^b}$.
4. Predicted values are $\hat{\boldsymbol{Y}}_{T^B}$.

In order not to over-fit the model, $\lambda \geq 0$ is introduced - the **learning rate**. Note $\lambda = 0$ no learning, while $\lambda = 1$ over-fit is likely.

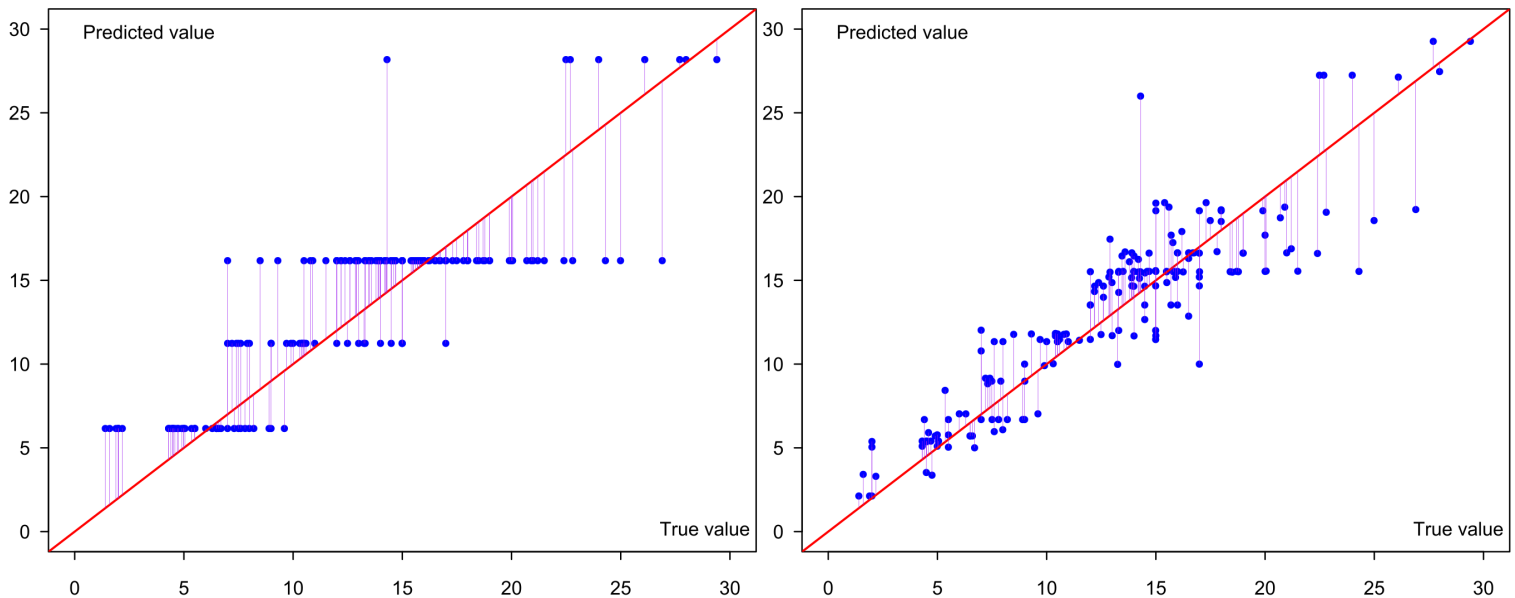
# Boosting: Introduction

As with other **Tree-Based** methods, we need to tune parameters.

- Number of trees $B$.
- Learning parameter $\lambda$, with typical values of 0.01, 0.001 or even smaller. Note that the smaller the $\lambda$ the larger needs to be $B$. Why?
- Complexity of the tree; tree depth or number of splits.

# Boosting: Example

The process of learning:




Within our example, the optimum parameters were depth = 2, $B = 1000$ and $\lambda = 0.01$. The MSE = 6.56 and MAE = 1.92. Among the best results.

# Toolset

What **methods** can we use?

- Linear regression.
- LASSO, Ridge and Elastic Net.
- Complete subset regressions.
- Random Forest.
- Boosted trees.

How to **increase accuracy** of models?

- Use interaction terms.
- Consider adding 'smart' features.
- Cross-validate model specifications, not just parameters.

[1] B Efron. “Bootstrap Methods: Another Look at the Jackknife”. In: *The Annals of Statistics* (1979), pp. 1–26.

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

[3] Dimitris N Politis and Joseph P Romano. “The stationary bootstrap”. In: *Journal of the American Statistical association* 89.428 (1994), pp. 1303–1313.

[4] Dimitris N Politis, Joseph P Romano, and T-L Lai. “Bootstrap confidence bands for spectra and cross-spectra”. In: *Sixth Multidimensional Signal Processing Workshop,* IEEE. 1989, pp. 88–90.

[1] B Efron. “Bootstrap Methods: Another Look at the Jackknife”. In: *The Annals of Statistics* (1979), pp. 1–26.

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

[3] Dimitris N Politis and Joseph P Romano. “The stationary bootstrap”. In: *Journal of the American Statistical association* 89.428 (1994), pp. 1303–1313.

[4] Dimitris N Politis, Joseph P Romano, and T-L Lai. “Bootstrap confidence bands for spectra and cross-spectra”. In: *Sixth Multidimensional Signal Processing Workshop,* IEEE. 1989, pp. 88–90.

MUNI
ECON

# Artificial Intelligence in Finance

## Supervised learning - continuous outcome part C

**Štefan Lyócsa**

Department of Finance, Faculty of Economics and Administration

October 23, 2024

MASARYK
UNIVERSITY