WWW.DAUPHINE.PSL.EU

ML Event Studies

Fabrice Riva

fabrice.riva@dauphine.psl.eu

Dauphine | PSL
UNIVERSITÉ PARIS

Objective

The objective is to analyze the ability of various return-generating models to predict reliable event-date returns for single securities (not portfolios)

- Model:
  - Linear models
  - Regularized linear models (Ridge, LASSO, Elastic Net)
  - Regression trees
  - Random forest regression
- Features
  - Standard features
    - Market portfolio returns
    - Fama-French 3 factors
    - Fama-French 5 factors
  - Non standard features
    - Stock's peers
    - Stock's nearest neighbors

Data

- Set of 256 stocks from the CRSP universe
  - Randomly selected stocks
  - Randomly selected "event" date between January 9, 2017 and March 29, 2018

	PERMNO	date	SICCD	RET	vwretd	SICCD_peers	RET_peers	date_rel	RET_neigh	Mkt_RF	SMB	HML	RMW	CMA	RF
0	10032	20170227	3670.0	-0.003292	0.001999	3670.0	0.007154	-250	-0.002337	0.0022	0.0085	-0.0033	-0.0038	-0.0015	0.00002
1	10032	20170228	3670.0	-0.025209	-0.004651	3670.0	-0.022114	-249	-0.008818	-0.0042	-0.0136	0.0016	-0.0001	-0.0020	0.00002
2	10032	20170301	3670.0	0.022650	0.013383	3670.0	0.021281	-248	0.024245	0.0147	0.0052	0.0077	-0.0054	0.0054	0.00001
3	10032	20170302	3670.0	-0.007848	-0.006866	3670.0	-0.008909	-247	-0.009514	-0.0070	-0.0054	-0.0090	0.0063	-0.0043	0.00001
4	10032	20170303	3670.0	-0.005273	0.000944	3670.0	-0.001480	-246	-0.000025	0.0009	-0.0018	0.0014	-0.0006	-0.0010	0.00001

- **PERMNO**: stock unique identifier
- **date** (yyyymmdd)
- **SICCD**: stock's SIC code (industrial classification)
- **RET**: daily return
- **wwretd**: market portfolio daily return

- **SICCD_peers**: peers' SICCD
- **RET_peers**: daily return of peers' equally-weighted portfolio
- **date_rel**: date relative to event (date 0)
- **RET_neigh**: daily return of peers' equally-weighted portfolio

- Mkt_RF: daily excess return of market portfolio
- **SMB**: daily return of SMB factor
- **HML**: daily return of HML factor
- **RMW**: daily return of RMW factor
- **CMA**: daily return of CMA factor
- RF: daily risk-free rate

Models

- Linear models

$$R_{i,t} = \theta_{0,i} + \theta_{1,i}\mathbf{x}_{i,t} + \epsilon_{i,t}$$

  where $\mathbf{x}$ is one of the following single feature / set of features:

  - vwretd
  - $[\mathrm{vwretd}, \mathrm{RET\_peers}]$
  - $[\mathrm{vwretd}, \mathrm{RET\_neigh}]$
  - $[\mathrm{vwretd}, \mathrm{SMB}, \mathrm{HML}]$
  - $[\mathrm{vwretd}, \mathrm{SMB}, \mathrm{HML}, \mathrm{RMW}, \mathrm{CMA}]$

- Penalized linear models, decision tree and random forest models. For these models, $\mathbf{x} = [\mathrm{vwretd}, \mathrm{RET\_peers}, \mathrm{RET\_neigh}, \mathrm{SMB}, \mathrm{HML}, \mathrm{RMW}, \mathrm{CMA}]$

Model test procedure

- For each model and each stock, training is performed on the estimation window running from relative date -250 to relative date -1
- Specification and power of the models are evaluated as follows:
  - A shock of size δ is artificially added to date 0's return
  - For the specification test, the value of δ is 0
  - For the power test, different shock values are used : -10%, -5%, -2%, -1%, +1%, +2%, +5%, +10%
- Assuming returns are normally distributed, conducting either the specification or the power test can be made as follows:
  - For each stock i, compute z-score $z_i = (AR_{i,0} + \delta)/\hat{\sigma}(\epsilon_i)$
  - For a α% test, compare z_i with $z_{\alpha/2}$
  - Reject the null hypothesis $AR_{i,0} + \delta = 0$ if $|z_i| \geq z_{\alpha/2}$
  - Iterate over stocks and compute the empirical rejection frequency of the null. Compare with the theoretical rejection frequency α

Regularized linear models

Regularization and model complexity

- Complex models overfit the data
- Regularization aims at reducing model complexity by putting constraints on how large the parameters of a model can be $\Rightarrow$ parameters are **shrunk** towards 0
- Why is it interesting to get small parameter values?
  - Encouraging the loss minimization procedure to target small parameter values encourages the model to focus on the most important features (higher explanatory power)
  - As a result the obtained model displays lower complexity and is less prone to overfitting the data
- Putting constraints on the parameters implies that those are not unbiased anymore
- Bias-variance tradeoff: accept some bias to achieve lower variance

Ridge regularization

- Definition:

$$L_n(\mathbf{X}, \mathbf{y}, \boldsymbol{\theta})_{\text{Ridge}} = L_n(\mathbf{X}, \mathbf{y}, \boldsymbol{\theta}) + \lambda \sum_{j=1}^{p} \theta_j^2$$

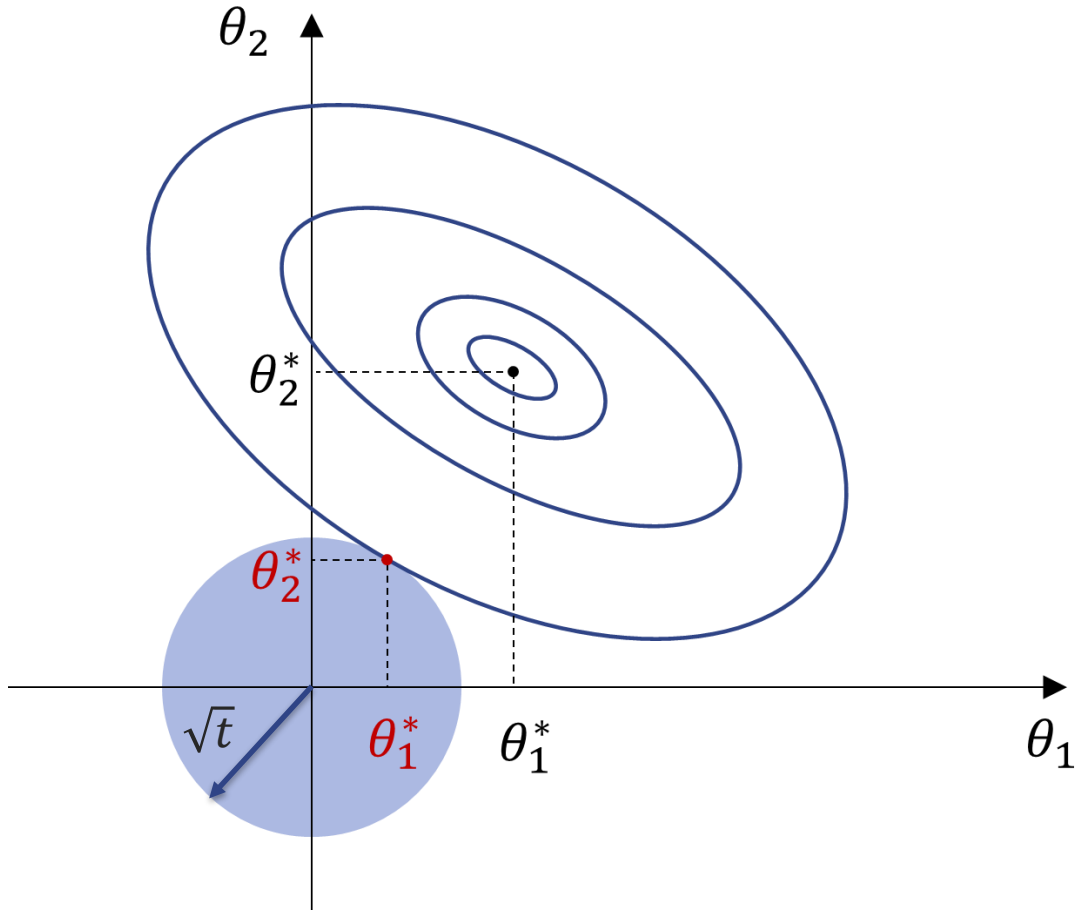
- Comments:
  - $L_n(\mathbf{X}, \mathbf{y}, \boldsymbol{\theta})$ is a baseline loss function, e.g. squared error loss for regression
  - Ridge adds a penalty term that is proportional to the sum of the **squared** θ_j parameters
  - Bias θ_0 does not enter the penalty
  - The intensity of the penalty is controlled by (positive) hyperparameter λ
  - The penalized loss reduces to the baseline loss if $\lambda = 0$

Ridge regularization: constrained optimization program

- Ridge penalty can also be expressed as a constrained optimization program

$$\begin{cases} \min_{\boldsymbol{\theta}} L_n(\mathbf{X}, \mathbf{y}, \boldsymbol{\theta}) \\ \text{s. t: } \sum_{j=1}^{p} \theta_j^2 \leq t \end{cases}$$

Graphical representation



LASSO regularization

- Definition:

$$L_n(\mathbf{X}, \mathbf{y}, \boldsymbol{\theta})_{\text{Lasso}} = L_n(\mathbf{X}, \mathbf{y}, \boldsymbol{\theta}) + \lambda \sum_{j=1}^{p} |\theta_j|$$

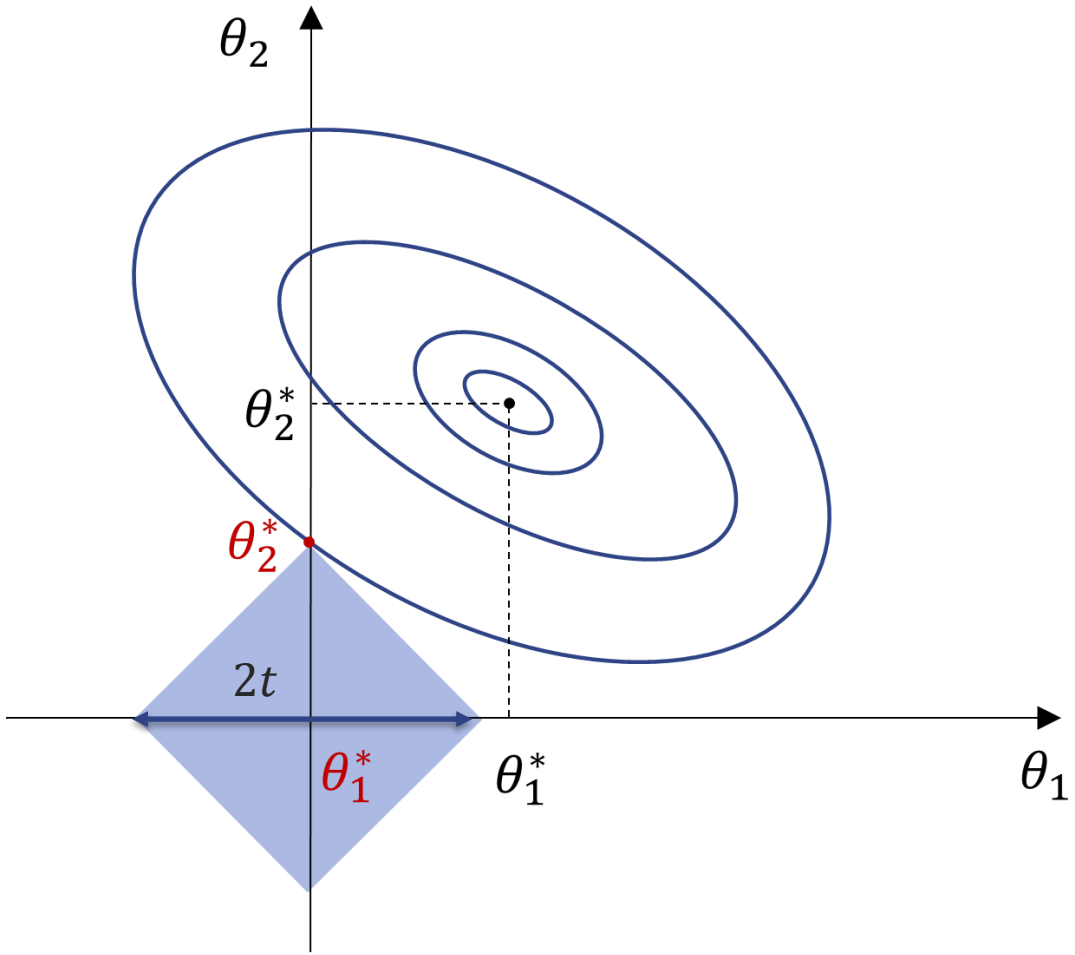
- Comments:
  - LASSO stands for Least Absolute Shrinkage and Selection Operator
  - Differs from Ridge by the fact that the penalty term is the sum of the **absolute value** of the θ_j parameters
  - Similar to Ridge, LASSO encourages small parameter values but goes one step further as, potentially, some parameter values can bet set to **exactly 0** $\Rightarrow$ LASSO promotes sparse models

LASSO regularization: constrained optimization program

- LASSO penalty can also be expressed as a constrained optimization program

$$\begin{cases} \min_{\boldsymbol{\theta}} L_n(\mathbf{X}, \mathbf{y}, \boldsymbol{\theta}) \\ \text{s.t:} \sum_{j=1}^{p} |\theta_j| \leq t \end{cases}$$

Graphical representation



Elactic Net

- Definition:

$$L_n(\mathbf{X}, \mathbf{y}, \boldsymbol{\theta})_{\text{EN}} = L_n(\mathbf{X}, \mathbf{y}, \boldsymbol{\theta}) + \lambda \left[\alpha \sum_{j=1}^{p} |\theta_j| + (1-\alpha) \sum_{j=1}^{p} \theta_j^2 \right]$$

- Comments:
  - Linear combination of Ridge and LASSO, where hyperparameter α controls the balance between both types of penalties and λ controls the overall intensity of the combined penalties
  - Aims at combining the strengths and weaknesses of Ridge and LASSO
    - LASSO: sparse models but difficulty in selecting correlated features
    - Ridge: no issue with correlated features but no sparsity
  - Shortcoming: requires the tuning of an additional hyperparameter

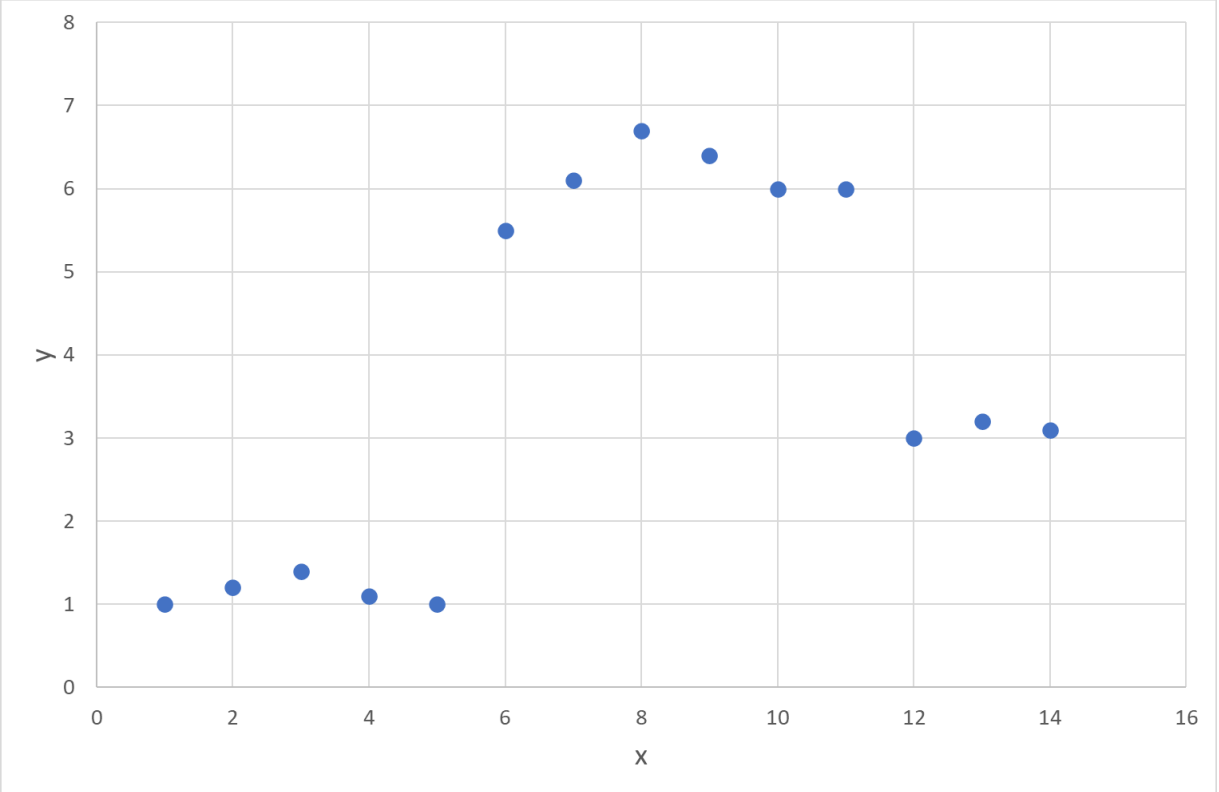
Decision tree regression

Example with one feature

- Data

x	1	2	3	4	5	6	7	8	9	10	11	12	13	14
y	1.0	1.2	1.4	1.1	1.0	5.5	6.1	6.7	6.4	6.0	6.0	3.0	3.2	3.1

- Graphical representation



Growing the tree - I

- Let denote $x_1, x_2, \ldots, x_n$ the sorted values of feature x. Corresponding labels are $y_1, y_2, \ldots, y_n$
- Average the first two x values to get threshold $t_1 = (x_1 + x_2)/2$. Denote:
  - $y_{t_1}^{\text{left}}$: labels of examples such that $x_i < t_1$
  - $y_{t_1}^{\text{right}}$: labels of examples such that $x_i > t_1$
- Next compute:
  - $\overline{y_{t_1}^{\text{left}}}$: average label of examples such that $x_i < t_1$
  - $\overline{y_{t_1}^{\text{right}}}$: average label of examples such that $x_i > t_1$
- Finally, compute MSE for t_1:

$$MSE_{t_1} = \sum_{x_i < t_1} \left(y_i - \overline{y_{t_1}^{\text{left}}}\right)^2 + \sum_{x_i > t_1} \left(y_i - \overline{y_{t_1}^{\text{right}}}\right)^2$$

Growing the tree - II

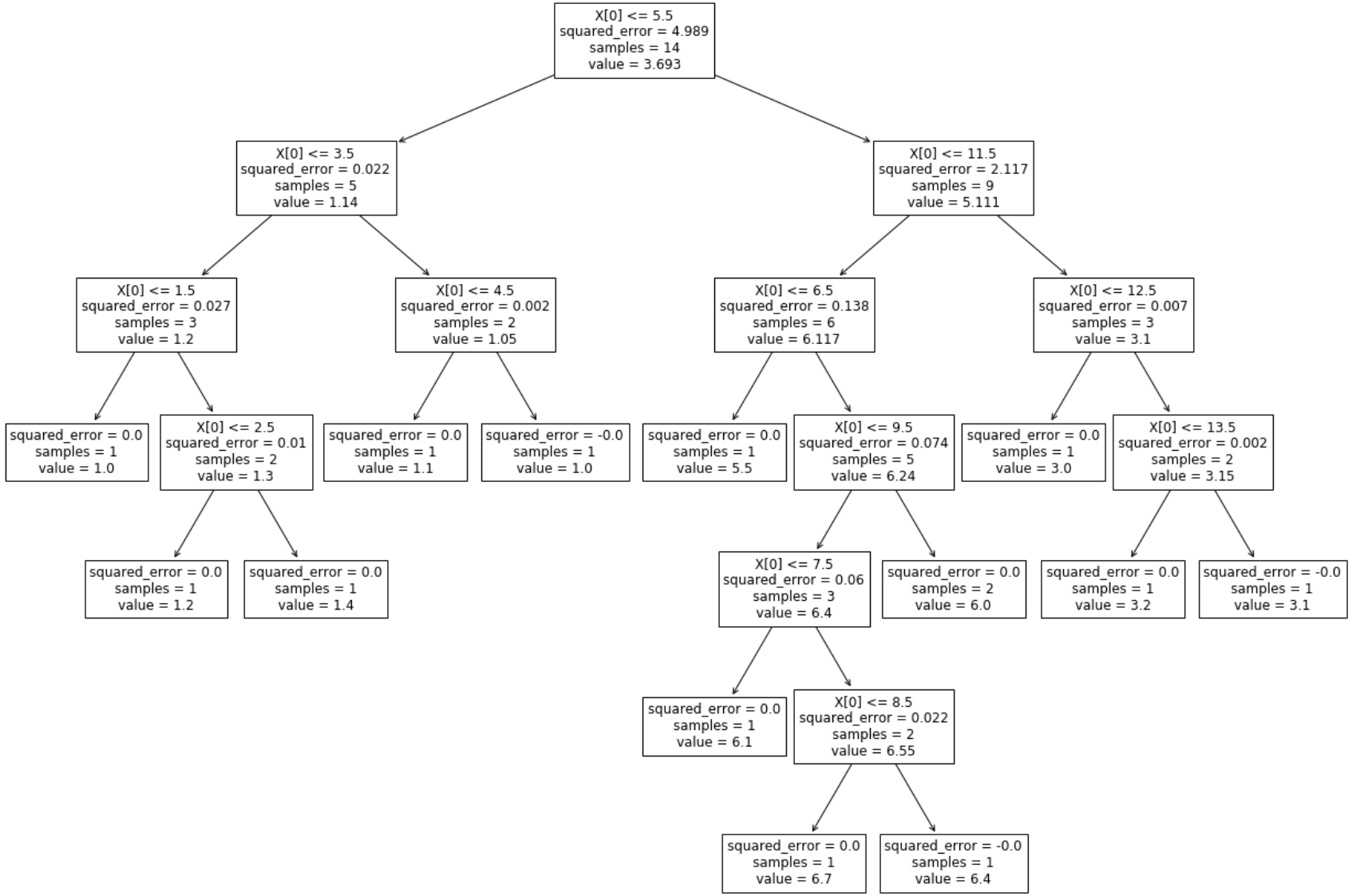
- Repeat the process with thresholds $t_2, t_3, \ldots, t_{n-1}$
- Choose threshold t_k that achieves the lowest MSE
- In the above example, the threshold that minimizes the MSE is $x = 5.5$
- The root of the tree is thus $x = 5.5$
- Next nodes are built by recursively applying the above steps until a given stopping criterion is hit (e.g. max depth)

```python
import numpy as np
from sklearn import tree
from sklearn.tree import DecisionTreeRegressor
from matplotlib import pyplot as plt
```

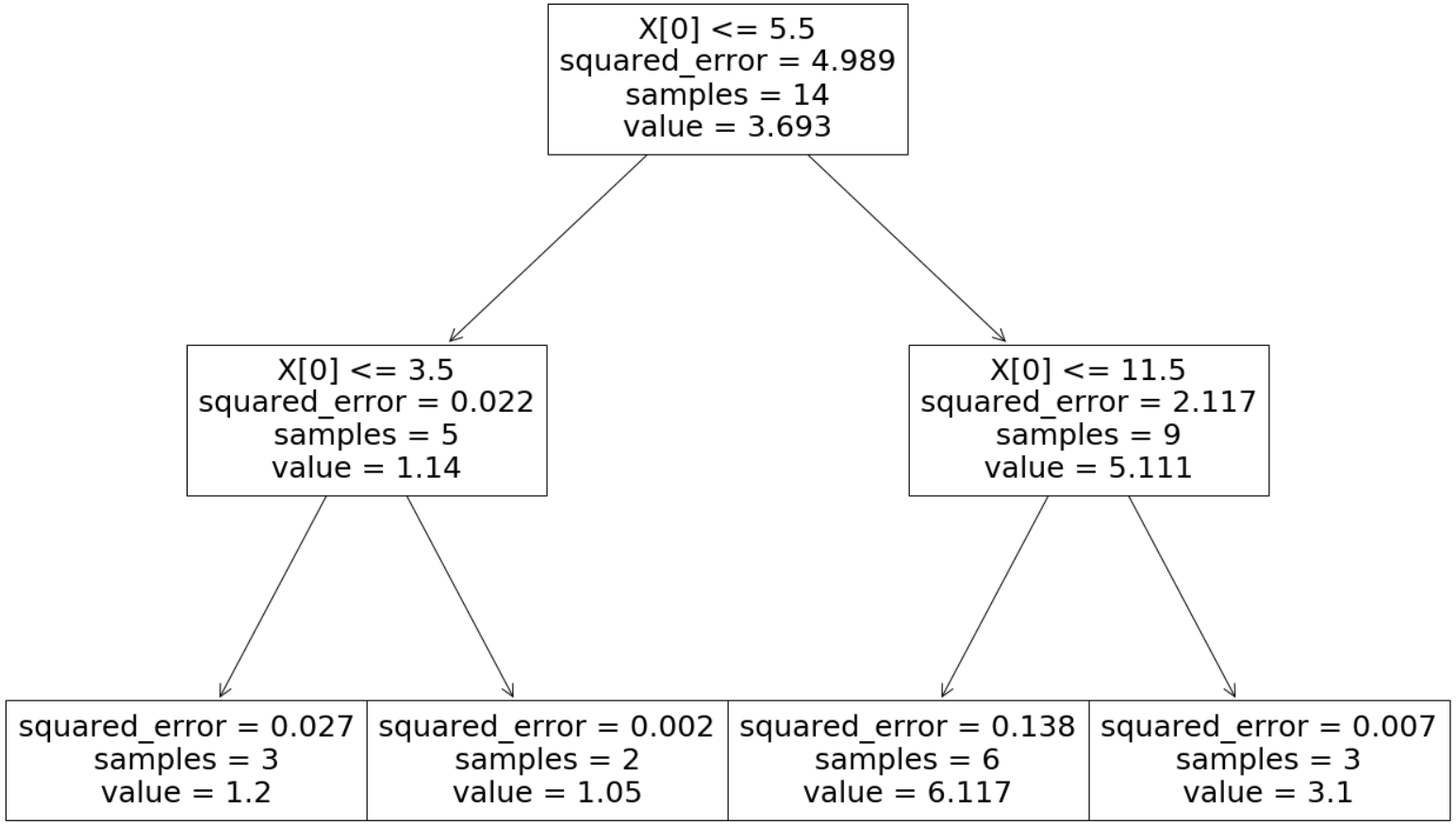
```python
X = np.array([1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14]).reshape(-1,1)
y = np.array([1.0, 1.2, 1.4, 1.1, 1.0, 5.5, 6.1, 6.7, 6.4, 6.0, 6.0, 3.0, 3.2, 3.1]).reshape(-1,1)

regTree = DecisionTreeRegressor()
reg = regTree.fit(X, y)
```

```python
plt.rcParams["figure.figsize"] = (22, 16)
tree.plot_tree(reg)
plt.show()
```

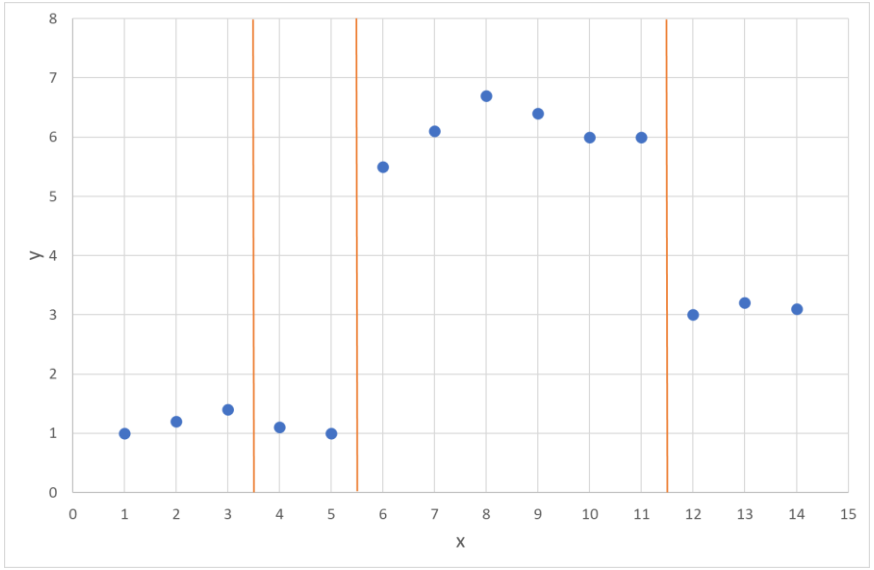
Tree representation - I

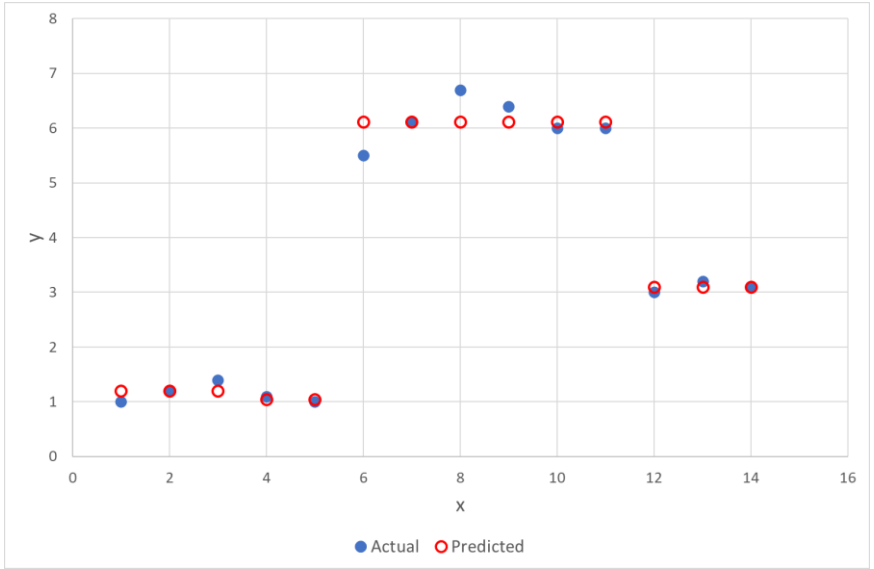


Tree representation - II



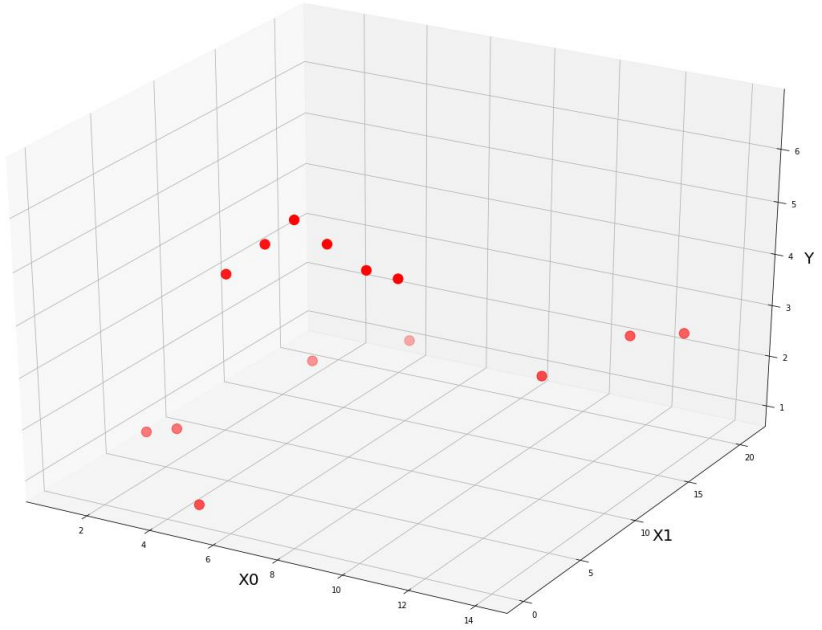

```
regTree = DecisionTreeRegressor(max_depth=2)
```

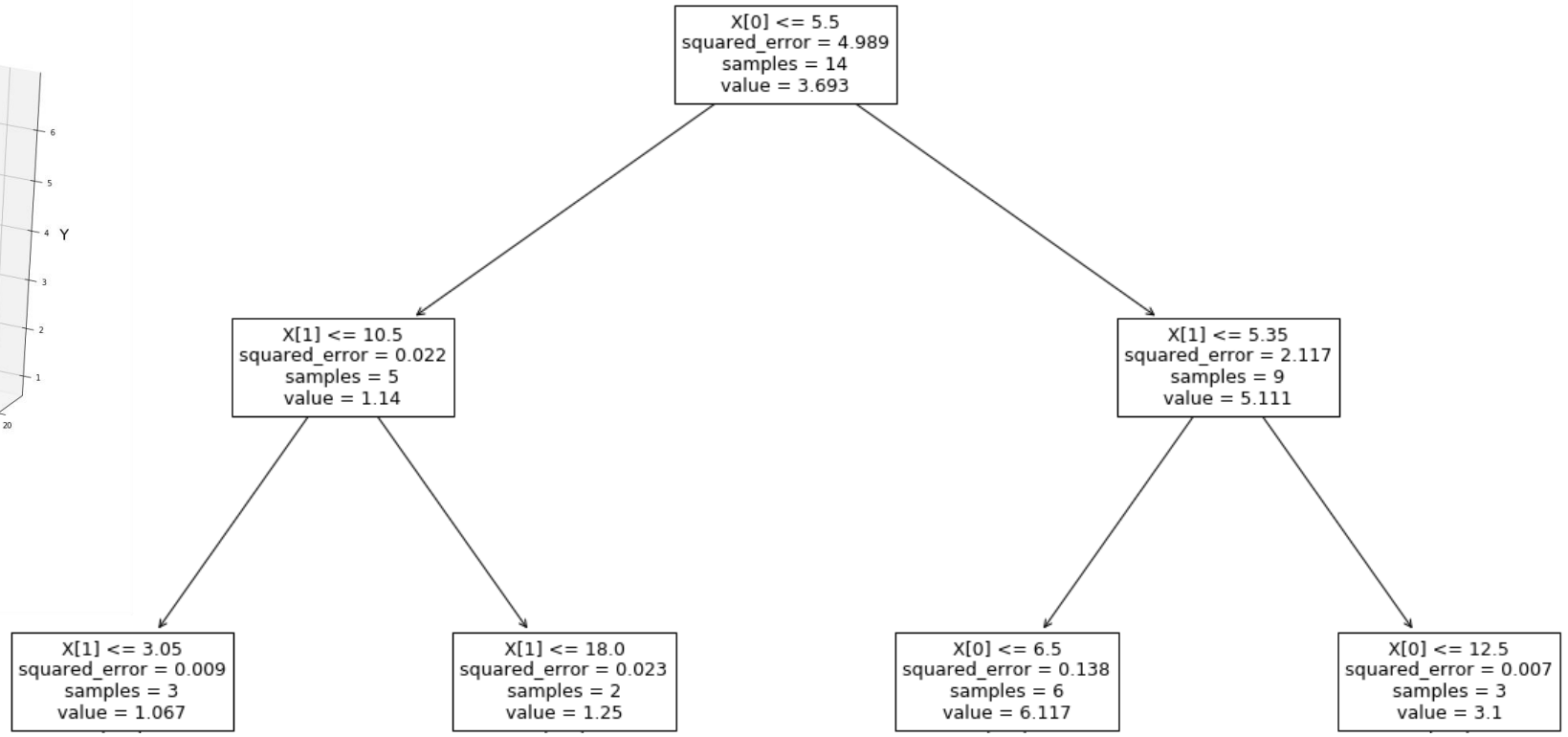




Multiple features

- Proceed as per previous steps for each feature
- Select the (feature, threshold) combination that minimizes the MSE
- Example:





Random forest regression

Principle

- A random forest (RF) is an ensemble of decision trees (DT) that are generally trained via bagging (random sampling with replacement)
- Compared with standard bagging, RF introduce an extra layer of randomness when growing trees
- Instead of splitting the examples by finding the best feature among all available ones it determines the best feature among a randomly selected subset of the features
- This extra randomness increases the independence and the diversity across trees, which should result in better ensemble learning
- The final prediction of a random forest prediction is often the mean of the predictions from all individual trees

Example

```
from sklearn.ensemble import RandomForestRegressor

regRF = RandomForestRegressor(max_depth=2)
reg3 = regRF.fit(X, y)
reg3.predict(X)
```