

# PSY9511: Seminar 5

Beyond linearity: Extensions of linear models and  
tree-based models

---

Esten H. Leonardsen

24.10.24



UNIVERSITETET  
I OSLO

# Outline

1. Exercise 3
2. Exercise 4
3. Recap
4. Extensions of linear models
  - 4.1 Generalized linear models (GLMs)
  - 4.2 Generalized additive models (GAMs)
5. Tree-based models
  - 5.1 Decision trees
  - 5.2 Random forests
  - 5.3 Gradient boosting (XGBoost)
6. Exercise 5



# Exercise 3

---



UNIVERSITETET  
I OSLO

# Exercise 3: Backward stepwise selection

<http://localhost:8888/notebooks/notebooks%2FBackward%20selection.ipynb>



# Exercise 3: Lasso

<http://localhost:8888/notebooks/notebooks/Lasso.ipynb>



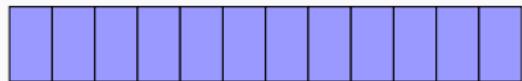
# Exercise 4

---

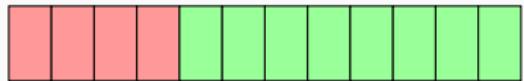
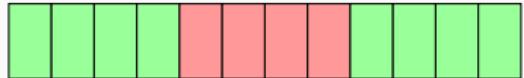
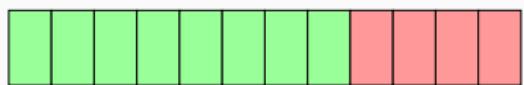
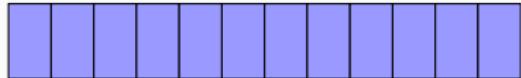


UNIVERSITETET  
I OSLO

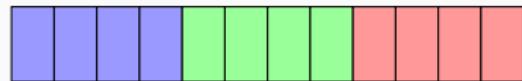
## Exercise 4: Data splitting



## Exercise 4: Data splitting



## Exercise 4: Data splitting



# Recap

---



UNIVERSITETET  
I OSLO

- Lecture 1: Introduction to machine learning
  - Supervised vs unsupervised learning
  - The goal of supervised learning: Find  $\hat{y} = f(x)$
- Lecture 2: Basics of regression and classification
  - Linear regression:  $\hat{y} = \beta_0 + \sum \beta_j x_j$
  - Logistic regression:  $p(y = 1) = \frac{e^{\beta_0 + \sum \beta_j x_j}}{1 + e^{\beta_0 + \sum \beta_j x_j}}$
- Lecture 3: Variable selection and regularization
  - Lasso:  $\beta = \min_{\beta} \left( \sum (y_i - \hat{y}_i)^2 + \lambda \sum |\beta_j| \right)$
  - Ridge:  $\beta = \min_{\beta} \left( \sum (y_i - \hat{y}_i)^2 + \lambda \sum \beta_j^2 \right)$
- Lecture 4: Model selection, validation, and testing
  - Whenever you make a decision based on results from data, subsequent results on the same data will be inflated



# Extensions of linear models

---



UNIVERSITETET  
I OSLO

# Non-linear models: Nothing to be afraid of

```
formula <- ...
result <- glm(formula, family=Gamma(link="log"), data=data)
```

```
formula <- ...
result <- gam(formula, data=data)
```

```
In[2]: from xgboost import XGBClassifier

model = XGBClassifier()
model.fit(X, y)
```

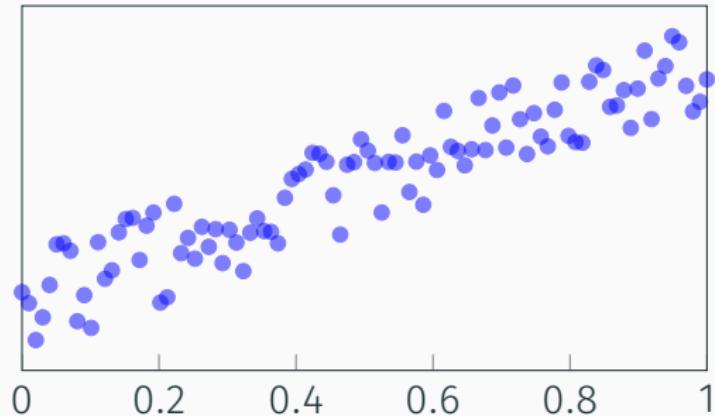


# Extensions of linear models: Generalized linear models

$$\hat{y} = \beta_0 + \sum_{i=0}^p \beta_i x_i$$



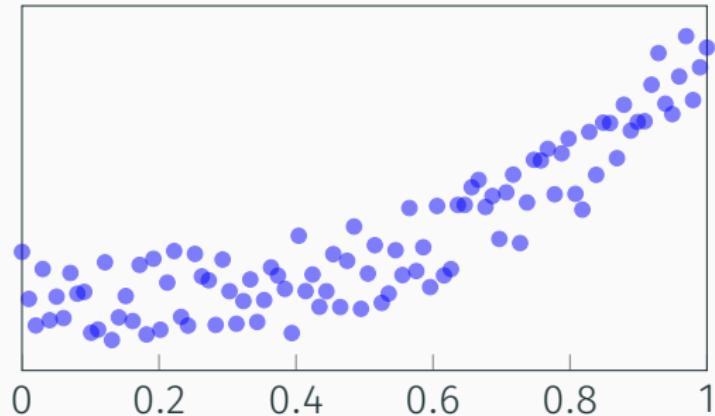
# Extensions of linear models: Generalized linear models



$$\hat{y} = \beta_0 + \sum_{i=0}^p \beta_i x_i$$



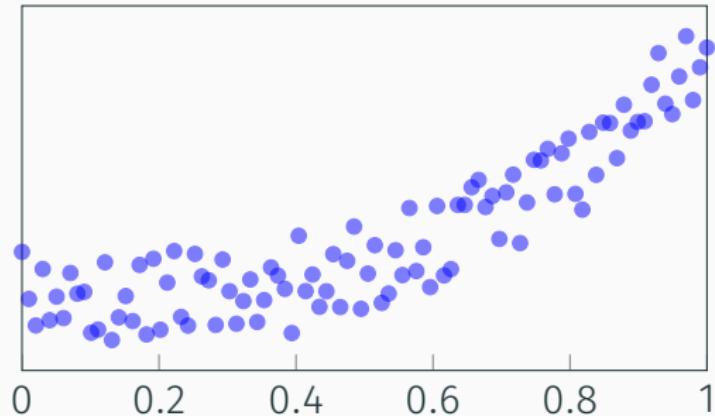
# Extensions of linear models: Generalized linear models



$$\hat{y} = \beta_0 + \sum_{i=0}^p \beta_i x_i$$



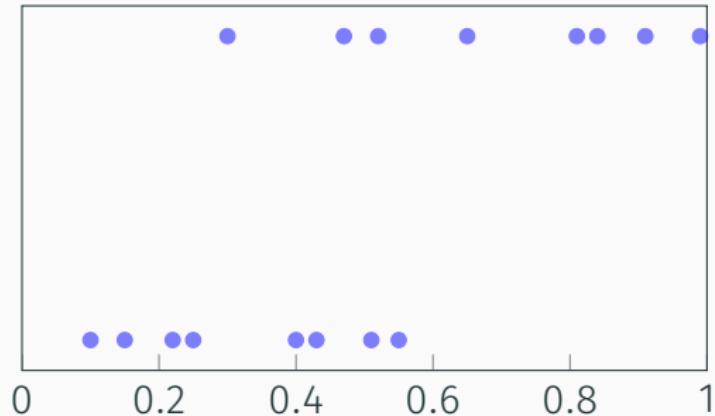
# Extensions of linear models: Generalized linear models



$$\hat{y} = \beta_0 + \sum_{i=0}^p \beta_i x_i$$



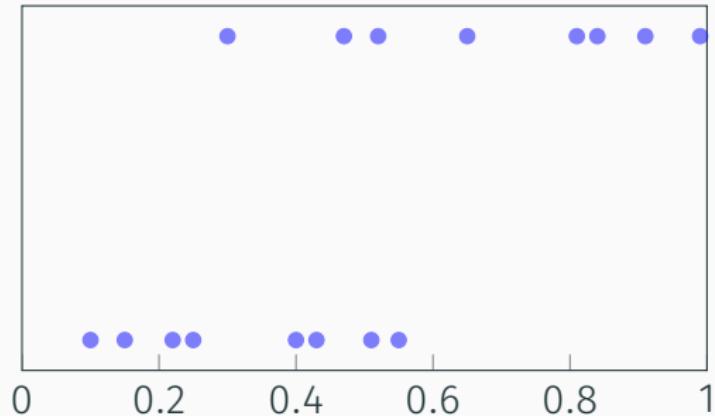
# Extensions of linear models: Generalized linear models



$$\hat{y} = \beta_0 + \sum_{i=0}^p \beta_i x_i$$



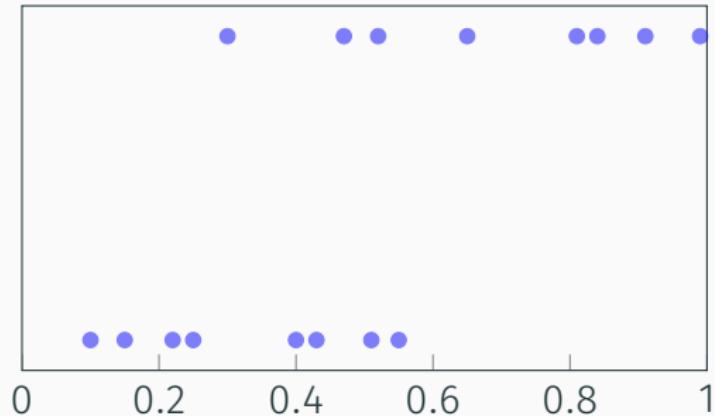
# Extensions of linear models: Generalized linear models



$$\log \left( \frac{p(X)}{1 - p(X)} \right) = \beta_0 + \sum_{i=0}^p \beta_i X_i$$



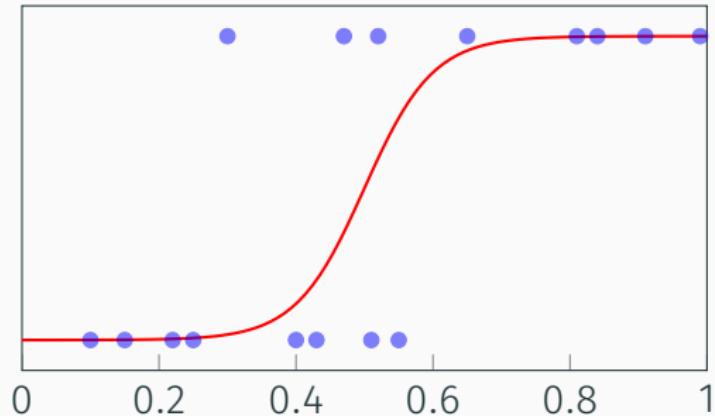
# Extensions of linear models: Generalized linear models



$$p(X) = \frac{e^{\left(\beta_0 + \sum_{i=0}^p \beta_i x_i\right)}}{1 + e^{\left(\beta_0 + \sum_{i=0}^p \beta_i x_i\right)}}$$



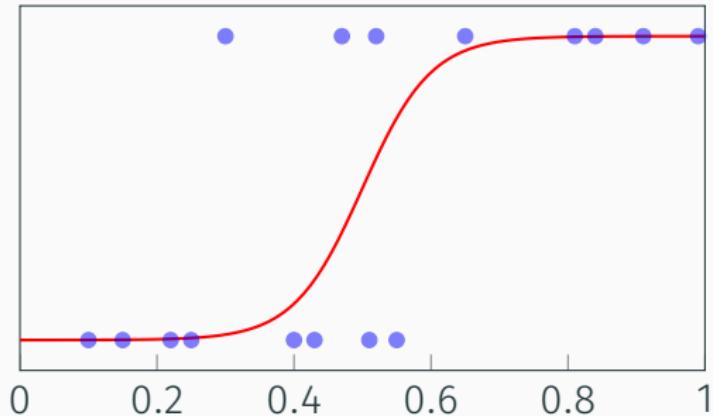
# Extensions of linear models: Generalized linear models



$$p(X) = \frac{e^{\left(\beta_0 + \sum_{i=0}^p \beta_i x_i\right)}}{1 + e^{\left(\beta_0 + \sum_{i=0}^p \beta_i x_i\right)}}$$



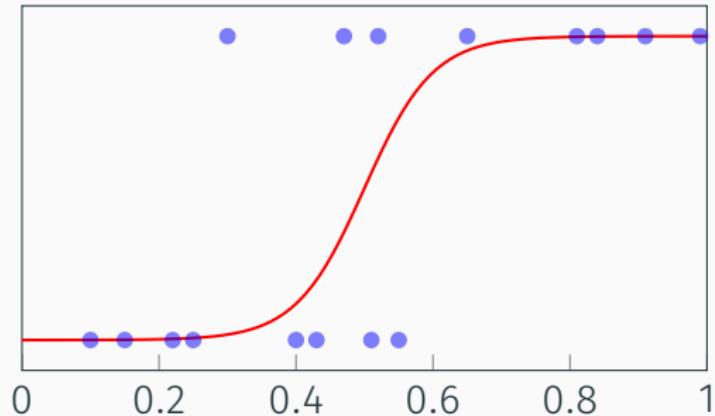
# Extensions of linear models: Generalized linear models



$$p(X) = \frac{e^{\left(\beta_0 + \sum_{i=0}^p \beta_i x_i\right)}}{1 + e^{\left(\beta_0 + \sum_{i=0}^p \beta_i x_i\right)}}$$



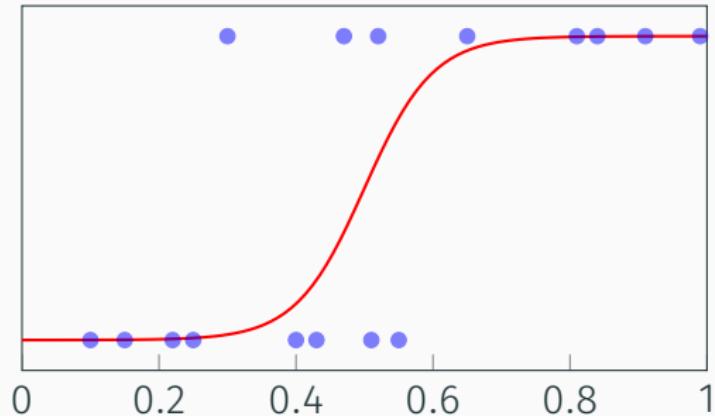
# Extensions of linear models: Generalized linear models



$$p(X) = f(\beta_0 + \sum_{i=0}^p \beta_i x_i)$$



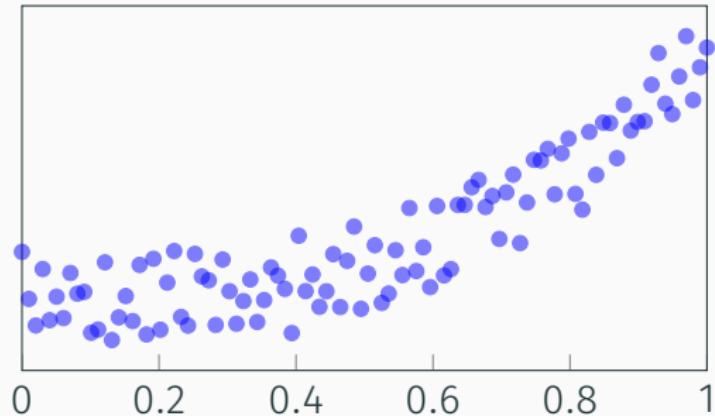
# Extensions of linear models: Generalized linear models



$$f(\hat{y}) = \beta_0 + \sum_{i=0}^p \beta_i x_i$$



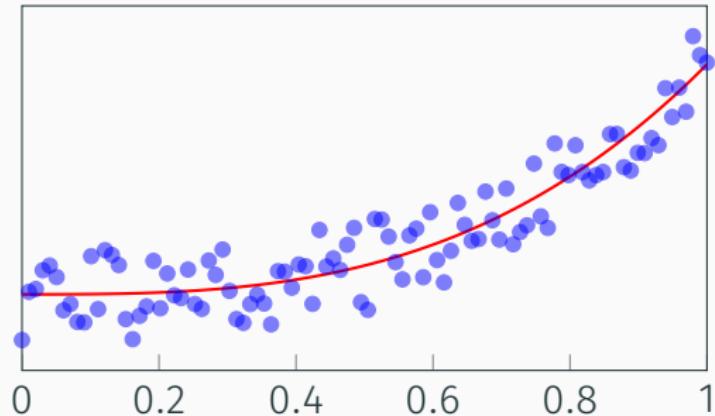
# Extensions of linear models: Generalized linear models



$$f(\hat{y}) = \beta_0 + \sum_{i=0}^p \beta_i x_i$$



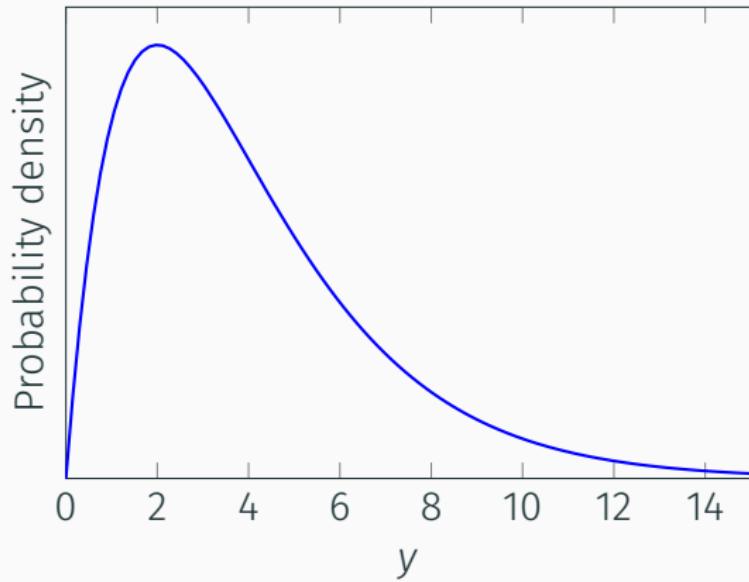
# Extensions of linear models: Generalized linear models



$$\log(\hat{y}) = \beta_0 + \sum_{i=0}^p \beta_i x_i$$



# Extensions of linear models: Generalized linear models



## Generalized linear models (GLMs):

Extends upon the regular linear model by associating the predictors to the response via a non-linear link function  $f$ .

- Requires us to specify  $f$  (often determined by investigating the distribution of the response).



# Extensions of linear models: Generalized linear models

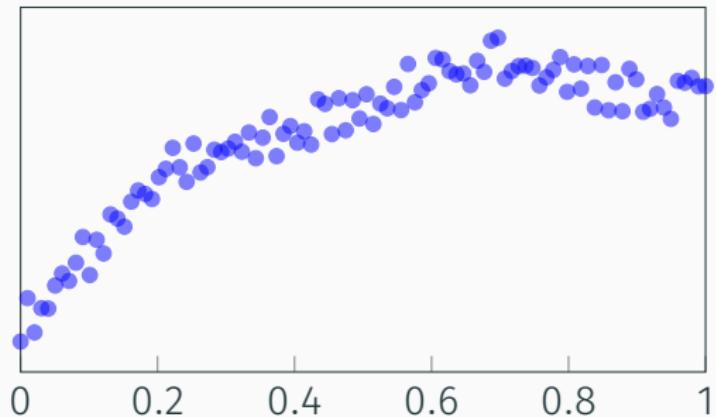
```
In[1]: from sklearn.linear_model import GammaRegressor  
  
model = GammaRegressor()  
model.fit(X, y)
```

```
In[2]: import statsmodels.api as sm  
  
link = sm.genmod.families.links.Log()  
model = sm.GLM(y, X, family=sm.families.Gamma(link=link))  
model.fit()
```

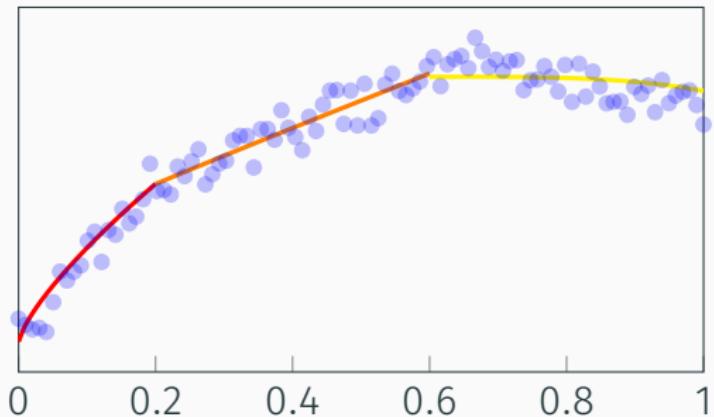
```
formula <- ...  
result <- glm(formula, family=Gamma(link="log"), data=data)
```



# Extensions of linear models: Generalized additive models

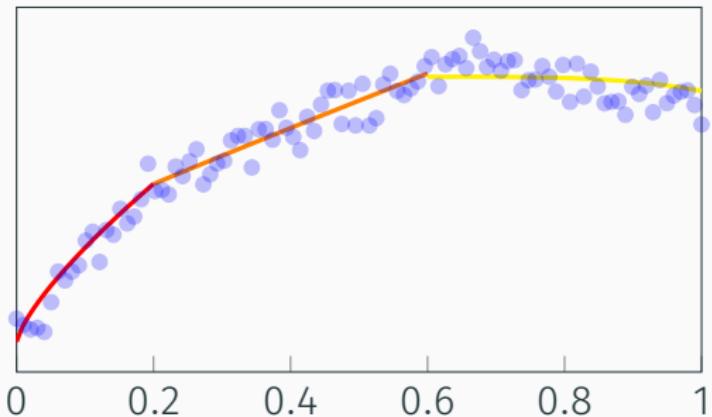


# Extensions of linear models: Generalized additive models



Splines: Piecewise polynomial functions

# Extensions of linear models: Generalized additive models



## Splines: Piecewise polynomial functions

- Regression splines (ISL, Chapter 7.4)
- Smoothing splines (This lecture)



# Extensions of linear models: Generalized additive models

$$\sum_{i=1}^n (y_i - g(x_i))^2 + \lambda \int g''(t)^2 dt$$



# Extensions of linear models: Generalized additive models

$$\hat{y}_i = g(x_i)$$
$$\sum_{i=1}^n (y_i - \underset{\downarrow}{g(x_i)})^2 + \lambda \int g''(t)^2 dt$$



# Extensions of linear models: Generalized additive models

$$\sum_{i=1}^n (y_i - g(x_i))^2 + \lambda \int g''(t)^2 dt$$

$\uparrow$

$$\sum (y_i - \hat{y}_i)^2$$



# Extensions of linear models: Generalized additive models

$$\sum_{i=1}^n (y_i - g(x_i))^2 + \lambda \int g''(t)^2 dt$$

↑  
Large when  $g$  is wiggly



# Extensions of linear models: Generalized additive models

Balances what is  
most important



$$\sum_{i=1}^n (y_i - g(x_i))^2 + \lambda \int g''(t)^2 dt$$

↑  
Tries to  
fit data

↑  
Tries to  
simplify  $g$



# Extensions of linear models: Generalized additive models

<http://localhost:8888/notebooks/notebooks/Smoothing%20spline.ipynb>



# Extensions of linear models: Generalized additive models

$$\hat{y} = \beta_0 + \beta_1 x$$

$$\hat{y} = g(x)$$



# Extensions of linear models: Generalized additive models

$$\hat{y} = \beta_0 + \beta_1 x$$



$$\hat{y} = \beta_0 + \sum_{j=1}^p \beta_j x_j$$

$$\hat{y} = g(x)$$



$$\hat{y} = \beta_0 + \sum_{j=1}^p f_j(x_j)$$



## Generalized additive models (GAMs):

Extends upon the regular linear model by allowing for non-linear functions  $f_j$  to be fitted for each predictor  $x_j$ .

- Does not allow for interactions between predictors.



# Extensions of linear models: Generalized additive models

[scripts/gam.R](#)



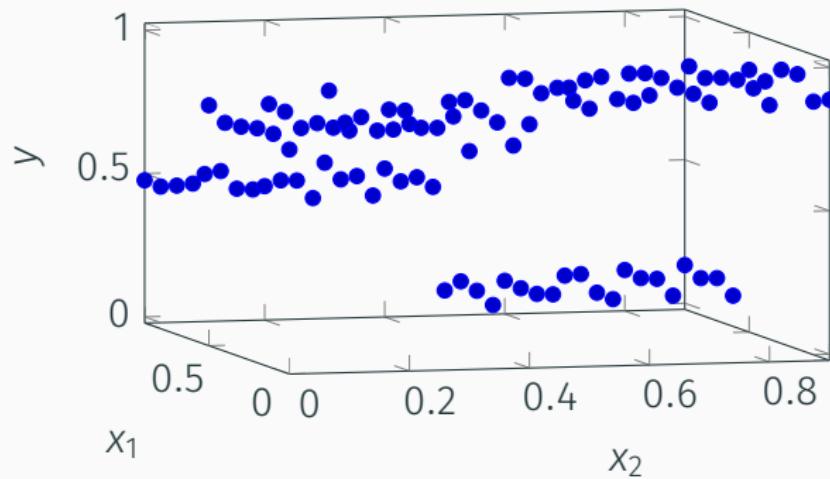
# Tree-based models

---

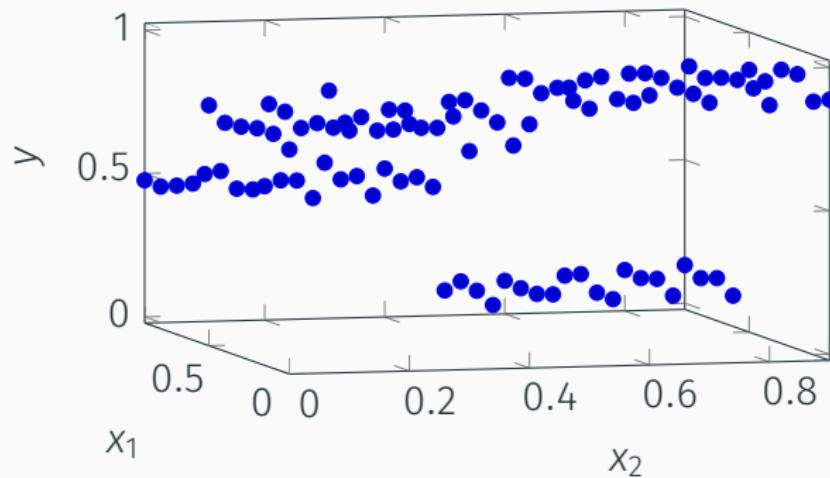


UNIVERSITETET  
I OSLO

# Tree-based models: Motivation



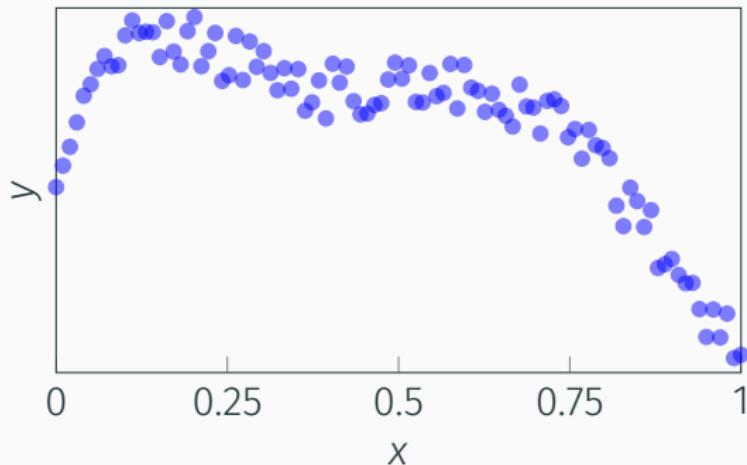
# Tree-based models: Motivation



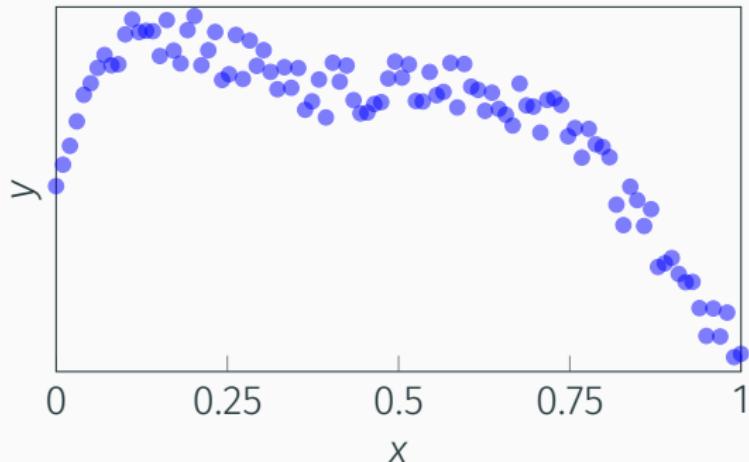
$$y = \begin{cases} 0.8 & x_1 \leq 0.6 \text{ & } x_2 \leq 0.5 \\ 0.9 & x_1 \leq 0.6 \text{ & } x_2 > 0.5 \\ 0.5 & x_1 > 0.5 \text{ & } x_2 \leq 0.5 \\ 0.1 & x_1 > 0.5 \text{ & } x_2 > 0.5 \end{cases}$$



# Tree-based models: Motivation



# Tree-based models: Motivation



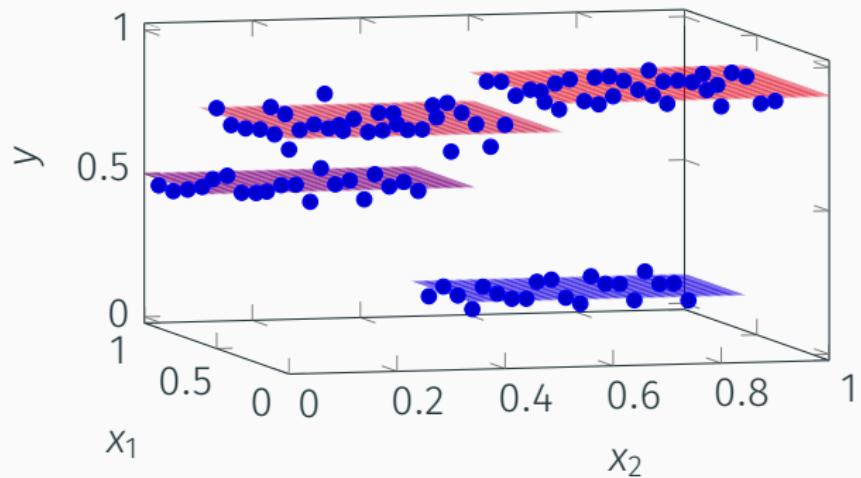
$$y = ?x^5 + ?x^4 + ?x^3 + ?x^2 + ?x + ?$$



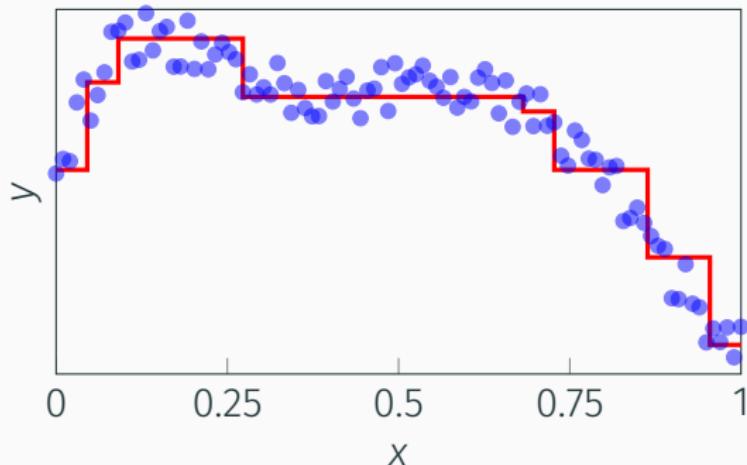
# Tree-based models: Motivation



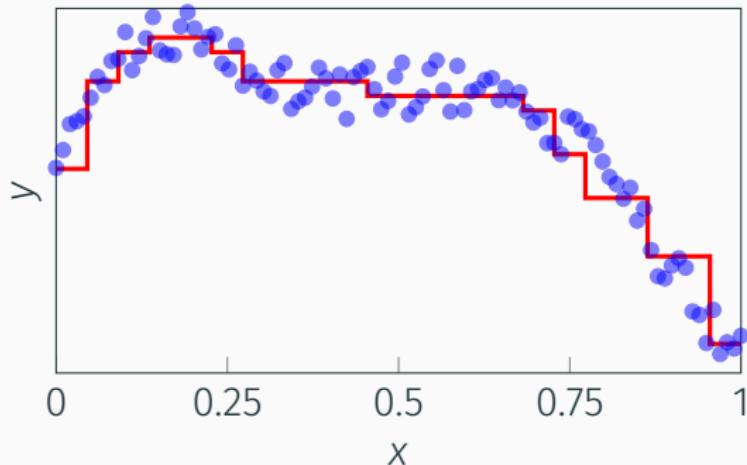
# Tree-based models: Motivation



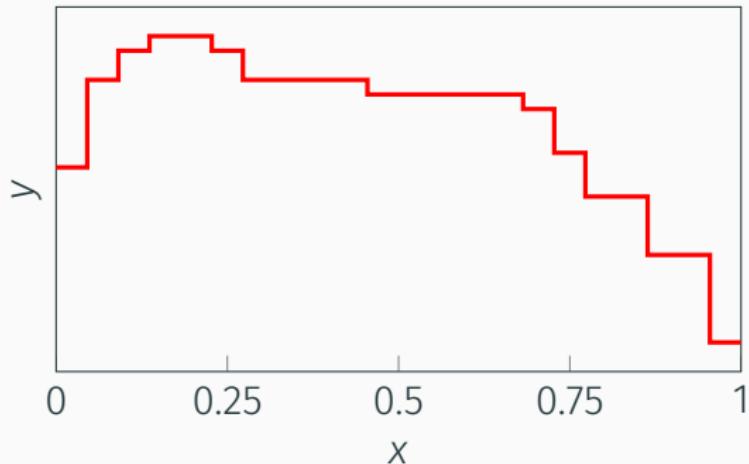
# Tree-based models: Motivation



# Tree-based models: Motivation



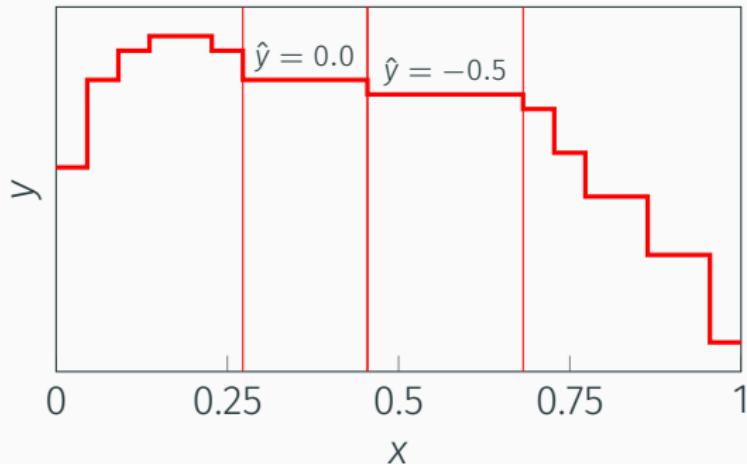
# Tree-based models: Motivation



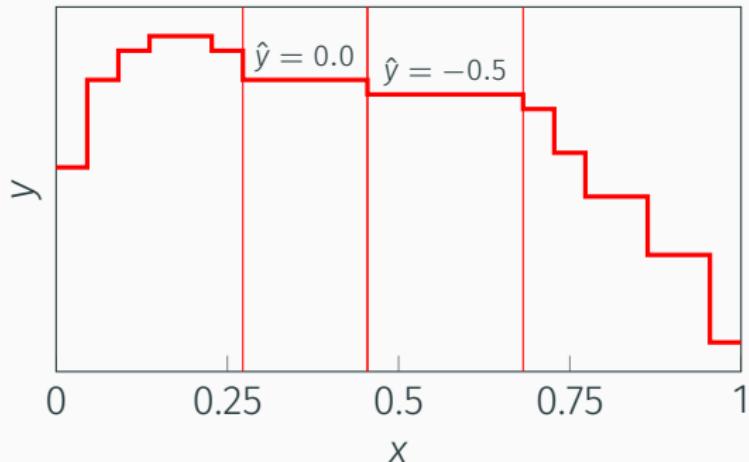
Piecewise constant function



# Tree-based models: Motivation



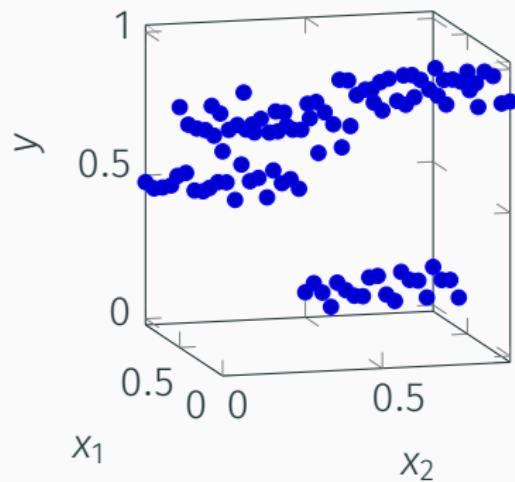
# Tree-based models: Motivation



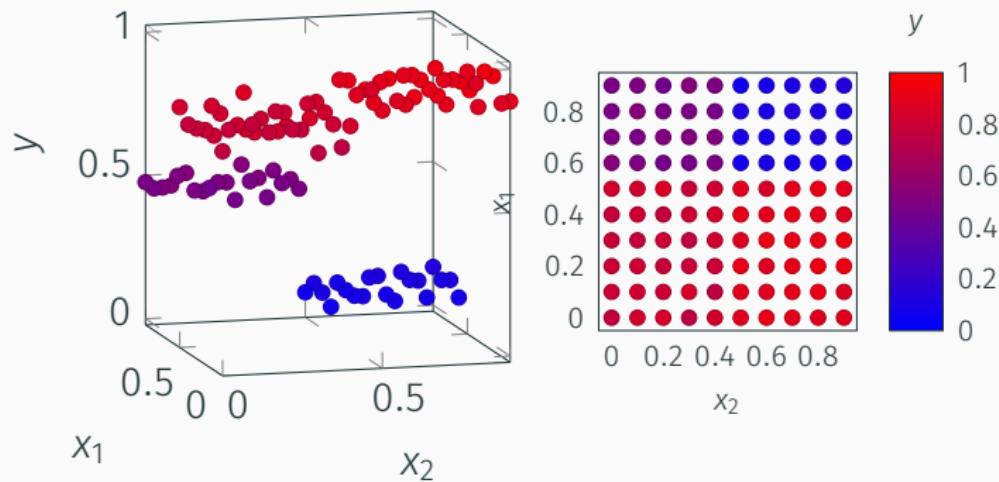
$$\hat{y} = \begin{cases} \dots \\ 0.0 & x \geq 0.27 \text{ \& } x < 0.45 \\ -0.5 & x \geq 0.45 \text{ \& } x < 0.69 \\ \dots \end{cases}$$



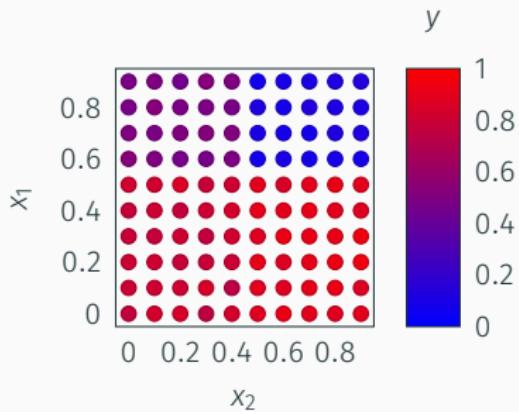
# Tree-based models: Decision trees



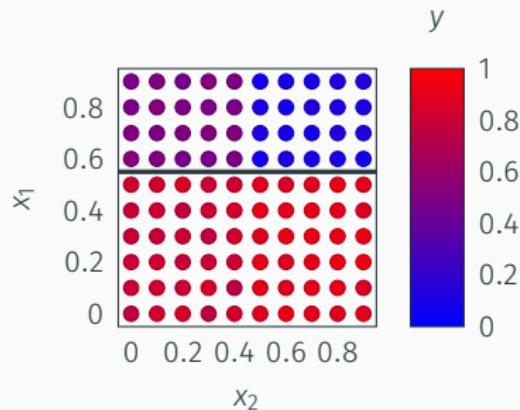
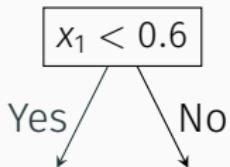
# Tree-based models: Decision trees



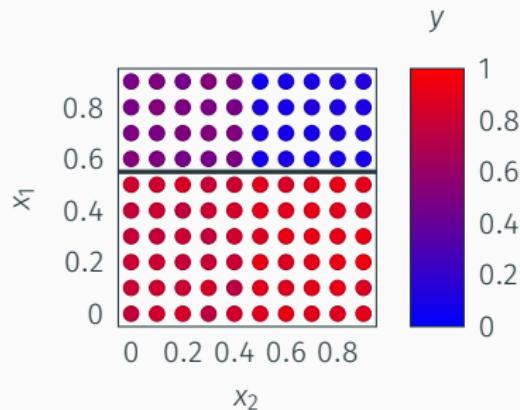
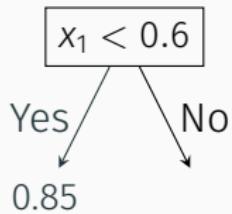
# Tree-based models: Decision trees



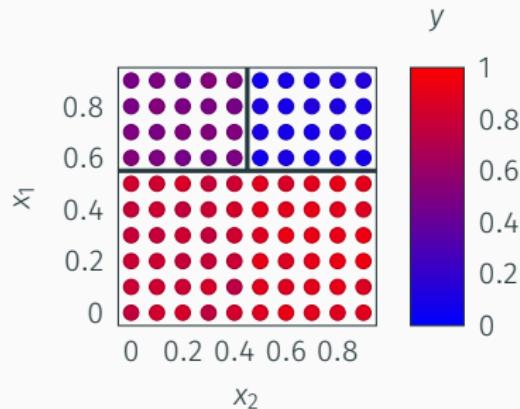
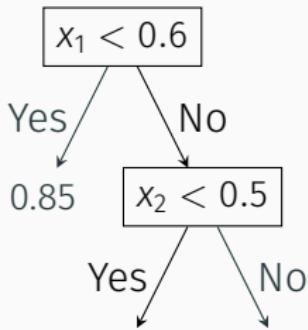
# Tree-based models: Decision trees



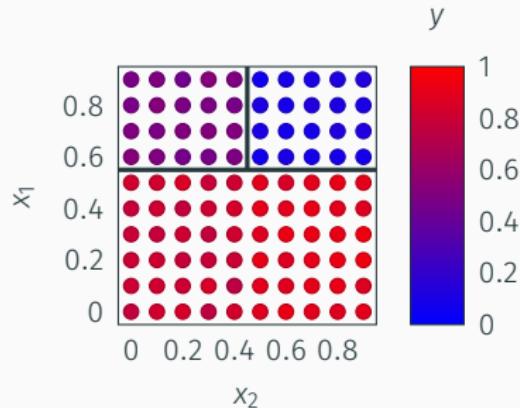
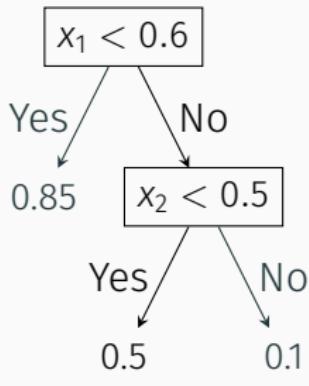
# Tree-based models: Decision trees



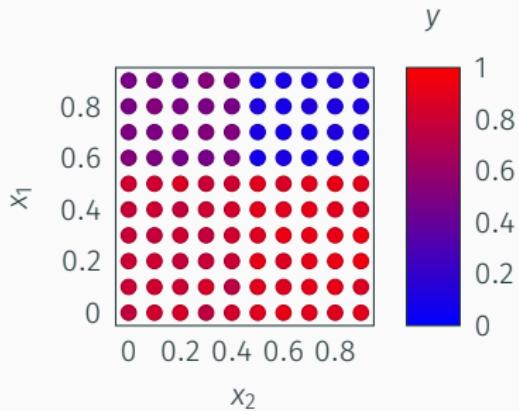
# Tree-based models: Decision trees



# Tree-based models: Decision trees

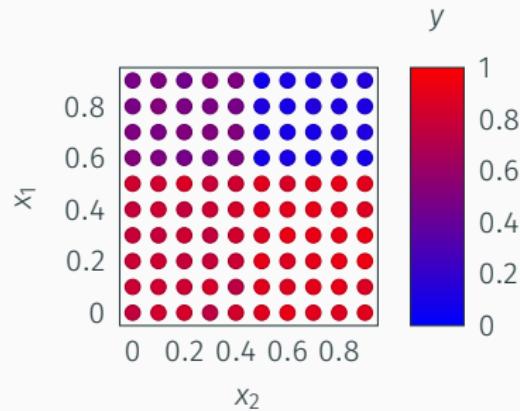


# Tree-based models: Decision trees



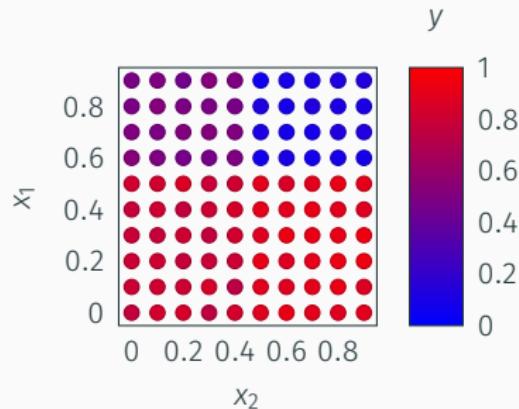
# Tree-based models: Decision trees

$x_1$  or  $x_2$ ?

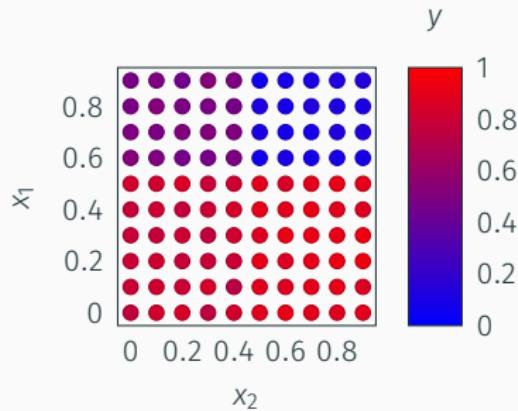
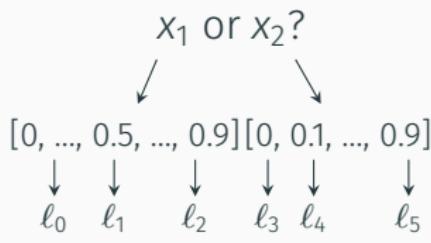


# Tree-based models: Decision trees

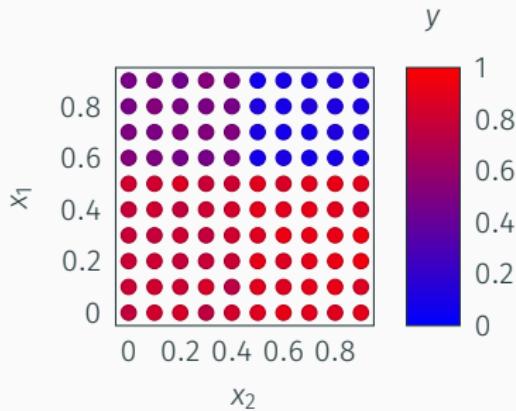
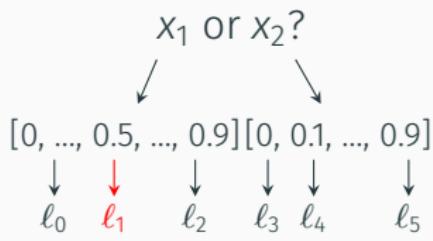
$x_1$  or  $x_2$ ?  
[0, ..., 0.5, ..., 0.9] [0, 0.1, ..., 0.9]



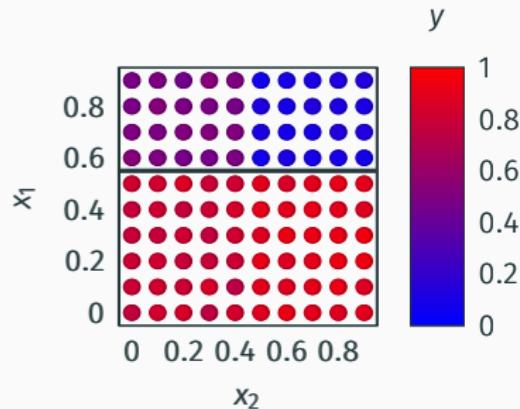
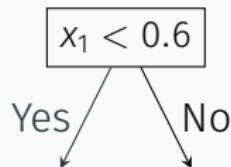
# Tree-based models: Decision trees



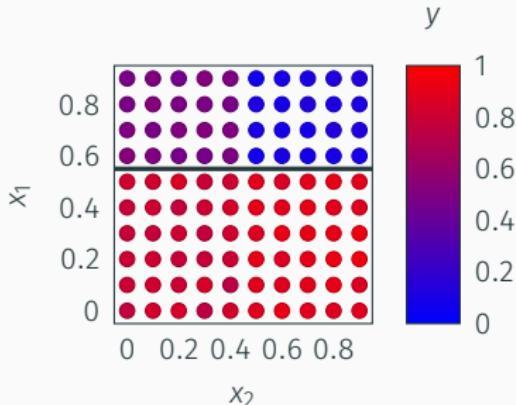
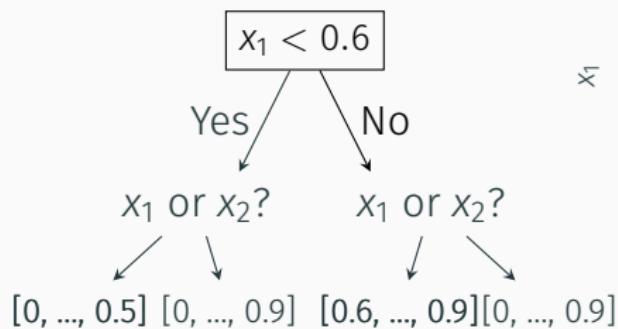
# Tree-based models: Decision trees



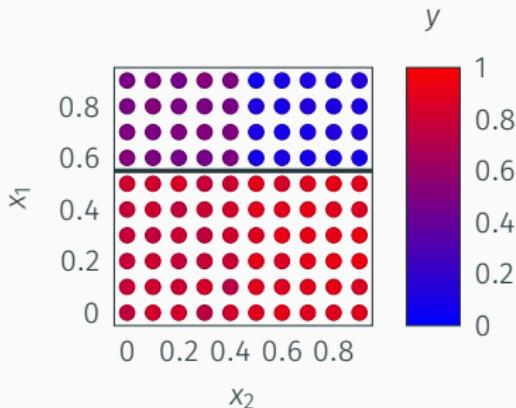
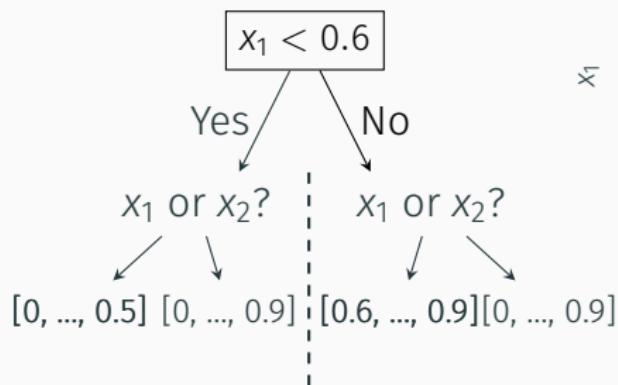
# Tree-based models: Decision trees



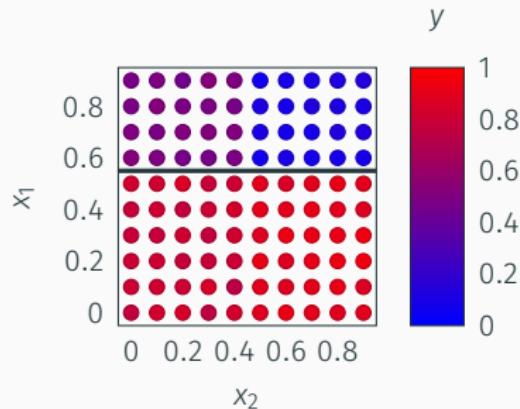
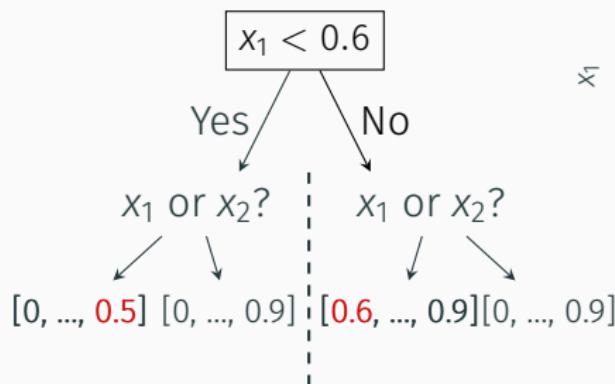
# Tree-based models: Decision trees



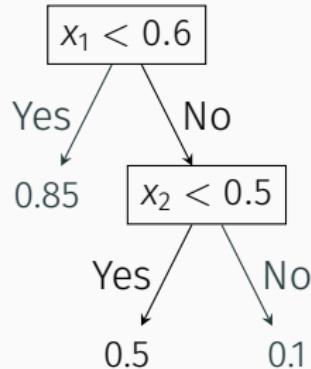
# Tree-based models: Decision trees



# Tree-based models: Decision trees

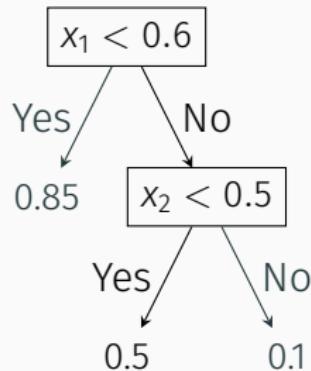


# Tree-based models: Decision trees



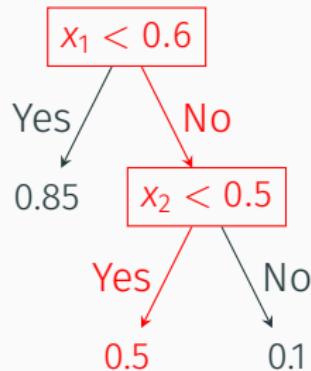
# Tree-based models: Decision trees

$$x = (0.8, 0.3)$$



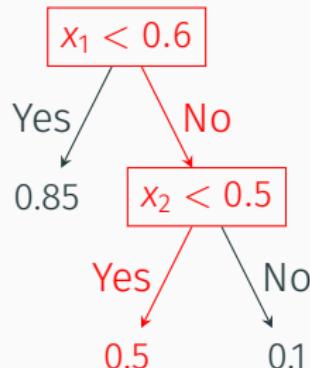
# Tree-based models: Decision trees

$$x = (0.8, 0.3)$$



# Tree-based models: Decision trees

$$x = (0.8, 0.3)$$

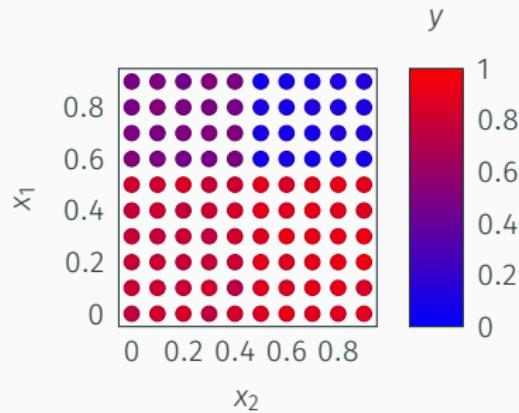


Why was  $\hat{y} = 0.5$ ?

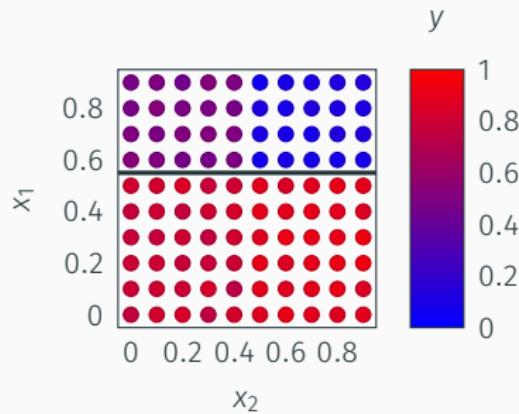
Because  $x_1 \geq 0.6$  and  $x_2 < 0.5$ .



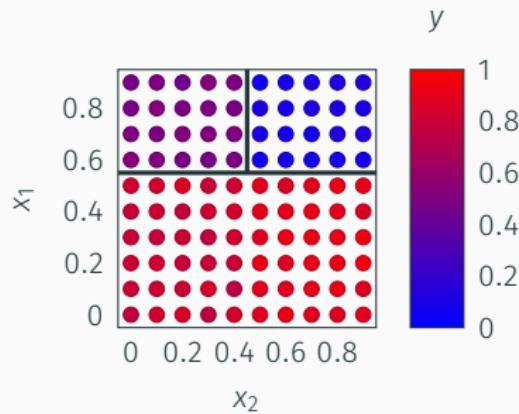
# Tree-based models: Decision trees



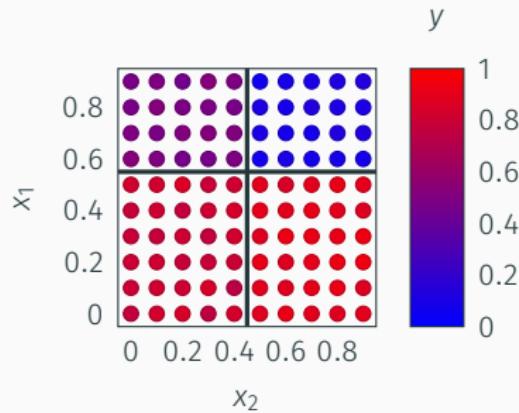
# Tree-based models: Decision trees



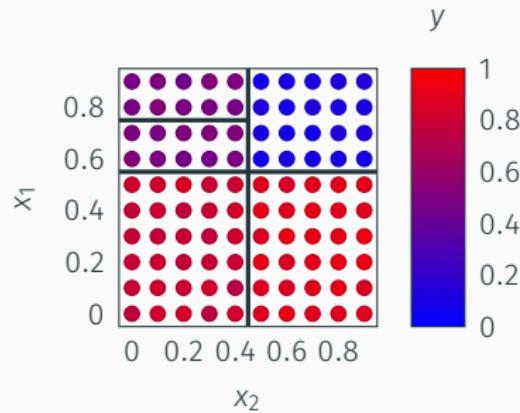
# Tree-based models: Decision trees



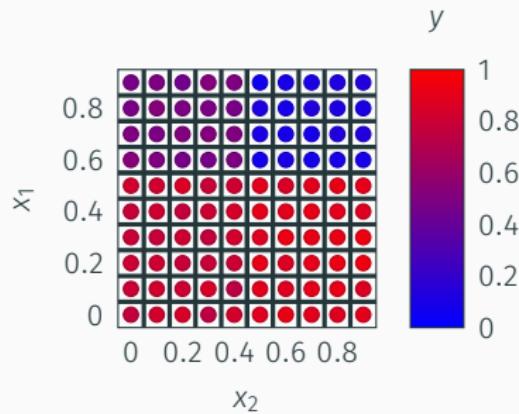
# Tree-based models: Decision trees



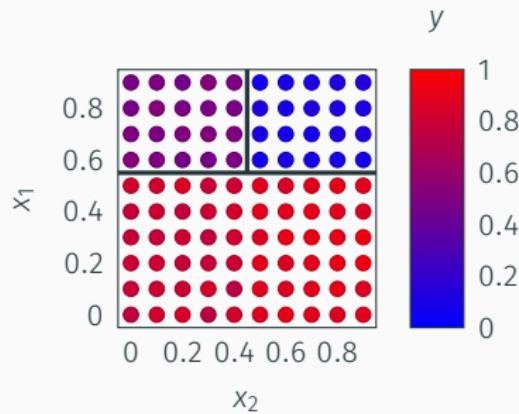
# Tree-based models: Decision trees



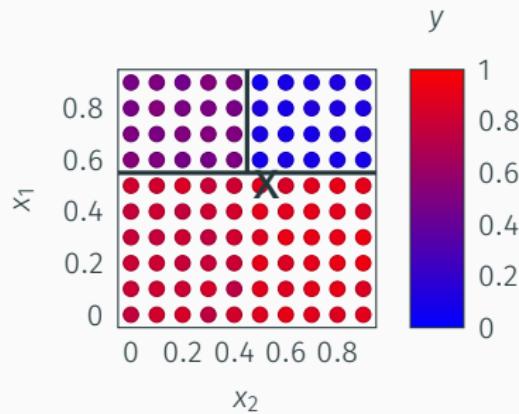
# Tree-based models: Decision trees



# Tree-based models: Decision trees



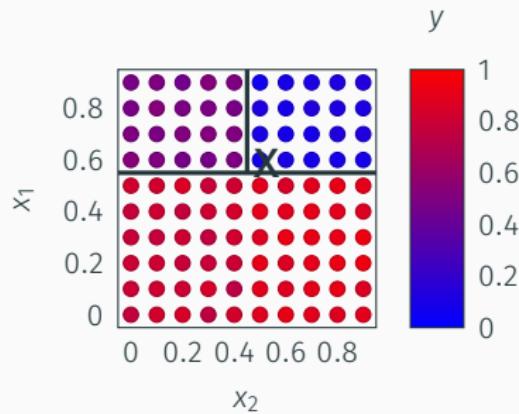
# Tree-based models: Decision trees



$$\hat{y} = 0.8$$



# Tree-based models: Decision trees



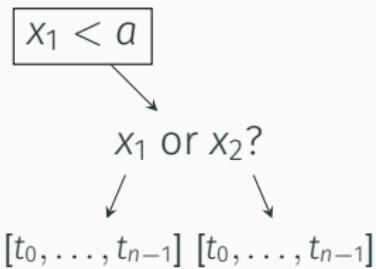
$$\hat{y} = 0.1$$



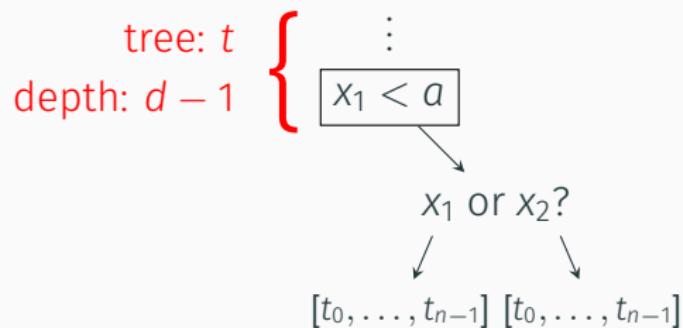
# Tree-based models: Decision trees



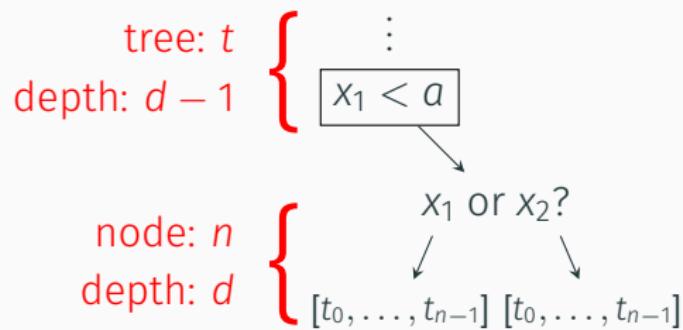
# Tree-based models: Decision trees



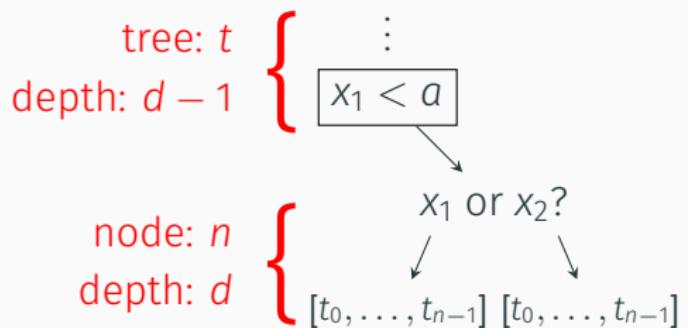
# Tree-based models: Decision trees



# Tree-based models: Decision trees



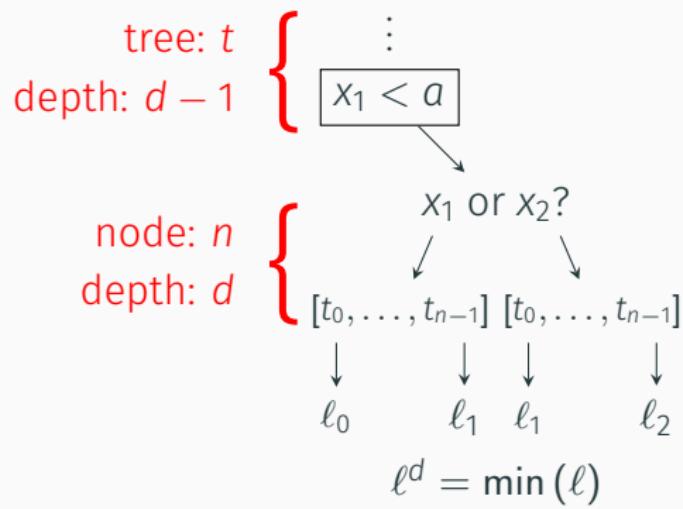
# Tree-based models: Decision trees



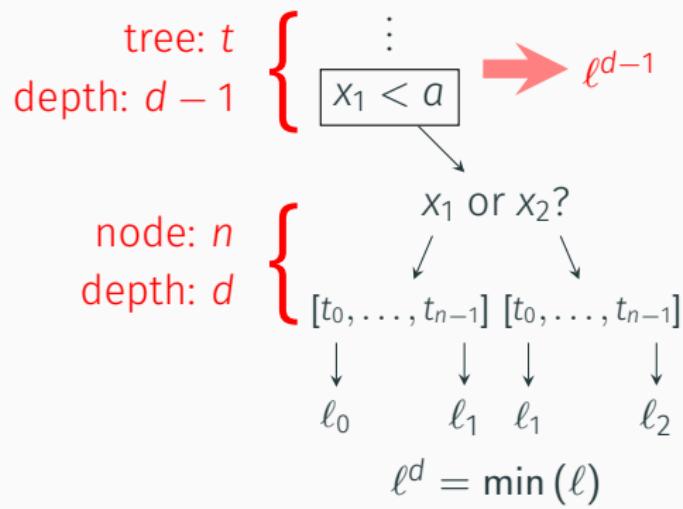
```
if  $d > \text{max\_depth}$ :  
    stop  
else:  
    continue
```



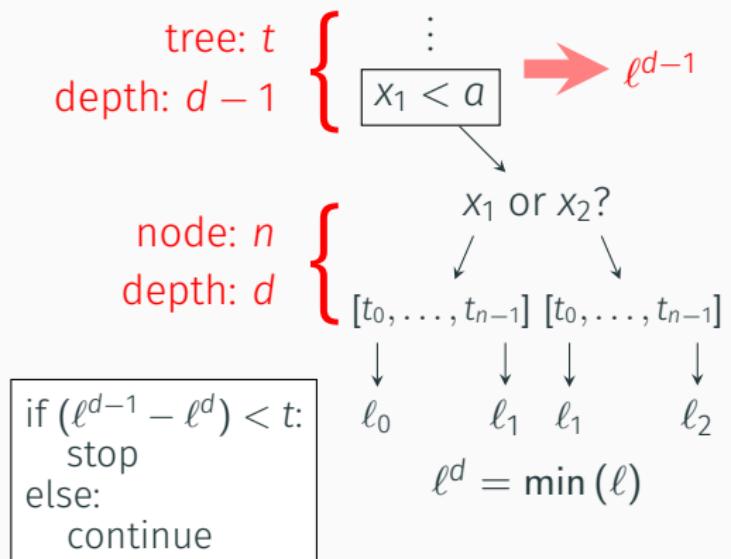
# Tree-based models: Decision trees



# Tree-based models: Decision trees



# Tree-based models: Decision trees

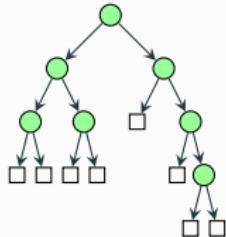


# Tree-based models: Decision trees

<https://scikit-learn.org/dev/modules/generated/sklearn.tree.DecisionTreeClassifier.html>



# Tree-based models: Decision trees



# Tree-based models: Decision trees

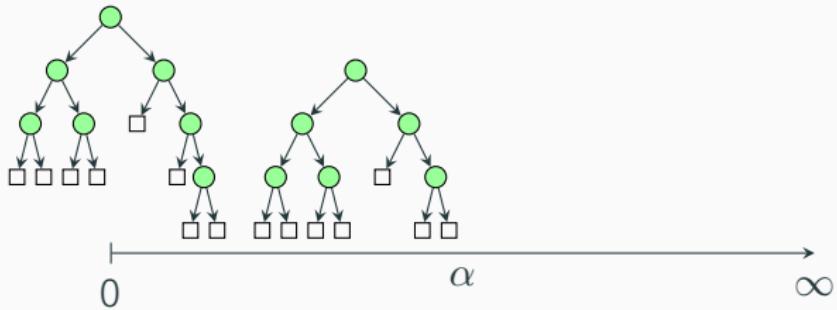


$$\sum_{i=0}^n (y_i - \hat{y}_i)^2 + \alpha|T|,$$

$|T|$ =number of leaf nodes



# Tree-based models: Decision trees

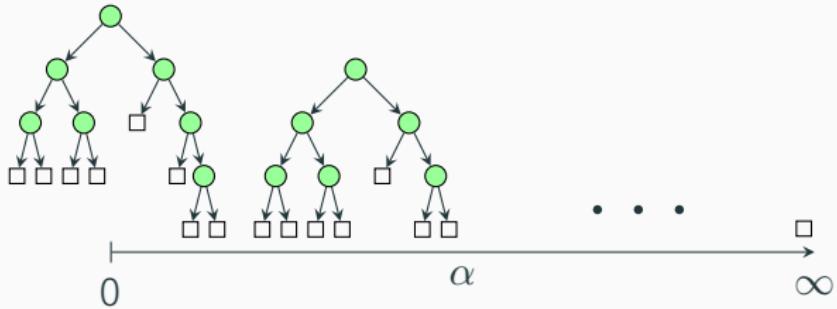


$$\sum_{i=0}^n (y_i - \hat{y}_i)^2 + \alpha|T|,$$

$|T|$ =number of leaf nodes



# Tree-based models: Decision trees

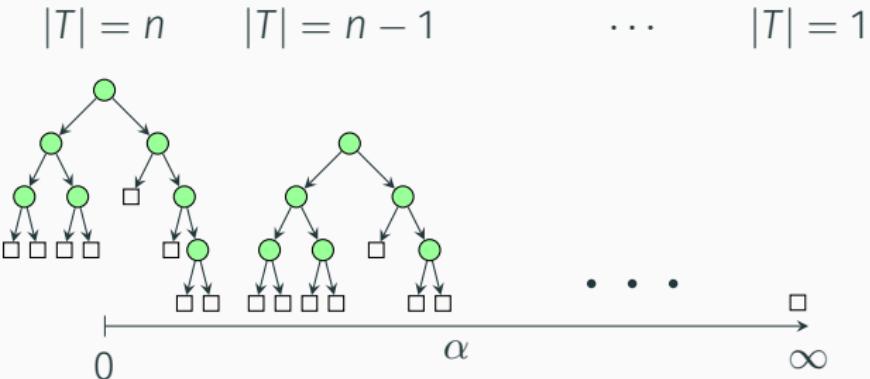


$$\sum_{i=0}^n (y_i - \hat{y}_i)^2 + \alpha|T|,$$

$|T|$ =number of leaf nodes



# Tree-based models: Decision trees

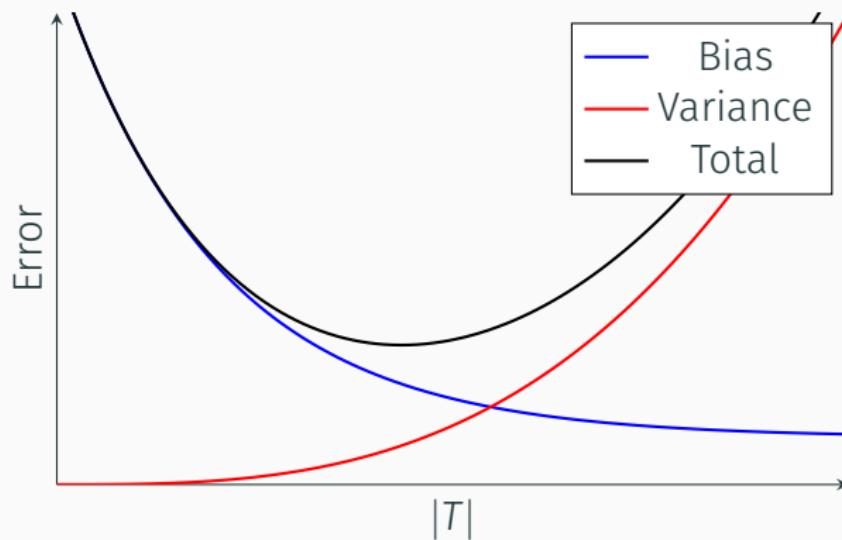


$$\sum_{i=0}^n (y_i - \hat{y}_i)^2 + \alpha|T|,$$

$|T|$ =number of leaf nodes



# Tree-based models: Decision trees



# Tree-based models: Decision trees

Decision trees: Segments the feature space into regions via the application of multiple binary splits.

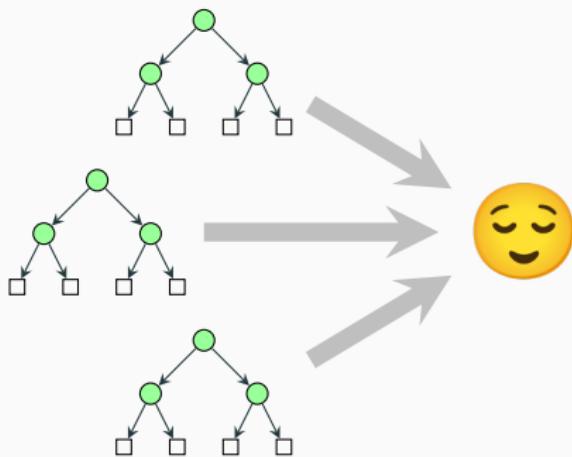
- Can be used for classification and regression
- + Very interpretable
- Quite bad
  - Prone to overfitting
  - Small changes in  $x$  can yield large differences in  $\hat{y}$
- Complexity can be controlled via limiting tree size during fitting
- Can be pruned afterwards to determine optimal size



# Tree-based methods: Ensembles



# Tree-based methods: Ensembles



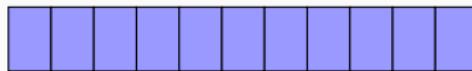
# Tree-based methods: Ensembles

Ensembling: Combining multiple weak learners can improve performance.

- A single prediction is achieved by averaging or majority voting
- Models should ideally be diverse



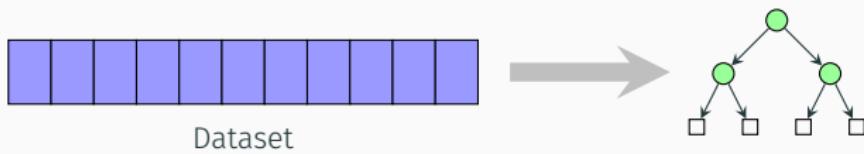
# Tree-based methods: Ensembles



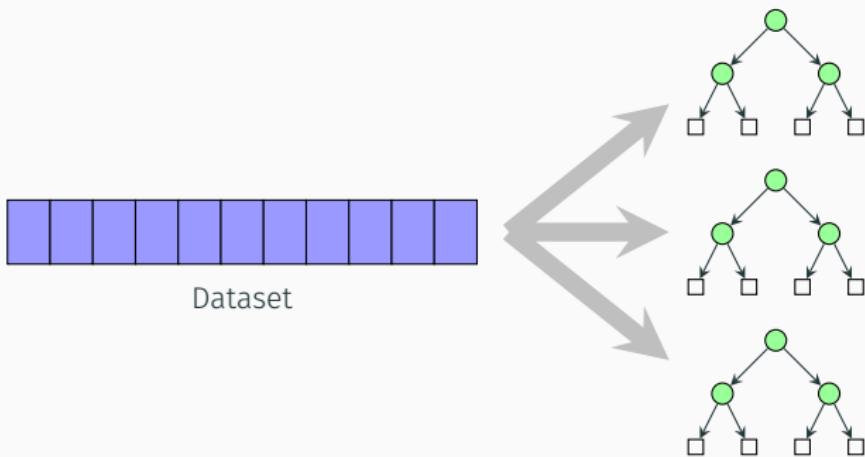
Dataset



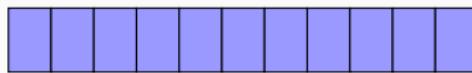
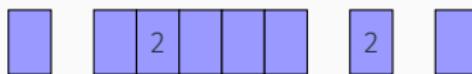
# Tree-based methods: Ensembles



# Tree-based methods: Ensembles



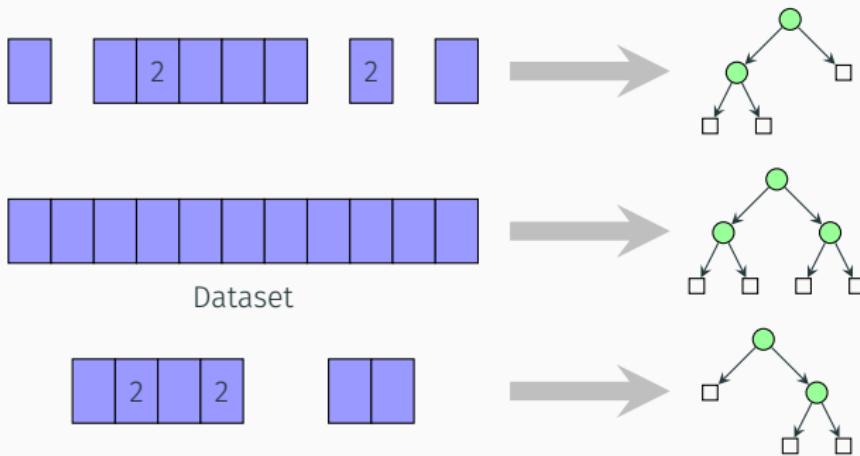
# Tree-based methods: Ensembles



Dataset



# Tree-based methods: Ensembles



# Tree-based methods: Ensembles

Bootstrap aggregation (bagging): Create a set of datasets by sampling with replacement, and train individual models on each.



# Tree-based methods: Ensembles

$x_1$	$x_2$	$x_3$	$x_4$	$x_5$	$x_6$	$y$
0.1	0.7	0.3	0.2	0.8	0.4	0.5
0.4	0.5	0.9	0.3	0.8	0.2	0.6
0.2	0.3	0.8	0.1	0.7	0.5	0.7
0.3	0.6	0.2	0.4	0.9	0.1	0.8
0.5	0.8	0.1	0.6	0.2	0.3	0.9
0.6	0.9	0.7	0.5	0.1	0.4	0.8



# Tree-based methods: Ensembles

$x_1$	$x_2$	$x_3$	$x_4$	$x_5$	$x_6$	$y$
0.1	0.7	0.3	0.2	0.8	0.4	0.5
0.4	0.5	0.9	0.3	0.8	0.2	0.6
0.2	0.3	0.8	0.1	0.7	0.5	0.7
0.3	0.6	0.2	0.4	0.9	0.1	0.8
0.5	0.8	0.1	0.6	0.2	0.3	0.9
0.6	0.9	0.7	0.5	0.1	0.4	0.8



# Tree-based methods: Ensembles

$x_1$	$x_2$	$x_3$	$x_4$	$x_5$	$x_6$	$y$
0.1	0.7	0.3	0.2	0.8	0.4	0.5
0.4	0.5	0.9	0.3	0.8	0.2	0.6
0.2	0.3	0.8	0.1	0.7	0.5	0.7
0.3	0.6	0.2	0.4	0.9	0.1	0.8
0.5	0.8	0.1	0.6	0.2	0.3	0.9
0.6	0.9	0.7	0.5	0.1	0.4	0.8



# Tree-based methods: Ensembles

$x_1$	$x_2$	$x_3$	$x_4$	$x_5$	$x_6$	$y$
0.1	0.7	0.3	0.2	0.8	0.4	0.5
0.4	0.5	0.9	0.3	0.8	0.2	0.6
0.2	0.3	0.8	0.1	0.7	0.5	0.7
0.3	0.6	0.2	0.4	0.9	0.1	0.8
0.5	0.8	0.1	0.6	0.2	0.3	0.9
0.6	0.9	0.7	0.5	0.1	0.4	0.8



# Tree-based methods: Ensembles

$x_1$	$x_2$	$x_3$	$x_4$	$x_5$	$x_6$	$y$
0.1	0.7	0.3	0.2	0.8	0.4	0.5
0.4	0.5	0.9	0.3	0.8	0.2	0.6
0.2	0.3	0.8	0.1	0.7	0.5	0.7
0.3	0.6	0.2	0.4	0.9	0.1	0.8
0.5	0.8	0.1	0.6	0.2	0.3	0.9
0.6	0.9	0.7	0.5	0.1	0.4	0.8



# Tree-based methods: Ensembles

$x_1$	$x_2$	$x_3$	$x_4$	$x_5$	$x_6$	$y$
0.1	0.7	0.3	0.2	0.8	0.4	0.5
0.4	0.5	0.9	0.3	0.8	0.2	0.6
0.2	0.3	0.8	0.1	0.7	0.5	0.7
0.3	0.6	0.2	0.4	0.9	0.1	0.8
0.5	0.8	0.1	0.6	0.2	0.3	0.9
0.6	0.9	0.7	0.5	0.1	0.4	0.8

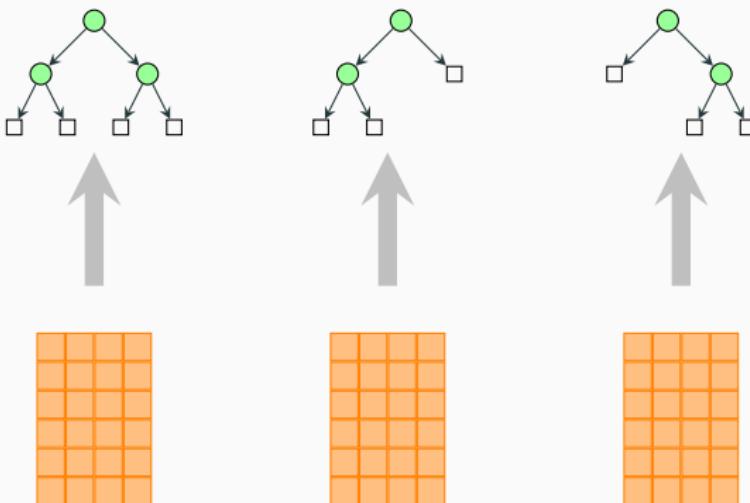


# Tree-based methods: Ensembles

$x_1$	$x_2$	$x_3$	$x_4$	$x_5$	$x_6$	$y$
0.1	0.7	0.3	0.2	0.8	0.4	0.5
0.4	0.5	0.9	0.3	0.8	0.2	0.6
0.2	0.3	0.8	0.1	0.7	0.5	0.7
0.3	0.6	0.2	0.4	0.9	0.1	0.8
0.5	0.8	0.1	0.6	0.2	0.3	0.9
0.6	0.9	0.7	0.5	0.1	0.4	0.8



# Tree-based methods: Ensembles



# Tree-based methods: Ensembles

Random forests: Fits multiple decision trees on random subsets of **both** predictors and observations.

- Produces more diverse models than bagging
- The number of predictors per tree are usually  $\approx \sqrt{p}$
- Requires us to select the number of trees. More trees  $\rightarrow$  better performance ( $\approx$  same bias, less variance)



# Tree-based methods: Boosting

<https://kaggle.com>



# Tree-based methods: Boosting

## 1st place solution

Thanks to Ennfit and Kaggle for hosting this great competition. And thanks to the great notebooks and discussions, I learned a lot. I have a feeling this comp is going to be slightly more volatile than optiver-one-and-wish-i-could-end-up-in-top-three. I am so happy to win my third solo win! 🎉🎉

### Overview

My final solution consists of 4 XGBoost models (for two targets and two `is_consumption`) and 2 GRU models (for two targets). These models share same 600 features.

model name	validation set	test set w/o online training	test set w/ update one time	test set w/ update three times
XGBoost	41.5912 (2 targets: 42.2531/42.8607)	55.6231	54.1343	54.1213

## [1st Place Solution] My Betting Strategy

### Foreword

Hello, this is yuuniee.

Thank you to the competition host Home Credit and staff, Kaggle staff and everyone involved.

The image below is a summary of my solution.

LGBM	DNN	CATBOOST
644 Features	661 Features	661 Features
CV : 0.707	CV : 0.699	CV : 0.706
Public LB : 0.58	Public LB : 0.56	Public LB : 0.59
0.4	0.24	0.36

## 1st place solution

Thanks to Optiver and Kaggle for hosting this great financial competition. And thanks to the great notebooks and discussions, I learned a lot. I am so happy to win my second solo win! 🎉🎉

### Overview

My final model [CV/Private LB of 5.8117/5.4030] was a combination of CatBoost (5.8240/5.4165), GRU (5.8440/5.4250), and Transformer (5.8610/5.4795), with respective weights of 0.5, 0.3, 0.2 searched from validation set. And these models share same 300 features.

Besides, online learning[DL] and post-processing[PP] also play an important role in my final submission.

model name	validation set w/o PP	validation set w/ PP	test set w/o OL w/o PP	test set w/ OL one time w/ PP	test set w/ OL five times w/ PP
CatBoost	5.8287	5.8240	5.4523	5.4291	5.4165



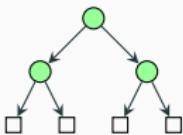
# Tree-based methods: Boosting

<i>Method</i>	RMSE
CatBoost	<b>3.00 ± 0.91</b>
XGBoost	3.25 ± 0.63
NPT	3.25 ± 1.31
Gradient Boosting	4.00 ± 1.08
Random Forest	4.50 ± 0.87
MLP	5.00 ± 1.22
LightGBM	6.50 ± 1.55
TabNet	6.75 ± 0.95
k-NN	8.75 ± 0.25

Self-Attention Between Datapoints: Going Beyond Individual Input-Output Pairs in Deep Learning, Kossen et al., preprint at arXiv, 2022



# Tree-based methods: Boosting



$$f_0 : \min_f \sum (y_i - f(x_i))^2$$



# Tree-based methods: Boosting

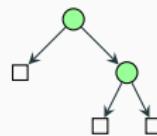
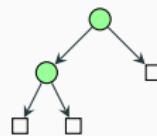
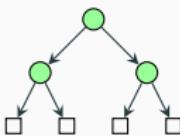


$$f_0 : \min_f \sum (y_i - f(x_i))^2$$

$$f_1 : \min_f \sum (y_i - f(x_i))^2$$



# Tree-based methods: Boosting

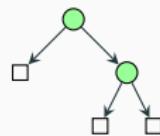
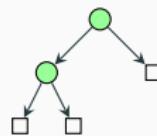
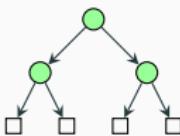


$$f_0 : \min_f \sum (y_i - f(x_i))^2$$

$$f_1 : \min_f \sum (y_i - f(x_i))^2$$

$$f_2 : \min_f \sum (y_i - f(x_i))^2$$

# Tree-based methods: Boosting

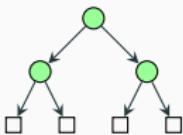


$$f_0 : \min_f \sum (y_i - f(x_i))^2 \quad f_1 : \min_f \sum (y_i - f(x_i))^2 \quad f_2 : \min_f \sum (y_i - f(x_i))^2$$

$$\hat{y} = \frac{1}{3} \sum_{i=0}^3 f_i(x)$$



# Tree-based methods: Boosting



$$f_0 : \min_f \sum (y_i - f(x_i))^2$$

$$\hat{f}(x) = \lambda f_0(x)$$

$$y^* = y - (\hat{f}(x))$$



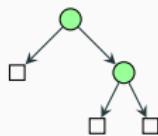
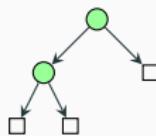
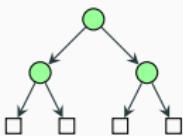
# Tree-based methods: Boosting



$$\begin{array}{ll} f_0 : \min_f \sum (y_i - f(x_i))^2 & f_1 : \min_f \sum (y_{*i} - f(x_i))^2 \\ \hat{f}(x) = \lambda f_0(x) & \hat{f}(x) = \lambda(f_0(x) + f_1(x)) \\ y* = y - (\hat{f}(x)) & y* = y - (\hat{f}(x)) \end{array}$$



# Tree-based methods: Boosting



$$f_0 : \min_f \sum (y_i - f(x_i))^2$$

$$\hat{f}(x) = \lambda f_0(x)$$

$$y^* = y - (\hat{f}(x))$$

$$f_1 : \min_f \sum (y_{*i} - f(x_i))^2$$

$$\hat{f}(x) = \lambda(f_0(x) + f_1(x))$$

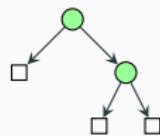
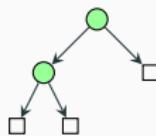
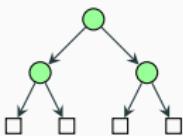
$$y^* = y - (\hat{f}(x))$$

$$f_2 : \min_f \sum (y_{*i} - f(x_i))^2$$

$$\hat{f}(x) = \lambda(f_0(x) + f_1(x) + f_2(x))$$

$$y^* = y - (\hat{f}(x))$$

# Tree-based methods: Boosting



$$f_0 : \min_f \sum (y_i - f(x_i))^2$$

$$\hat{f}(x) = \lambda f_0(x)$$

$$y^* = y - (\hat{f}(x))$$

$$f_1 : \min_f \sum (y_{*i} - f(x_i))^2$$

$$\hat{f}(x) = \lambda(f_0(x) + f_1(x))$$

$$y^* = y - (\hat{f}(x))$$

$$f_2 : \min_f \sum (y_{*i} - f(x_i))^2$$

$$\hat{f}(x) = \lambda(f_0(x) + f_1(x) + f_2(x))$$

$$y^* = y - (\hat{f}(x))$$

# Tree-based methods: Boosting

Boosting: Fits a sequence of trees  $t_0, \dots, t_n$ . Each tree  $t_i$  tried to predict the residual of the preceding model, e.g. finding the signal that the model so far has missed.

- Works *really* well



# Tree-based methods: Boosting

<https://xgboost.readthedocs.io/en/stable/parameter.html#parameters-for-tree-booster>

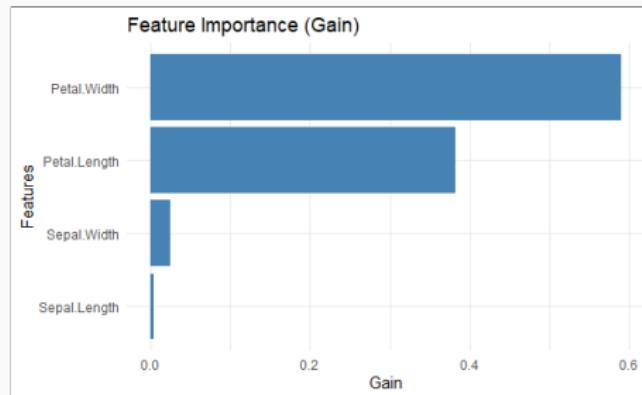


# Tree-based methods: Boosting

[https://xgboost.readthedocs.io/en/latest/python/examples/cross\\_validation.html](https://xgboost.readthedocs.io/en/latest/python/examples/cross_validation.html)



# Tree-based methods: Boosting



[https://xgboost.readthedocs.io/en/stable/python/python\\_api.html#module-xgboost.plotting](https://xgboost.readthedocs.io/en/stable/python/python_api.html#module-xgboost.plotting)



## Exercise 5

1. Download the Credit.csv dataset from ISL:  
<https://www.statlearning.com/s/Credit.csv>
2. Preprocess the dataset as you see fit.
3. Fit non-linear models (GLM, GAM, RF, XGBoost) to predict *Balance* from the other variables in the dataset. **You are not allowed to use the *Income* variable as a predictor, due to its high collinearity with *Balance*.** Whether you fit one or more types of models is up to you, the same goes for the choice of hyperparameters and how you find them. However, you should aim for achieving good predictive performance, and being relatively certain the performance holds up in new data. In other words, using a train/validation/test-split or a (nested) cross-validation is probably a good idea. When you are satisfied, report model performance in a comprehensible manner.
4. Reflect on the choices you made: Why did you choose the model(s) you did? What about the hyperparameters? How did you tune them?
5. Reflect upon the performance of the model(s): How well did it/they perform? What would you expect the performance to be on new data? How can you argue this is the case? What other choices could have made you more or less certain about this?

