PSY9511: Seminar 5

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

Esten H. Leonardsen

24.10.24

UNIVERSITETET
I OSLO

Outline

Outline

Exercise solutions

https://uio.instructure.com/courses/54846

Exercise 3

UNIVERSITETET
I OSLO

Exercise 3: Standardization

Why do we standardize the validation set based on information from the training set?

Exercise 3: Standardization

Why do we standardize the validation set based on information from the training set?

- To avoid data leakage
- To minimize the impact of distributional differences

Exercise 3: Standardization

Train

x	y
-1	1.1
1	1.5
3	1.7

Validation

x	y
2	1.9
1	1.5
5	2.4

Exercise 3: Standardization

Train

x	y
-1	1.1
1	1.5
3	1.7

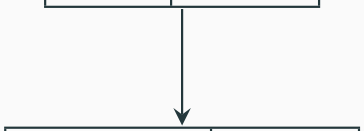
Validation

x	y
2	1.9
1	1.5
5	2.4

Exercise 3: Standardization

Train

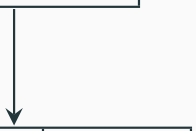
x	y
-1	1.1
1	1.5
3	1.7



x	y
-1.22	1.1
0	1.5
1.22	1.7

Validation

x	y
2	1.9
1	1.5
5	2.4

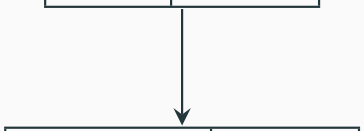


x	y
-0.39	1.9
-0.98	1.5
1.37	2.4

Exercise 3: Standardization

Train

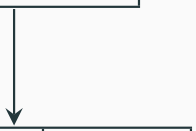
x	y
-1	1.1
1	1.5
3	1.7



x	y
-1.22	1.1
0	1.5
1.22	1.7

Validation

x	y
2	1.9
1	1.5
5	2.4



x	y
-0.39	1.9
-0.98	1.5
1.37	2.4

Exercise 3: Solution

http://localhost:8888/notebooks/notebooks/Solution%203.ipynb

Exercise 4

UNIVERSITETET
I OSLO

Exercise 4: Stratification

http://localhost:8888/notebooks/notebooks%2FStratification.ipynb

Exercise 4: Docstrings

https://numpydoc.readthedocs.io/en/latest/format.html

Exercise 4: Solution

http://localhost:8888/notebooks/notebooks/Solution%204.ipynb

Recap

UNIVERSITETET
I OSLO

Recap

Whenever a modelling choice is made on the basis of performance in a dataset, **we have to assume the performance achieved by the chosen model is inflated**

! Don't use a single train/validation split, cross-validation or bootstrapping with only train/validation sets to select the best model and then reports its performance

→ Hold out a test set

→ Use a nested cross-validation

→ Bootstrap train/validation/test sets

Extensions of linear models

UNIVERSITETET
I OSLO

Non-linear models: Nothing to worry about

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

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

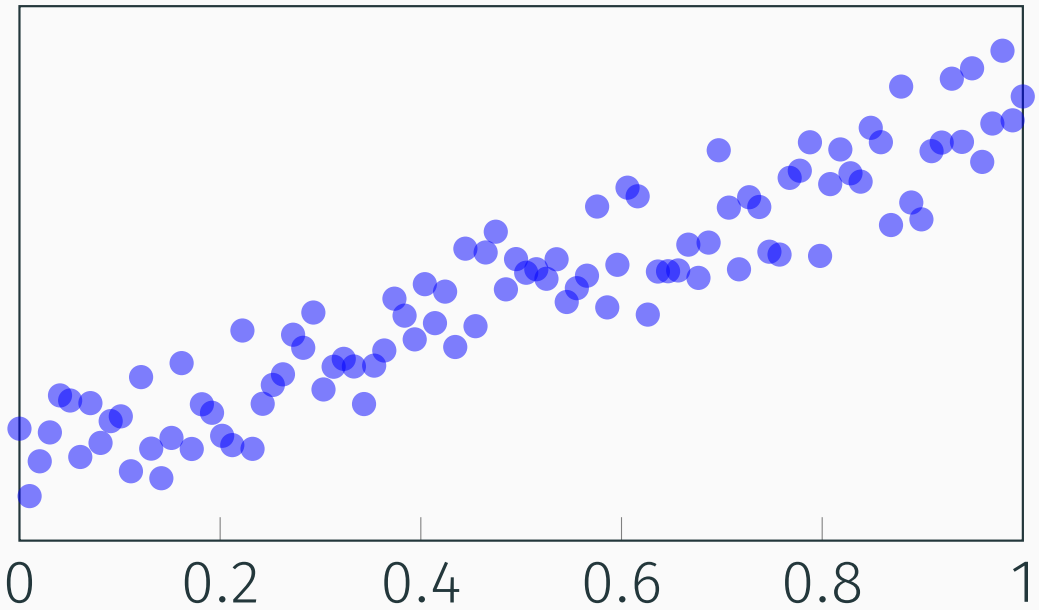
In[2]:

```python
from xgboost import XGBClassifier

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

Extensions of linear models: Generalized linear models

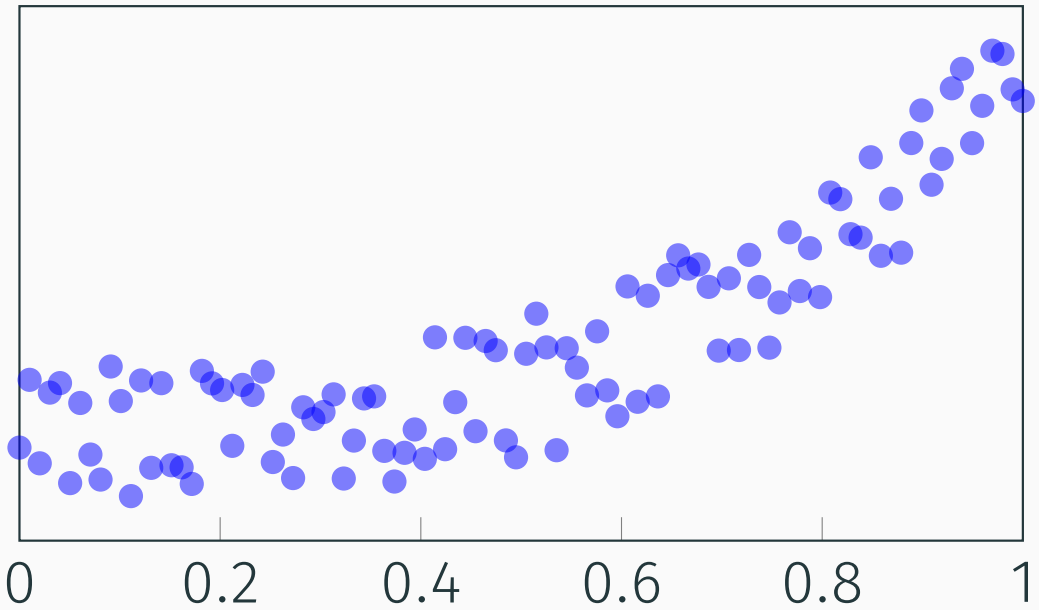
$$\hat{y} = \beta_0 + \sum_{i=0}^{p} \beta_i x_i$$

Extensions of linear models: Generalized linear models



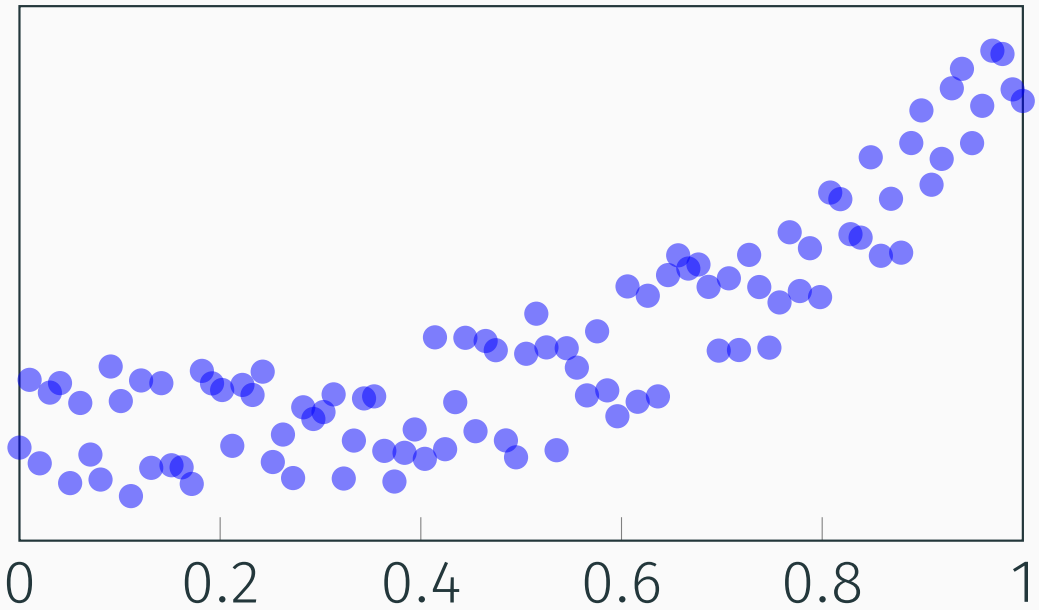
$$\hat{y} = \beta_0 + \sum_{i=0}^{p} \beta_i x_i$$

Extensions of linear models: Generalized linear models



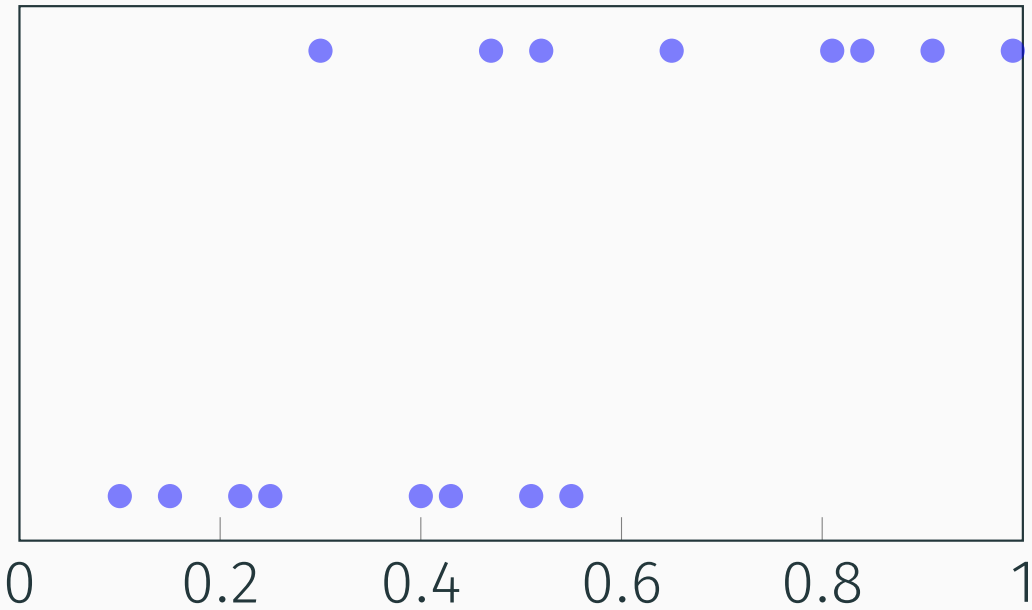
$$\hat{y} = \beta_0 + \sum_{i=0}^{p} \beta_i x_i$$

Extensions of linear models: Generalized linear models



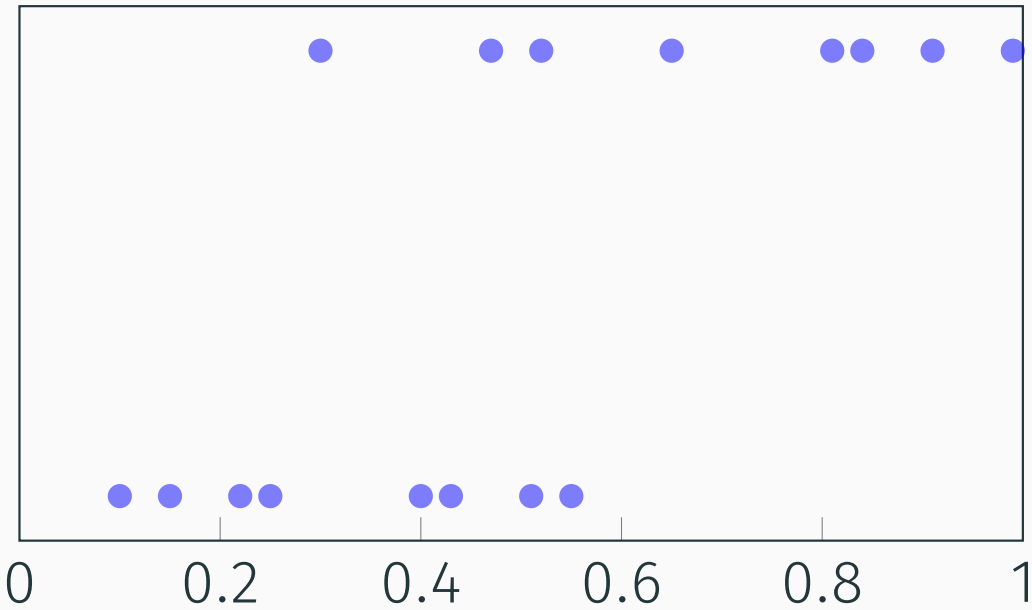
$$\hat{y} = \beta_0 + \sum_{i=0}^{p} \beta_i x_i$$

Extensions of linear models: Generalized linear models



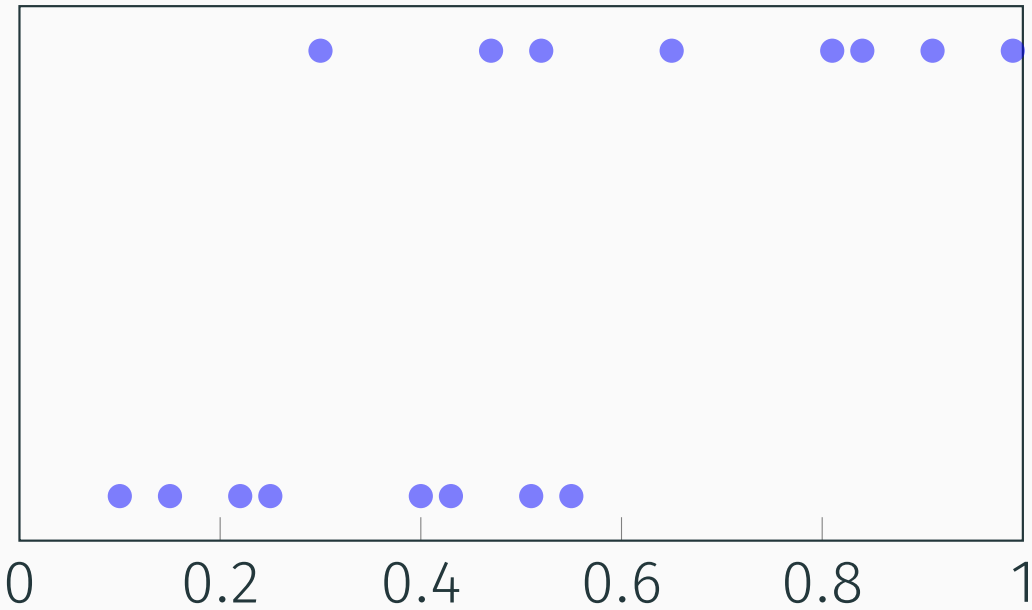
$$\hat{y} = \beta_0 + \sum_{i=0}^{p} \beta_i x_i$$

Extensions of linear models: Generalized linear models



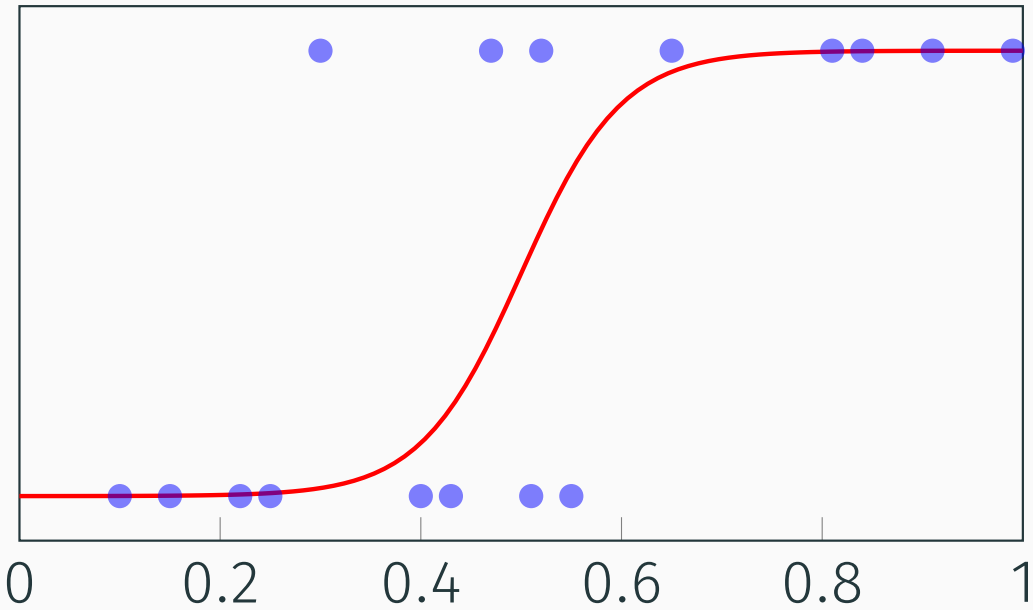
$$\log\left(\frac{p(X)}{1-p(X)}\right) = \beta_0 + \sum_{i=0}^{p} \beta_i x_i$$

Extensions of linear models: Generalized linear models



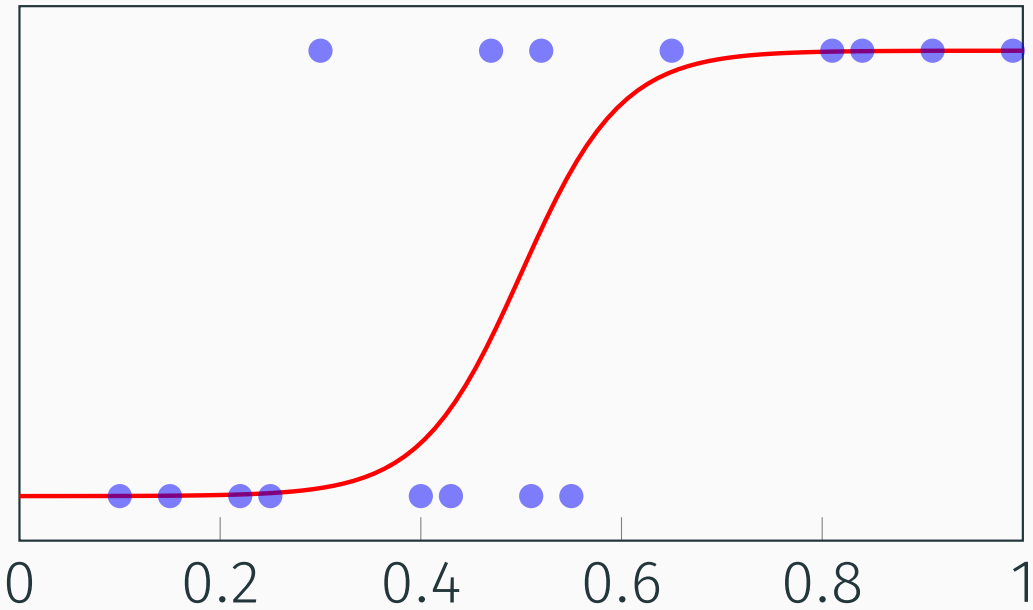
$$p(X) = \frac{e^{\left(\beta_0 + \sum_{i=0}^{p} \beta_i x_i\right)}}{1 + e^{\left(\beta_0 + \sum_{i=0}^{p} \beta_i x_i\right)}}$$

Extensions of linear models: Generalized linear models



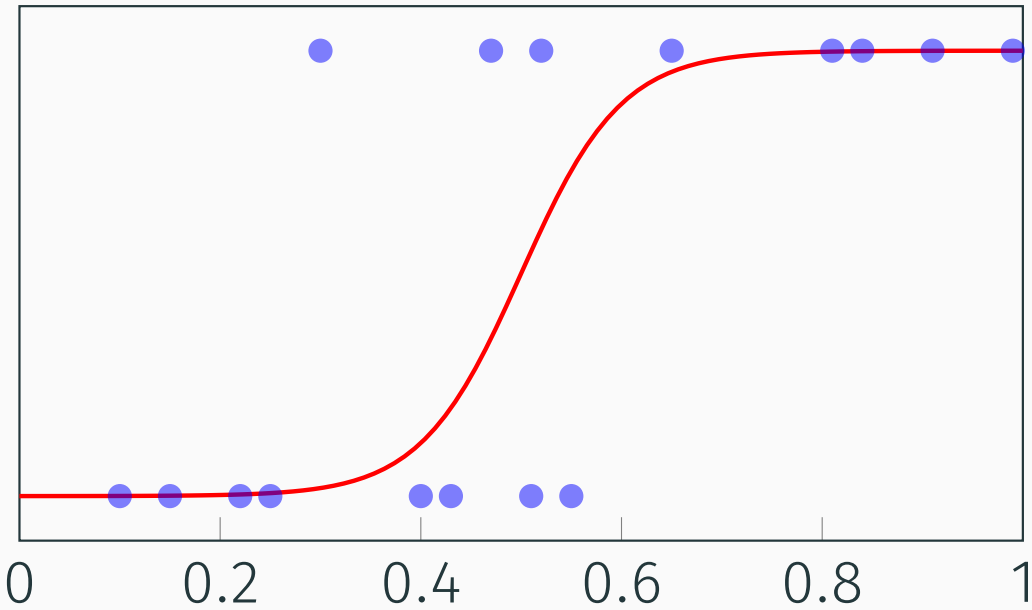
$$p(X) = \frac{e^{\left(\beta_0 + \sum_{i=0}^{p} \beta_i x_i\right)}}{1 + e^{\left(\beta_0 + \sum_{i=0}^{p} \beta_i x_i\right)}}$$

Extensions of linear models: Generalized linear models



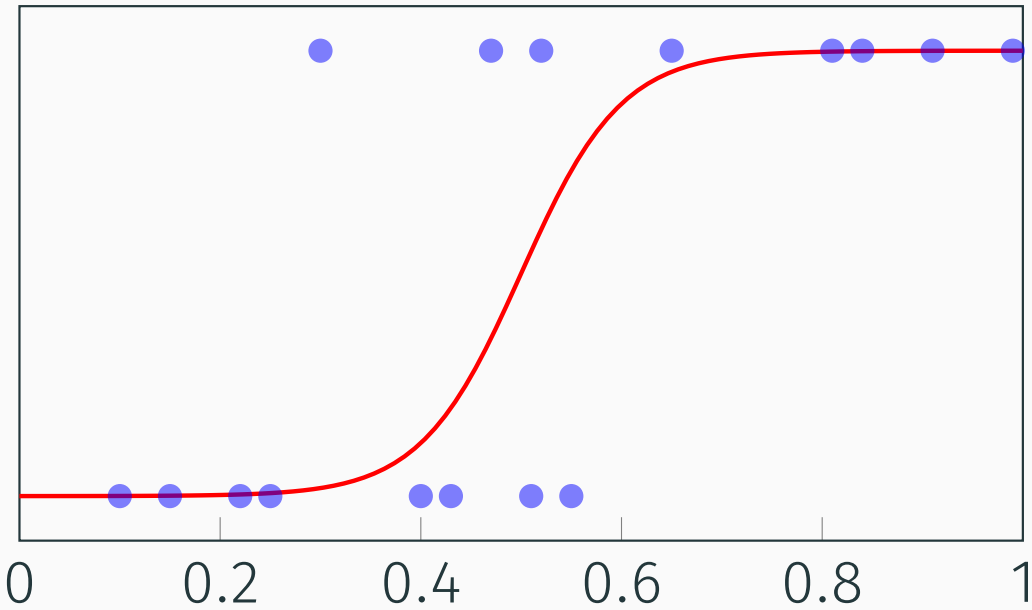
$$p(X) = \frac{e^{\left(\beta_0 + \sum_{i=0}^{p} \beta_i x_i\right)}}{1 + e^{\left(\beta_0 + \sum_{i=0}^{p} \beta_i x_i\right)}}$$

Extensions of linear models: Generalized linear models



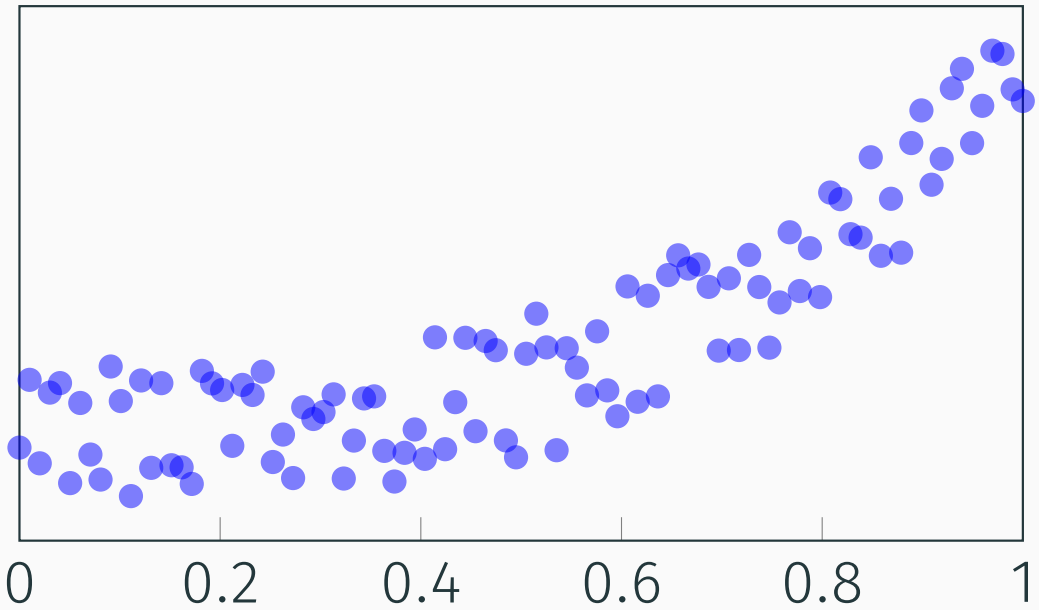
$$p(X) = f(\beta_0 + \sum_{i=0}^{p} \beta_i x_i)$$

Extensions of linear models: Generalized linear models



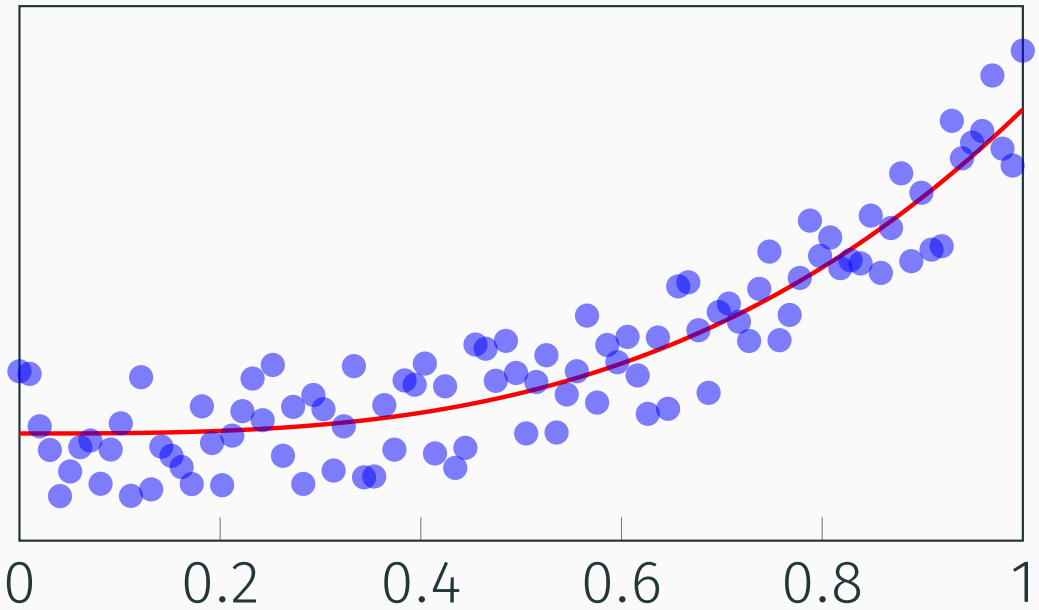
$$f(\hat{y}) = \beta_0 + \sum_{i=0}^{p} \beta_i x_i$$

Extensions of linear models: Generalized linear models



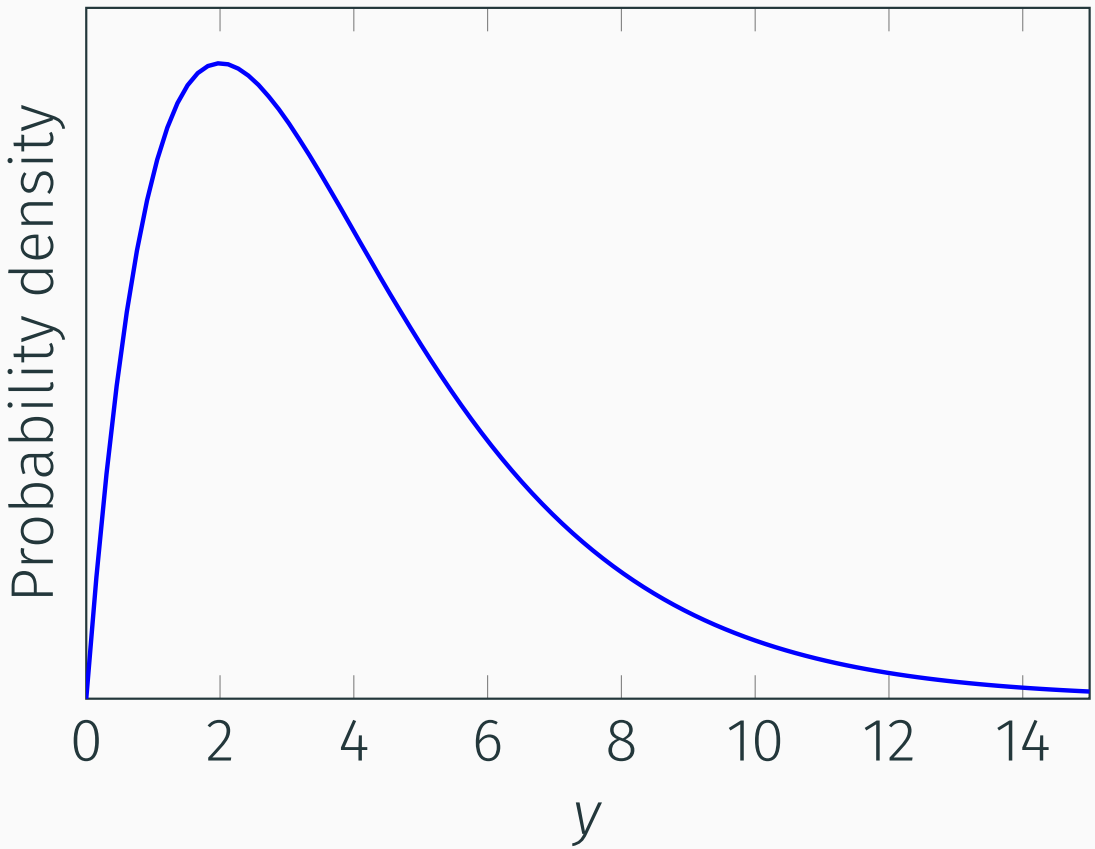
$$f(\hat{y}) = \beta_0 + \sum_{i=0}^{p} \beta_i x_i$$

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



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).

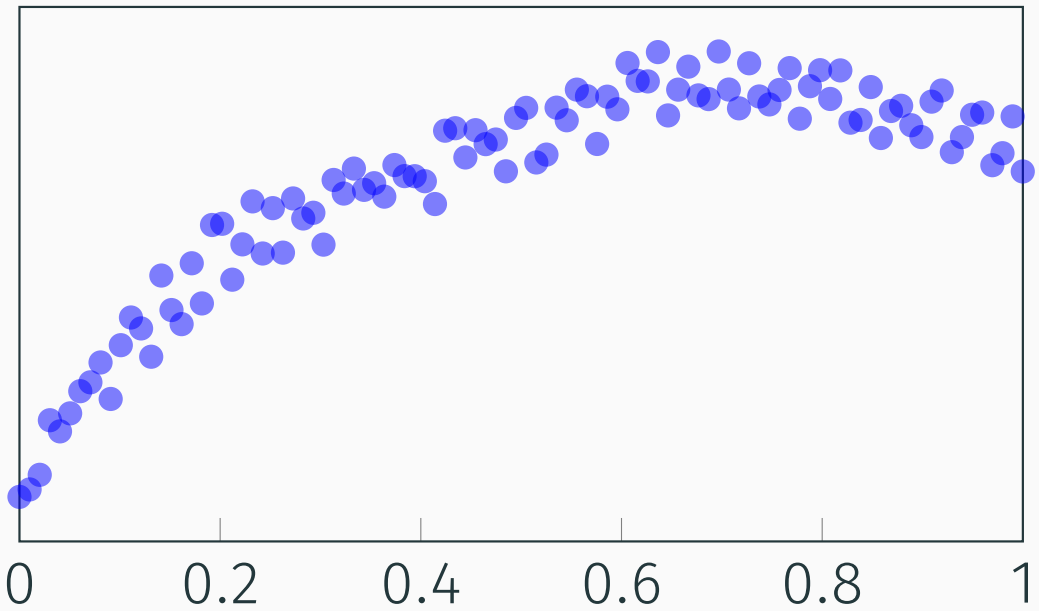
Extensions of linear models: Generalized linear models

In[1]:
```python
from sklearn.linear_model import GammaRegressor

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

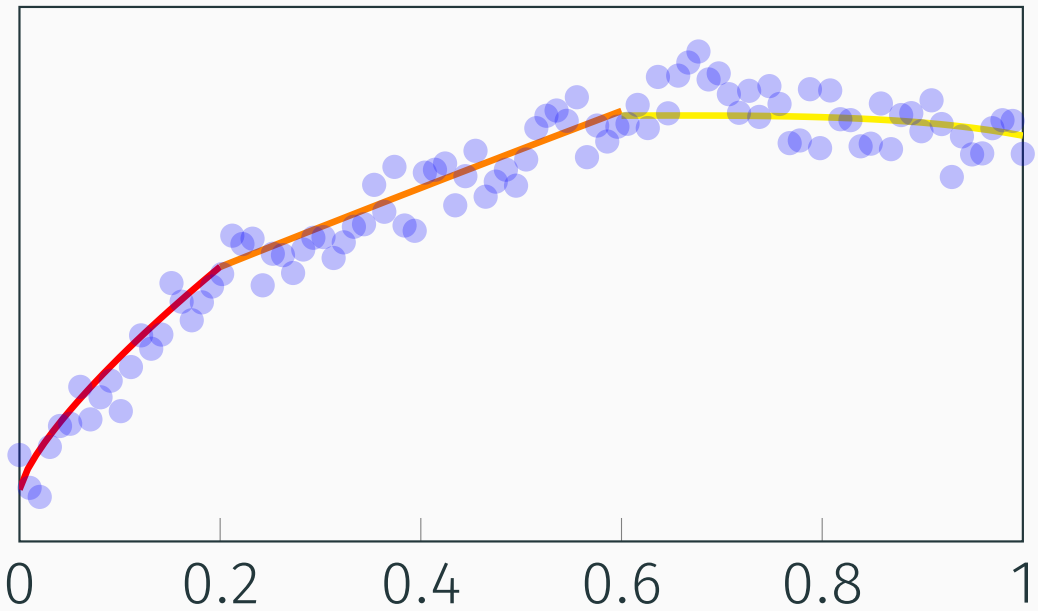
In[2]:
```python
import statsmodels.api as sm

link = sm.genmod.families.links.Log()
model = sm.GLM(y, X, family=sm.families.Gamma(link=link))
model.fit()
```

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

Extensions of linear models: Generalized additive models



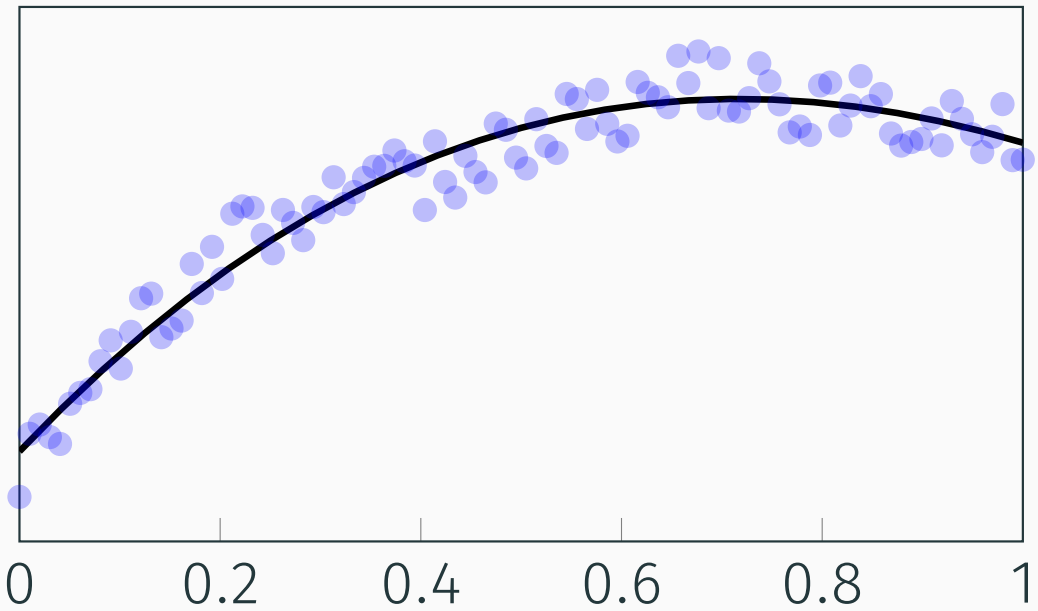
Extensions of linear models: Generalized additive models



Piecewise polynomial functions:

- Regression splines (ISL, Chapter 7.4)

Extensions of linear models: Generalized additive models



A single, complex, polynomial function:

- 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

$$\hat{y}_i = g(x_i)$$

$$\downarrow$$

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

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

$$\hat{y} = \beta_0 + \beta_1 x \qquad\qquad \hat{y} = g(x)$$

Extensions of linear models: Generalized additive models

$$\hat{y} = \beta_0 + \beta_1 x \qquad \hat{y} = g(x)$$

$$\downarrow \qquad \downarrow$$

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

Extensions of linear models: Generalized additive models

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.

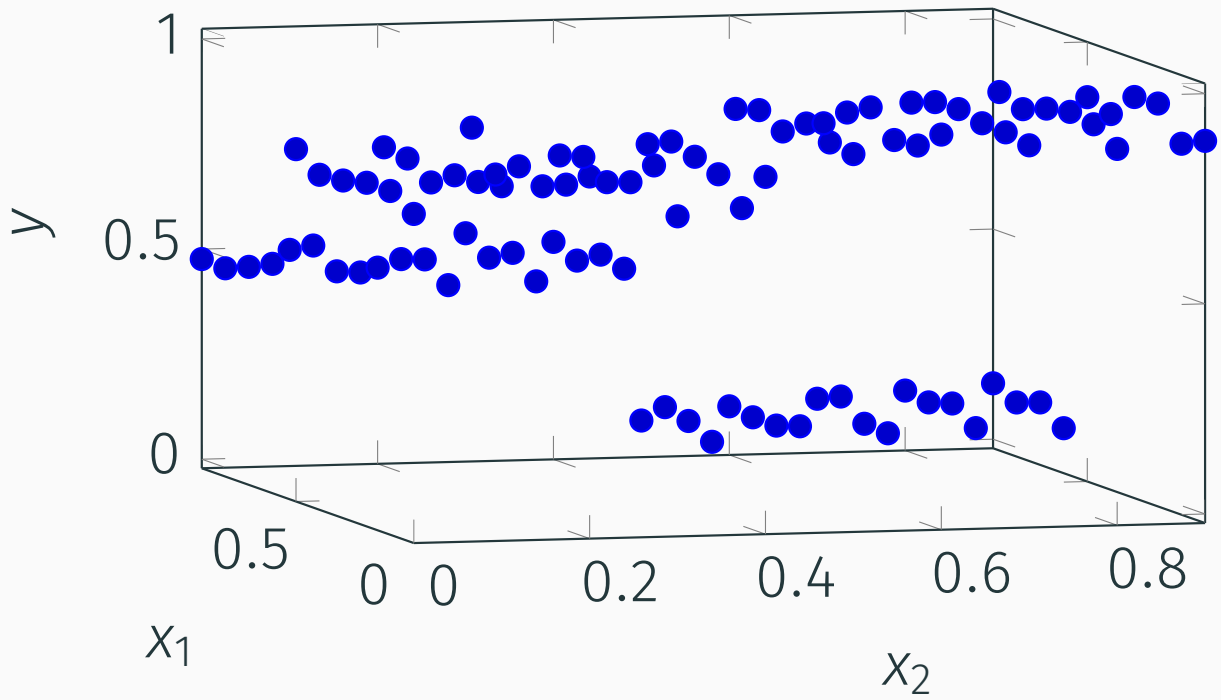
Extensions of linear models: Generalized additive models

scripts/gam.R

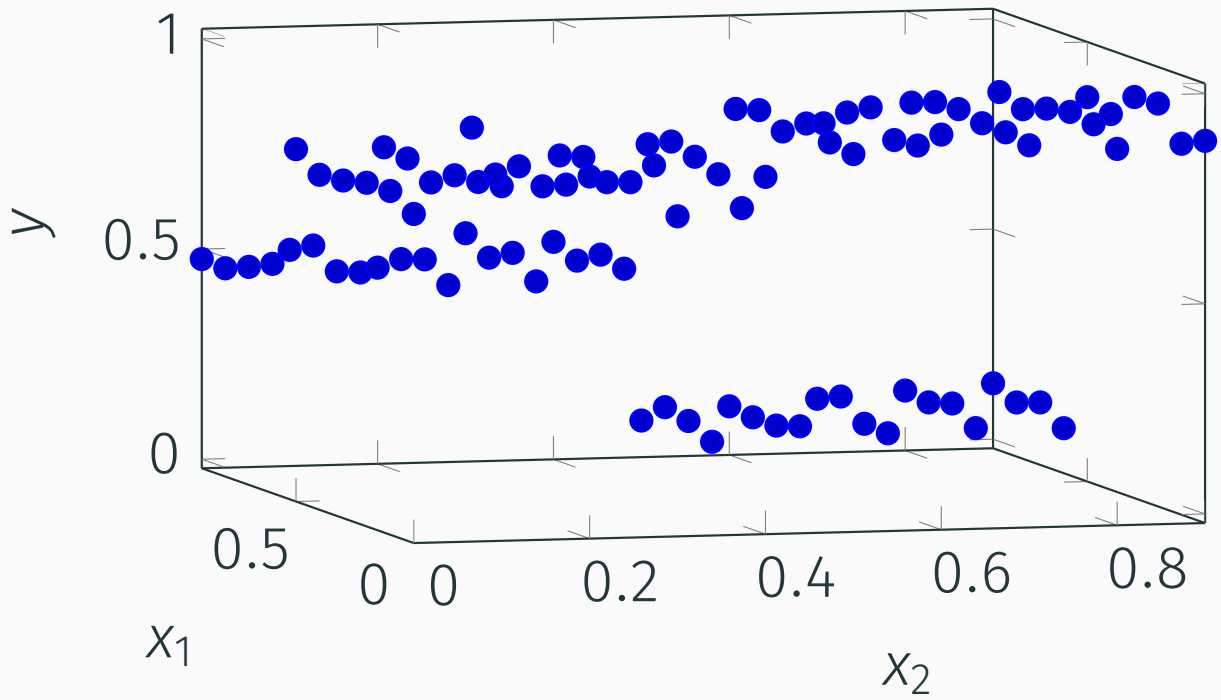
Tree-based models

UNIVERSITETET
I OSLO

Tree-based models: Motivation

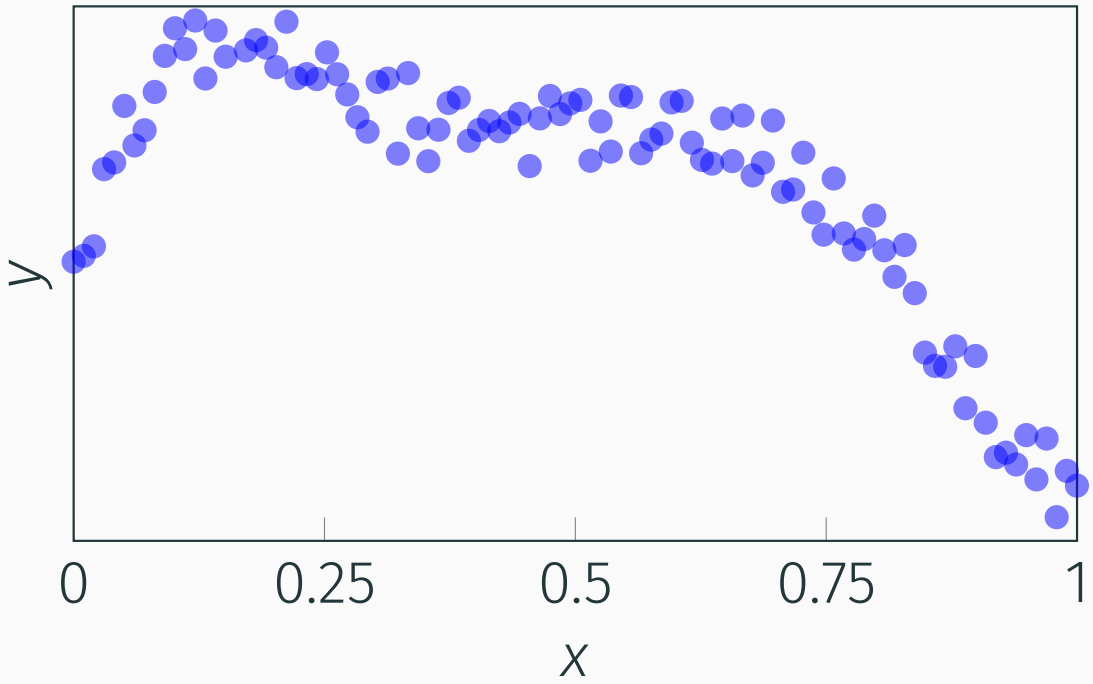


Tree-based models: Motivation

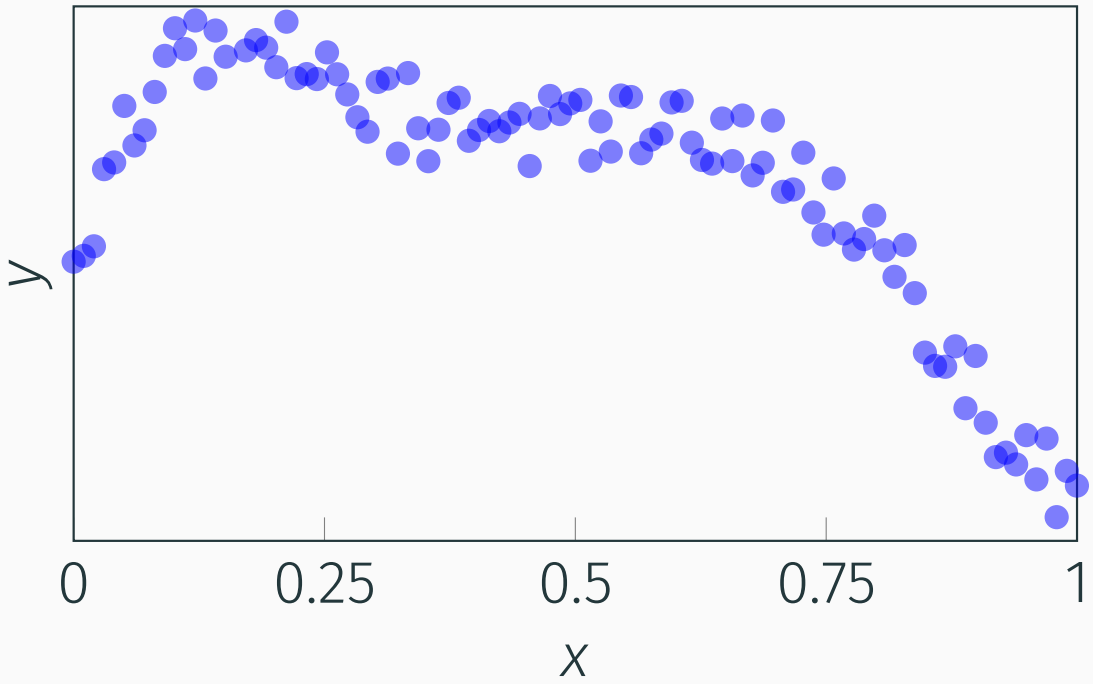


$$y = \begin{cases} 0.8 & x_1 \leq 0.6 \;\&\; x_2 \leq 0.5 \\ 0.9 & x_1 \leq 0.6 \;\&\; x_2 > 0.5 \\ 0.5 & x_1 > 0.5 \;\&\; x_2 \leq 0.5 \\ 0.1 & x_1 > 0.5 \;\&\; x_2 > 0.5 \end{cases}$$

Tree-based models: Motivation

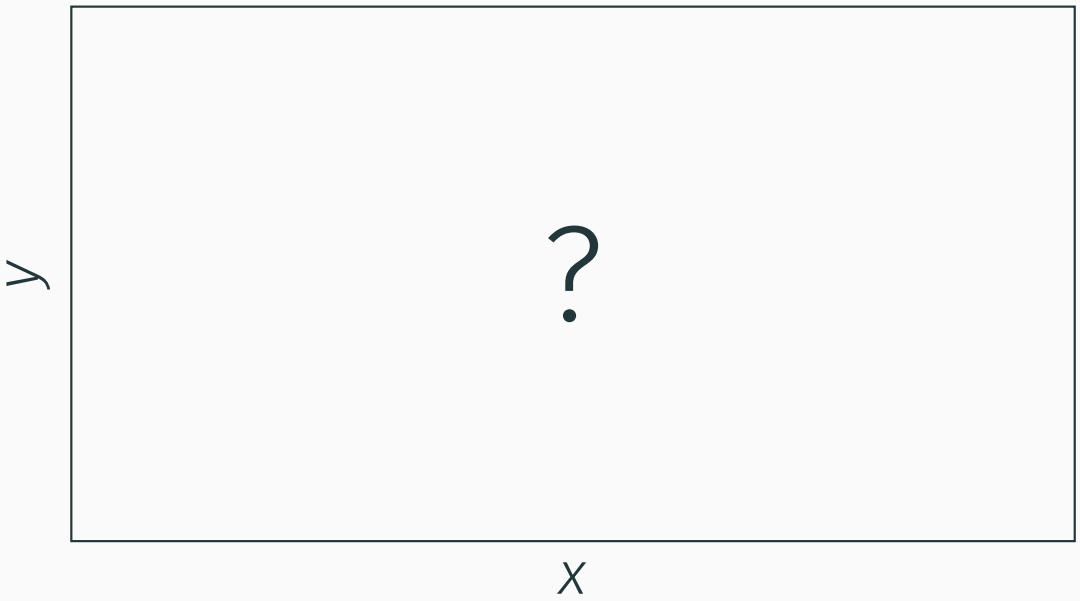


Tree-based models: Motivation

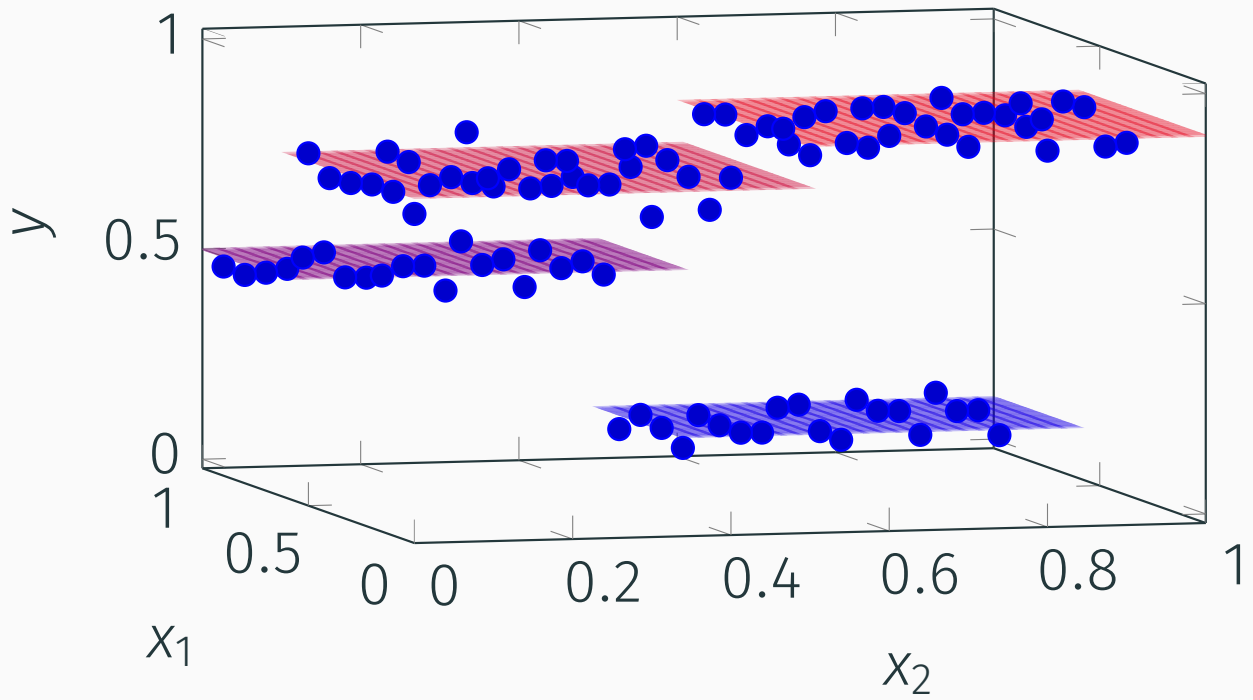


$$y = ?x^5 + ?x^4 + ?x^3 + ?x^2 + ?x + ?$$

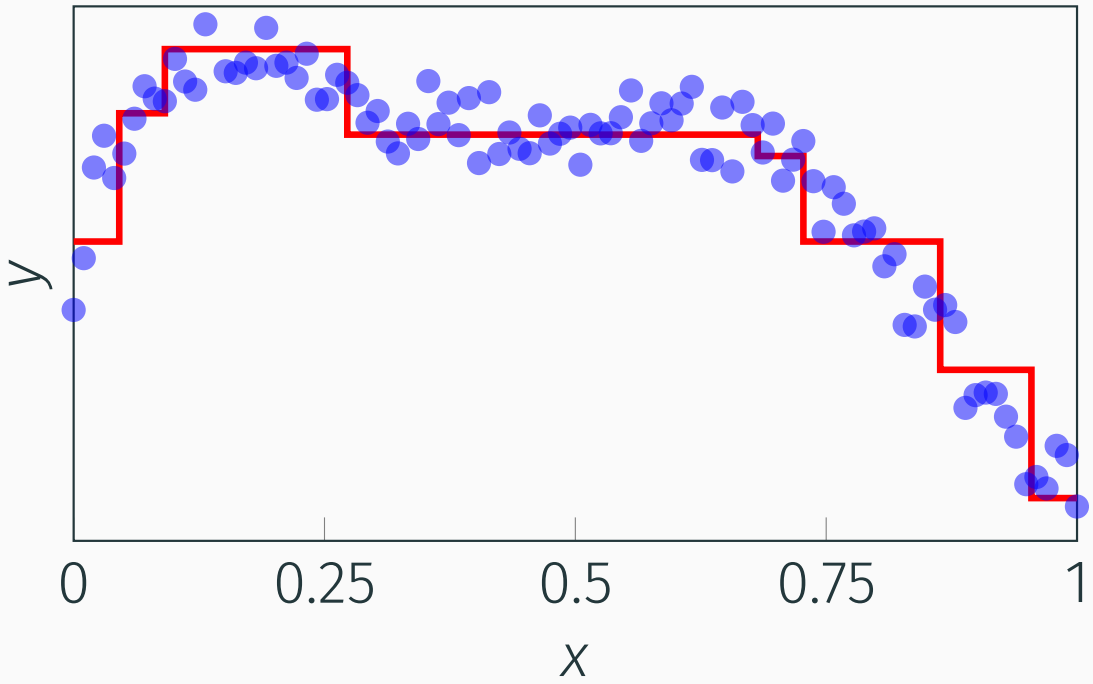
Tree-based models: Motivation



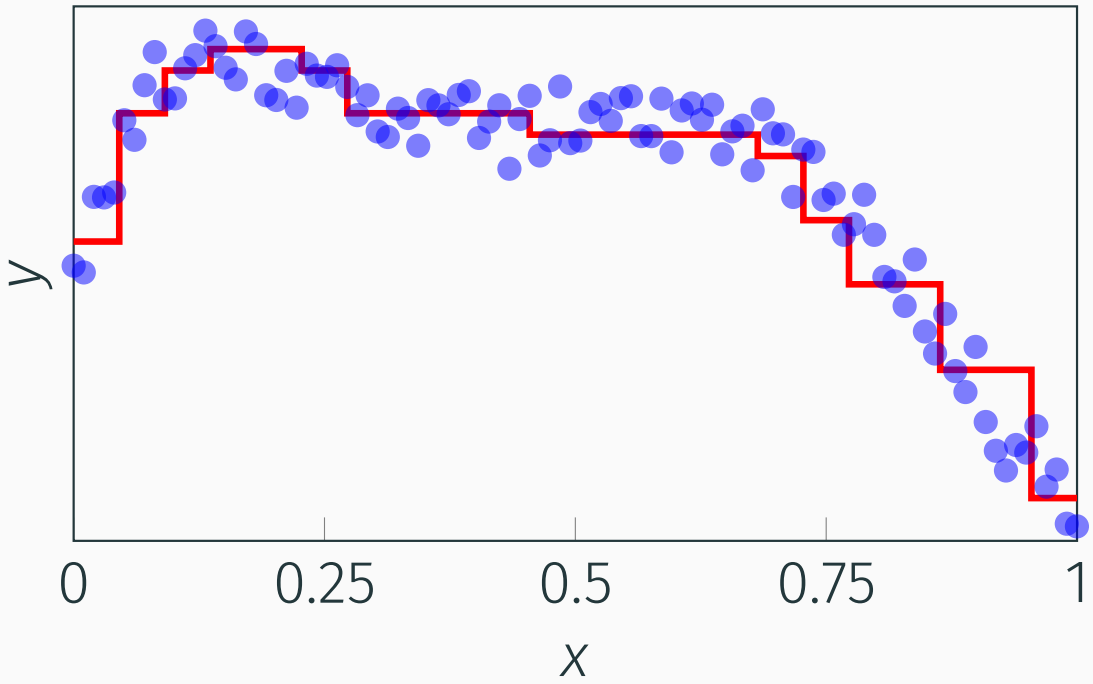
Tree-based models: Motivation



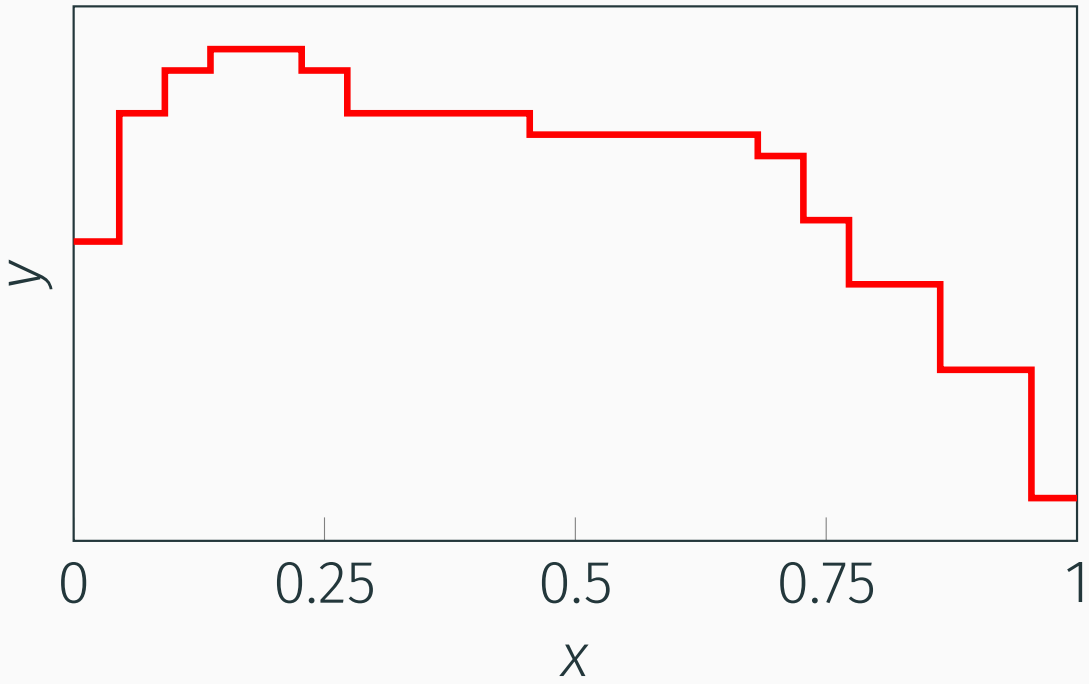
Tree-based models: Motivation



Tree-based models: Motivation

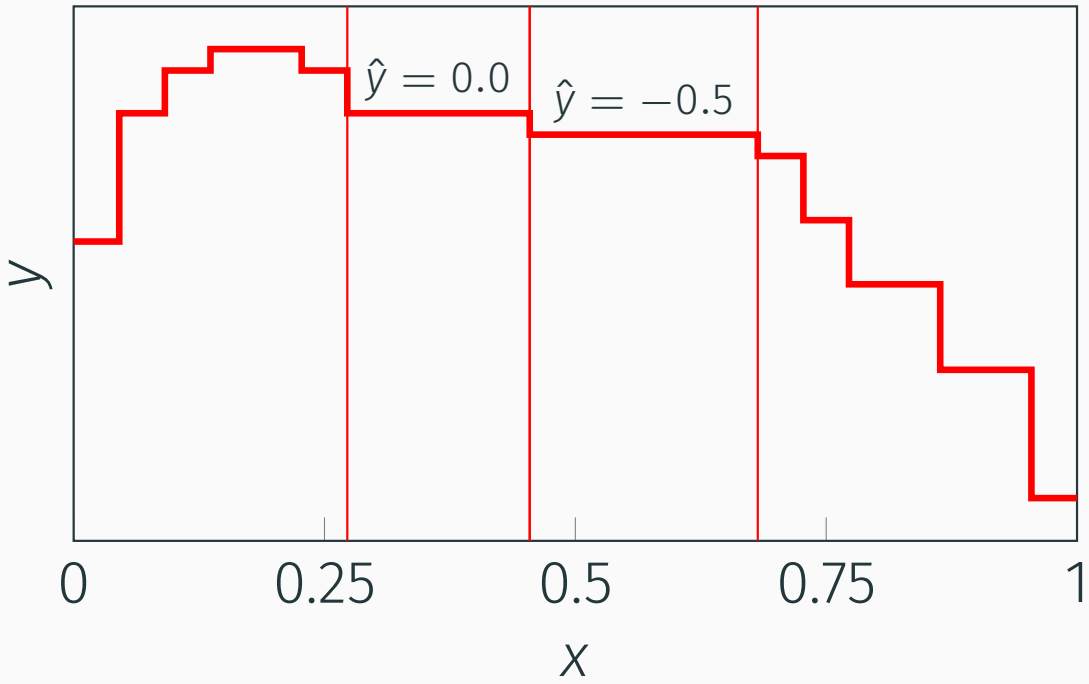


Tree-based models: Motivation

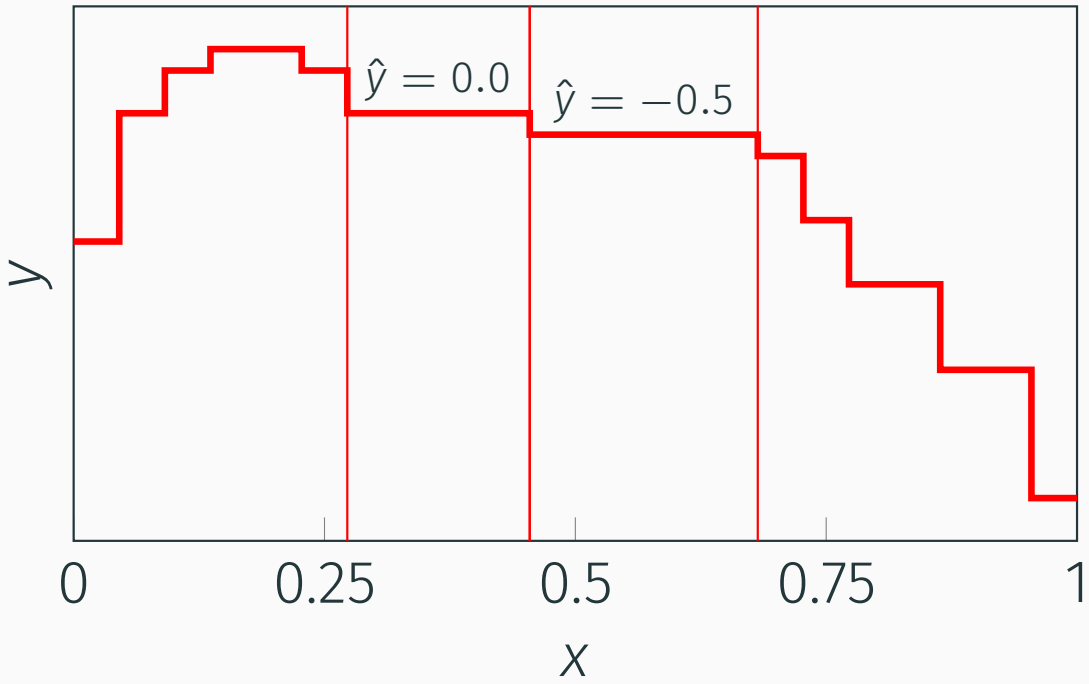


Piecewise constant function

Tree-based models: Motivation

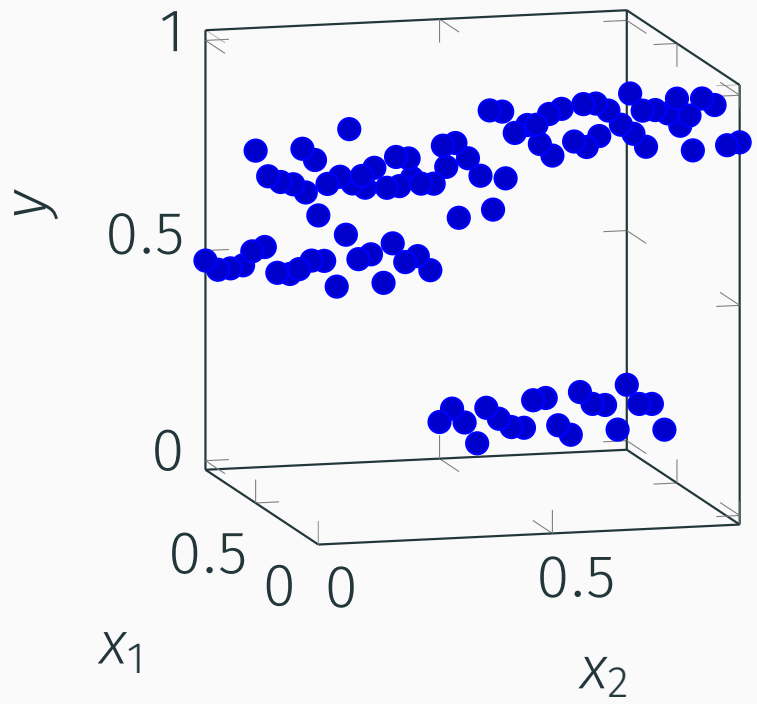


Tree-based models: Motivation

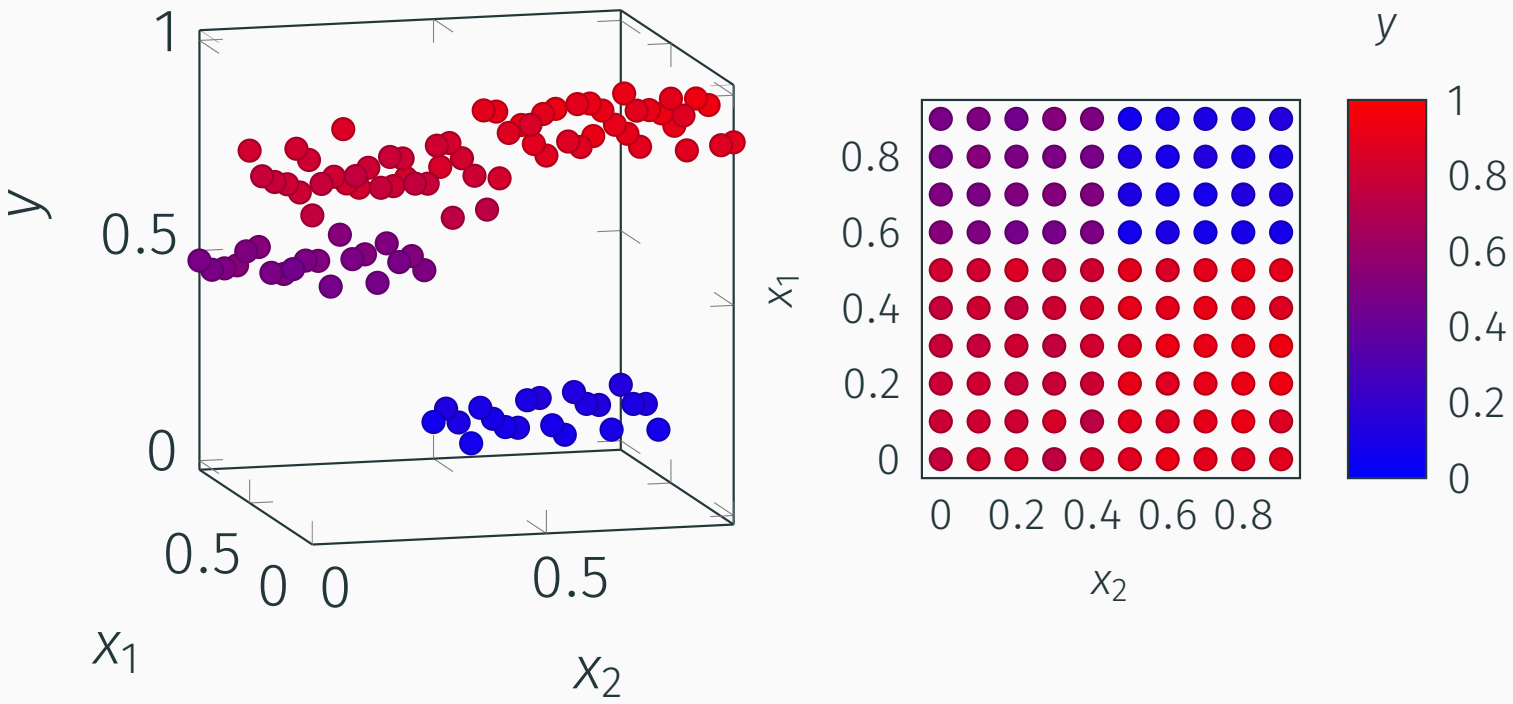


$$\hat{y} = \begin{cases} \dots & \\ 0.0 & x \geq 0.27 \ \& \ x < 0.45 \\ -0.5 & x \geq 0.45 \ \& \ x < 0.69 \\ \dots & \end{cases}$$

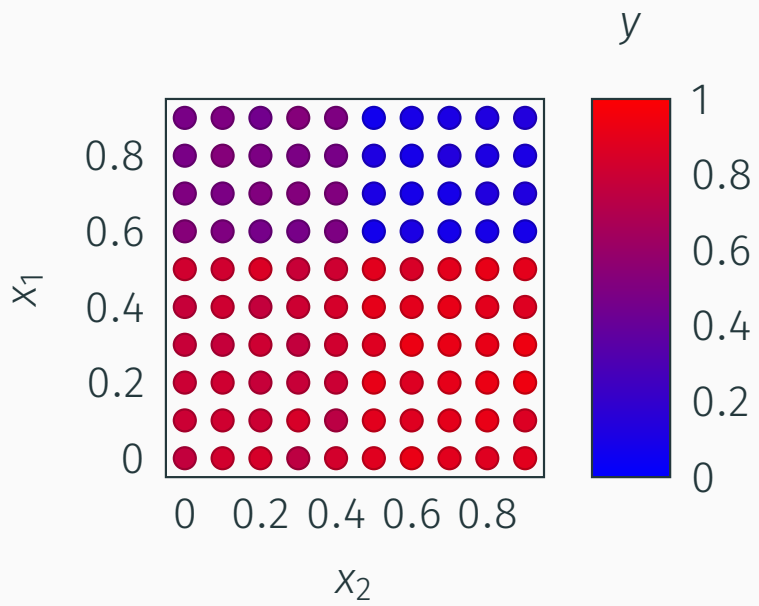
Tree-based models: Decision trees



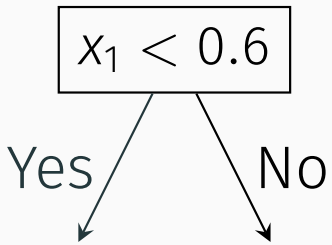
Tree-based models: Decision trees

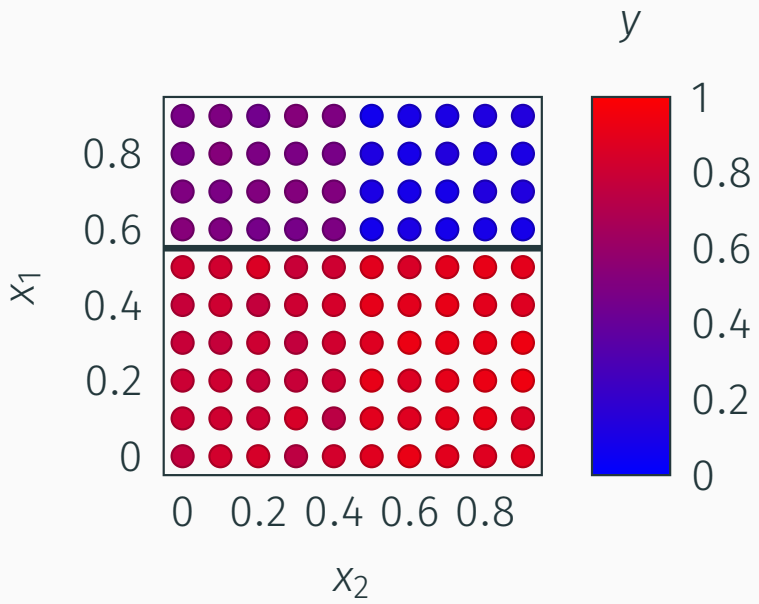


Tree-based models: Decision trees

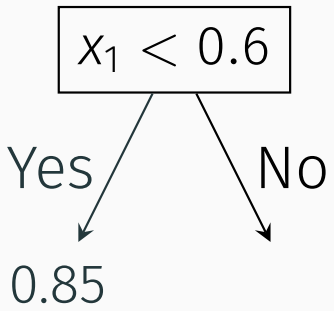


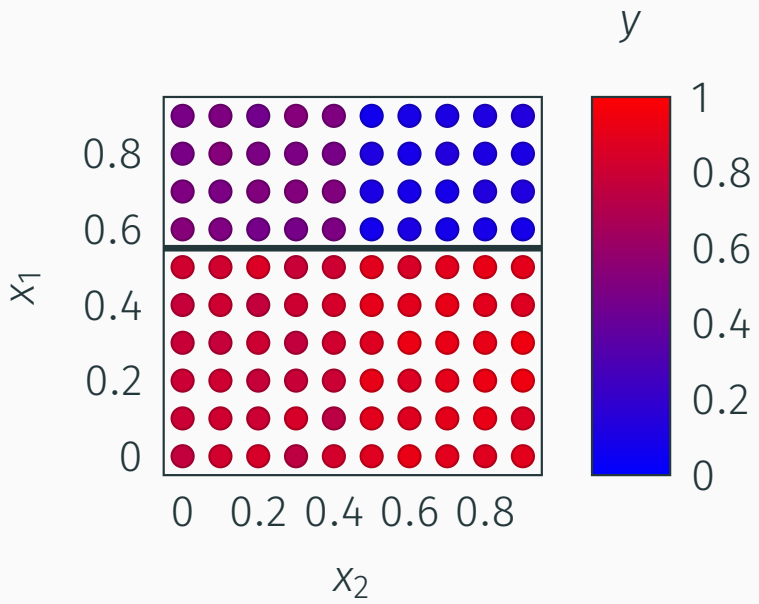
Tree-based models: Decision trees



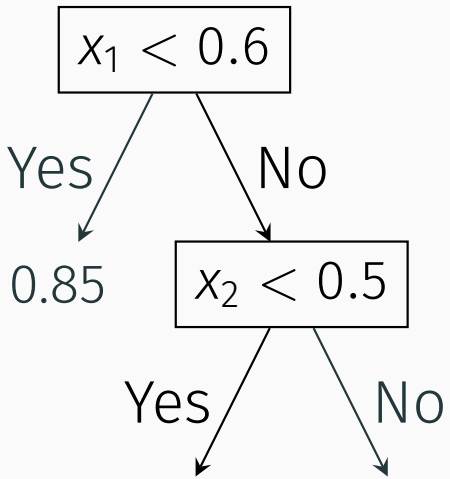


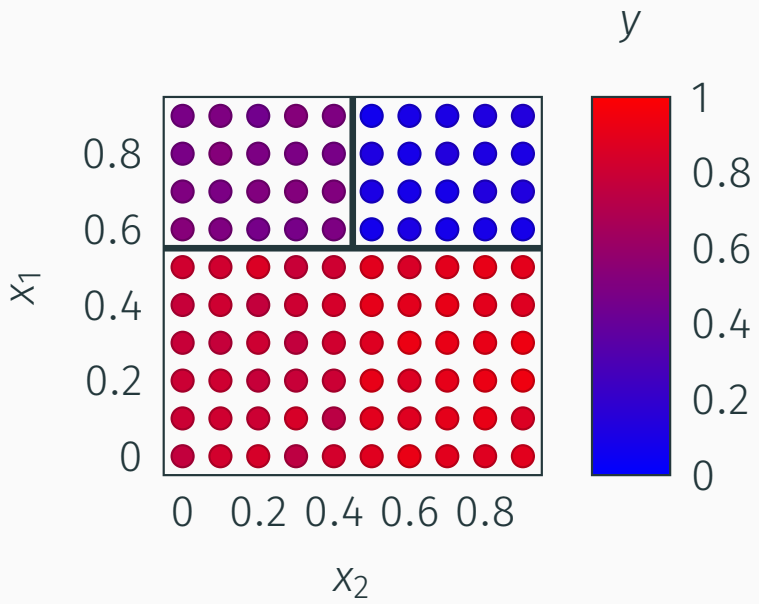
Tree-based models: Decision trees



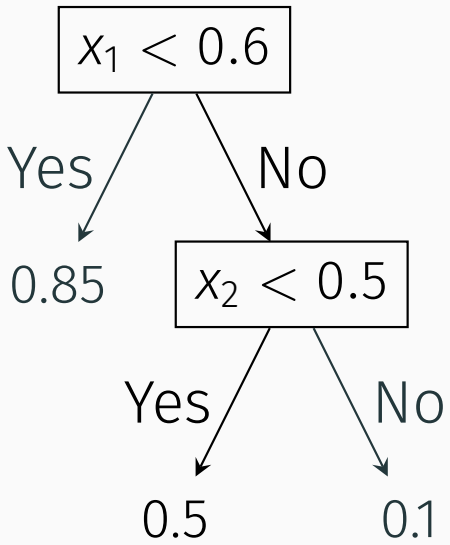


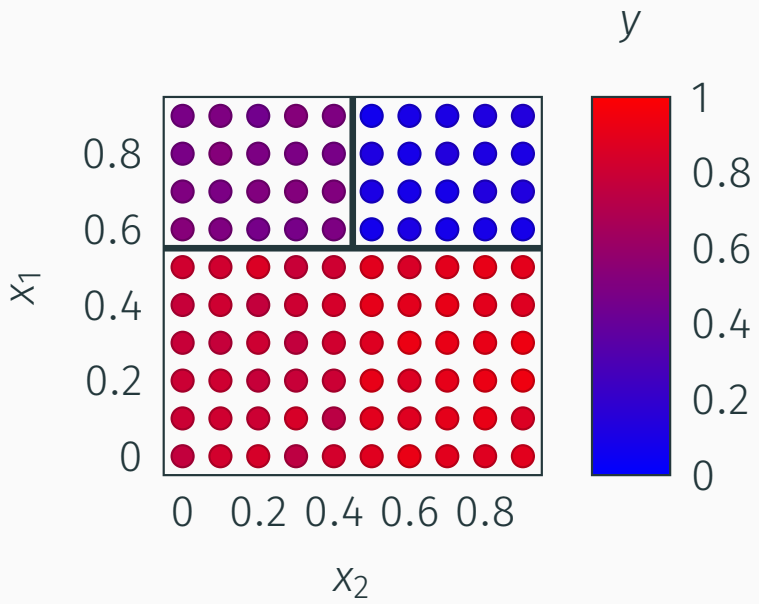
Tree-based models: Decision trees



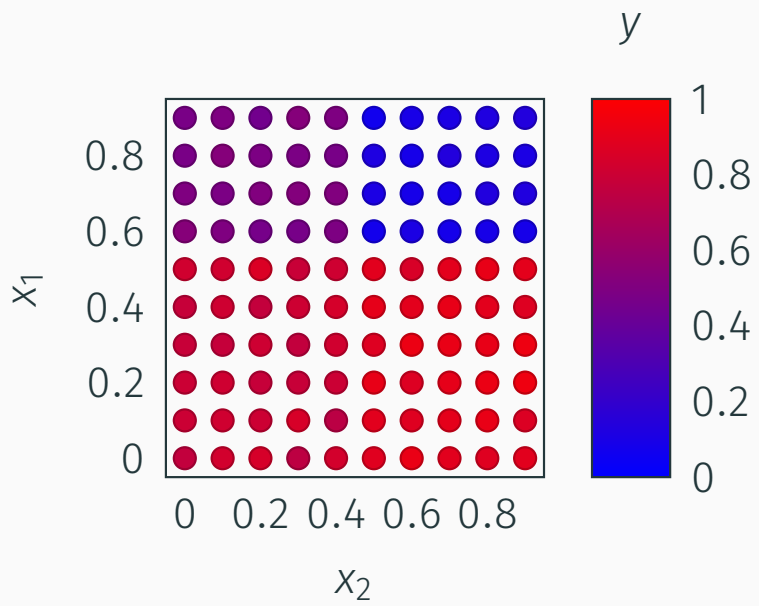


Tree-based models: Decision trees



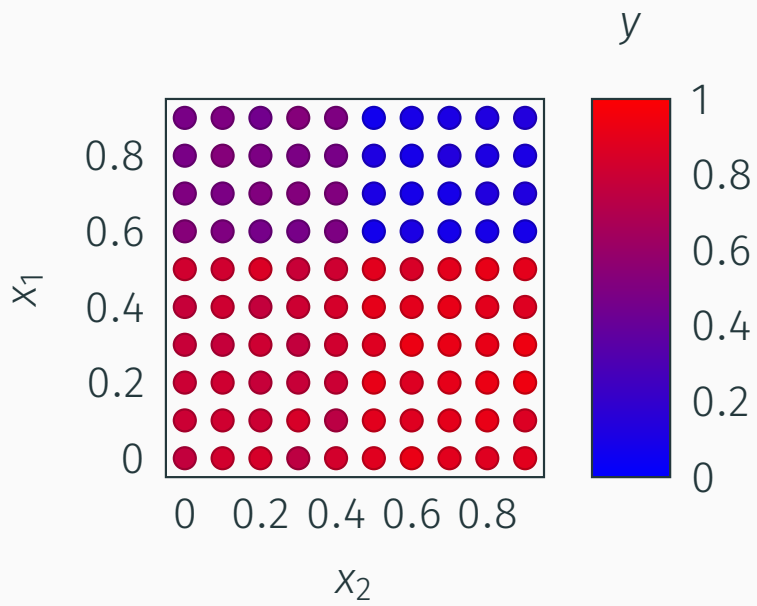


Tree-based models: Decision trees

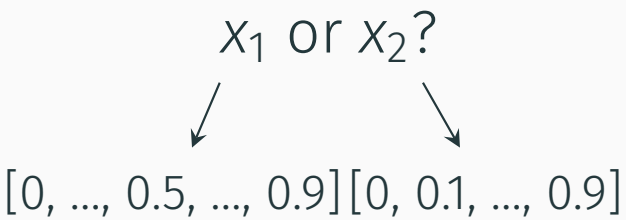


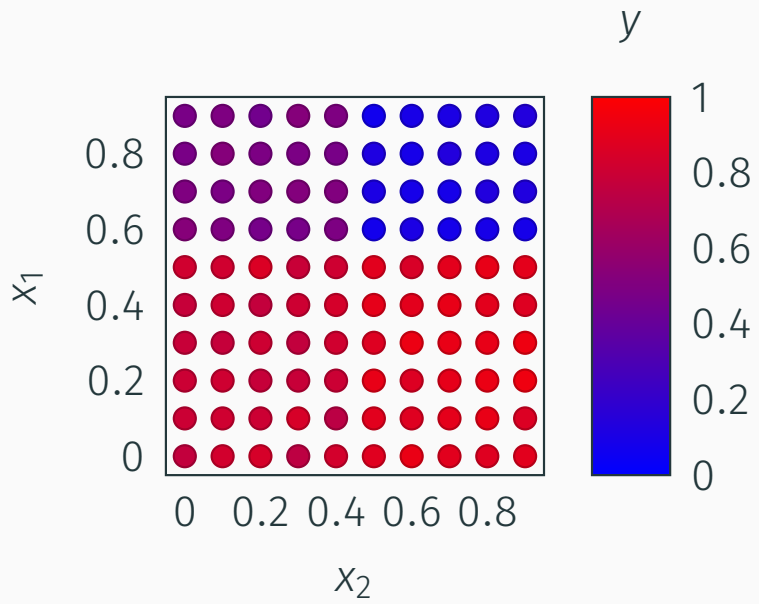
Tree-based models: Decision trees

x_1 or x_2?

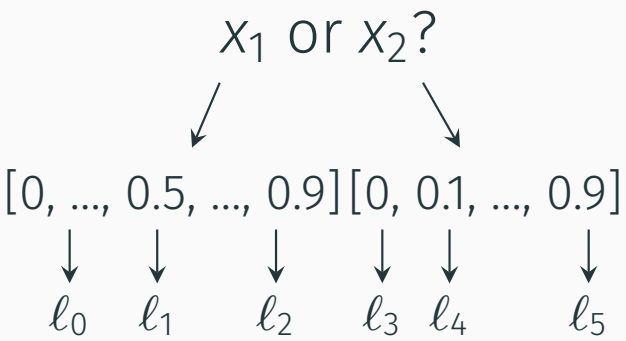


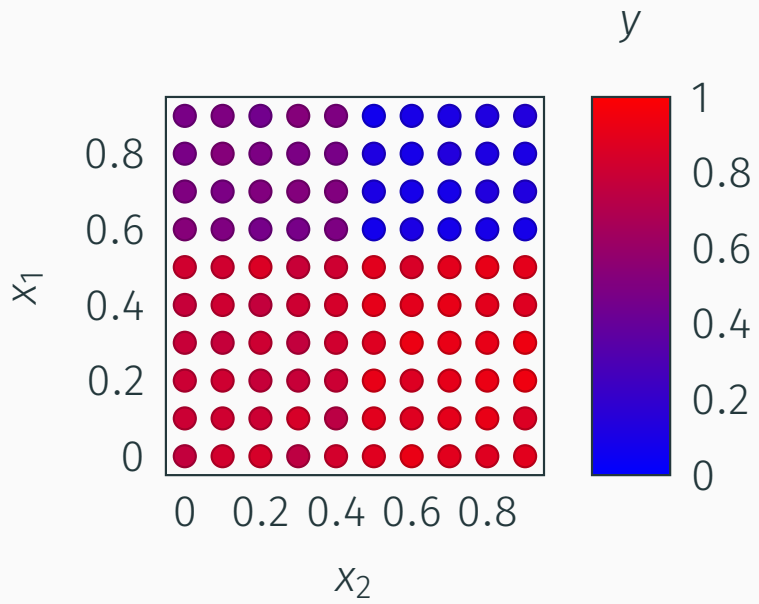
Tree-based models: Decision trees



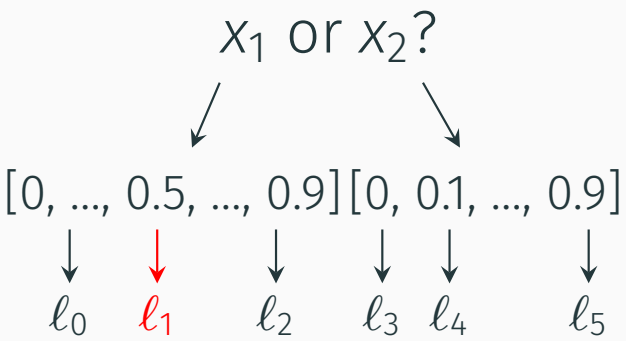


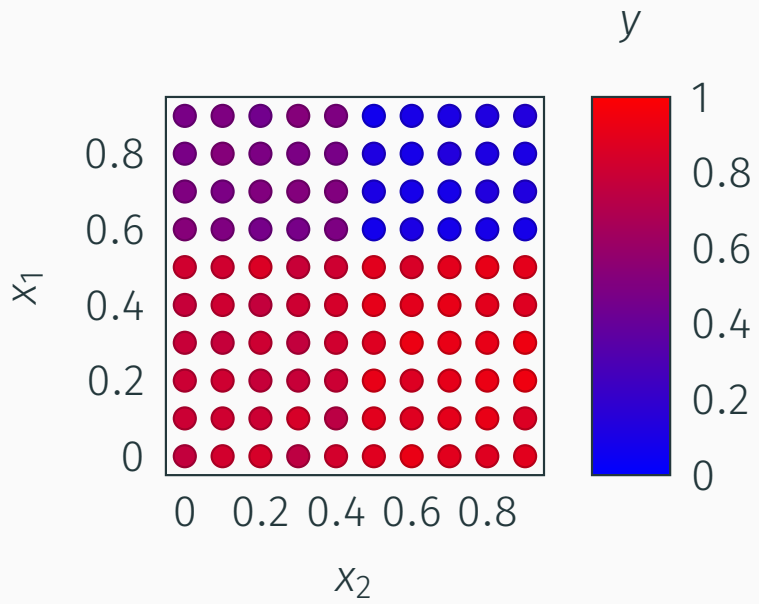
Tree-based models: Decision trees



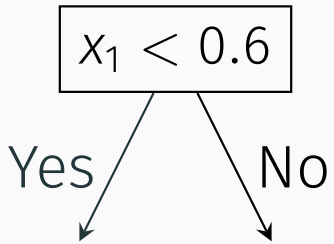


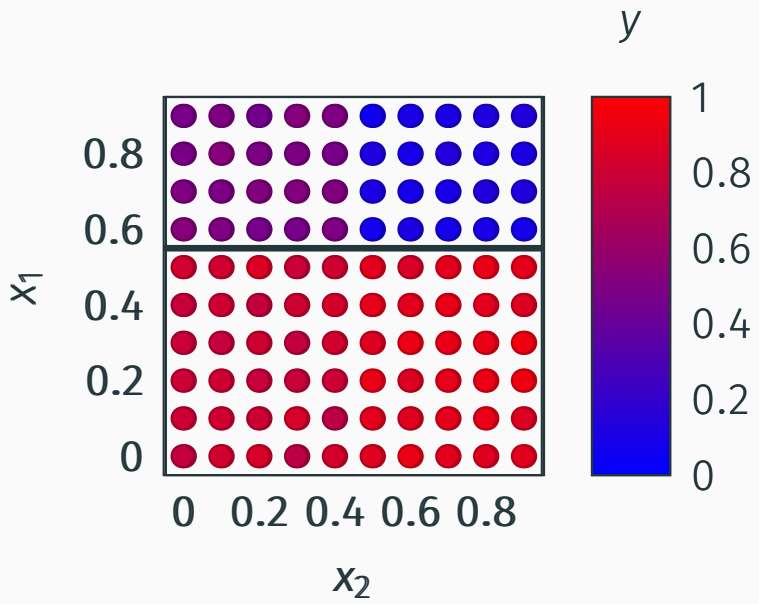
Tree-based models: Decision trees



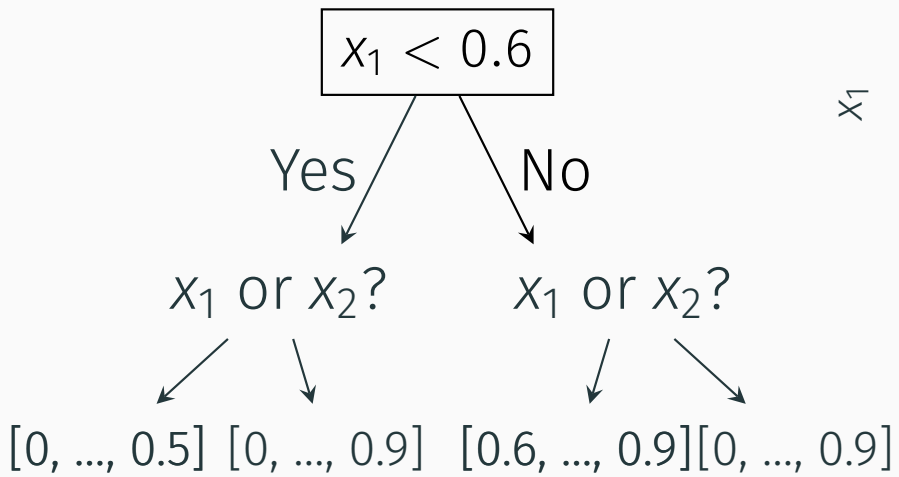


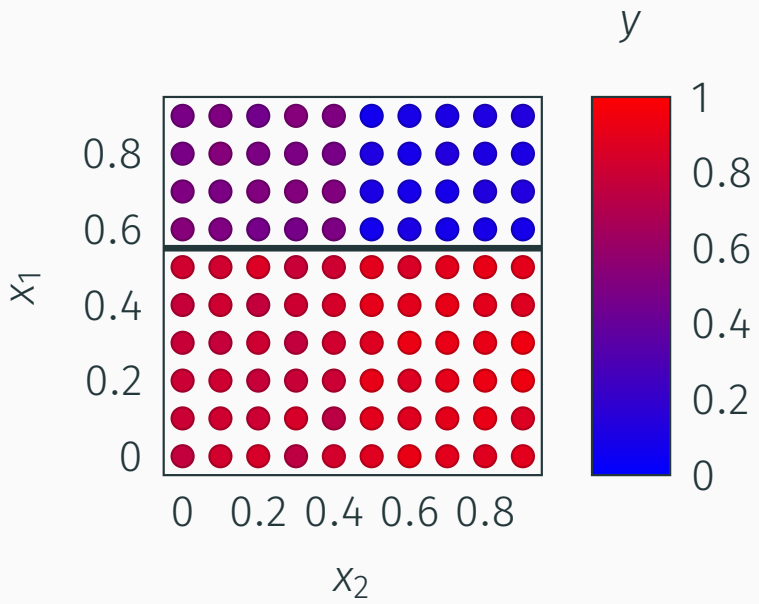
Tree-based models: Decision trees



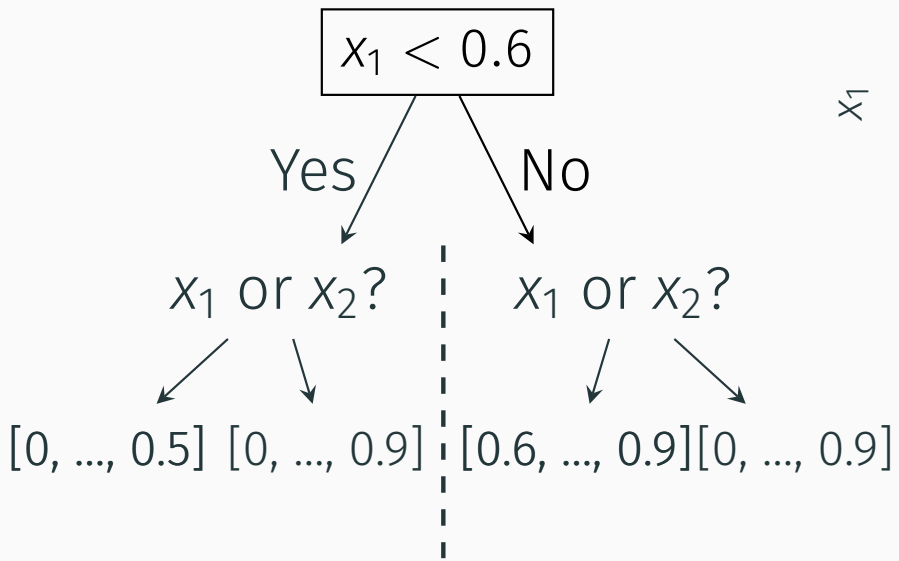


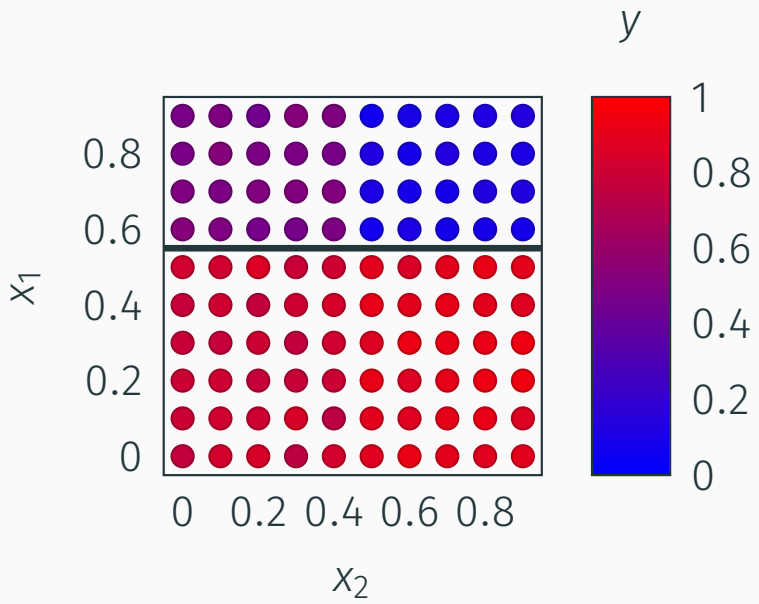
Tree-based models: Decision trees



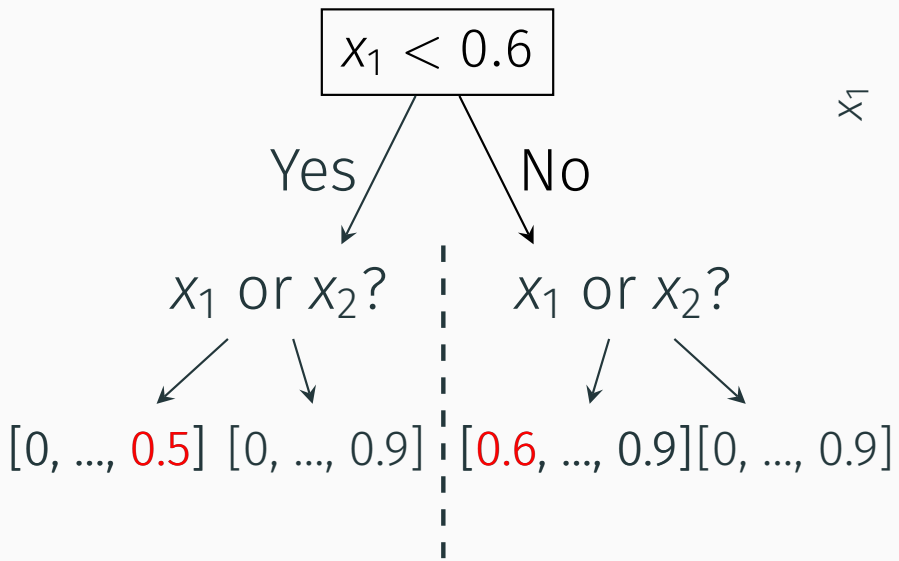


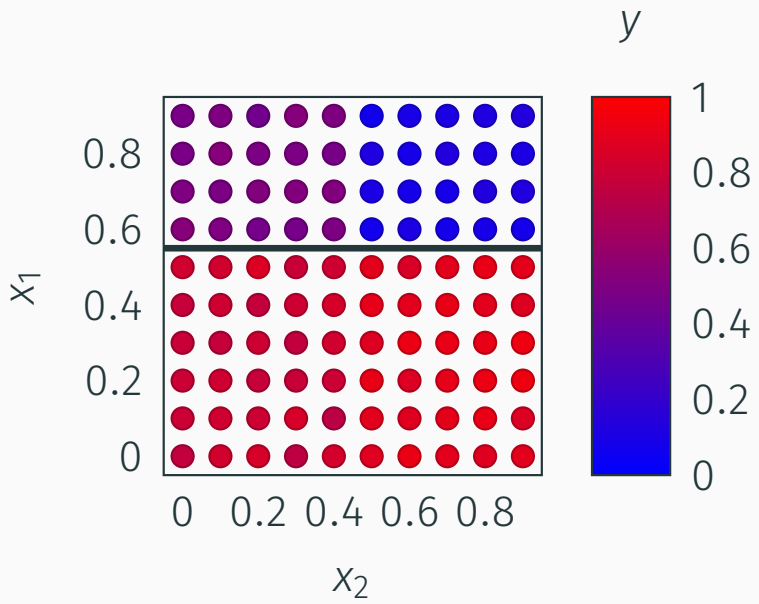
Tree-based models: Decision trees



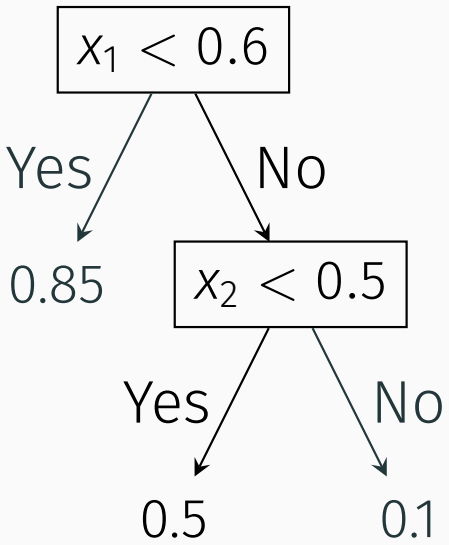


Tree-based models: Decision trees



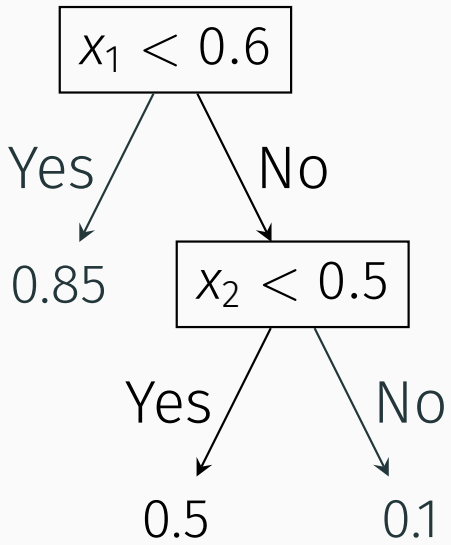


Tree-based models: Decision trees



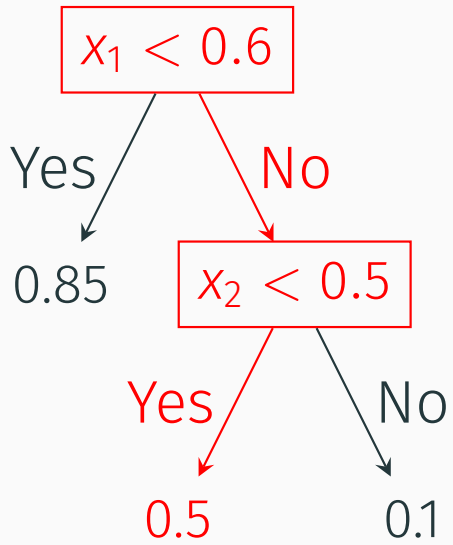
Tree-based models: Decision trees

$$x = (0.8, 0.3)$$



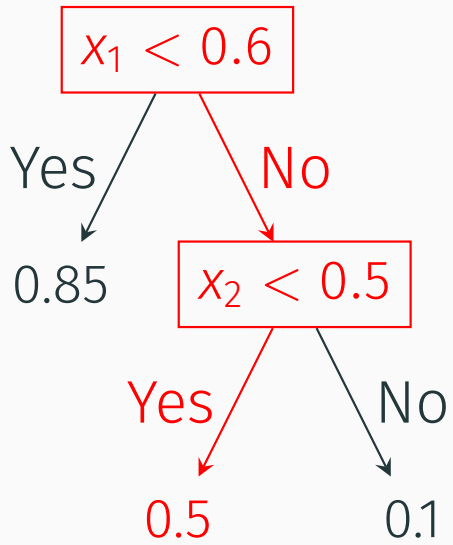
Tree-based models: Decision trees

$$x = (0.8, 0.3)$$



Tree-based models: Decision trees

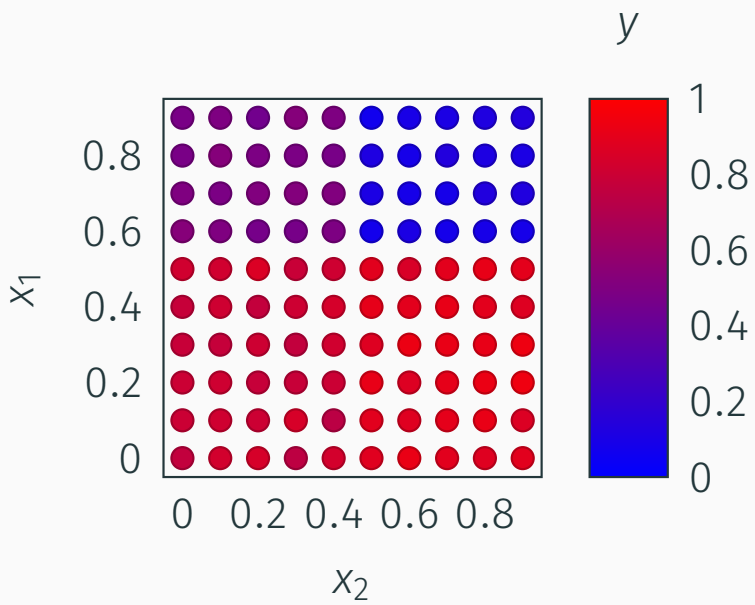
$$x = (0.8, 0.3)$$



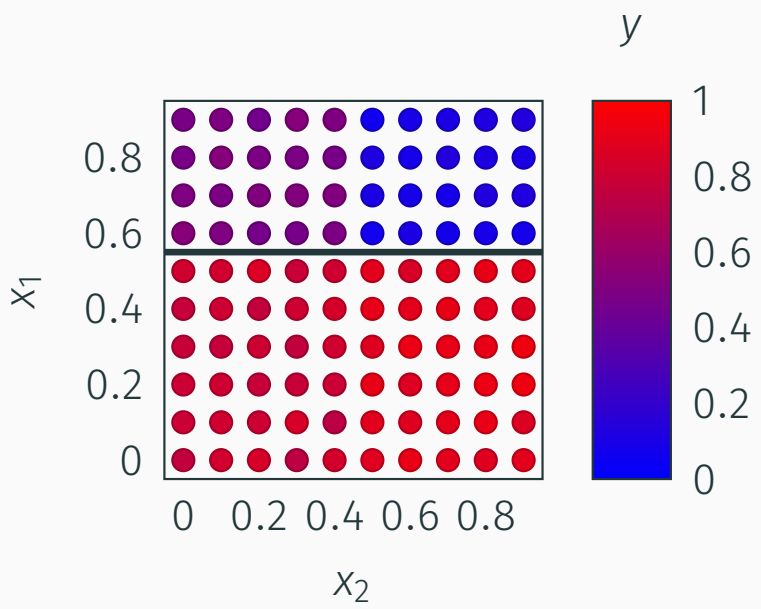
Why was $\hat{y} = 0.5$?

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

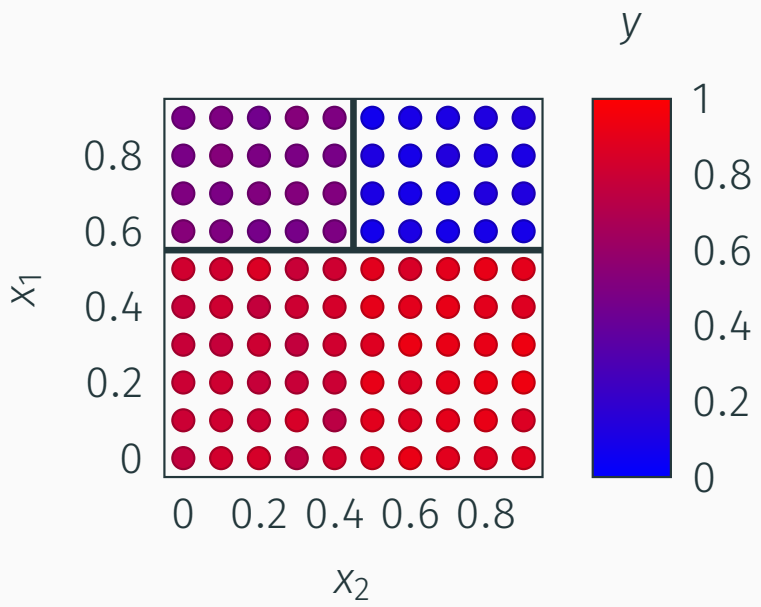
Tree-based models: Decision trees



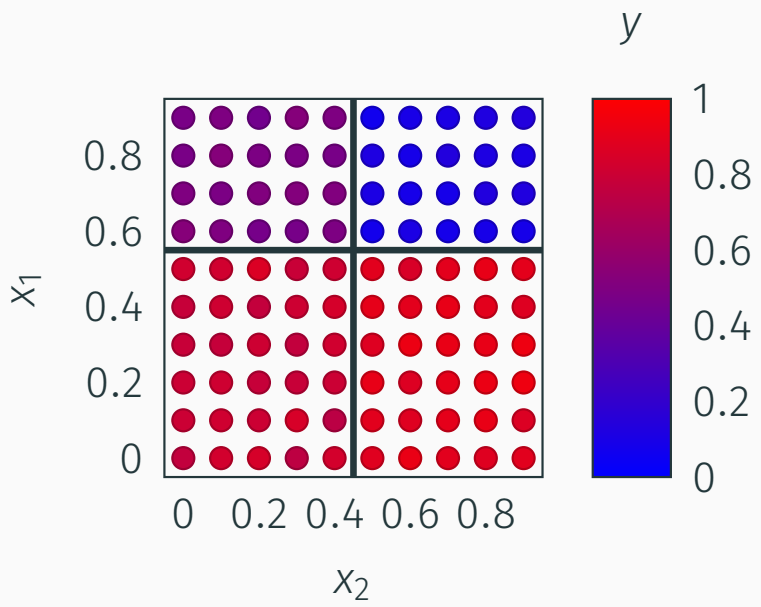
Tree-based models: Decision trees



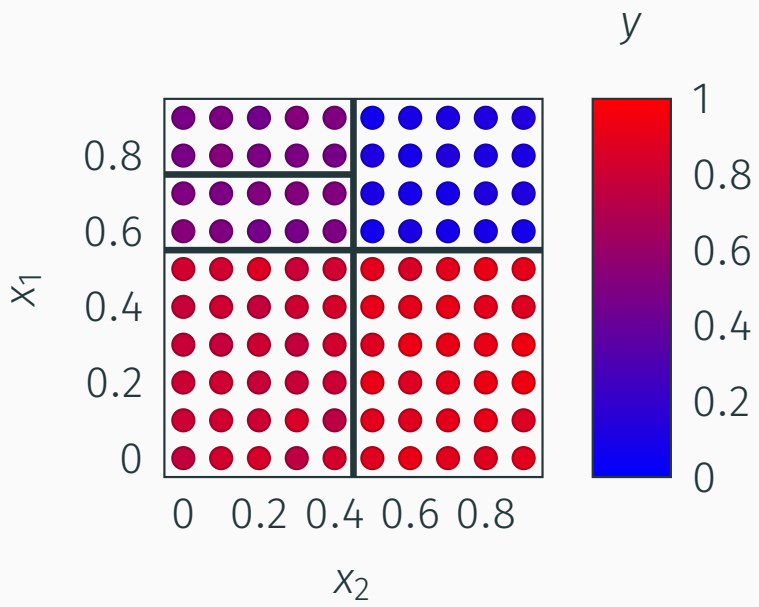
Tree-based models: Decision trees



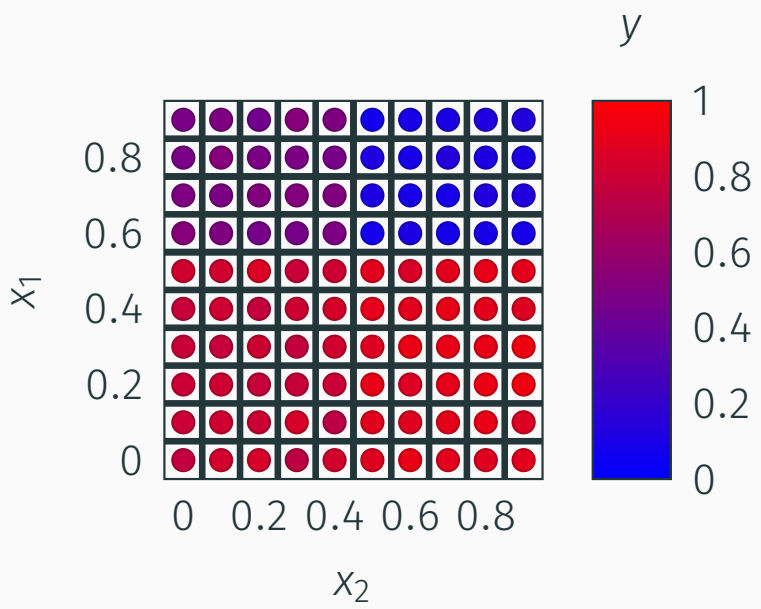
Tree-based models: Decision trees



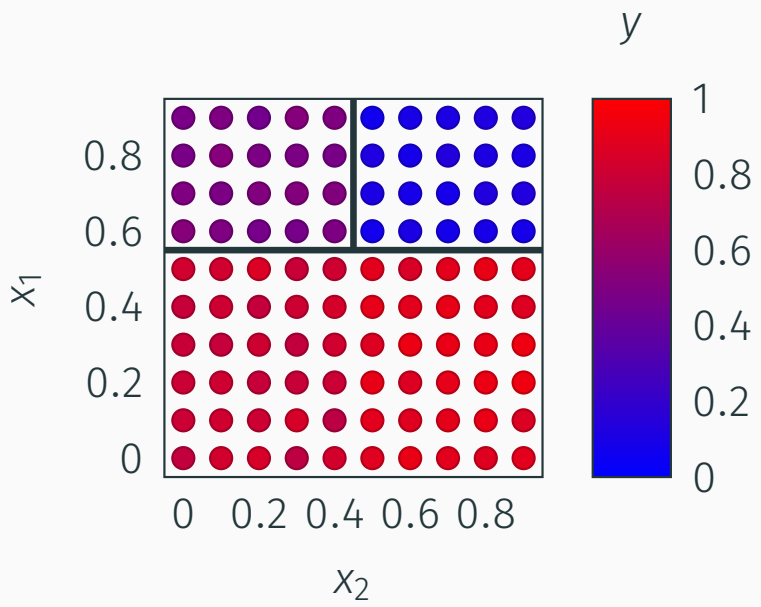
Tree-based models: Decision trees



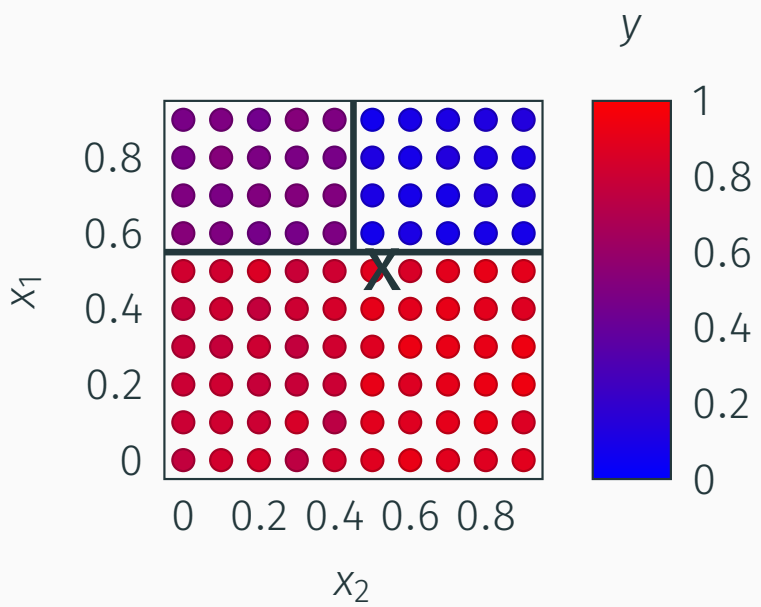
Tree-based models: Decision trees



Tree-based models: Decision trees

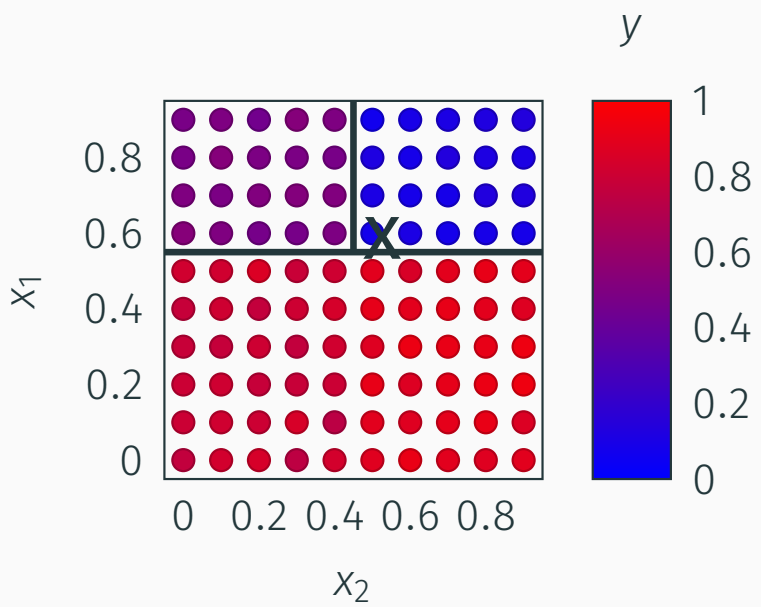


Tree-based models: Decision trees



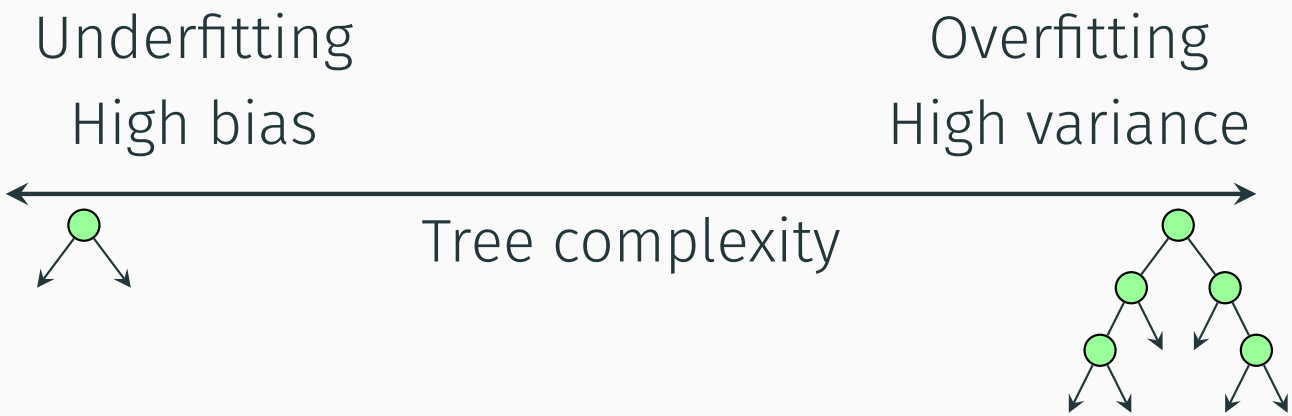
$$\hat{y} = 0.8$$

Tree-based models: Decision trees

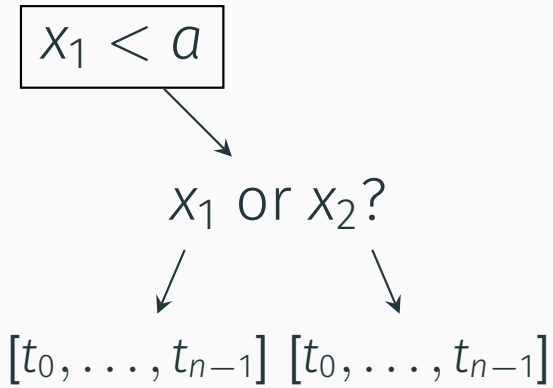


$$\hat{y} = 0.1$$

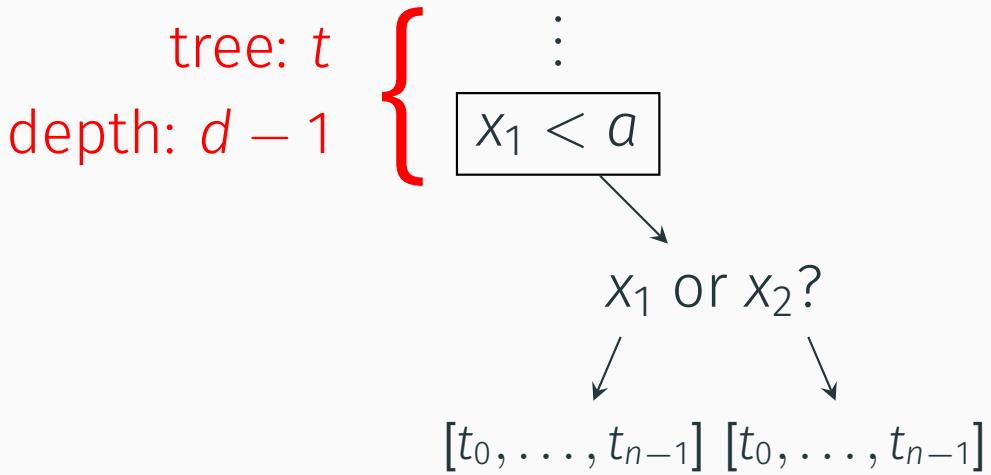
Tree-based models: Decision trees



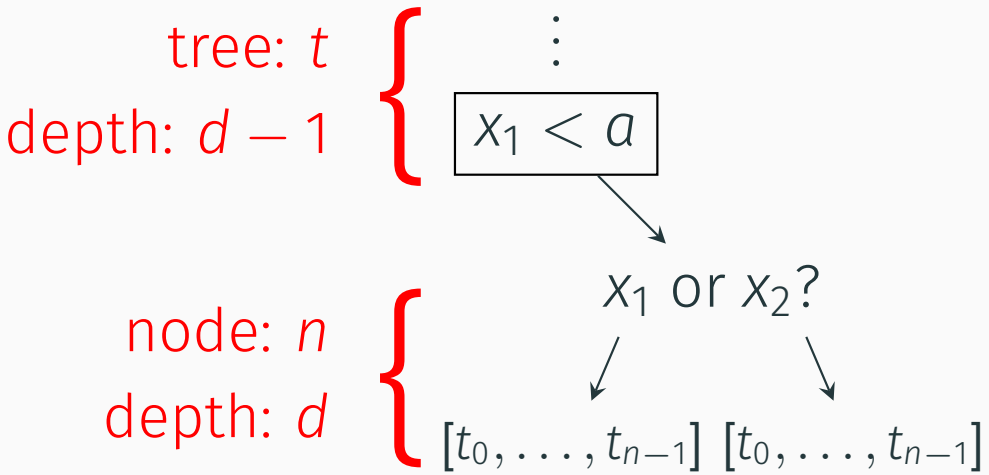
Tree-based models: Decision trees



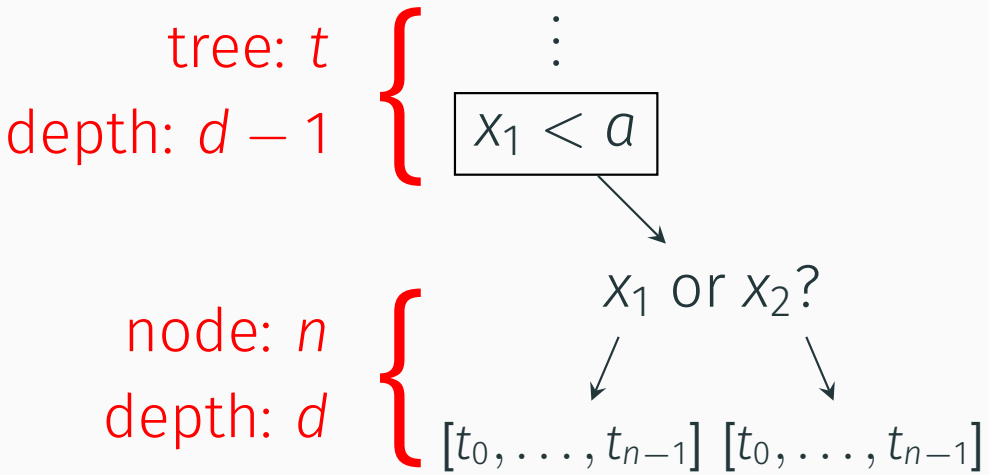
Tree-based models: Decision trees



Tree-based models: Decision trees

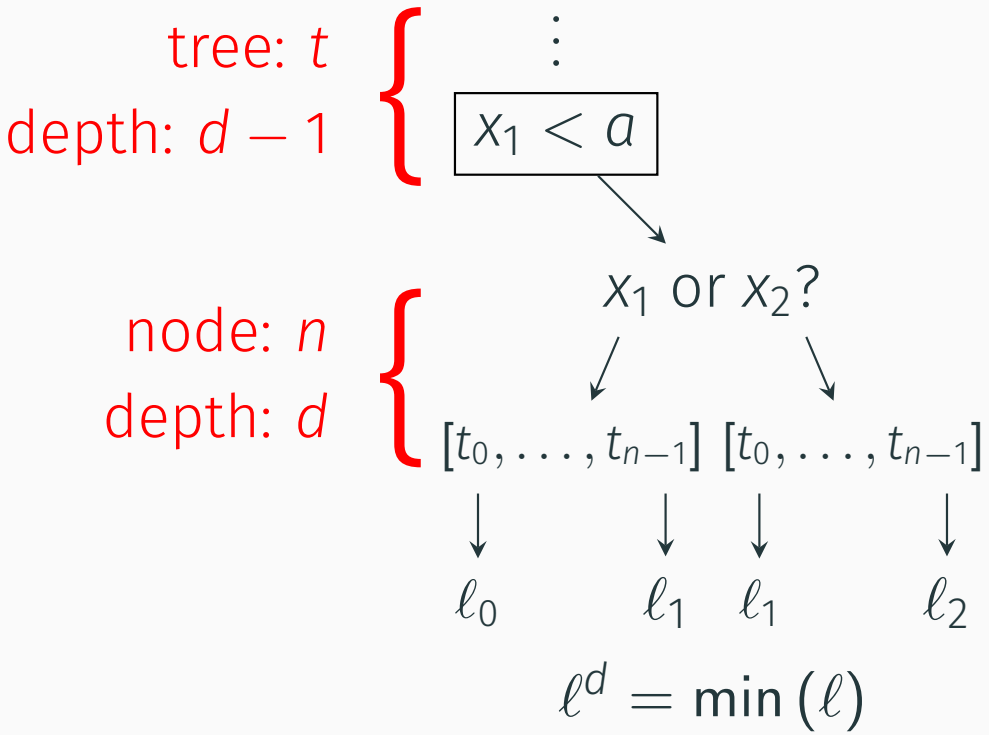


Tree-based models: Decision trees

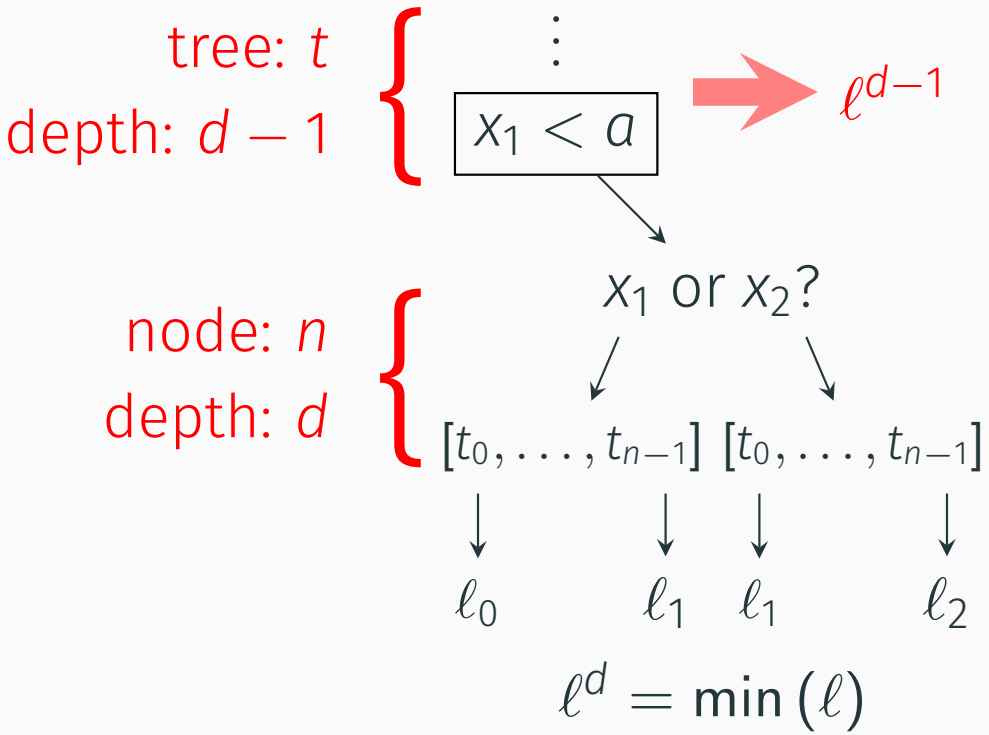



```
if d > max_depth:
    stop
else:
    continue
```

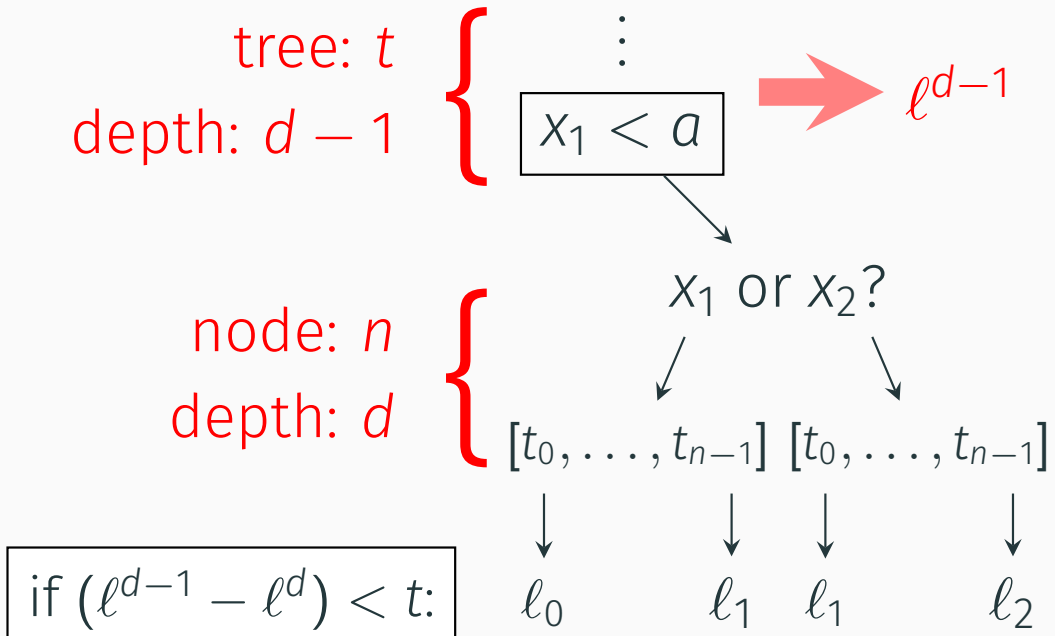
Tree-based models: Decision trees



Tree-based models: Decision trees



Tree-based models: Decision trees

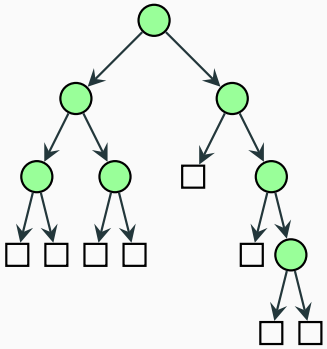


if $(\ell^{d-1} - \ell^d) < t$:
    stop
else:
    continue

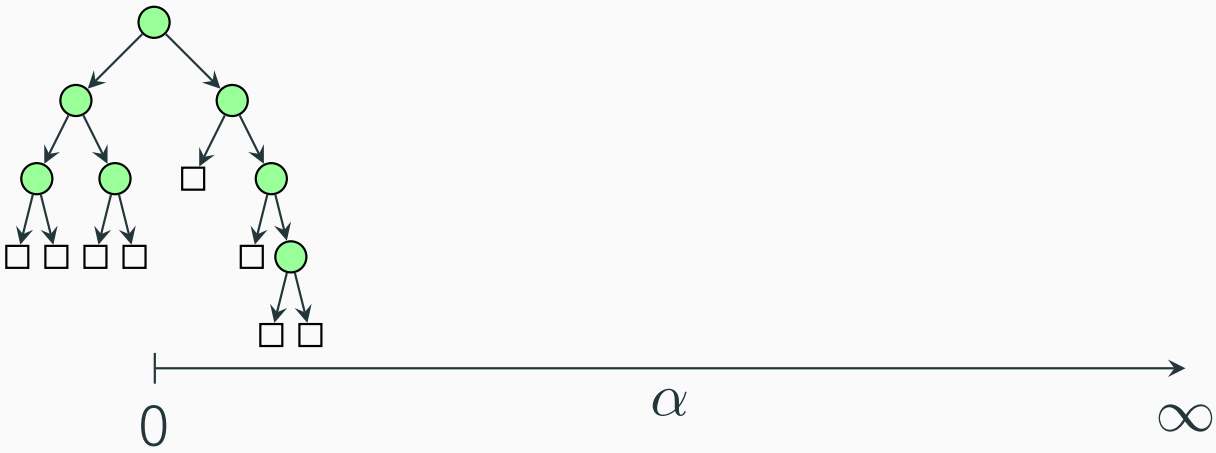
Tree-based models: Decision trees

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

Tree-based models: Decision trees



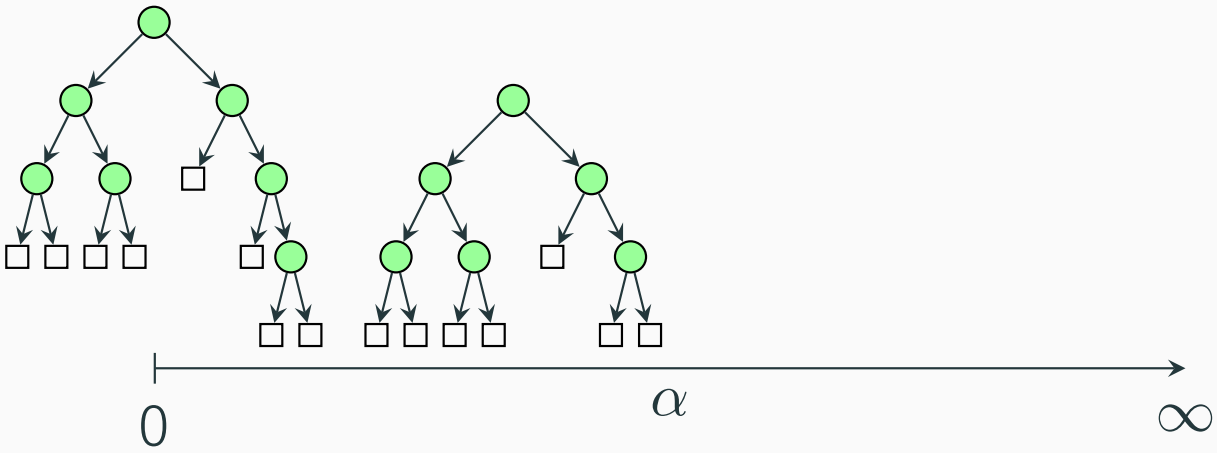
Tree-based models: Decision trees



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

$|T|$=number of leaf nodes

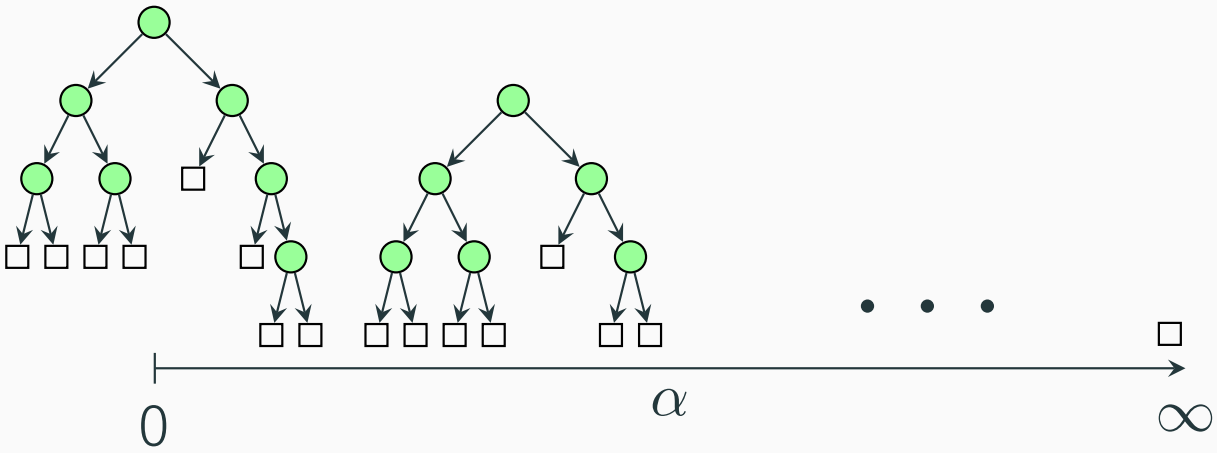
Tree-based models: Decision trees



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

$|T|$=number of leaf nodes

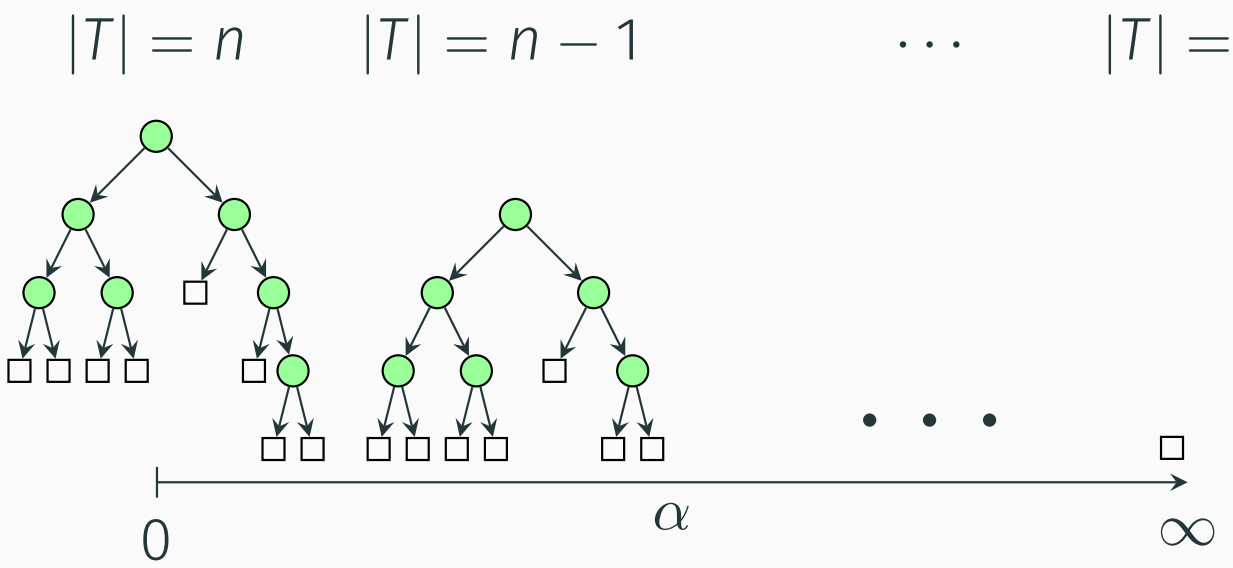
Tree-based models: Decision trees



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

$|T|$=number of leaf nodes

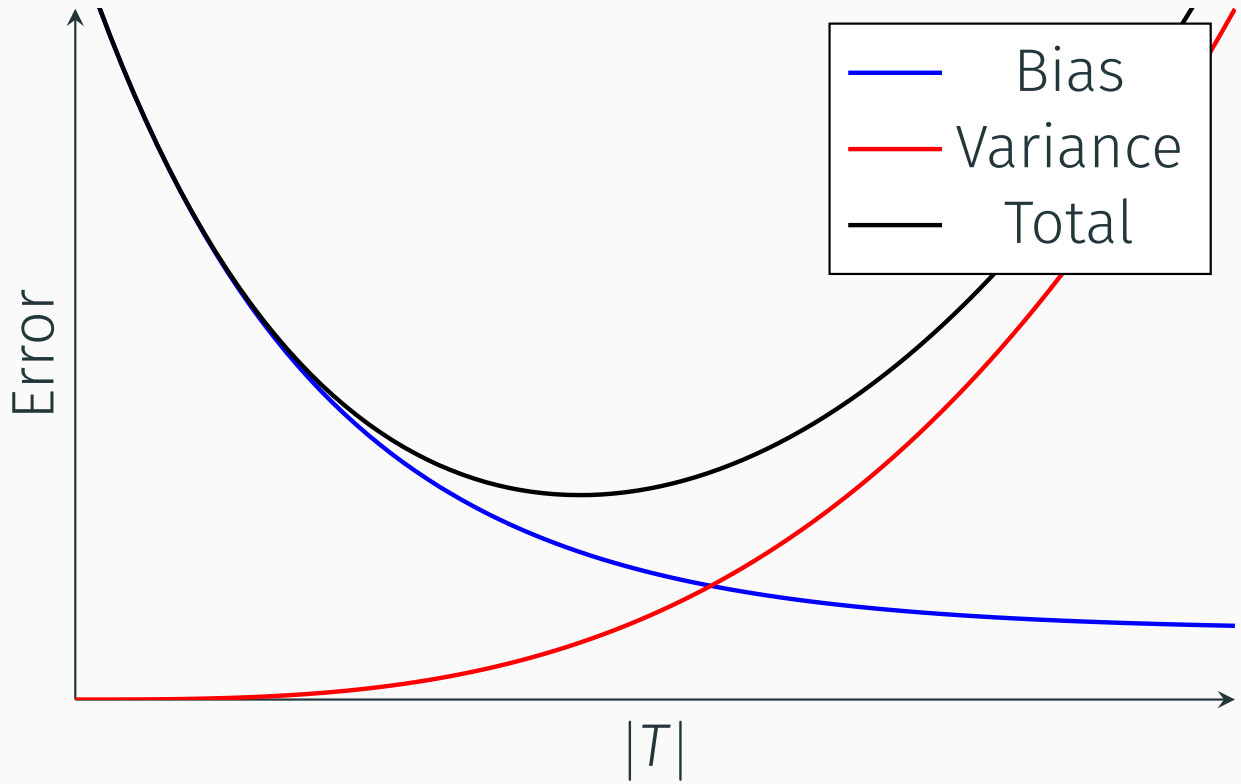
Tree-based models: Decision trees



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

$|T|$=number of leaf nodes

Tree-based models: Decision trees



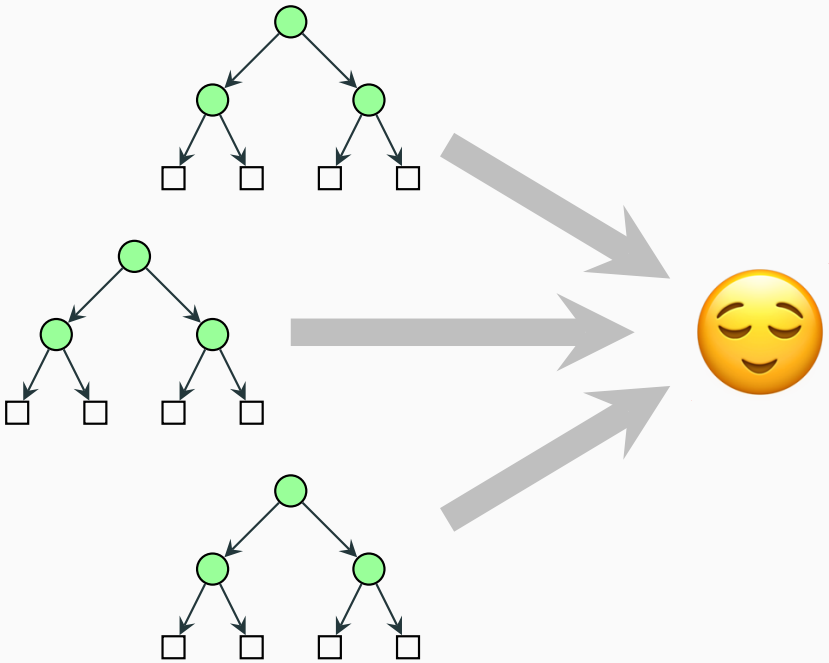
Tree-based models: Decision trees

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

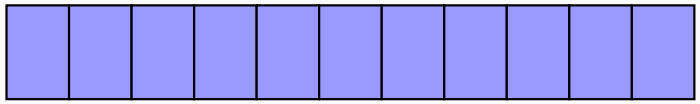


Tree-based methods: Ensembles

Ensembling: Combining multiple weak learners can improve performance.

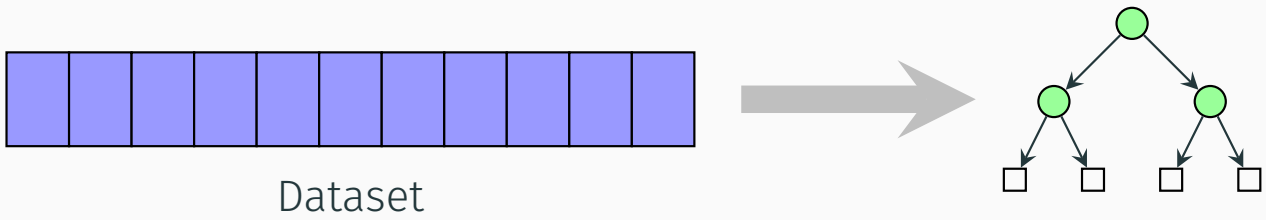
- A single prediction is achieved by averaging or majority voting
- Models should ideally be diverse

Tree-based methods: Ensembles

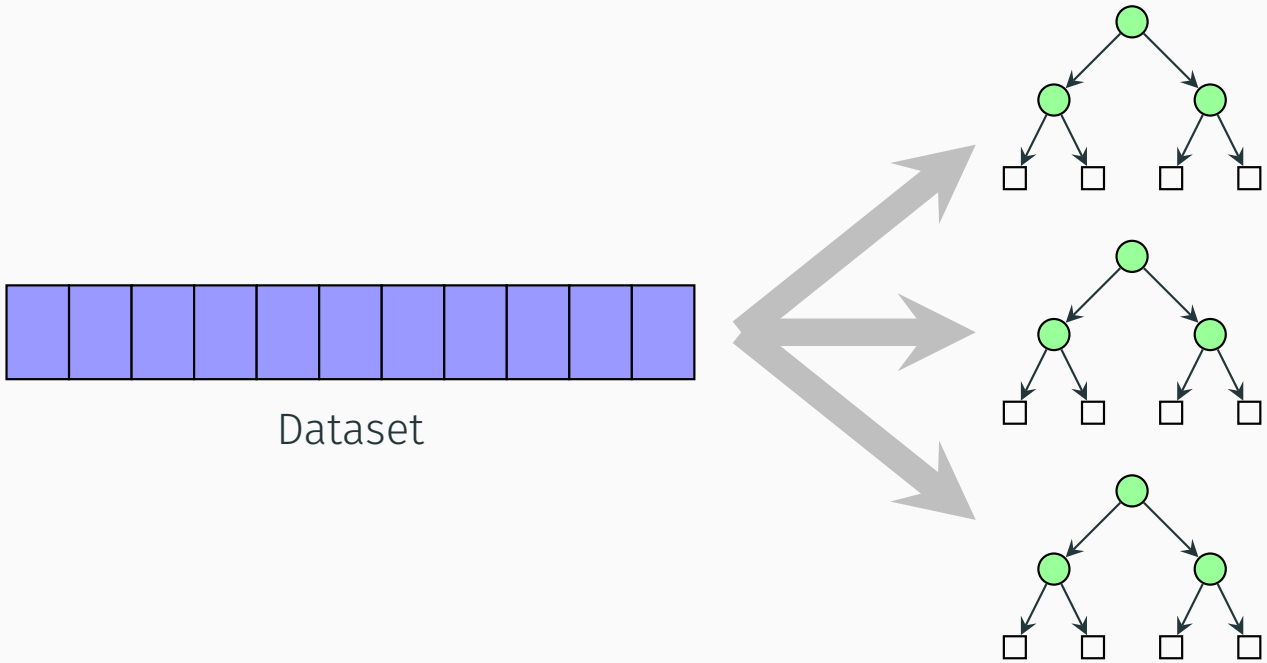


Dataset

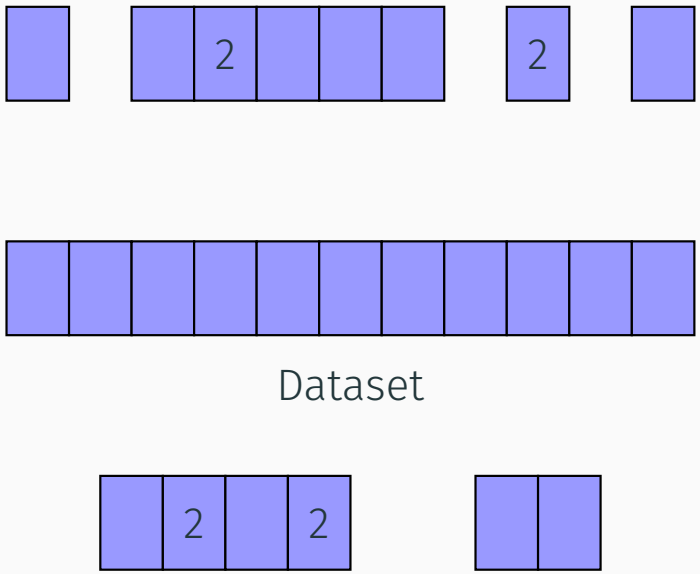
Tree-based methods: Ensembles



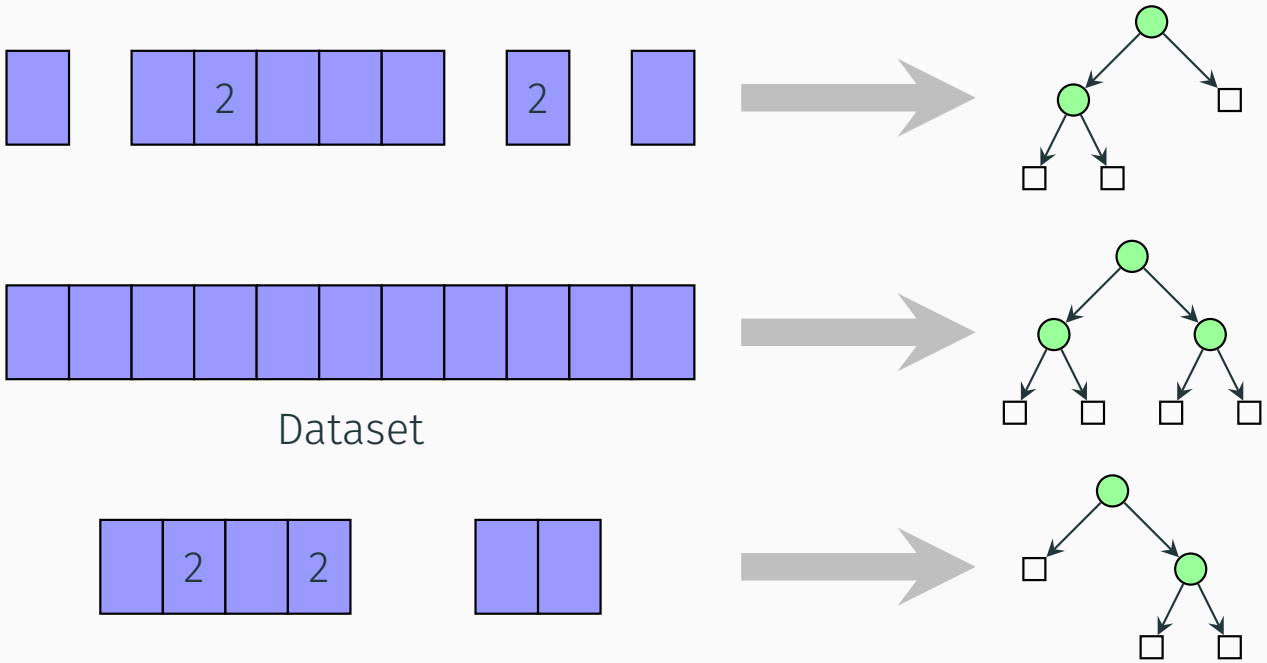
Tree-based methods: Ensembles



Tree-based methods: Ensembles



Tree-based methods: Ensembles



Tree-based methods: Ensembles

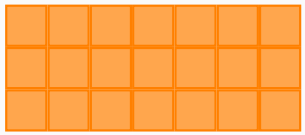
Bootstrap aggregation (bagging): Create a set of datasets by sampling with replacement, and train individual models on each.

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

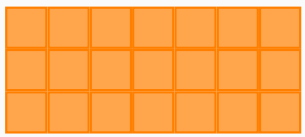
Tree-based methods: Ensembles

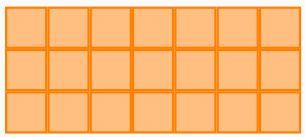
x_1	x_2	x_3	x_4	x_5	x_6	y
0.1	0.7	0.3	0.2	0.8	0.4	0.5
0.4	0.5	0.9	0.3	0.8	0.2	0.6
0.2	0.3	0.8	0.1	0.7	0.5	0.7
0.3	0.6	0.2	0.4	0.9	0.1	0.8
0.5	0.8	0.1	0.6	0.2	0.3	0.9
0.6	0.9	0.7	0.5	0.1	0.4	0.8



Tree-based methods: Ensembles

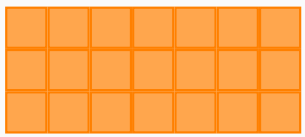
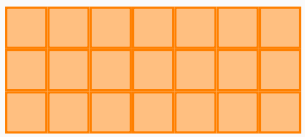
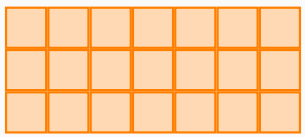
x_1	x_2	x_3	x_4	x_5	x_6	y
0.1	0.7	0.3	0.2	0.8	0.4	0.5
0.4	0.5	0.9	0.3	0.8	0.2	0.6
0.2	0.3	0.8	0.1	0.7	0.5	0.7
0.3	0.6	0.2	0.4	0.9	0.1	0.8
0.5	0.8	0.1	0.6	0.2	0.3	0.9
0.6	0.9	0.7	0.5	0.1	0.4	0.8





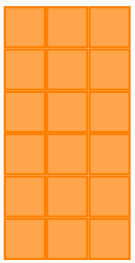
Tree-based methods: Ensembles

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

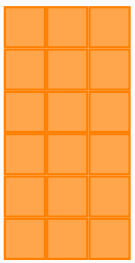
Tree-based methods: Ensembles

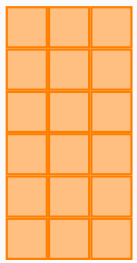
x_1	x_2	x_3	x_4	x_5	x_6	y
0.1	0.7	0.3	0.2	0.8	0.4	0.5
0.4	0.5	0.9	0.3	0.8	0.2	0.6
0.2	0.3	0.8	0.1	0.7	0.5	0.7
0.3	0.6	0.2	0.4	0.9	0.1	0.8
0.5	0.8	0.1	0.6	0.2	0.3	0.9
0.6	0.9	0.7	0.5	0.1	0.4	0.8



Tree-based methods: Ensembles

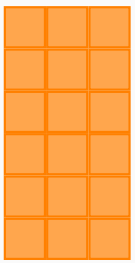
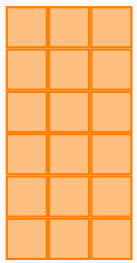
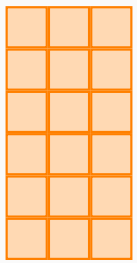
x_1	x_2	x_3	x_4	x_5	x_6	y
0.1	0.7	0.3	0.2	0.8	0.4	0.5
0.4	0.5	0.9	0.3	0.8	0.2	0.6
0.2	0.3	0.8	0.1	0.7	0.5	0.7
0.3	0.6	0.2	0.4	0.9	0.1	0.8
0.5	0.8	0.1	0.6	0.2	0.3	0.9
0.6	0.9	0.7	0.5	0.1	0.4	0.8



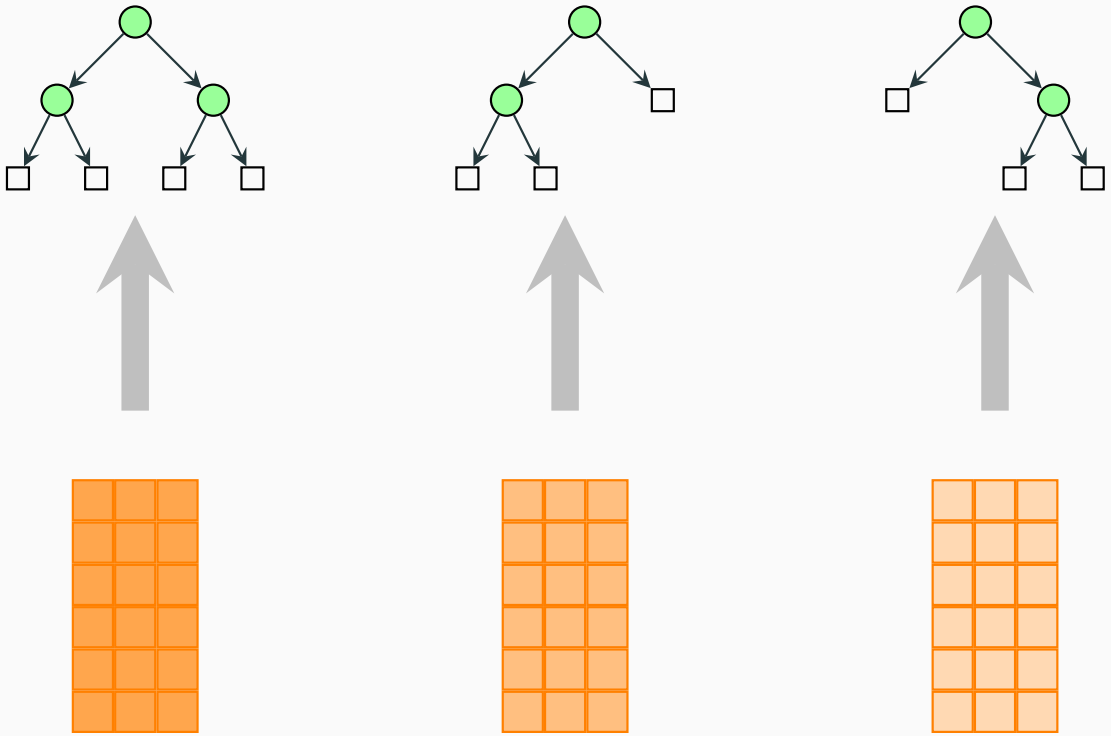


Tree-based methods: Ensembles

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 Enefit 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 place.~~ 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:42.2531/42.6607)	55.6231	54.1343	54.1213

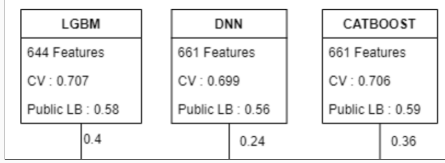
[1st Place Solution] My Betting Strategy

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.



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 (5.8481/5.4259), and Transformer (5.8619/5.4296), with respective weights of 0.5, 0.3, 0.2 searched from validation set. And these models share same 300 features.

Besides, online learning(OL) and post-processing(PP) also play an important role in my final submission.

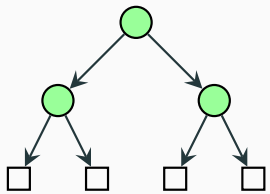
model name	validation set w/o PP	validation set w/ PP	test set w/o OL w/ PP	test set w/ OL one time w/ PP	test set w/ OL five times w/ PP
CatBoost	5.8287	5.8240	5.4523	5.4291	5.4165

Tree-based methods: Boosting

Method	RMSE
CatBoost	**3.00 ± 0.91**
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

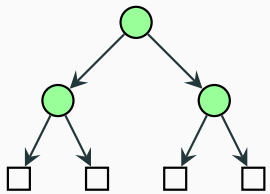
Self-Attention Between Datapoints: Going Beyond Individual Input-Output Pairs in Deep Learning, Kossen et al., preprint at arXiv, 2022

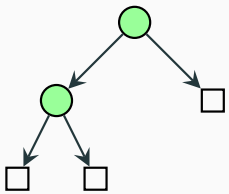
Tree-based methods: Boosting



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

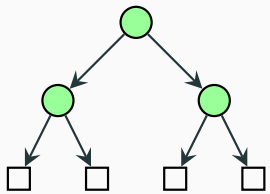
Tree-based methods: Boosting



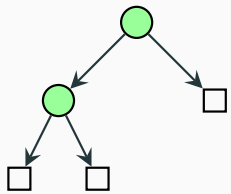


$$f_0 : \min_f \sum (y_i - f(x_i))^2 \qquad f_1 : \min_f \sum (y_i - f(x_i))^2$$

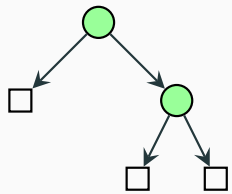
Tree-based methods: Boosting



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

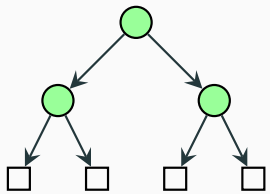


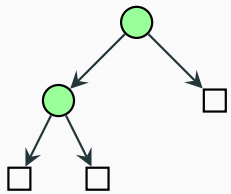
$$f_1 : \min_f \sum (y_i - f(x_i))^2$$

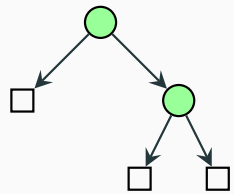


$$f_2 : \min_f \sum (y_i - f(x_i))^2$$

Tree-based methods: Boosting



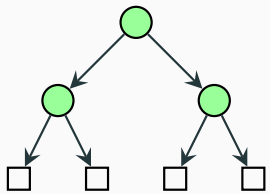




$$f_0 : \min_f \sum (y_i - f(x_i))^2 \qquad f_1 : \min_f \sum (y_i - f(x_i))^2 \qquad f_2 : \min_f \sum (y_i - f(x_i))^2$$

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

Tree-based methods: Boosting

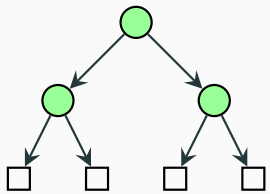


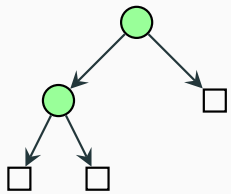
$$f_0 : \min_f \sum (y_i - f(x_i))^2$$

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

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

Tree-based methods: Boosting





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

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

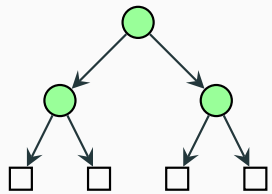
$$y* = y - (\hat{f}(x))$$

$$f_1 : \min_f \sum (y *_i - f(x_i))^2$$

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

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

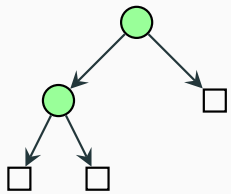
Tree-based methods: Boosting



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

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

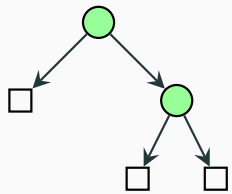
$$y* = y - (\hat{f}(x))$$



$$f_1 : \min_f \sum (y *_i - f(x_i))^2$$

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

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

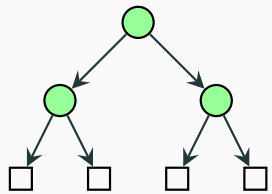


$$f_2 : \min_f \sum (y *_i - f(x_i))^2$$

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

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

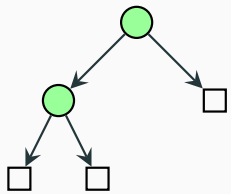
Tree-based methods: Boosting



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

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

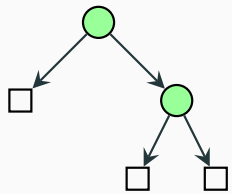
$$y* = y - (\hat{f}(x))$$



$$f_1 : \min_f \sum (y *_i - f(x_i))^2$$

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

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



$$f_2 : \min_f \sum (y *_i - f(x_i))^2$$

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

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

Tree-based methods: Boosting

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

- Works *really* well

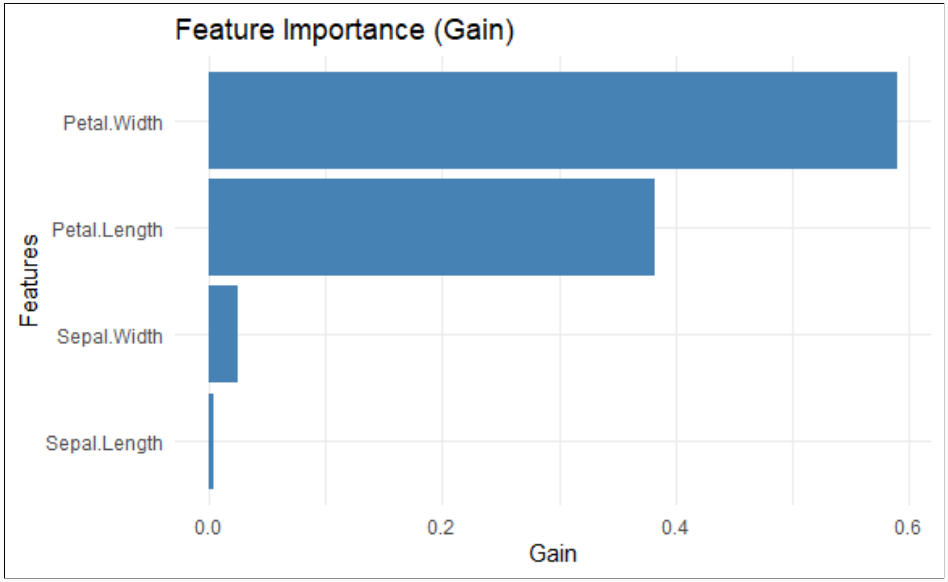
Tree-based methods: Boosting

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

Tree-based methods: Boosting



https://xgboost.readthedocs.io/en/stable/python/python_api.html#module-xgboost.plotting

Exercise 5

1. Download the Credit.csv dataset from ISL
2. Preprocess the dataset as you see fit.
3. Fit one or more non-linear models, choosing model types and hyperparameters according to what you've learned
4. Report the performance of the best model
5. Reflect on the choices you made
6. Reflect upon the performance of the model(s)