SUPERVISED ML

SUPERVISED ML

- Certainly the most successful branch of ML currently
- Training a computer program (algorithm) to learn through examples
- Tasks:
  - Predict the weather, the climate
  - Recognize objects/people in pictures
  - Evaluate the risks of recidivism of a convict (don't do that!)
  - What else ?

SUPERVISED ML

- Two main objectives, with similar solutions
- Regression: predict a numerical value
  - Temperature, cost, grade, etc.
- Classification: predict a class/label/category
  - Success/Failure, Blue/Red/Yellow, which animal among 1000 possibles, etc.

SUPERVISED ML: DNN

- Many recent successes thanks to Deep Neural Networks
- This class: only "classic" methods
- DNN are just an *evolution* of methods presented in this class, all principles stay the same.

FICTIONAL EXAMPLE

- Let's say we want to predict the price of apartments. We have a collection of examples, for now in comparable settings (same neighborhood of the same city…)

- We have access to some characteristics of apartments:
  - Surface Area, # of rooms, # of windows, Elevator…

- This is typically a Regression problem.

EVALUATION/OBJECTIVE

- Before applying any method, set up an objective/a quality score/an error measure

- We want to be able to compare several prediction methods to see which one is the most efficient. But how to compare them ?

- Typical scores:
  - MAE: Mean Absolute Error
  - MSE, RMSE: (Root) Mean Square Error
  - R^2

MEAN ABSOLUTE ERROR

- $$\mathrm{MAE} = \frac{1}{n}\sum_{i=1}^{n}\left|y_i - \hat{y}_i\right| = \frac{1}{n}\sum_{i=1}^{n}\left|e_i\right|$$

- Similarity with the MAD (Mean Absolute Deviation), comparing values with predictions instead of simple mean.

- Simple to interpret
  - lower the value, lower the error, better the prediction
  - 0: perfect prediction

MEAN SQUARED ERROR

- $\mathrm{MSE} = \frac{1}{n}\sum_{i=1}^{n}\left(y_i - \hat{y}_i\right)^2 = \frac{1}{n}\sum_{i=1}^{n} e_i^2$

- Similarity with the **Variance**
- Using *squared* errors give stronger importance to large errors
- $\mathrm{RMSE} = \sqrt{\mathrm{MSE}}$, can be easier to interpret

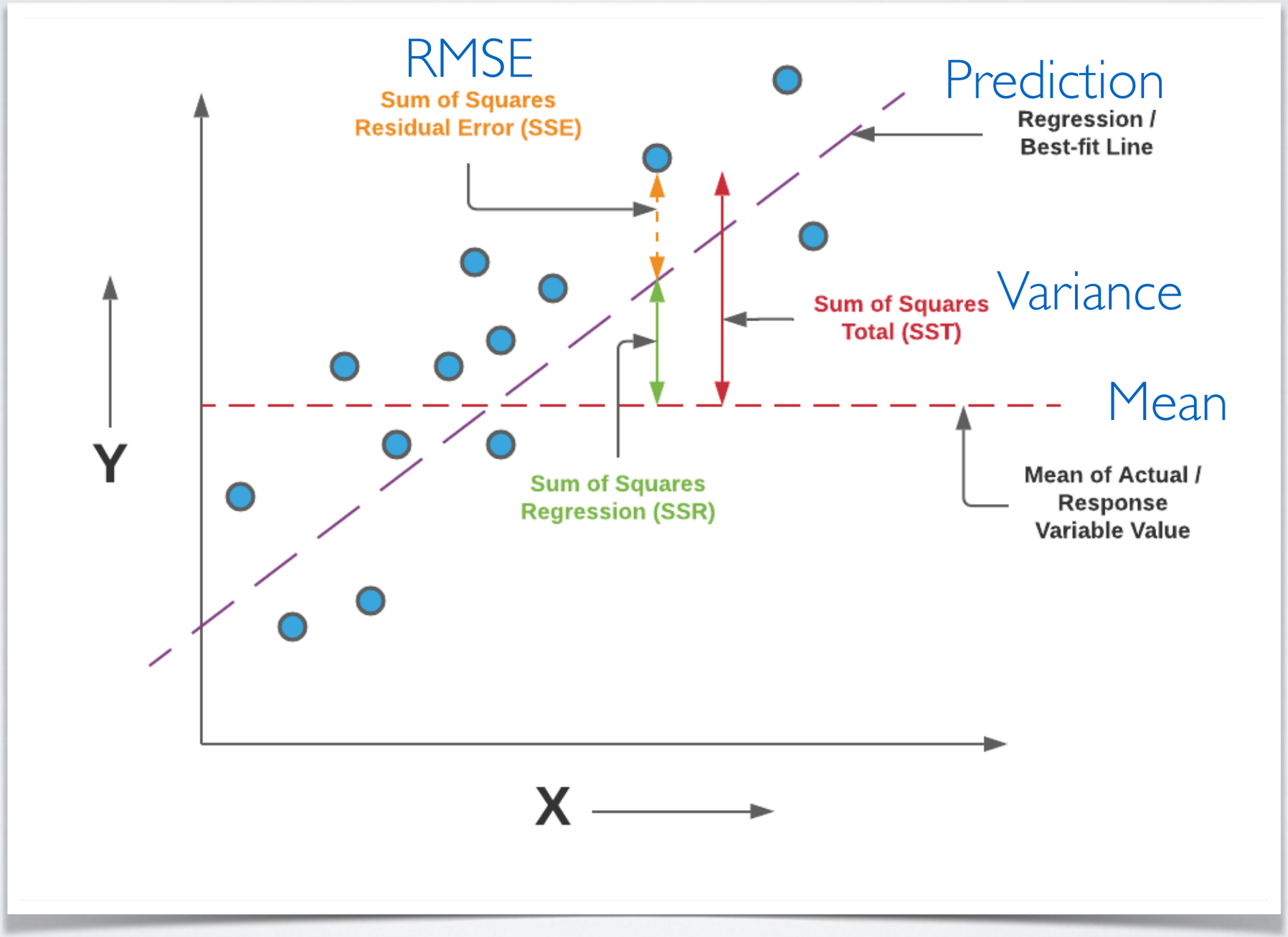
R^2 (R-SQUARED)

- $R^2 = 1 - \frac{\sum_i e_i^2}{\sum_i y_i - \bar{y}} = 1 - \frac{MSE}{Var(y)}$

- Quantifies the fraction of the variance that is explained by the prediction
  - Sometimes called *coefficient of determination* for linear regression

- 1=>Perfect prediction.
  - Negative if the prediction is worst than taking the average (=Variance)

R^2 (R-SQUARED)

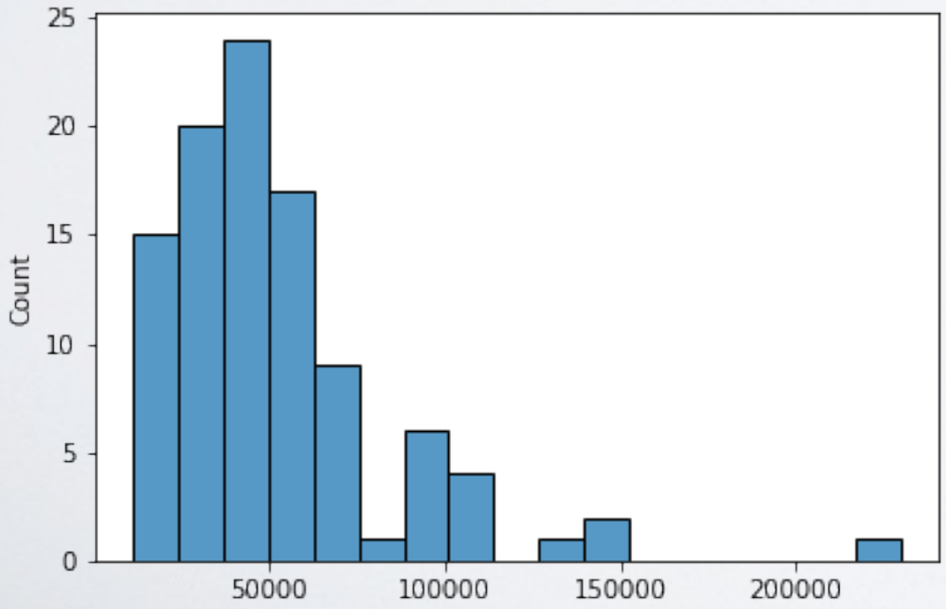

RMSE
Sum of Squares
Residual Error (SSE)
Prediction
Regression /
Best-fit Line
Variance
Sum of Squares
Total (SST)
Mean
Mean of Actual /
Response
Variable Value
Sum of Squares
Regression (SSR)
Y
X

EVALUATION/OBJECTIVE

- Which one should you use?
  - Different literature have their favorite one. RMSE is probably the most popular.
  - If your ML algorithm use the RMSE as objective (loss function), then you should probably use RMSE
- If you're not writing a paper or playing a competition, use all of them
  - More information can allow you to judge better. There is no "truth".

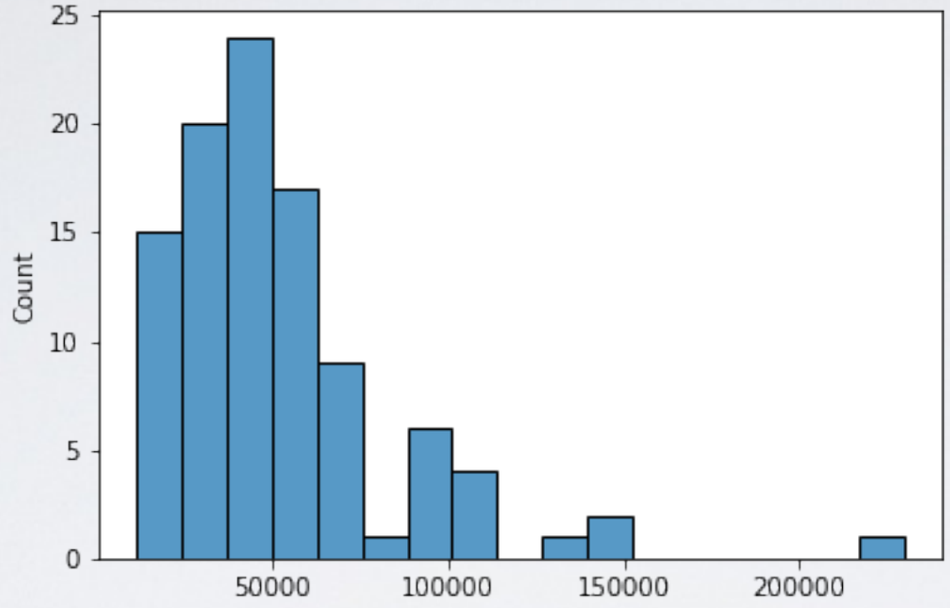
BASELINE

- Let's define our baseline, our reference to improve on
- Let's assume we only know the target variable values
- Using statistics, we know that the best "prediction" we can do for the price of a future apartment will be
  - The average (for MSE) =>Variance
  - The median (for MAE) => MAD



(Some imaginary values)

BASELINE



Using Mean=51676

```
MSE 1105345073.7155044
RMSE 33246.73027104326
MAE 22740.967725747014
R2 0.0
```

RMSE lower

Using Median=43086

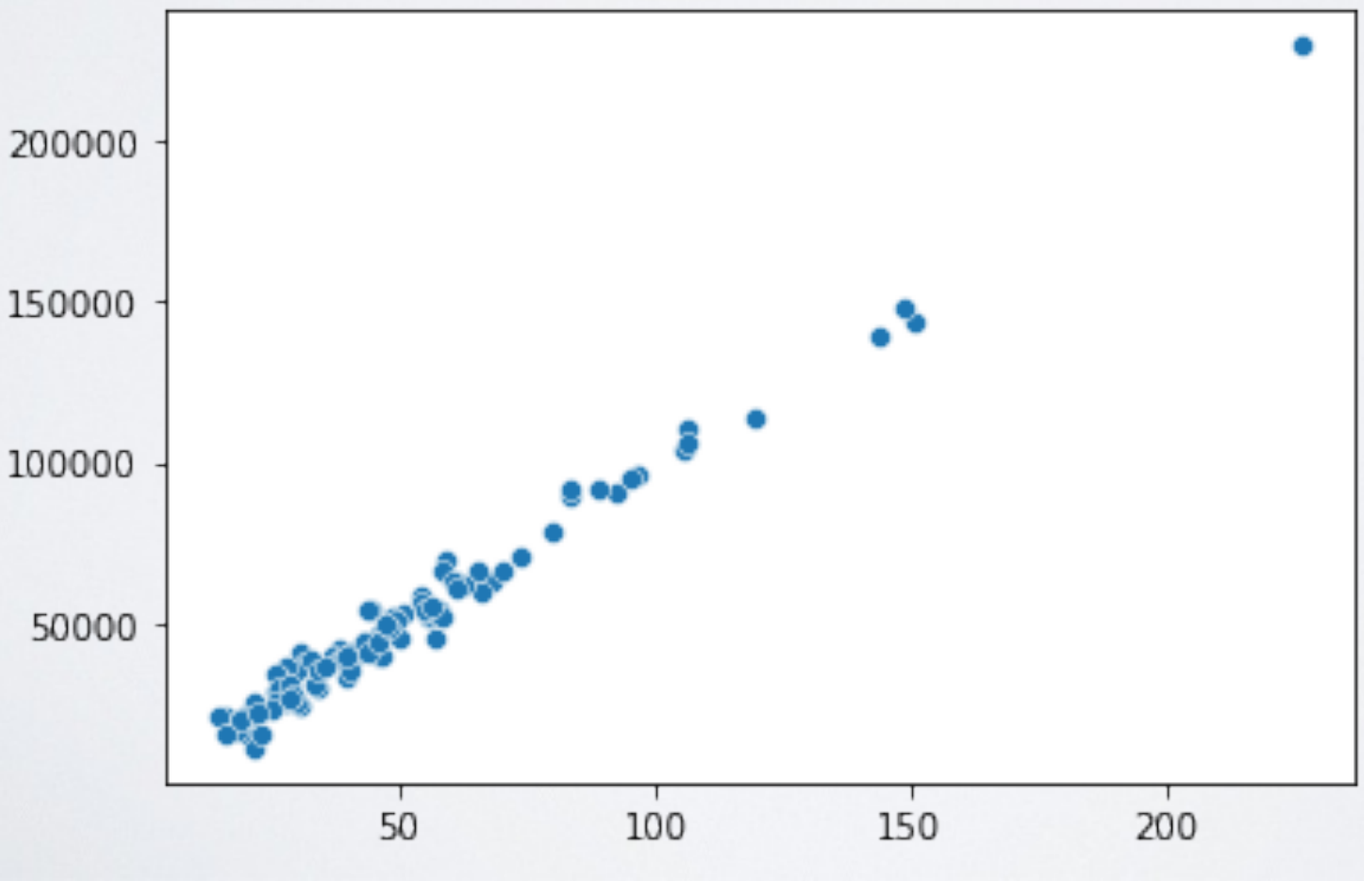
```
MSE 1179133659.4166086
RMSE 34338.51568452848
MAE 21658.66828240126
R2 -0.0667561537620748 9
```

MAE lower

LINEAR REGRESSION

- Let's assume that we know one apartment attribute: Surface area. We can plot the relation between Surface and Price

- There seems to be a linear relationship



LINEAR REGRESSION

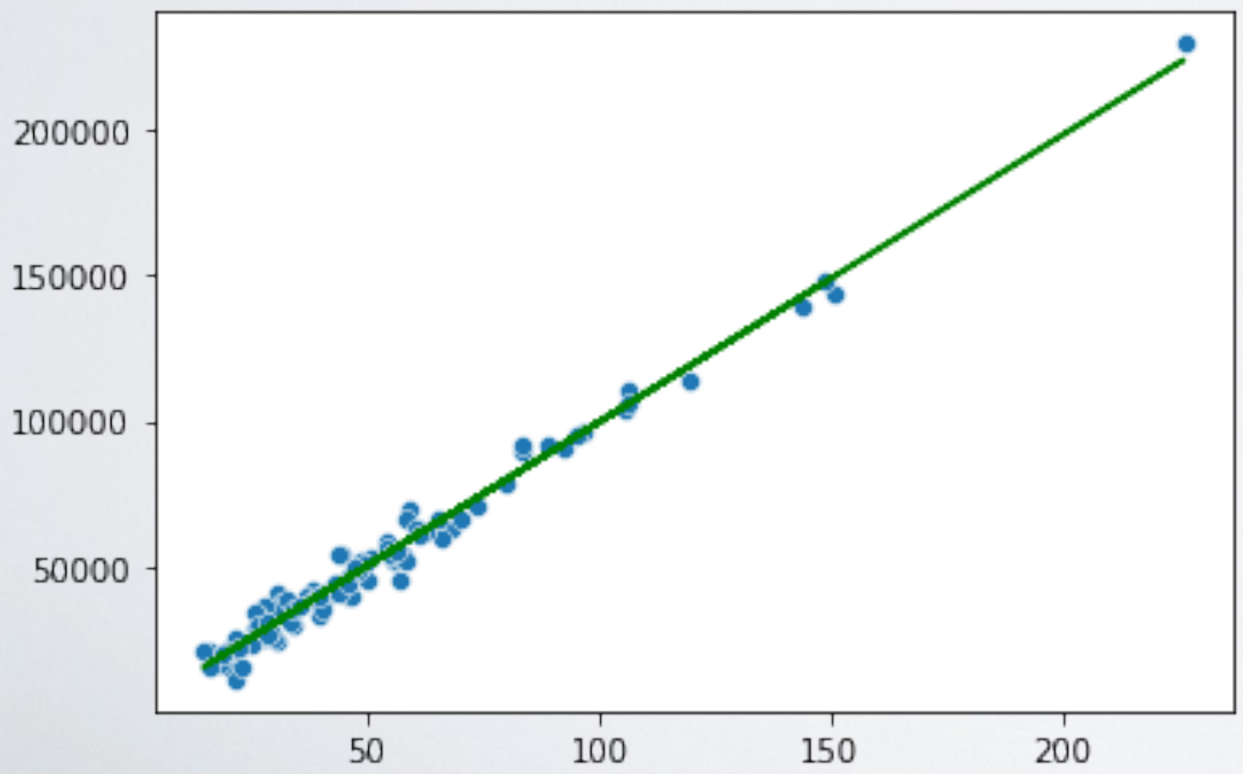
- We will use **linear regression** method, and more specifically **Ordinary Least Square**. First, with a single variable:

- We assume that: $y_i = \beta_0 + \beta_1 x_i + \epsilon$
  - Target value=constant+(constant*feature)+normally distributed (random) errors
  - i=>ith example in our dataset

- The objective of linear regression is to find parameters $\Theta = \{\beta_0, \beta_1\}$
  - Such as to minimize the **MSE**,
  - Considering that the prediction is: $\hat{y}_i = \beta_0 + \beta_1 x_i$
    - Equivalently: $\hat{y} = \beta_0 + \beta_1 x$

LINEAR REGRESSION

- We solve this problem, and obtain:
  - β_0=987
  - β_1=779




```
MSE 20668278.463901177
RMSE 4546.237836266508
MAE 3512.3861644882704
R2 0.981301514834252 8
```

LINEAR REGRESSION

- We solve this problem, and obtain:
  - β_0=987
  - β_1=779

Using Mean

```
MSE 1105345073.7155044
RMSE 33246.73027104326
MAE 22740.967725747014
R2 0.0
```

Using Median

```
MSE 1179133659.4166086
RMSE 34338.51568452848
MAE 21658.66828240126
R2 -0.0667561537620748 9
```

Using Linear Regression

```
MSE 20668278.463901177
RMSE 4546.237836266508
MAE 3512.3861644882704
R2 0.9813015148342528
```

LINEAR REGRESSION

- Note: To generate the data, I used indeed a linear model, with parameters
  - β_0=~~987~~ 0
  - β_1=~~779~~ 1000
-

Using
Linear Regression

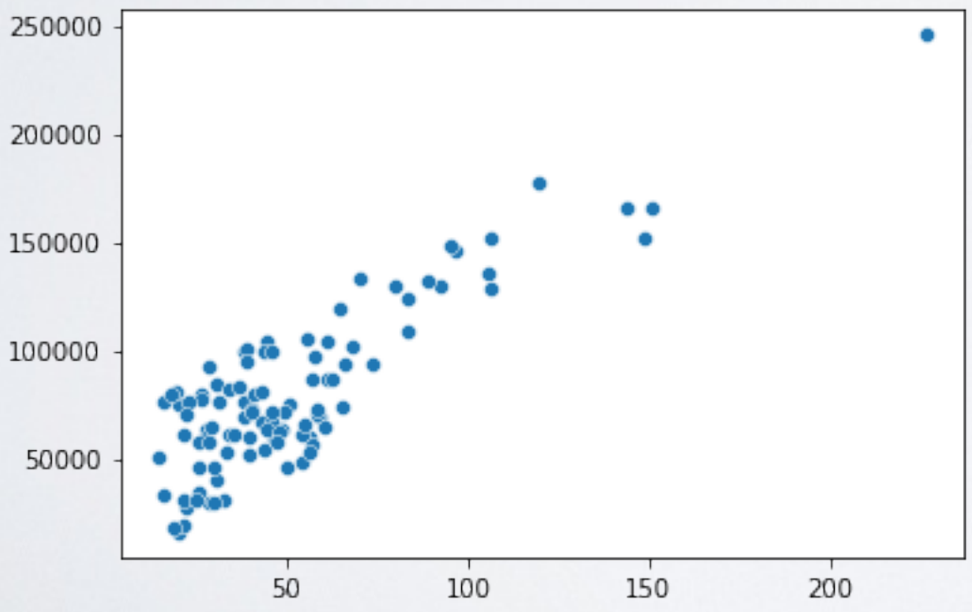
```
MSE 20668278.463901177
RMSE 4546.237836266508
MAE 3512.3861644882704
R2 0.981301514834252 8
```

Using
"Real" generative model

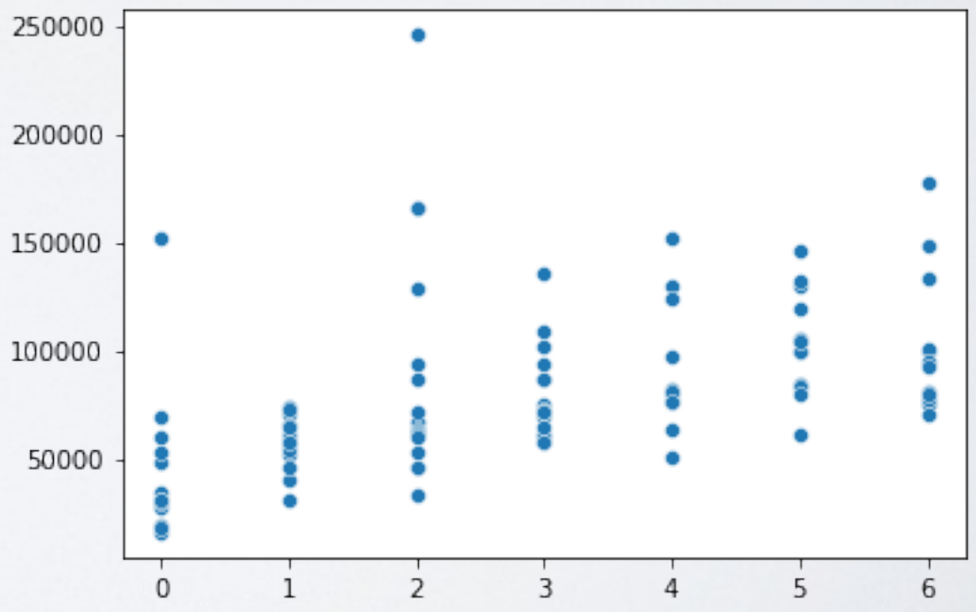
```
MSE 20863741.73315057
RMSE 4567.68450455486
MAE 3506.783422078361
R2 0.9811246802204318
```

LINEAR REGRESSION

- In real life, we usually have more than 1 parameter
  - New generator, prices depends on surface AND floor



Surface



Floor

LINEAR REGRESSION

- General formulation with any number of attribute
  - $y = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \ldots + \beta_n x_n + \epsilon$
  - Searching for the different coefficients

Surfaces only

```
MSE 388200345.3991482
RMSE 19702.800445600322
MAE 16757.480694933285
R2 0.7329146952183824
```

Floor only

```
MSE 785600976.607142
RMSE 28028.57428780747
MAE 22165.777484397917
R2 0.3422280780552575
```

All features

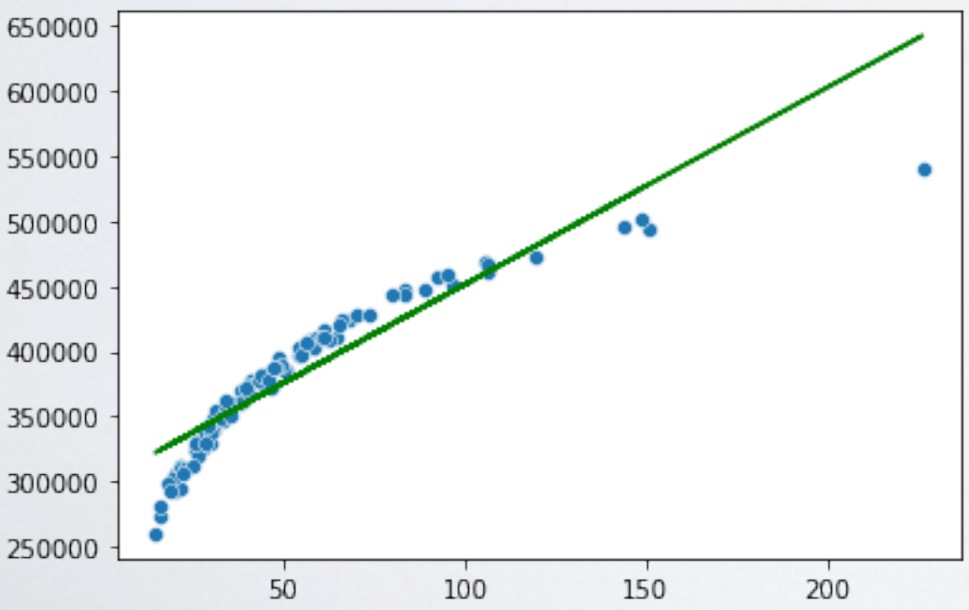
```
MSE 22157971.6387145
RMSE 4707.225471412486
MAE 3617.346073048316
R2 0.984755117612315 5
```

Generative Parameters
- $\beta_0 = 0\ \beta_1 = 1\,000, \beta_2 = 10\,000$

Found Parameters
- $\beta_0 = 579\ \beta_1 = 994, \beta_2 = 9\,821$

LINEAR REGRESSION

- Linear regression works :)
- But what happens if relations are not linear?
  - Assume that Price ≈ log(surface)*100 000 ?



Linear regression

```
MSE 474131230.6072998
RMSE 21774.554659218633
MAE 16958.426496791166
R2 0.8437196622358905
```

Real model

```
MSE 23408487.920127597
RMSE 4838.231900201518
MAE 4057.809620606243
R2 0.992842323758786
```

LINEAR REGRESSION

- Linear regression works if there are indeed linear relations
  - But there is no particular reason for relations to be linear

- In many scientific domains (e.g., epidemiology, biology, econometrics, etc.), linear regression is still widely used.
  - Why ?

OLS STRENGTH

- OLS (Ordinary Least Square Linear regression) is simple to understand and flexible:
  - If I know that a relation is log-linear, or other, I can transform my variables beforehand to fall back on a linear problem (but…)
- Linear regression is interpretable:
  - The meaning of each coefficient β can be interpreted (positive/negative, strength, significance), and thus the relative strength of the corresponding features
  - /!\ True only if strict conditions are respected:
    - No multicollinearity (correlations between variables)
    - No endogeneity (correlation between variables and errors (ϵ))
    - No heteroskedasticity (inhomogeneous distribution of errors)
    - Etc.

(OLS?)

- Common question: Is OLS machine learning or stat/econometrics?

- In my opinion:
  - Not a very meaningful question. *Is tomato a vegetable or a fruit?*
  - If we focus mainly on the coefficients, and not on the prediction => not classic machine learning
  - If we focus mainly on the prediction and compare with other models => Yes, a ML method like any other

OLS STRENGTH

- Analytical solution: $\hat{\beta} = (X^T X)^{-1} X^T y$
  - With X the feature matrix

- An analytical solution guarantees to find the optimal solution

- Possible to do before the generalization of computers

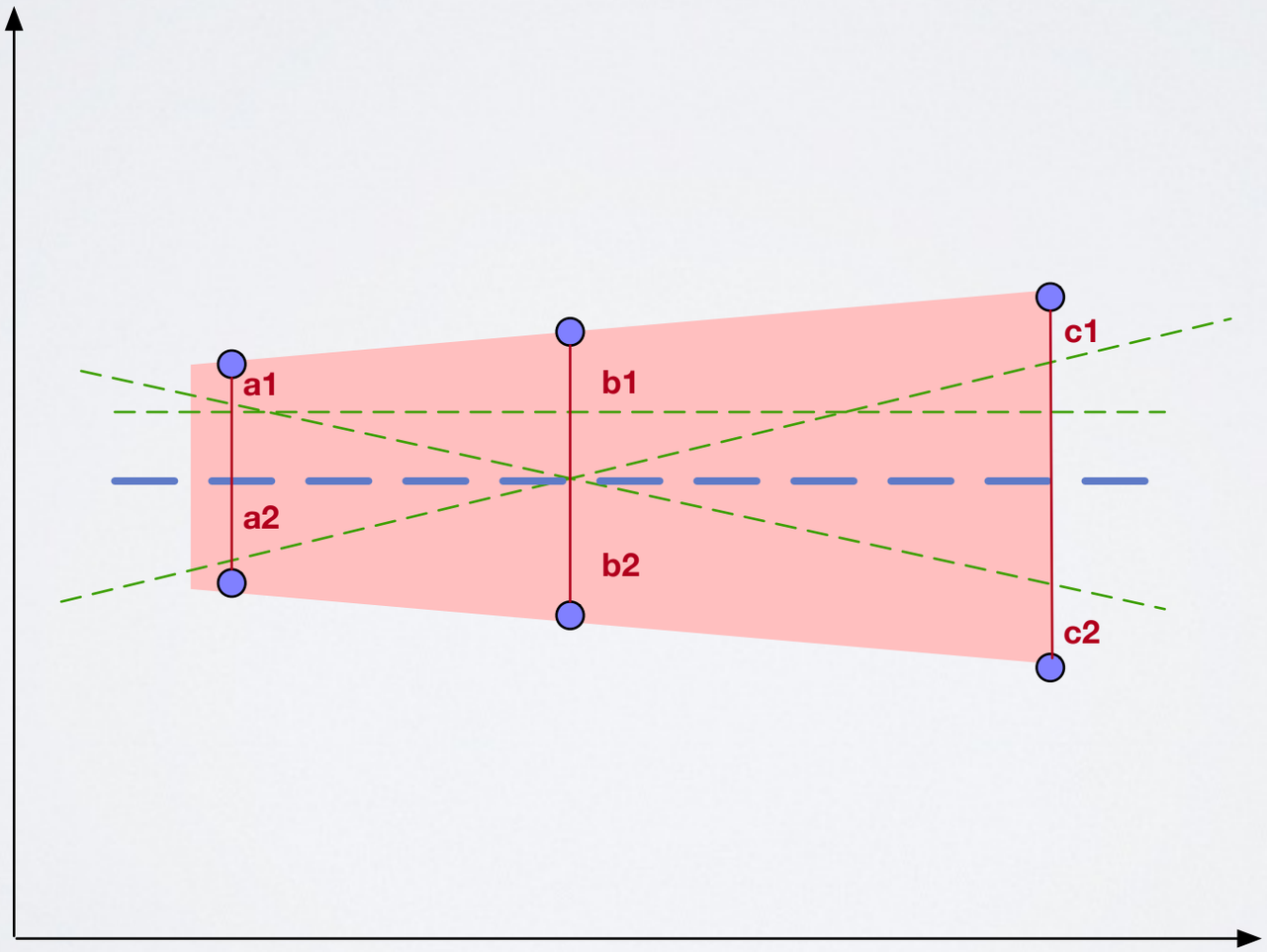
- If there are
  - Many variables, matrix inversion becomes a bottleneck $\mathcal{O}(v^3)$
  - Many observations, matrix multiplication goes $\mathcal{O}(nv)$
  - Solution=>Gradient descent, later during the class

OLS STRENGTH

- Another strength of OLS is that it is equivalent to MLE
- MLE (Maximum Likelihood Estimation) is a model-based approach
  - IF we make the assumption that our data has been generated by a random model of the form: $y = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \ldots + \beta_n x_n + \epsilon$
    - And that variables are normally distributed and independent
  - Then MLE finds the parameters of this model that generate the observed data with the highest probability (likelihood).
- It's a less "intuitive", more "scientifically grounded" way to arrive at the same method

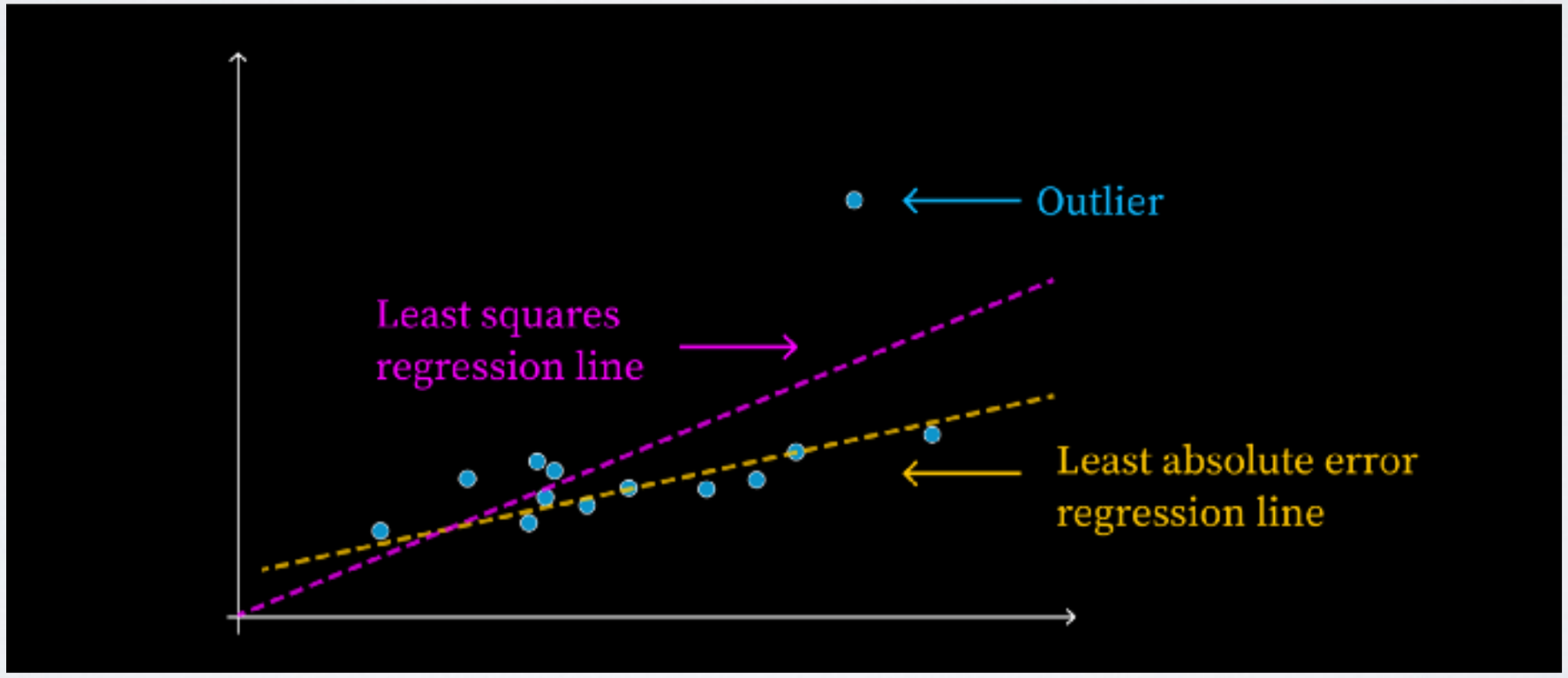
OLS STRENGTH

- Intuitive reason to use MSE instead of MAE
  - Single, "intuitive" solution (blue)
  - MSE: all the green line fits equally (a1+a2, b1+b2, c1+c2=constant)



OLS KNOWN WEAKNESS

- MSE is known to be sensitive to outliers



NON-LINEAR REGRESSION:
DECISION TREE REGRESSION

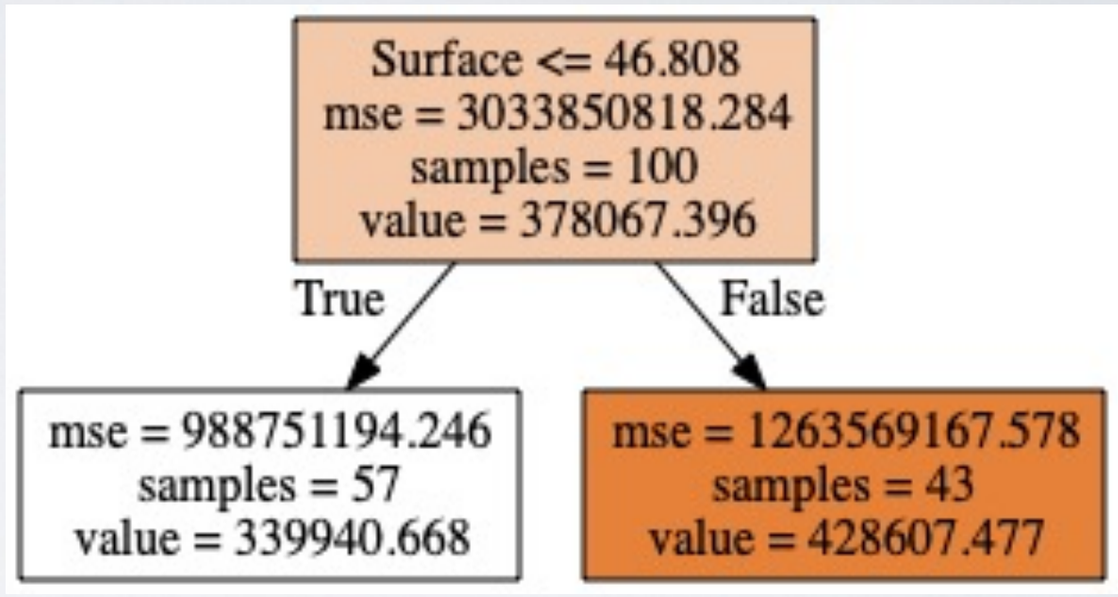
DECISION TREE

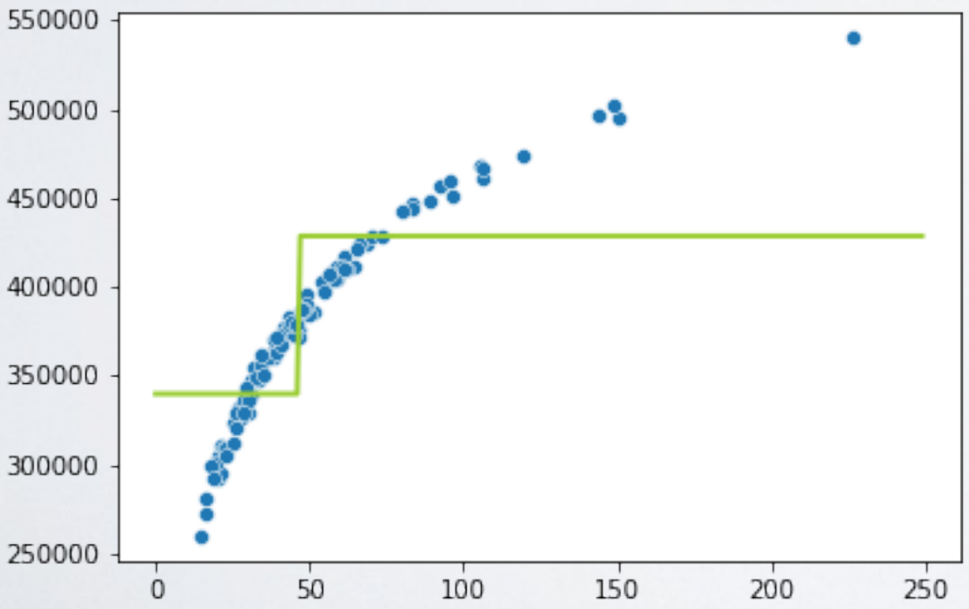
- Decision tree is a simple yet powerful way to do machine learning.

- Meta-algorithm:
  - Recursively split the data in 2 groups of items, based on a chosen attribute, so that elements in the same group have as close target values as possible
  - Predict that the value of a new item is the same as those of the group it belongs to.

DECISION TREE

- Ex: Using
  - MSE as split criteria
  - 1 Level of splitting

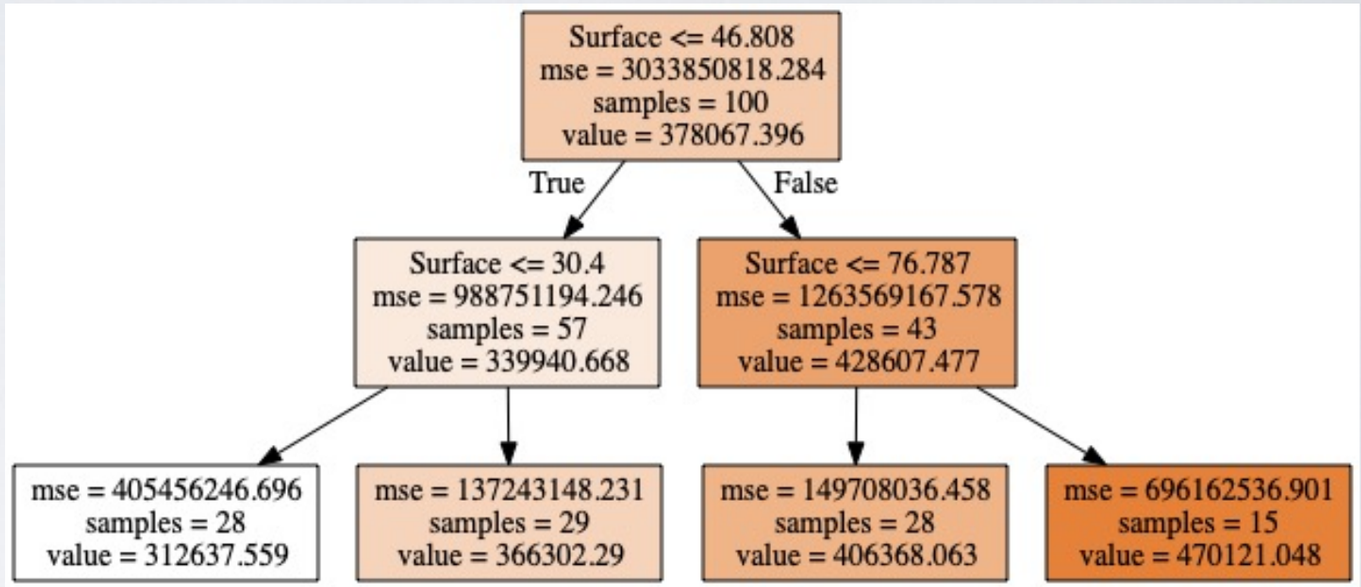


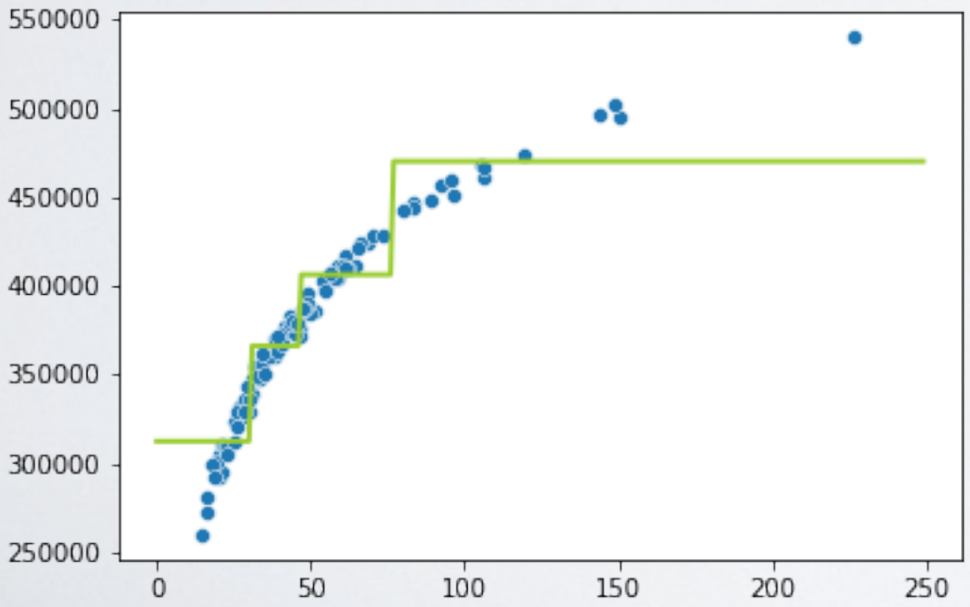



```
MSE 1106922922.7787206
RMSE 33270.45119589935
MAE 27836.40899704275
R2 0.6351425995939648
```

DECISION TREE

- Ex: Using
  - MSE as split criteria
  - 2 Level of splitting

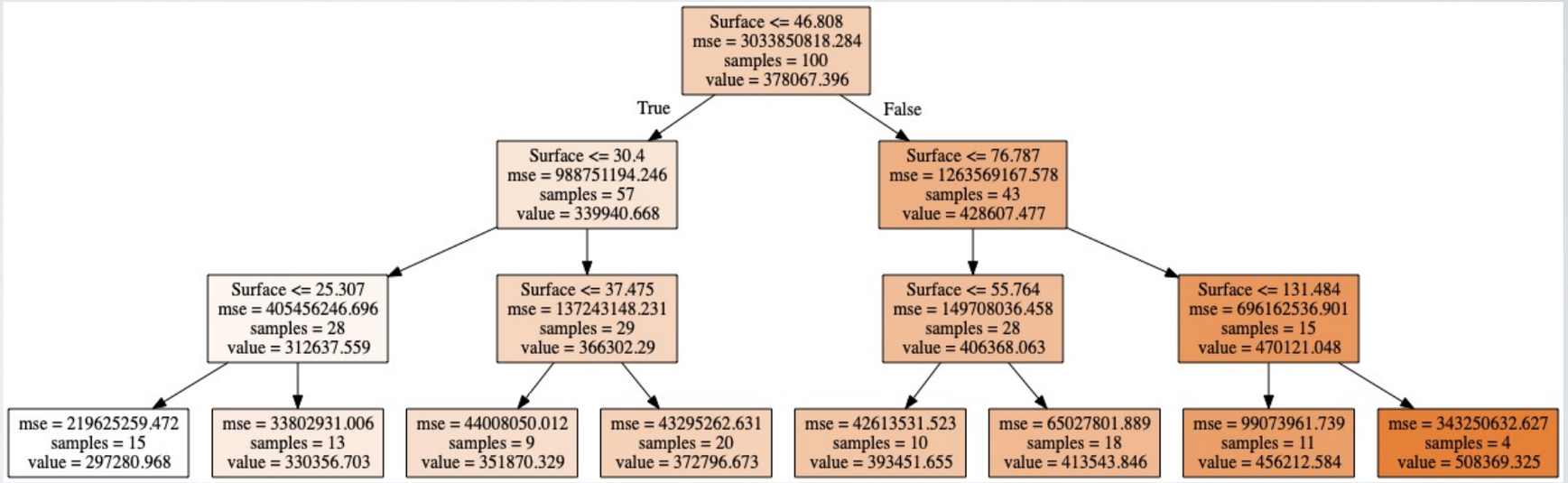


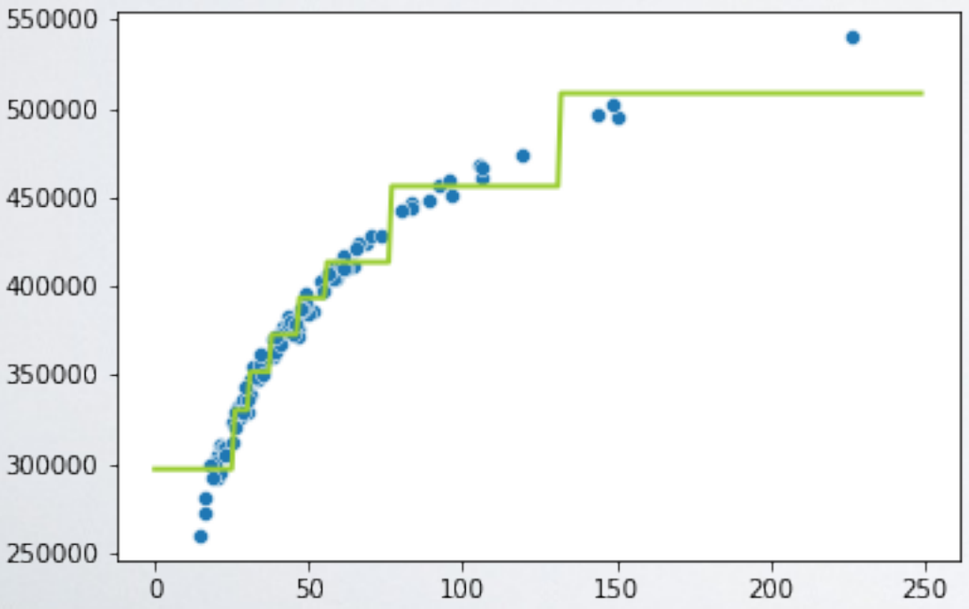



```
MSE 299670892.805488
RMSE 17311.00496232059
MAE 13262.652619929546
R2 0.901224490634346
```

DECISION TREE

- Ex: Using
  - MSE as split criteria
  - 3 Level of splitting

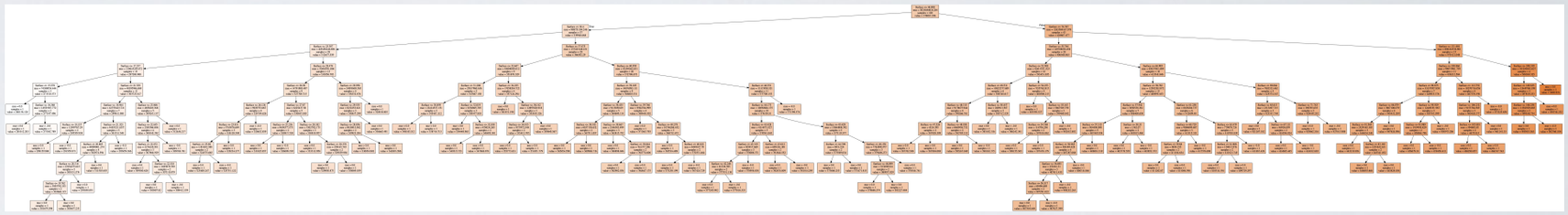


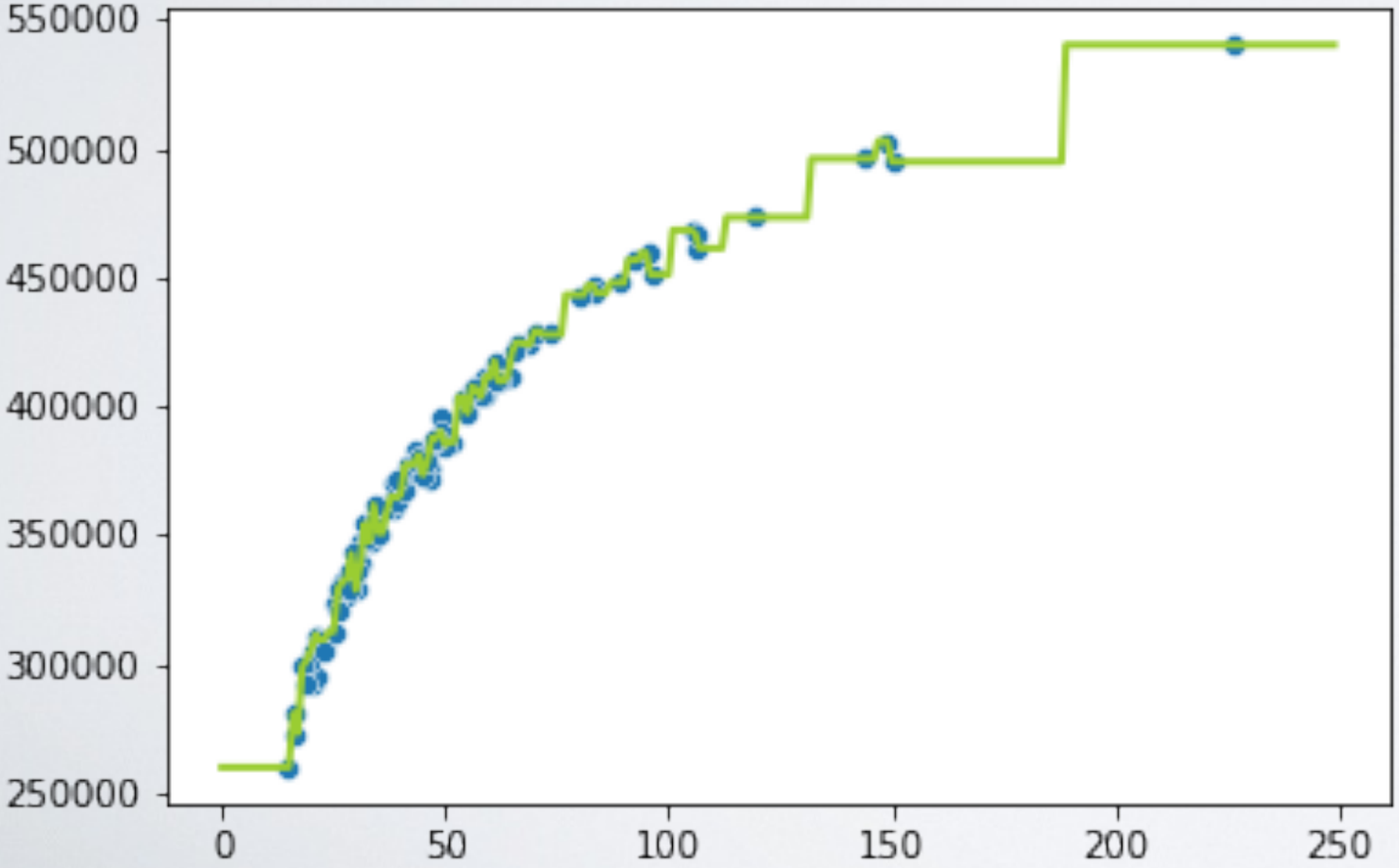



```
MSE 90552465.56733872
RMSE 9515.905924678886
MAE 7434.910779663157
R2 0.970152630768257 3
```

DECISION TREE

- Ex: Using
  - MSE as split criteria
  - 10 Level of splitting






```
MSE 0.0
RMSE 0.0
MAE 0.0
R2 1.0
```

MACHINE LEARNING: SOLVED

OR IS IT ?

OVERFITTING…

AVOIDING OVERFIT

- The most important rule of machine learning
  - And essential part of the scientific process

- Predicting what you already know is cheating
  - Even if you genuinely try not to cheat, you can cheat unintentionally
  - Experimental scientific experiments are done in **double blind:**
    - Neither the tested subject **nor the experimenter** know the placebo from real pill

- You must hide a **test set**, that you will **never** use when learning, and that you will **only use once**, for evaluating.

AVOIDING OVERFIT

- When your data is ready, before any learning, split your data into:
  - A **training** set
  - A **test** set

- You can train as many method with as many parameters as you want on the training set.

- Only when all your models are trained, you can evaluate it on the test set
  - You can never, ever reuse that (exactly same) test set.

AVOIDING OVERFIT

	Decision Tree, **levels=10**	Decision Tree, **levels=5**
Scores on Train Set	MSE 0.0 RMSE 0.0 MAE 0.0 R2 1.0	MSE 9675372.95170697 RMSE 3110.5261535159884 MAE 2364.5552169188454 R2 0.996810860674691 8
Scores on Test Set	MSE 60522590.58807978 RMSE 7779.626635519199 MAE 6427.594619486819 R2 0.968984922491333 6	MSE 47482936.48734139 RMSE 6890.786347532579 MAE 5748.307144423111 R2 0.975667152691510 4

TRAIN/EVALUATION/TEST

- In some cases, you need to see the results on the test set to know how to improve your prediction
  - Ex: how many levels in my decision tree? The right level is the one with the best results on the test set.
  - More generally: *hyperparameter tuning*.
    - If my learning method has parameters, how to fix those parameters? (Coefficient of learning, number of layers in deep NN, etc.)

- => Solution:
  - Use an evaluation set for intermediary steps (hidden like test set, but not for final evaluation)
    - You can do whatever you want with your evaluation set, it's part of your training process
  - Keep a test set, that you will use only once at the end

TRAIN/TEST SPLIT

- What size should your test set have?
  - No good answer. 66% Train, 33% Test is often a default choice

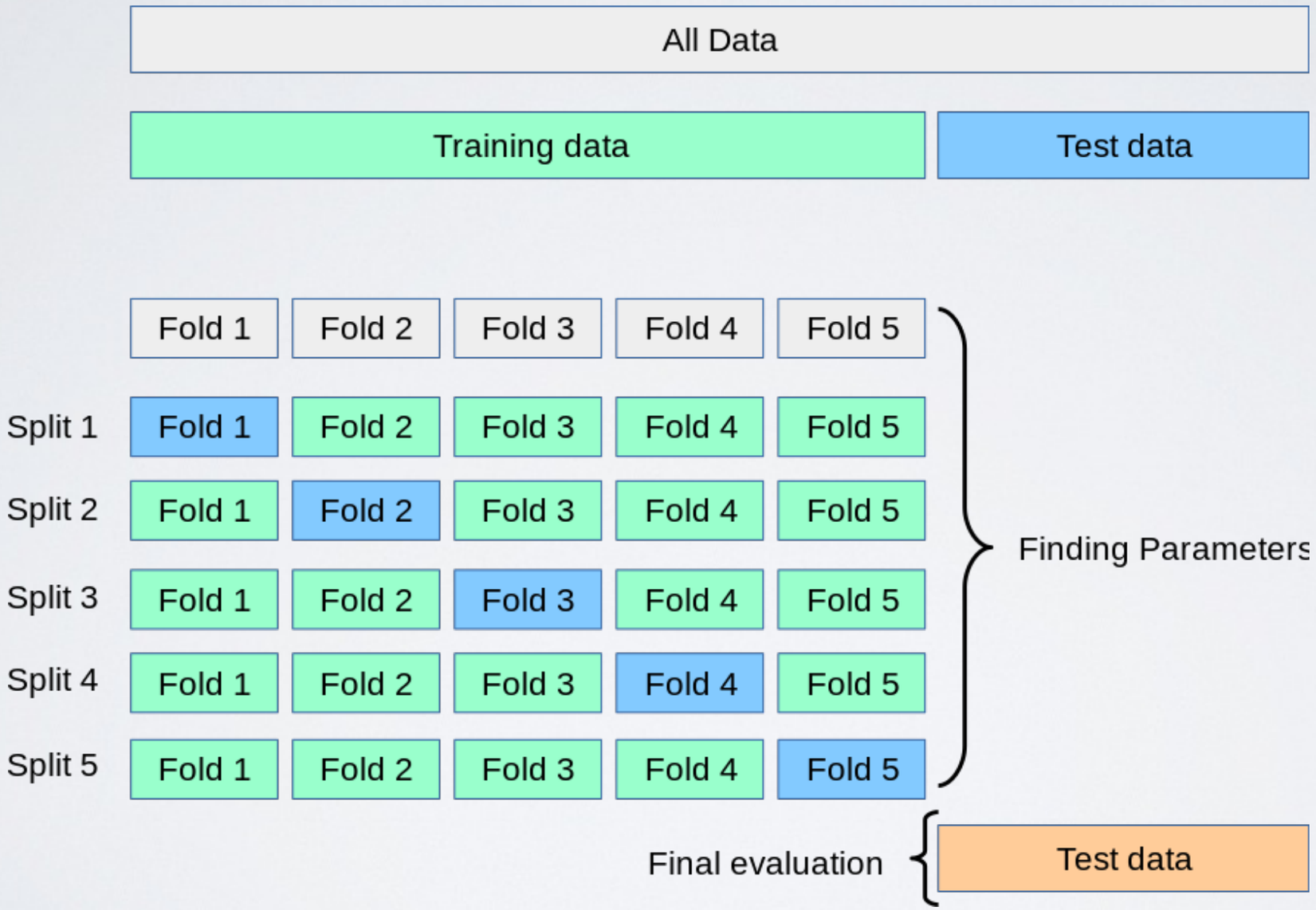
- Rule of thumb:
  - You need enough data for training. If your problem is simple (few features…) and you have many examples, then a random sample of 5% of it can be enough
    - e.g.: predict weight of a Tomato based on its species…

- Problem is if data is scarce
  - =>Cross validation

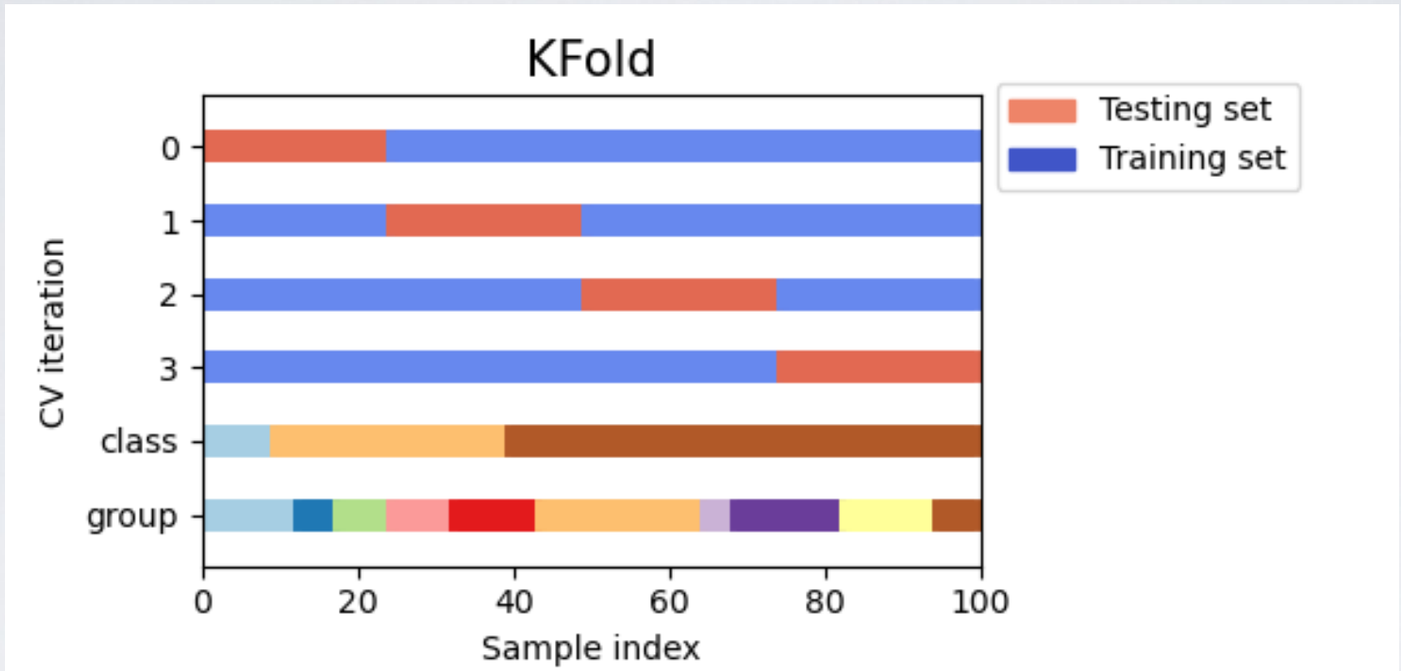
CROSS VALIDATION

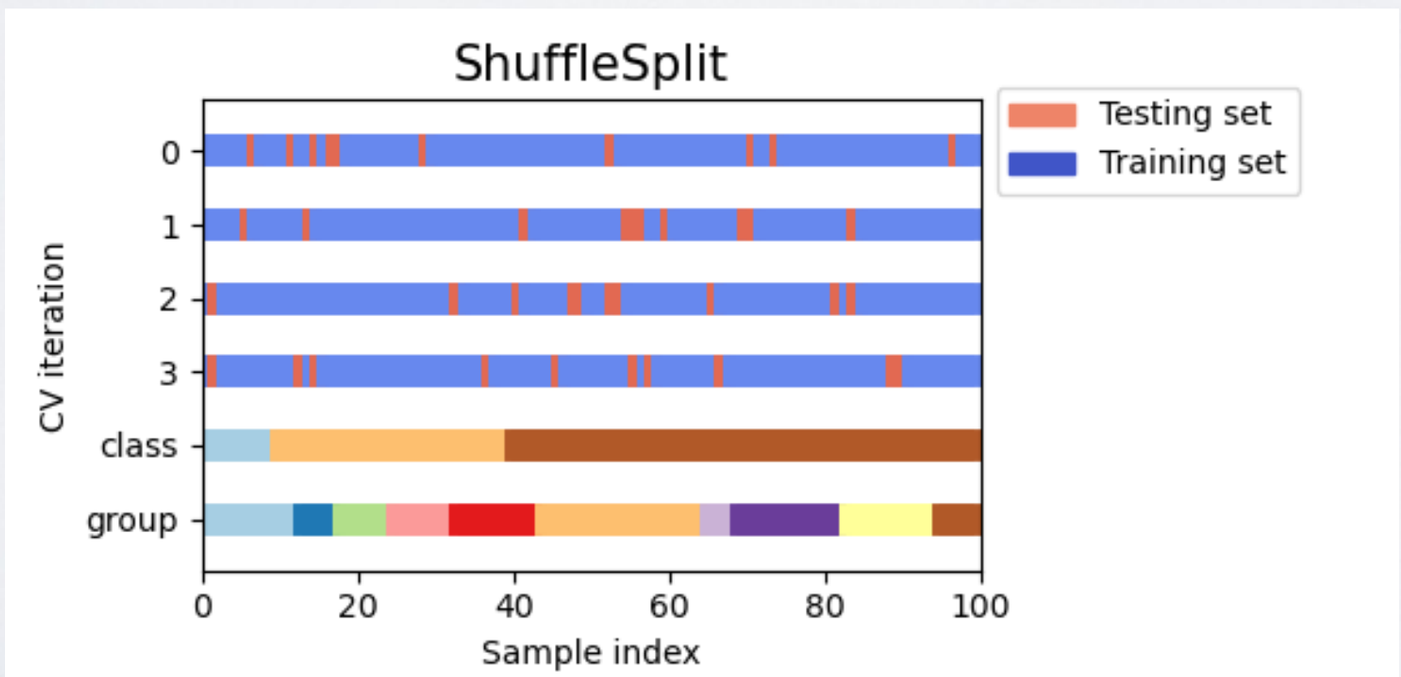
- Golden rule: One test set must be used only once
- But from a single dataset, you can create multiple test sets
  - If you just want to restart you training differently, make a new 66/33 split
- If you have too few data to put 33% aside for training:
  - Use only, e.g., 10% for the test set
  - But creates 10 train/test sets, each with different test sets
    - Then, compute the average scores over the 10 sets

CROSS VALIDATION

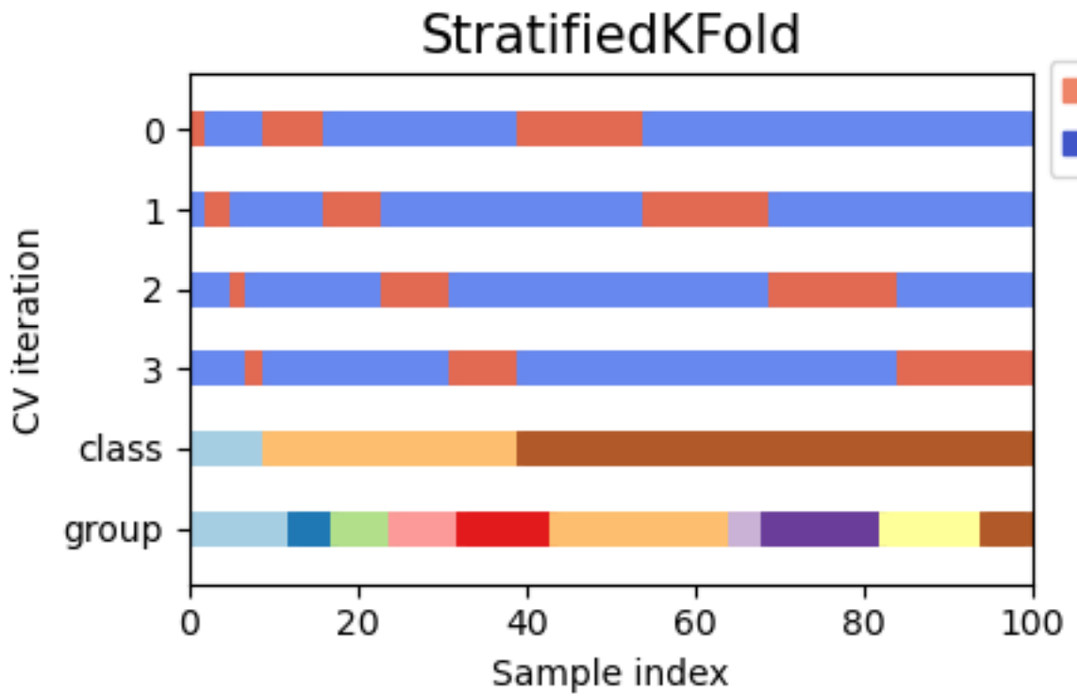


CROSS VALIDATION

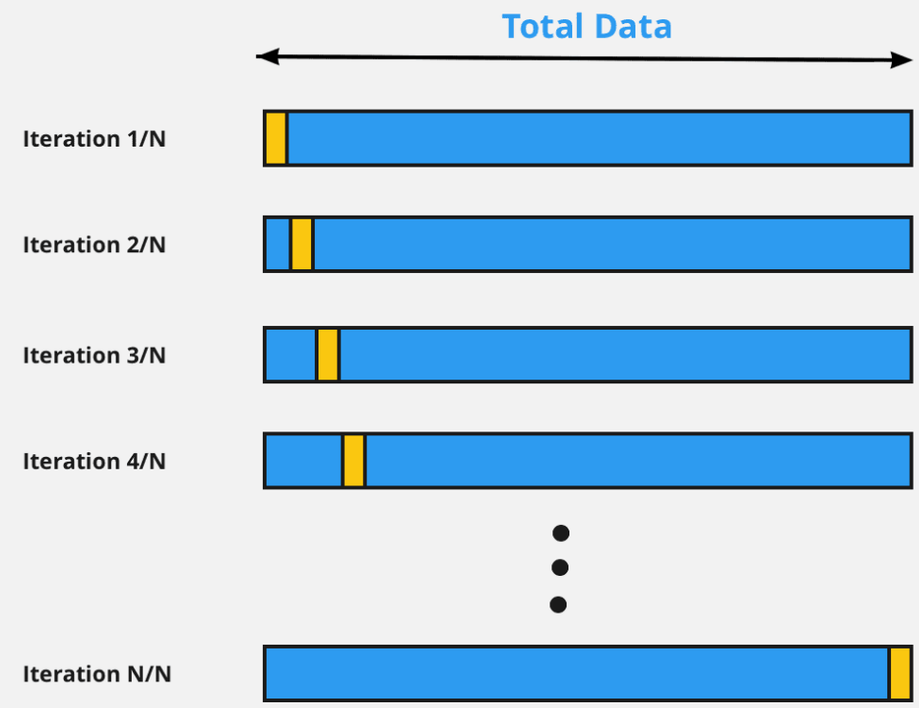




CROSS VALIDATION



LOOCV: Leave One Out Cross Validation



dataaspirant.com

CROSS VALIDATION

- Do you need Cross validation and which one?

- In my opinion:
  - If your data is very scarce, you need it from the beginning
  - If you are at a point in which every fraction of a % of accuracy count.

- A simple train/test is enough for most cases.

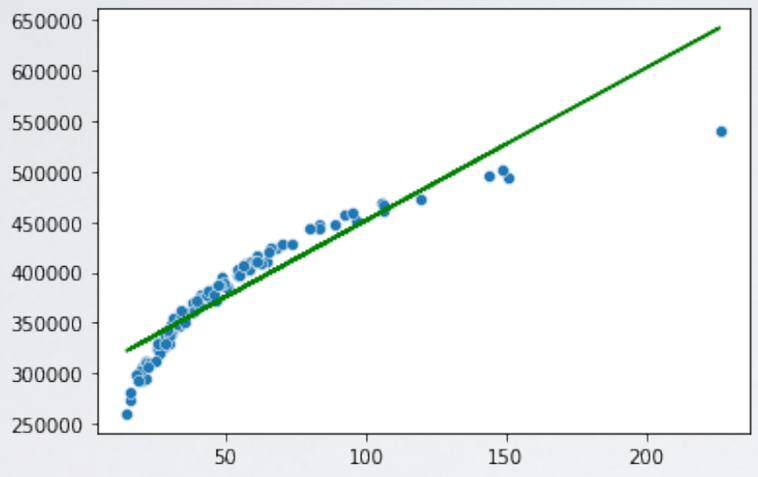
FIGHTING OVERFIT

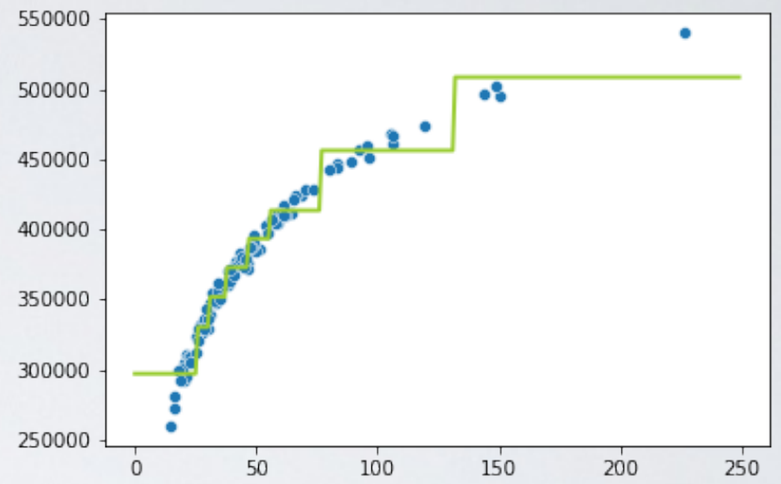
BACK TO THE METHOD

FIGHTING OVERFIT

- Implicit limit to overfit:
  - Because a method has a limited power of expression, it cannot overfit "too much".
    - Trivial solution: each point has its own prediction. No **generalization**
  - =>A linear regression method cannot overfit to the trivial solution, unlike decision tree
    - Unless there are enough variables…
- Explicit limit to overfit:
  - The method is not limited in its power of expression, but contains a safeguard against overfit
- Not necessarily a clear boundary between the 2

FIGHTING OVERFIT





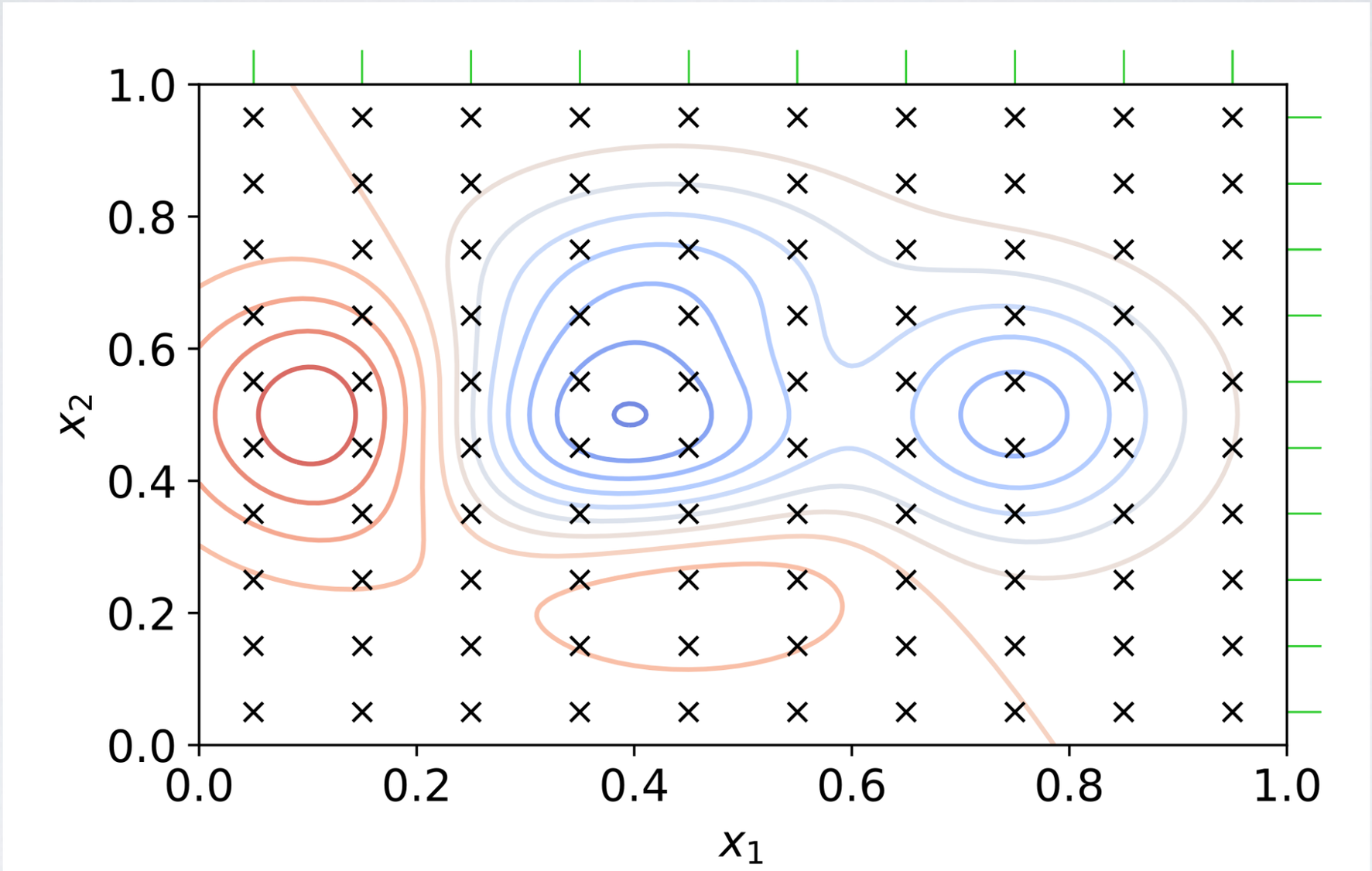
Train	MSE 474131230.6072998 RMSE 21774.554659218633 MAE 16958.426496791166 R2 0.843719662235890**5**	MSE 9675372.95170697 RMSE 3110.5261535159884 MAE 2364.5552169188454 R2 0.996810860674691**8**
Test	MSE 297361867.9984524 RMSE 17244.18359907051 MAE 14666.202886910516 R2 0.847615554878275**9**	MSE 47482936.48734139 RMSE 6890.786347532579 MAE 5748.307144423111 R2 0.975667152691510**4**

FIGHTING OVERFIT

- Avoiding overfit in decision trees: Pruning strategies
  - One way to see: Artificially limit the expressivity of the model
  - 1)Limit the number of levels (Simple but naive)
  - 2) Limit the number of leaves
    - =>Split nodes in priority where it improves the most
  - 3) Limit the size of leaves
    - => Explicitly forbids the naive solutioN

- Hyperparameter tuning/optimization
  - Typical approach: Grid search.
  - Fix a set of possible parameters. Test all possibilities on a validation test

GRID SEARCH



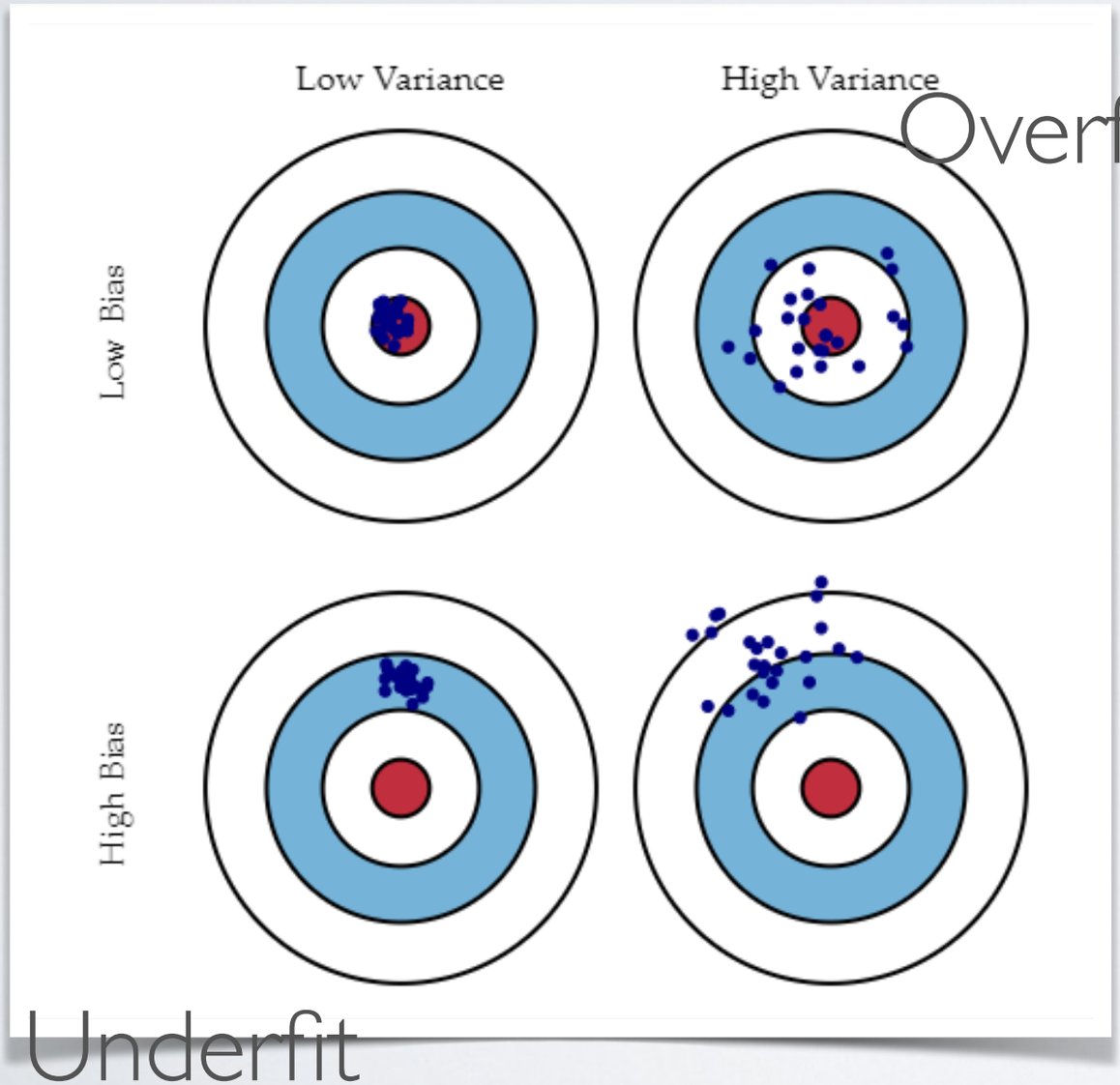
More clever methods exist: Bayesian optimization, etc.

NOTE: GENERALIZATION

- A very important notion in machine learning is Generalization
  - Can we extract generic principles underlying our data?
  - Can we generalize our observations to unseen cases?

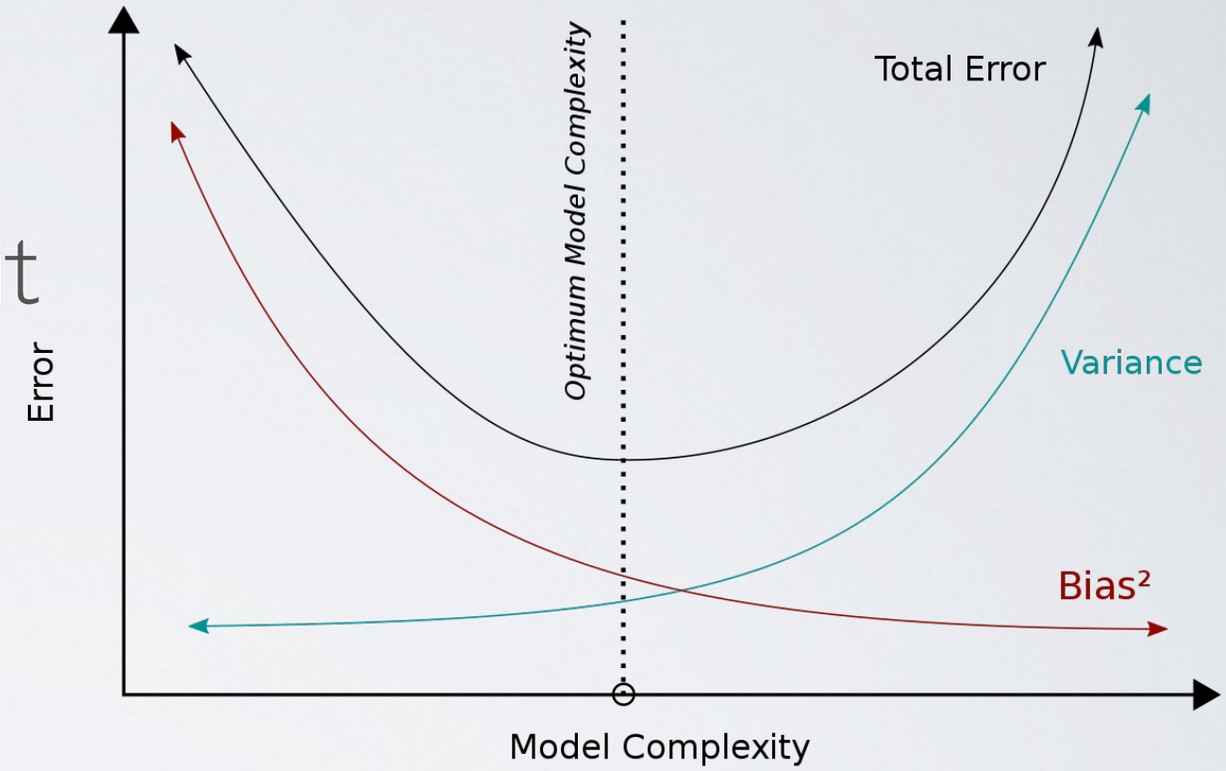
- Linear regression can predict an unseen value, while decision tree cannot.
  - What the weather be like in 5 years ? Extrapolation from current condition…

OVERFIT/UNDERFIT, HIGH BIAS HIGH VARIANCE



Overfit

Underfit



$$\mathrm{Bias}_D\left[\hat{f}(x;D)\right] = \mathrm{E}_D\left[\hat{f}(x;D)\right] - f(x)$$

$$\mathrm{Var}_D\left[\hat{f}(x;D)\right] = \mathrm{E}_D[\left(\mathrm{E}_D[\hat{f}(x;D)] - \hat{f}(x;D)\right)^2].$$

D: subsets.
X: all elements in all subsets

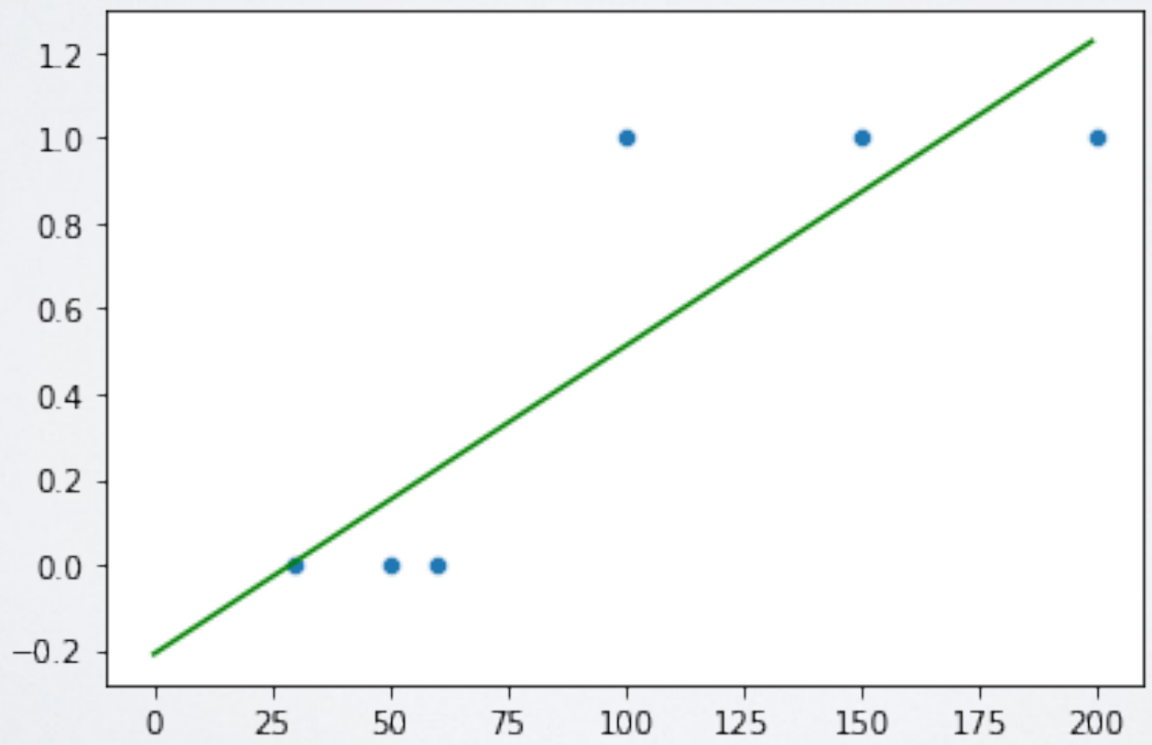
CLASSIFICATION

CLASSIFICATION

- Objective: predict the class of an item
- Methods for regression can be reused with some adaptations
  - Binary Classification is usually simple
  - Multiclass Classification might require more changes
- Evaluation methods change
- Imbalanced datasets might become a problem

LINEAR CLASSIFICATION

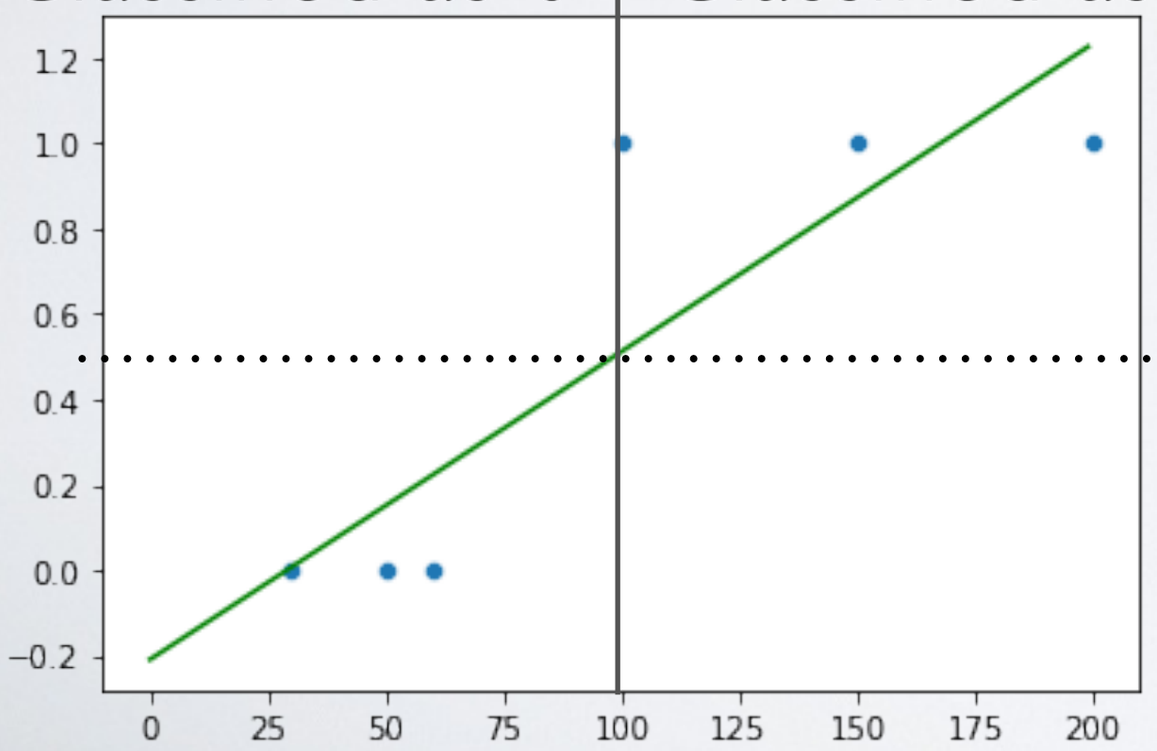
- We can easily adapt linear regression
- Imagine a 1 feature example:
  - We want to classify between apartments and houses
  - Our (unique) feature is dwelling surface



LINEAR CLASSIFICATION

- We can easily adapt linear regression
- Imagine a 1 feature example:
  - We want to classify between apartments and houses
  - Our (unique) feature is dwelling surface

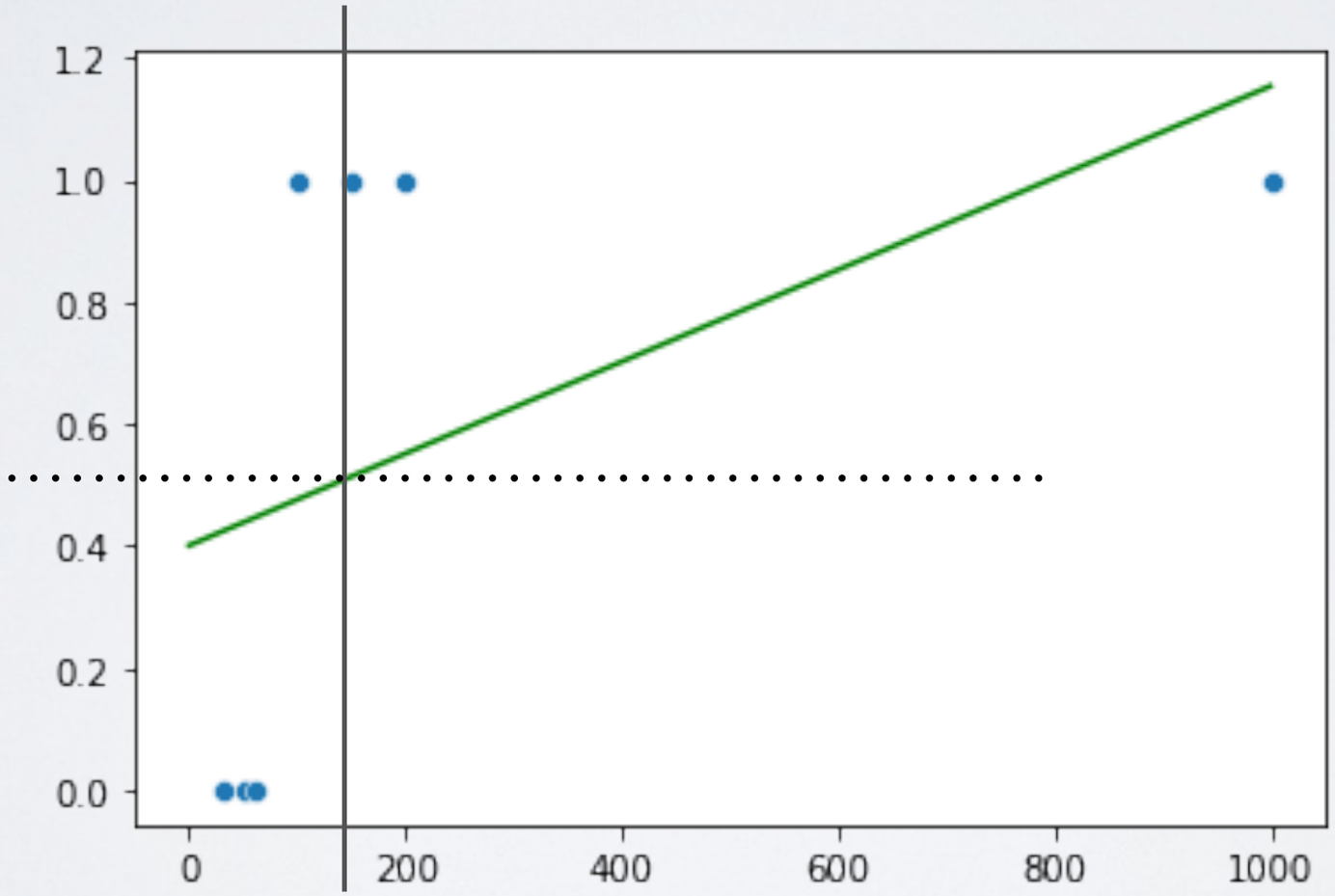
Classified as 0 | Classified as 1




```
MSE 0.0636152055857253 8
RMSE 0.252220549491363 6
MAE 0.205068528575122 92
R2 0.745539177657098 5
```

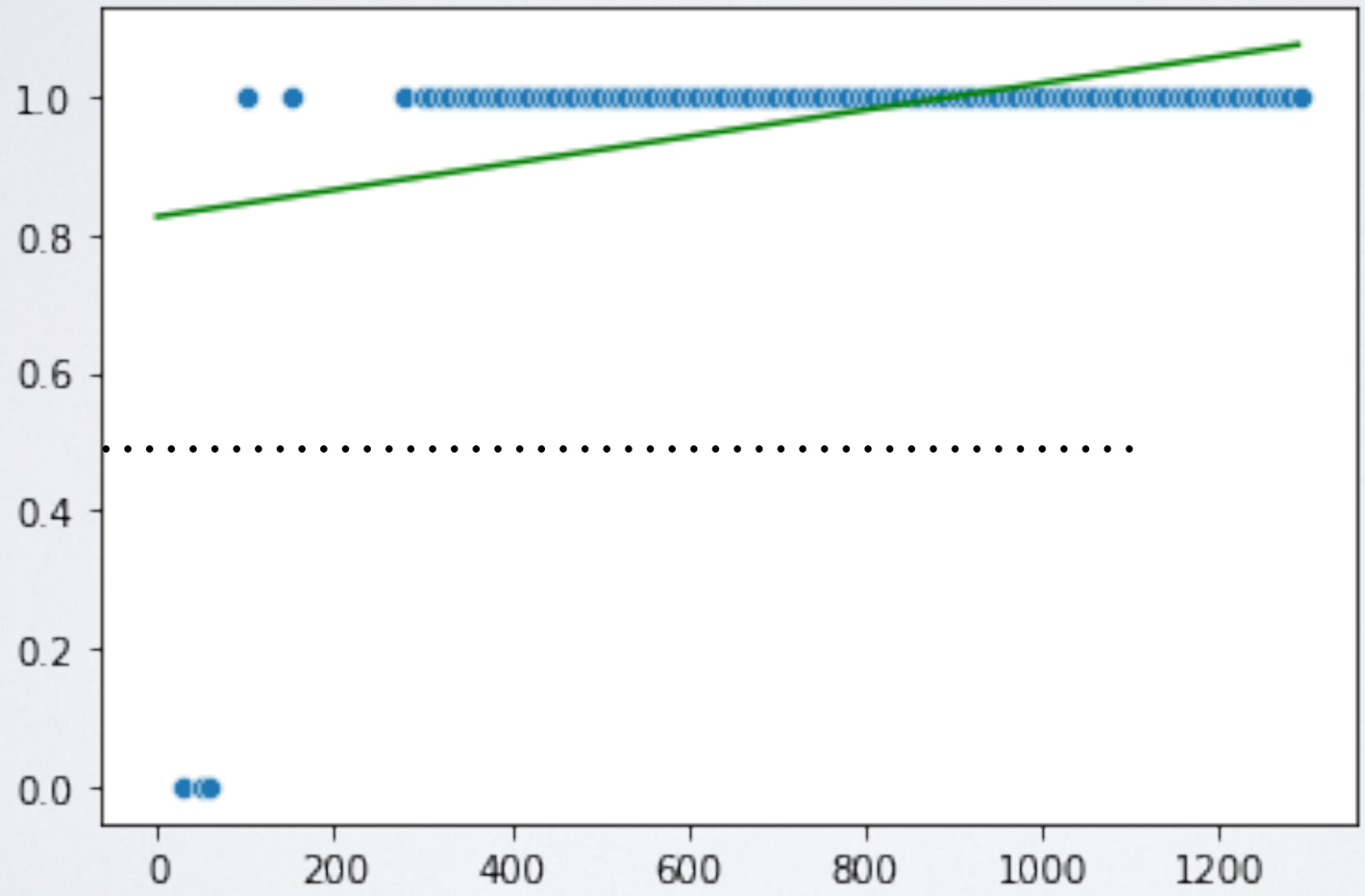
LINEAR CLASSIFICATION

- Weaknesses: Outliers



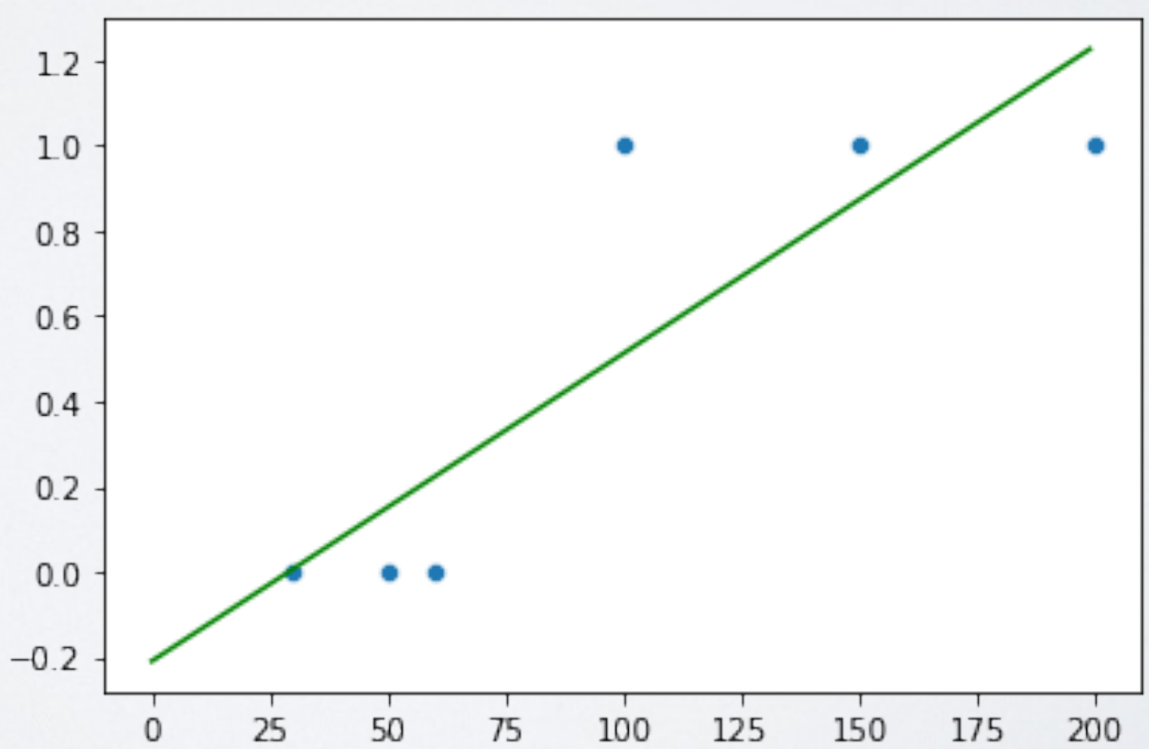
LINEAR CLASSIFICATION

- Weaknesses: Class imbalance

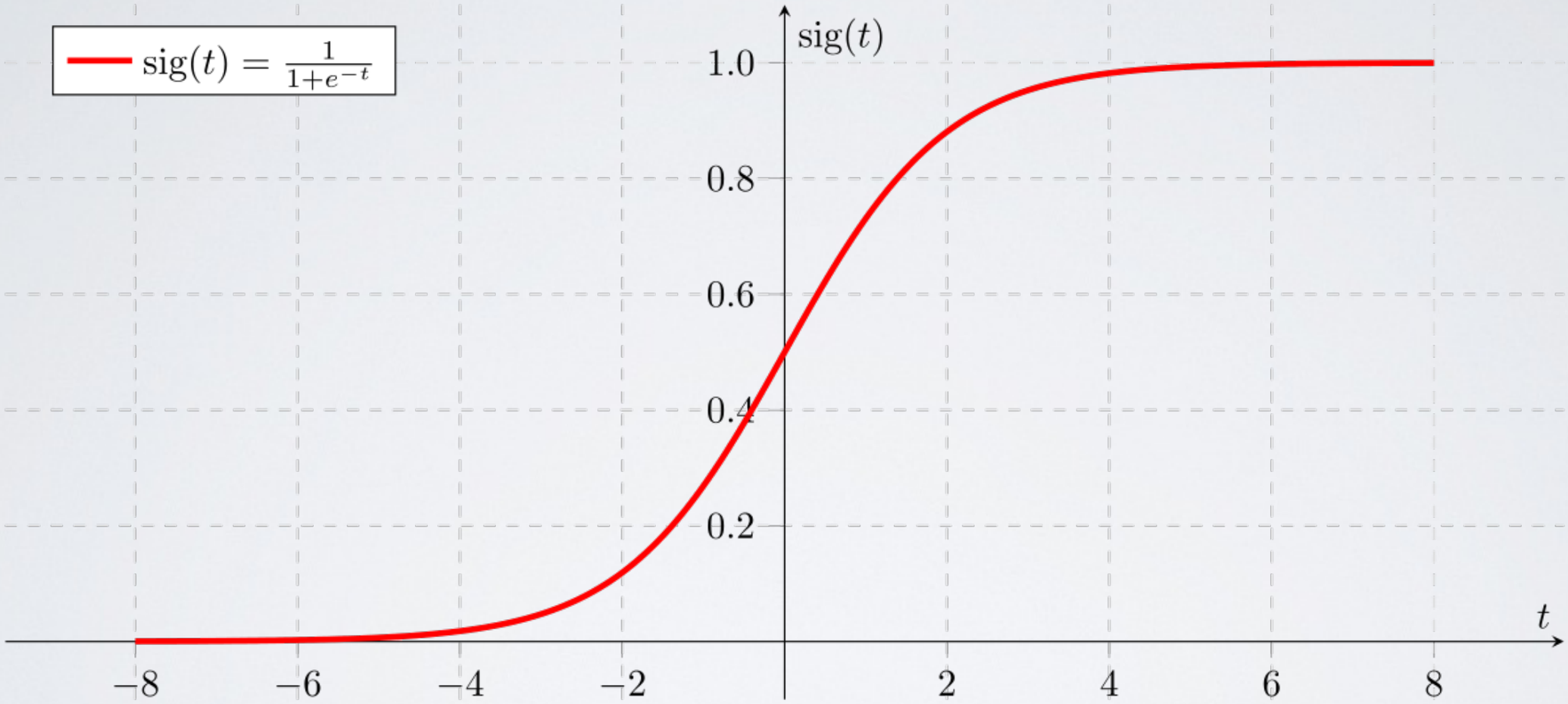


LINEAR CLASSIFICATION

- More generally, inadapted objective:
  - The relation is not linear
  - We minimize a cost function (MSE) which is not meaningful:
    - Some predictions go *beyond* possible values (prediction less than 0 or more than 1 adding error



SIGMOID FUNCTION



$$\lim_{t \to -\infty} sig(t) = 0 \qquad \lim_{t \to +\infty} sig(t) = 1 \qquad sig(0) = 0.5$$

LOGISTIC REGRESSION

Logisitic (Sigmoid) function:

$$Sig(x) = \frac{1}{1 + e^{-x}}$$

Linear regression:

$$\hat{y} = \beta_0 + \beta_1 x_i + \beta_2 x_2 + \ldots + \beta_n x_n$$

Logistic Regression:

$$P(y = 1) = Sig(\beta_0 + \beta_1 x_i + \beta_2 x_2 + \ldots + \beta_n x_n)$$

$$P(y = 1) = \frac{1}{1 + e^{-\beta_0 + \beta_1 x_i + \beta_2 x_2 + \ldots + \beta_n x_n}}$$

LOGISTIC REGRESSION

$$P(y=1) = \frac{1}{1+e^{-\beta_0+\beta_1 x_i+\beta_2 x_2+\ldots+\beta_n x_n}}$$

$$\frac{1}{P(y=1)} = 1+e^{-\beta_0+\beta_1 x_i+\beta_2 x_2+\ldots+\beta_n x_n}$$

$$\frac{1-P(y=1)}{P(y=1)} = e^{-\beta_0+\beta_1 x_i+\beta_2 x_2+\ldots+\beta_n x_n}$$

$$\frac{P(y=1)}{1-P(y=1)} = e^{\beta_0+\beta_1 x_i+\beta_2 x_2+\ldots+\beta_n x_n}$$

LOGISTIC REGRESSION

$$ln(\frac{P(y=1)}{1-P(y=1)}) = \beta_0 + \beta_1 x_i + \beta_2 x_2 + \ldots + \beta_n x_n$$

$$\frac{P(y=1)}{1-P(y=1)} = e^{\beta_0+\beta_1 x_i+\beta_2 x_2+\ldots+\beta_n x_n}$$

$$\frac{P(y=1)}{1-P(y=1)} = e^{\beta_0} e^{\beta_1 x_i} e^{\beta_2 x_2} (\ldots) e^{\beta_n x_n}$$

LOGISTIC REGRESSION

$$ln(\frac{P(y=1)}{1-P(y=1)}) = \beta_0 + \beta_1 x_i + \beta_2 x_2 + \ldots + \beta_n x_n$$

Problem to solve similar to a linear regression.
We minimize the error between true $y \in \{0,1\}$
and estimated probability of being 1

LOGISTIC REGRESSION

/!\ log transform of the target variable => multiplicative relation between variables

Interpretation as **odd ratios:**

$+1$in x_i =>prediction multiplied by e^{β_i}

$$\frac{P(y=1)}{1-P(y=1)} = e^{\beta_0} e^{\beta_1 x_i} e^{\beta_2 x_2} (\ldots) e^{\beta_n x_n}$$

https://christophm.github.io/interpretable-ml-book/logistic.html

MULTICLASS LOGISTIC REGRESSION

- In many cases, we have more than 2 classes
  - e.g.: {house, apartment, office, industrial}. {cat,dog,horse,…}
  - Categories are unordered=> conversion to numeric would be catastrophic

- Simple solution (often used): one VS all
  - Train a logistic classifier on one class VS all other classes.
  - Attribute the class with the largest confidence
    - e.g.: house: 20%. Apartment: 30%. Office: 70%. Industrial: 80%=>Industrial.
  - Rather a heuristic than principled method.

- Alternative approach: softmax regression
  - Later in class

DECISION TREE

- Trees can be easily adapted to the classification task
  - It is even more natural than for regression

- The principle is to divide observations in term of **class homogeneity**
  - We want items in the same branch/leaf to belong to the same class

DECISION TREE

- Most common homogeneity/diversity/inequality/purity scores
  - p_i: fraction of items of class i
  - Gini Coefficient: $1 - \sum_j p_j^2$
  - Entropy: $-\sum_j p_j \cdot log_2 p_j$

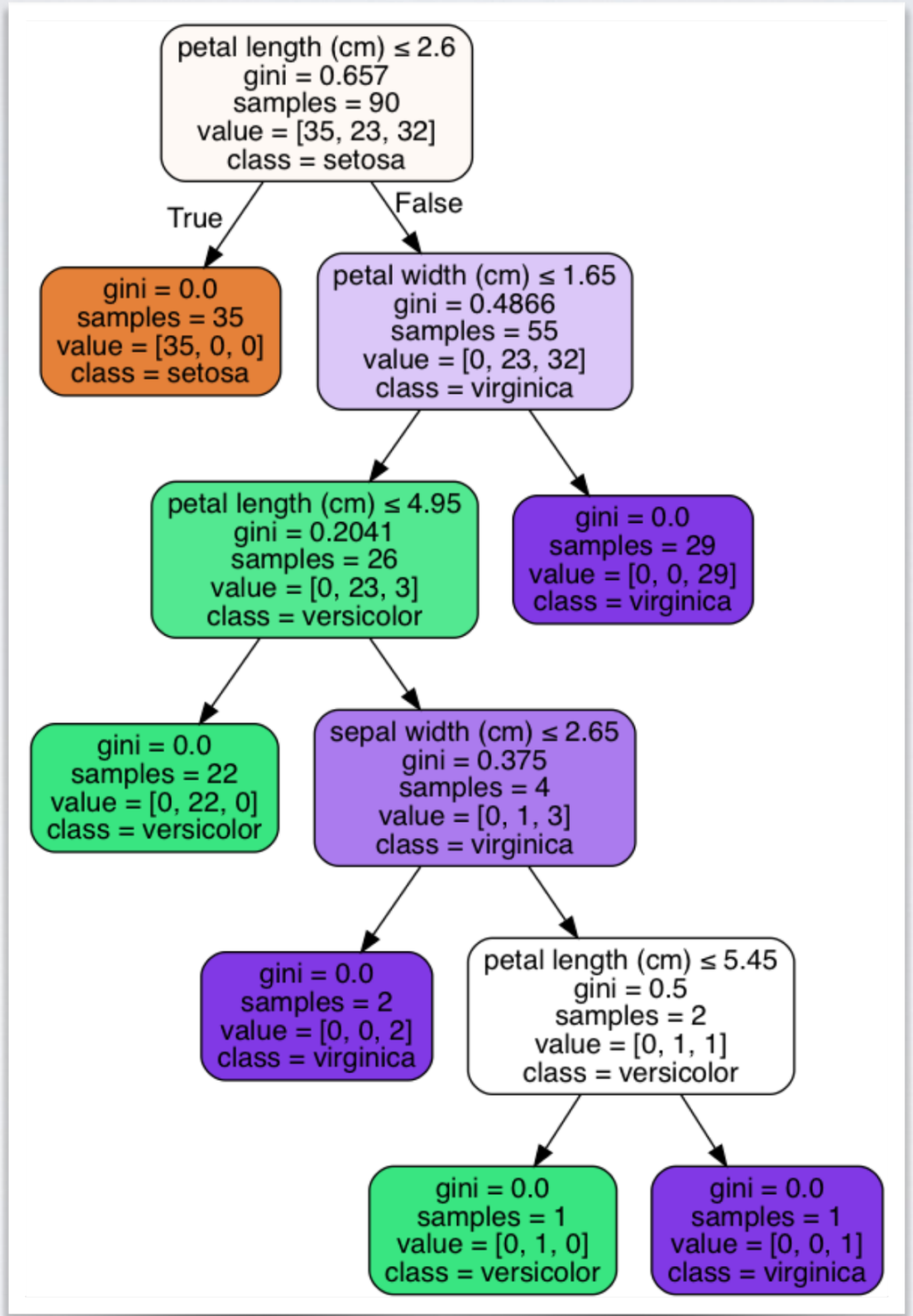
DECISION TREE

- Most common homogeneity/diversity/inequality/purity scores
  - p_i: fraction of items of class i
  - Gini Coefficient: $1 - \sum_j p_j^2$
    - Min: 0: 1 class only
    - Max: 0.5: (2 classes), 0.66(3classes), 0.75 (4classes), 0.875(8classes)
  - Interpretation:
    - If we classify by taking an element at random, probability to be wrong.

DECISION TREE

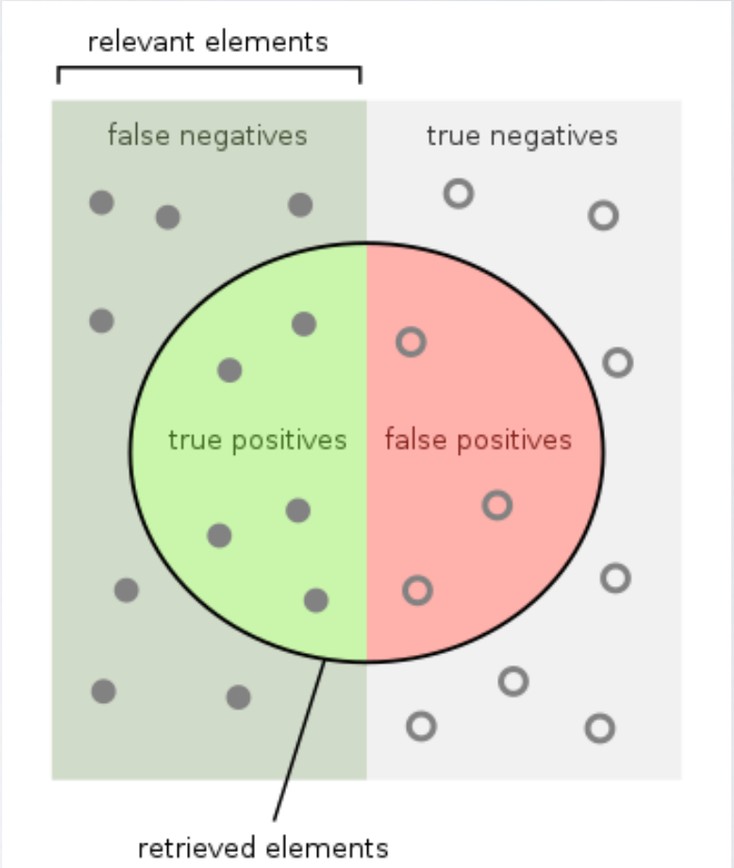
- Most common homogeneity/diversity/inequality/purity scores
  - p_i: fraction of items of class i
  - Entropy: $-\sum_j p_j \cdot log_2 p_j$
    - Min: 0: 1 class only
    - Max: 1 (2 classes), 1.584(3 classes), 2 (4 classes), 3 (8 classes), etc.
  - Interpretation: average # of bits required to encode the information of the class of each item

DECISION TREE



CLASSIFICATION: EVALUATION

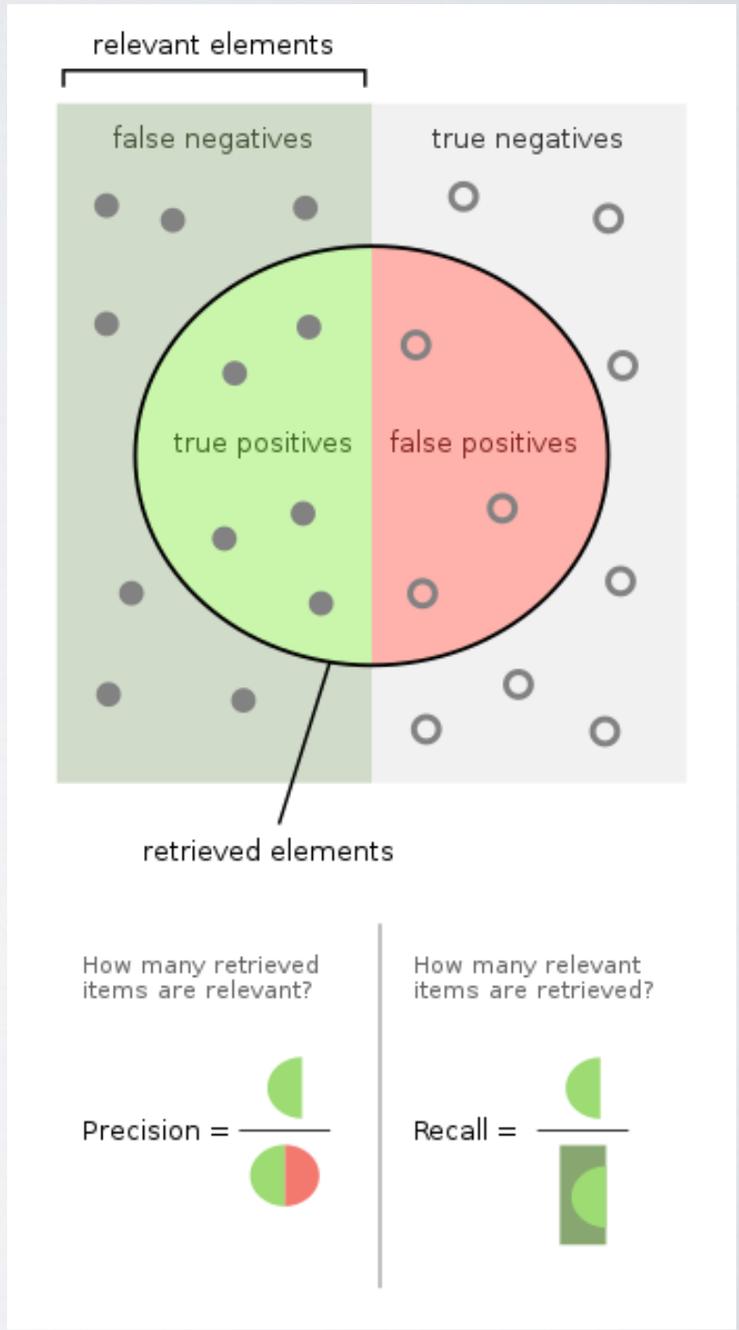
		Actual	
		Positive	Negative
Predicted	Positive	**True Positive**	**False Positive**
	Negative	**False Negative**	**True Negative**



CLASSIFICATION: EVALUATION

- Precision= $\frac{TP}{TP+FP}$
  - Among those predicted as True, fraction of really True

- Recall= $\frac{TP}{TP+FN}$
  - Among those really true, what fraction did we identity correctly



ACCURACY

- Accuracy: $\frac{TP + TN}{P + N}$

- Fraction of correct prediction, among all predictions
  - Simple to interpret

- Main drawback: class imbalance
  - Test whole city, 1 000 people, for Covid
    - 95% don't have covid, i.e., 50 people have covid, 950 don't have it
  - Our test (ML algorithm) is pretty good: TP: 45 - FN: 5 - TN: 900 -FP: 50
    - Accuracy= (45+900)/1 000=0.945
  - Dumb classifier: Always answer: not covid
    - Accuracy: (0+950)/1 000 = 0.95

F1 SCORE

- F1 score: $F_1 = 2\frac{precision * recall}{precision + recall}$
  - Harmonic mean between precision and recall
    - Harmonic mean more adapted for rates.
    - Gives more importance to the lower value

- Scores for the covid predictor:
  - Precision=45/95=0.47
  - Recall = 45/50=0.9
  - F1=0.65

- Score for the naive predictor impossible to compute…
  - You need at least some TP !
  - Assuming 1 "free" TP (Precision=1, Recall=1/50)
    - => F1=0.04

AUC

- Method adapted for strong class imbalance
  - Typical example: recommendation. Will user X buy product Z?
  - We are not really interested in having a correct classification, but of ranking correctly items
    - Top k strategy: Accuracy among the top-k items with higher score/probability?
- Will see in link prediction class