

# 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
- To begin with: only “classic” methods
- DNNs 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:
  - MSE, RMSE: (Root) Mean Square Error
  - MAE: Mean Absolute Error
  - $R^2$

# MEAN SQUARED ERROR

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

- Similarity with the **Variance**
- Using *squared* errors give stronger importance to large errors
  - Strength and weakness (outliers)
- $\text{RMSE} = \sqrt{\text{MSE}}$ , can be easier to interpret
- Simple to interpret
  - lower the value, lower the error, better the prediction

# MEAN ABSOLUTE ERROR

$$\bullet \text{ MAE} = \frac{1}{n} \sum_{i=1}^n |y_i - \hat{y}_i| = \frac{1}{n} \sum_{i=1}^n |e_i|$$

- Similarity with the MAD (Mean Absolute Deviation), comparing values with predictions instead of simple mean.
- Intuitive interpretation (not a root of a square)
- *Is being off by 10 more than twice as bad as being off by 5?*

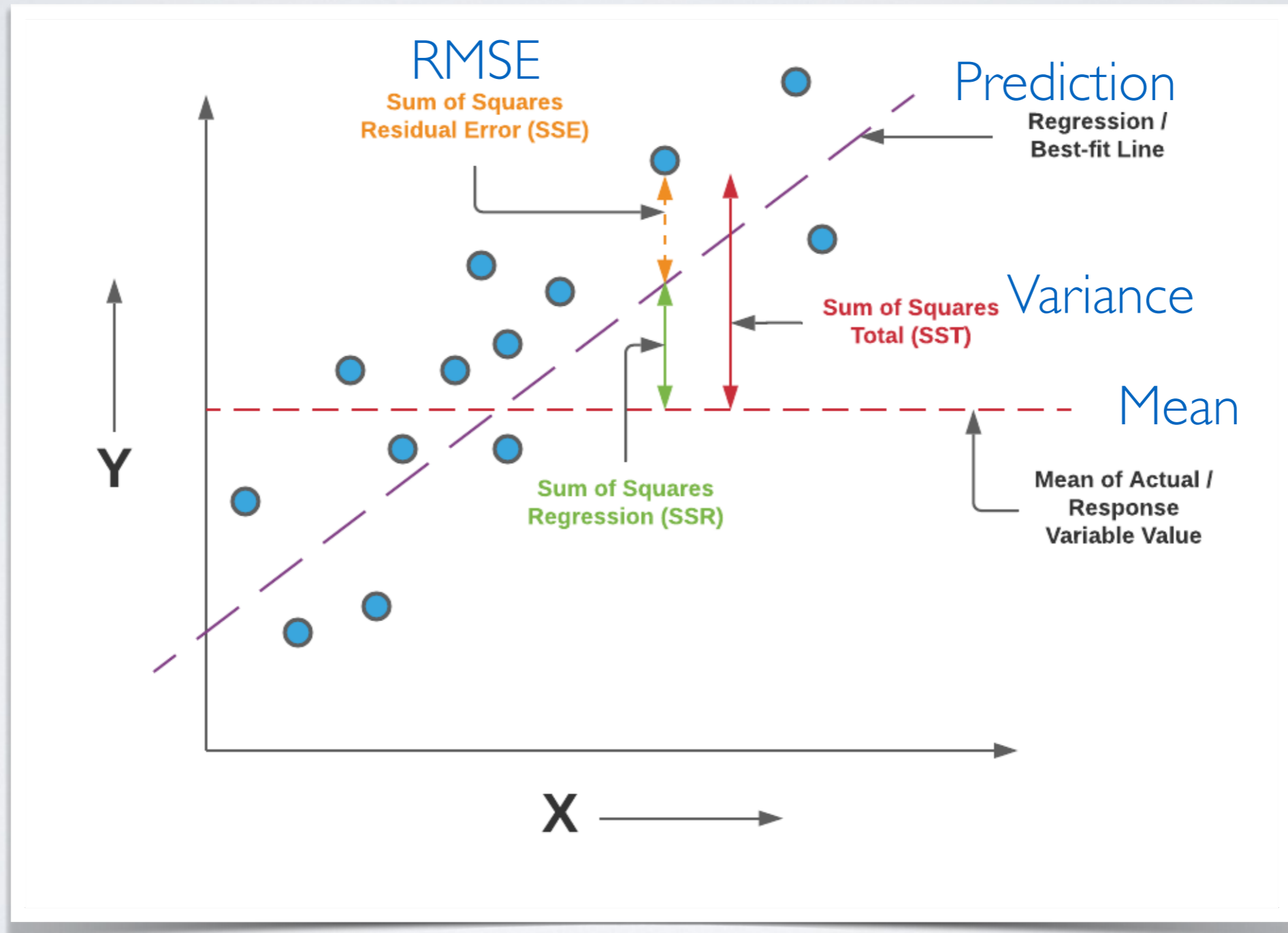


# $R^2$ (R-SQUARED)

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

- Quantifies the fraction of the variance that is explained by the prediction
  - Sometimes called *coefficient of determination* for linear regression
- $|R^2| = 1 \Rightarrow$  Perfect prediction.
  - Negative if the prediction is worse than taking the average (=Variance)

# $R^2$ (R-SQUARED)

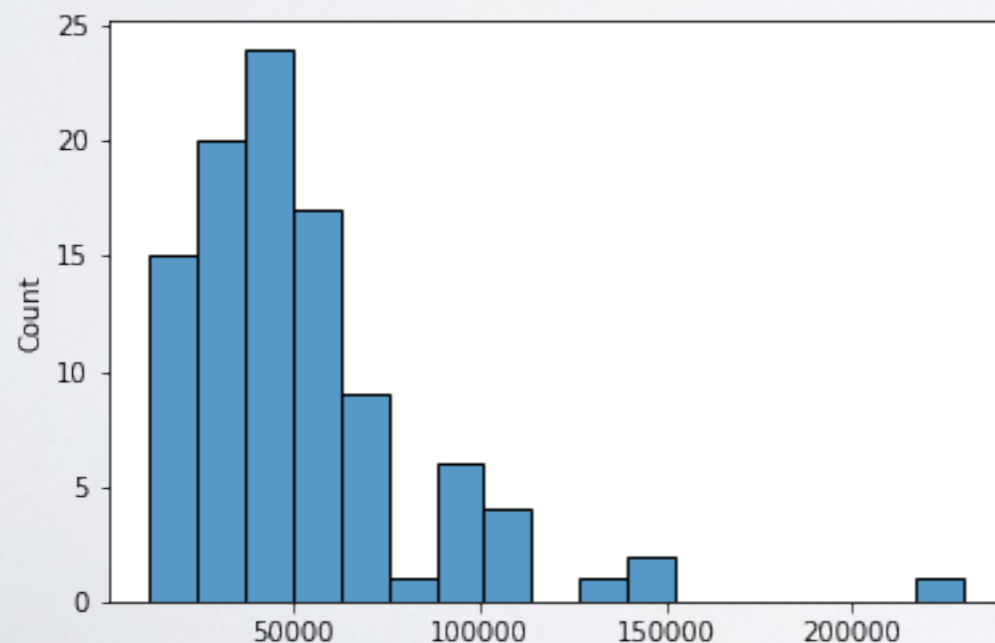


# EVALUATION/OBJECTIVE

- 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".

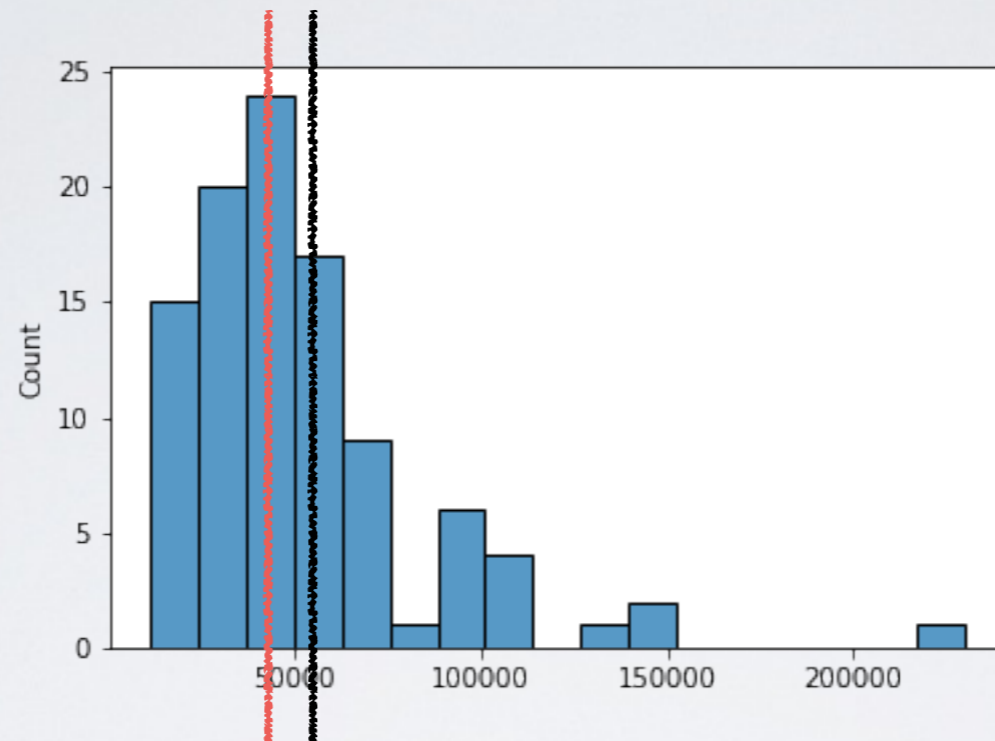
# BASELINE

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



(Some imaginary values)

# BASELINE



Using **Mean**=51676

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

RMSE lower

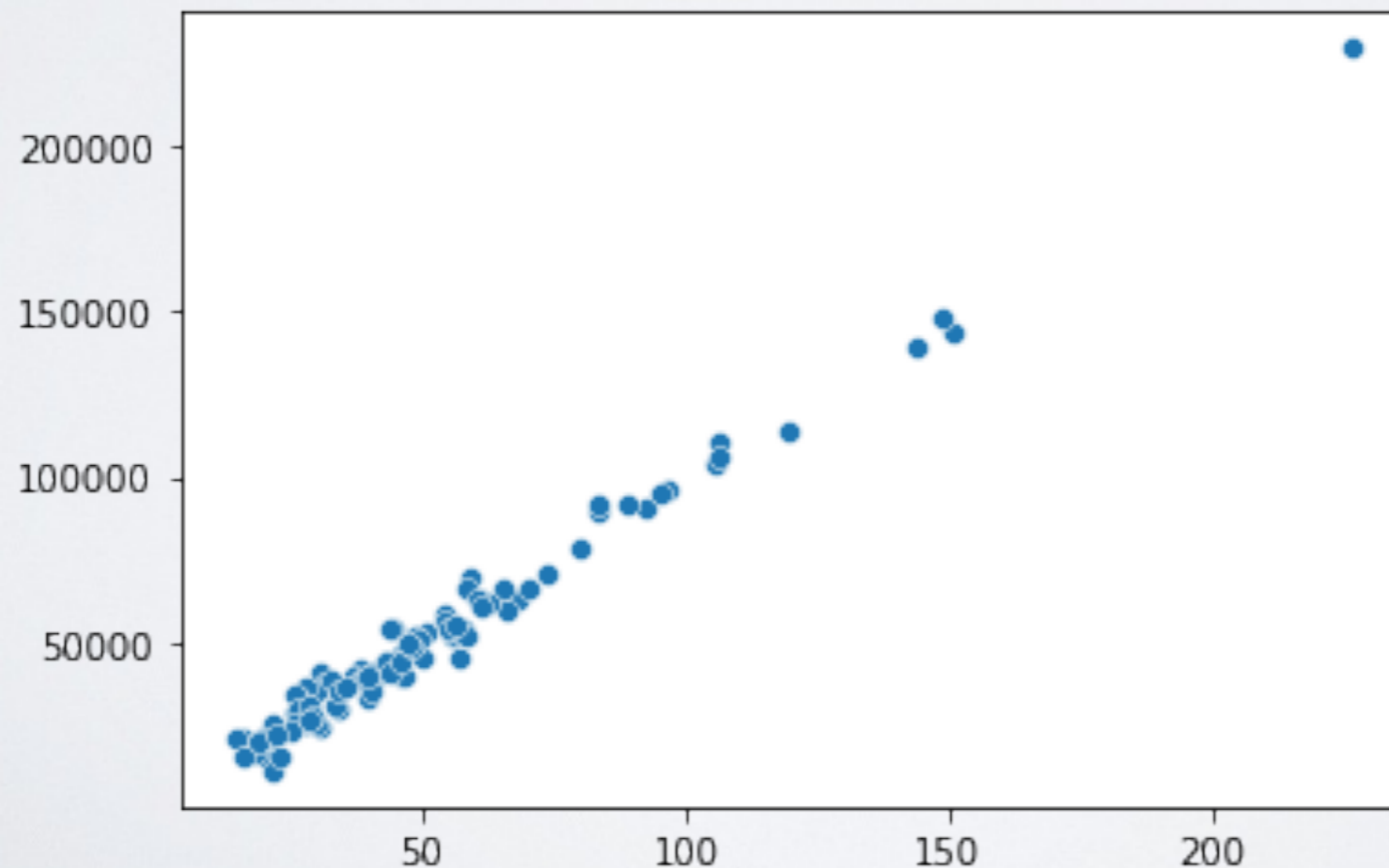
Using **Median**=43086

```
MSE 1179133659.4166086  
RMSE 34338.51568452848  
MAE 21658.66828240126  
R2 -0.06675615376207489
```

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

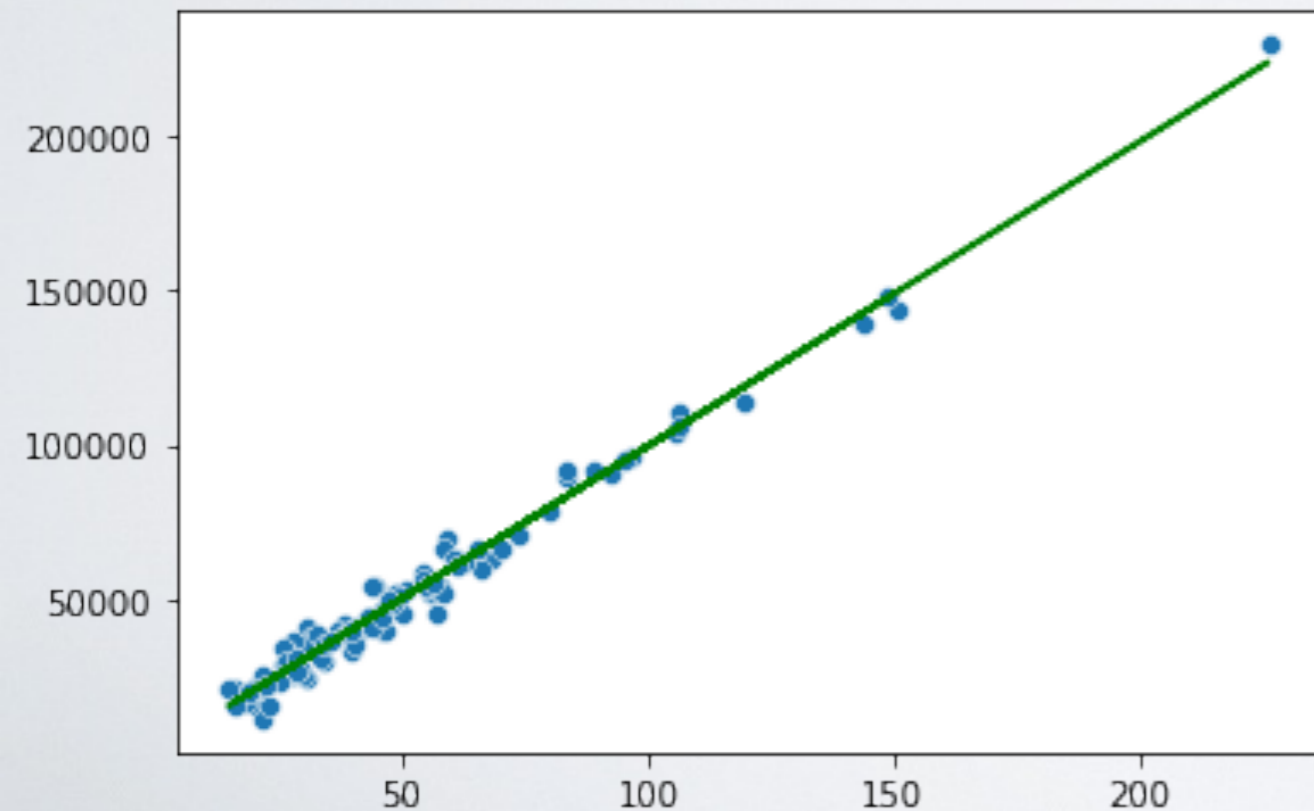


# LINEAR REGRESSION

- 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
  - $i \Rightarrow$   $i$ th 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:
  - $\beta_0=987$
  - $\beta_1=779$



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



# LINEAR REGRESSION

- We solve this problem, and obtain:
  - $\beta_0=987$
  - $\beta_1=779$

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 -0.06675615376207489
```

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
  - $\beta_0 = 987 \ 0$
  - $\beta_1 = 779 \ 1000$

Using  
Linear Regression

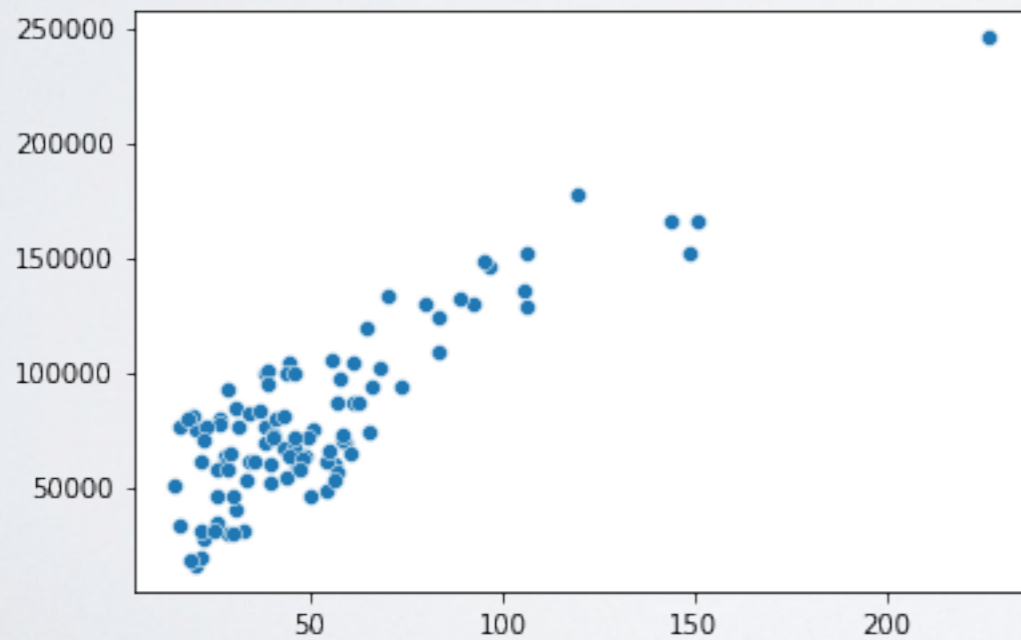
```
MSE 20668278.463901177  
RMSE 4546.237836266508  
MAE 3512.3861644882704  
R2 0.9813015148342528
```

Using  
“Real” generative model

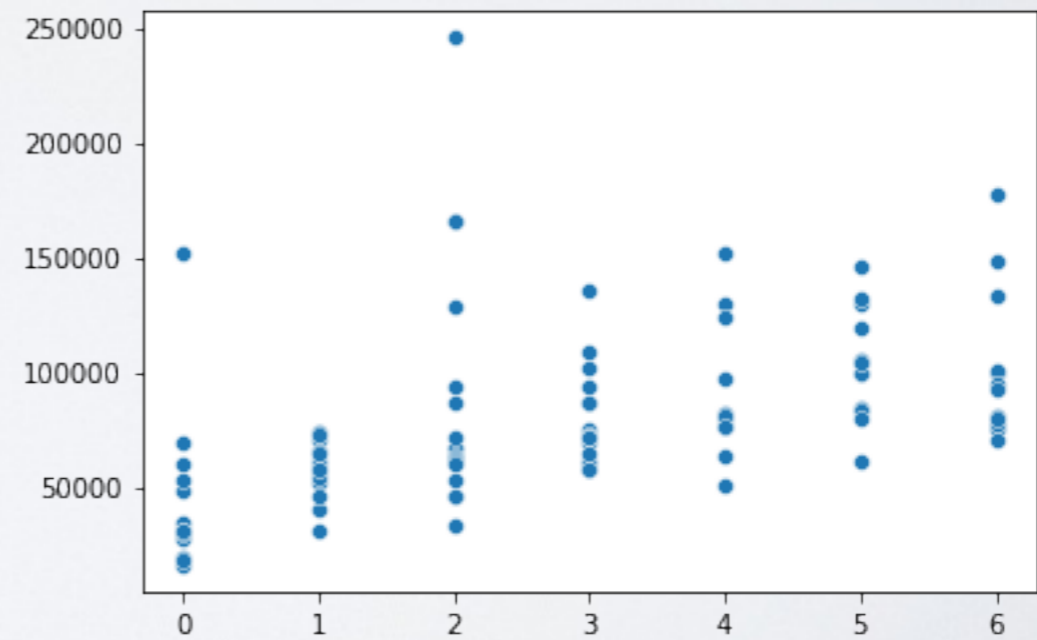
```
MSE 20863741.73315057  
RMSE 4567.68450455486  
MAE 3506.783422078361  
R2 0.9811246802204318
```

# 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
  - $y = \beta_0 + \beta_1x_1 + \beta_2x_2 + \dots + \beta_nx_n + \epsilon$
  - 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 0.34222807880552575

All features

MSE 22157971.6387145  
RMSE 4707.225471412486  
MAE 3617.346073048316  
R2 0.9847551176123155

Generative Parameters

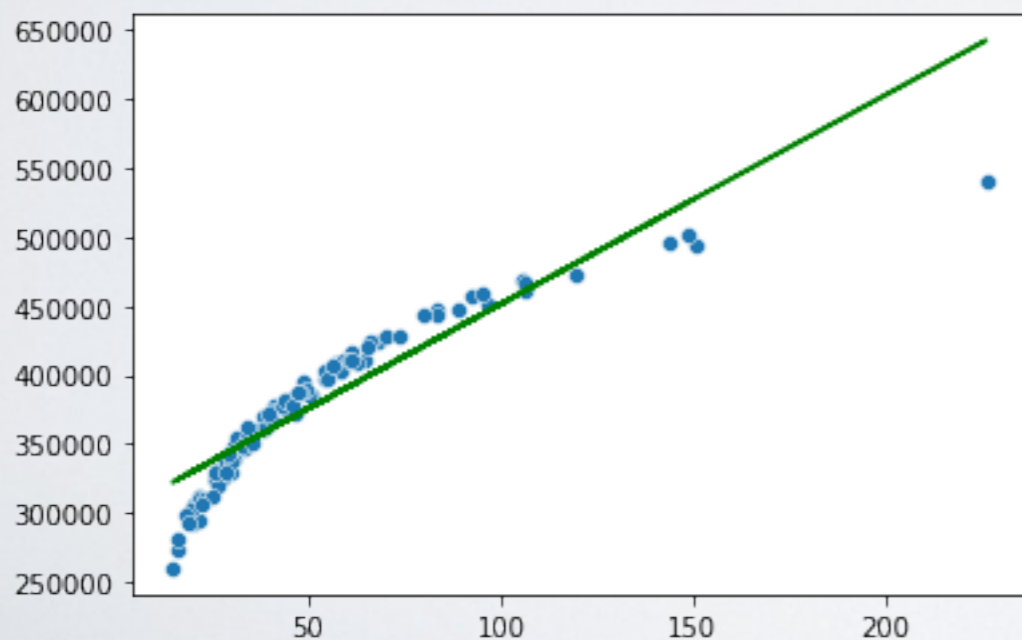
- $\beta_0 = 0$   $\beta_1 = 1\ 000$ ,  $\beta_2 = 10\ 000$

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  $\text{Price} \approx \log(\text{surface}) * 100\,000$  ?



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 0.9922842323758786

# 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 between variable and target 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  $\beta$  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 ( $\epsilon$ ))
    - No Homoscedasticity (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  $\Rightarrow$  Gradient descent, later during the class

# OLS STRENGTH

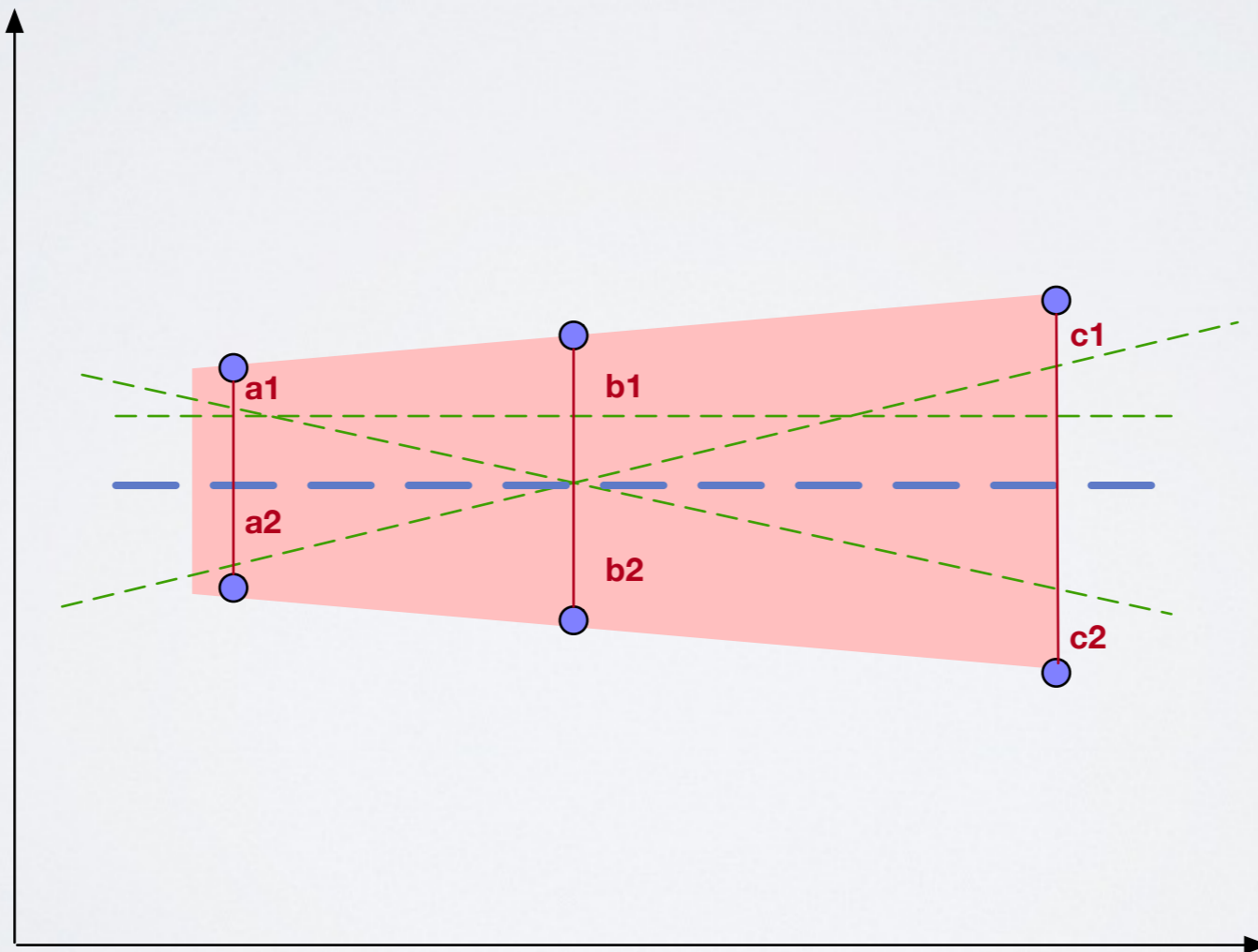
- Another strength of OLS is that it is equivalent to MLE
- MLE (Maximum Likelihood Estimation) is a model-based approach
  - If we make the assumption that our data has been generated by a random model of the form:  $y = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \dots + \beta_n x_n + \epsilon$ 
    - And that the errors are normally distributed and centered in zero
  - 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 OPTIMALITY

- If some conditions are respected, OLS is **optimal** to solve the problem. No other method can outperform it.
  - The problem is that these conditions are not realistic for many real problems
- OLS conditions for optimality
  - Relations between features and targets are really linear
  - All necessary features are present, and all present features are necessary
  - etc.

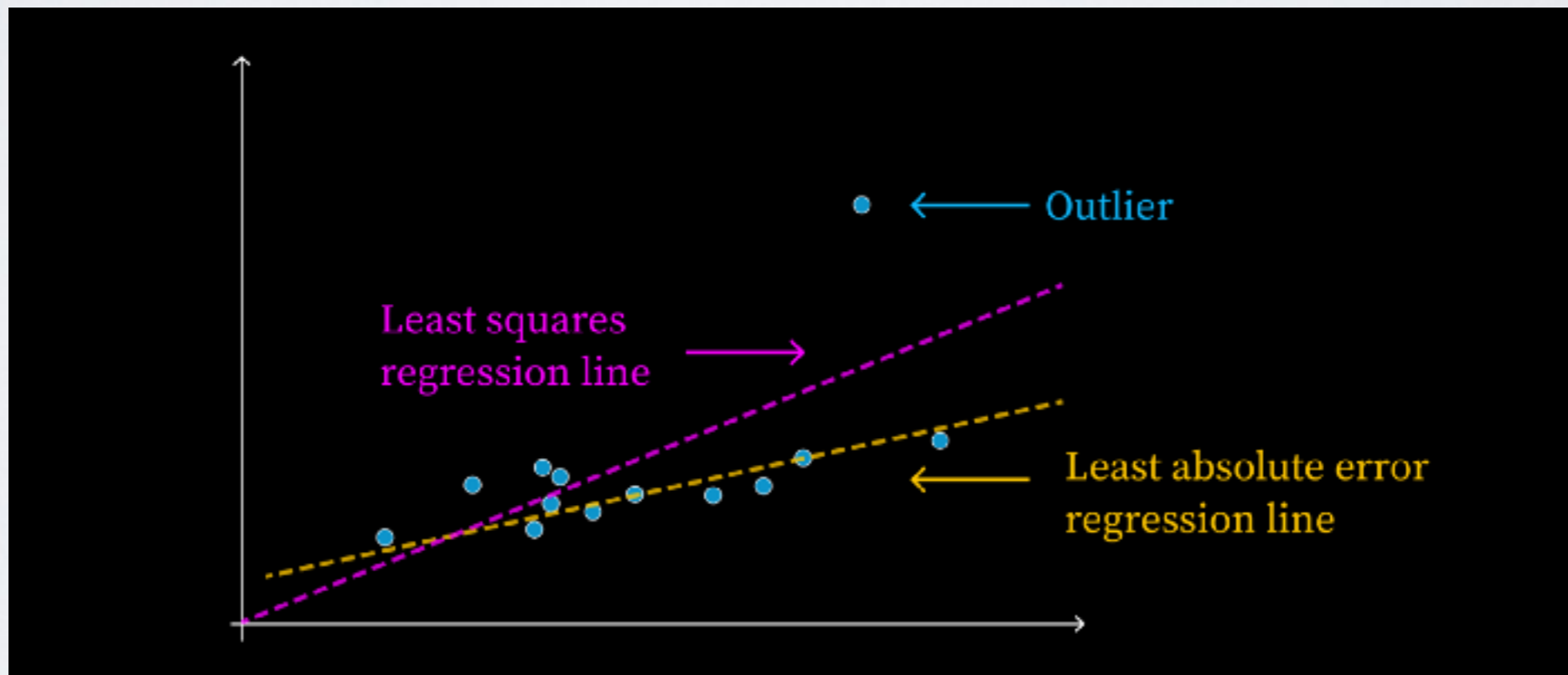
# OLS STRENGTH

- Intuitive reason to use MSE instead of MAE
  - Single, “intuitive” solution (blue)
  - MSE: all the green line fits equally ( $a_1 + a_2, b_1 + b_2, c_1 + c_2 = \text{constant}$ )



# OLS KNOWN WEAKNESS

- MSE is known to be sensitive to outliers



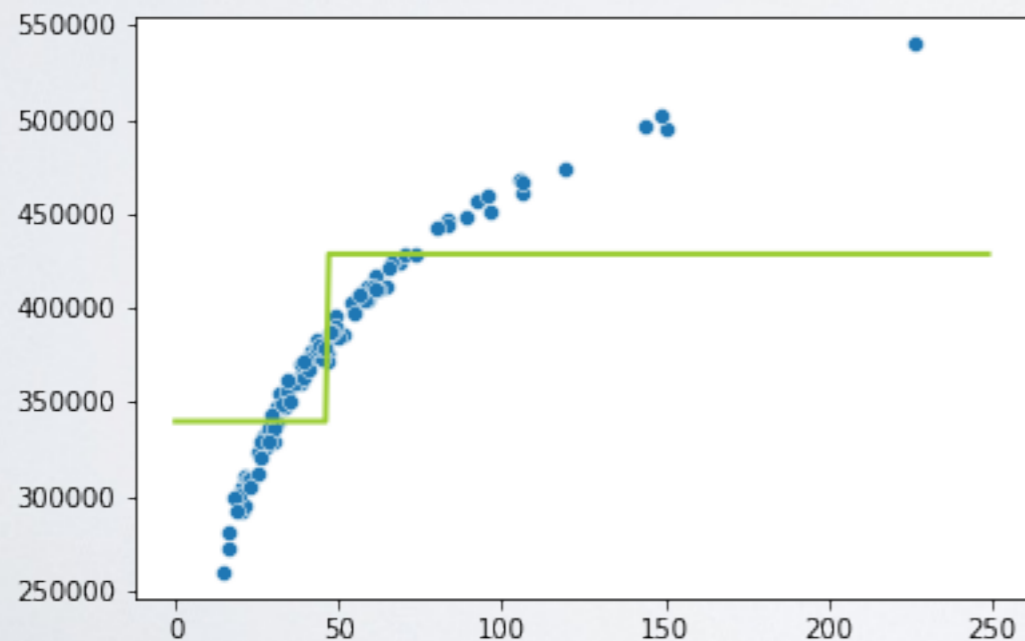
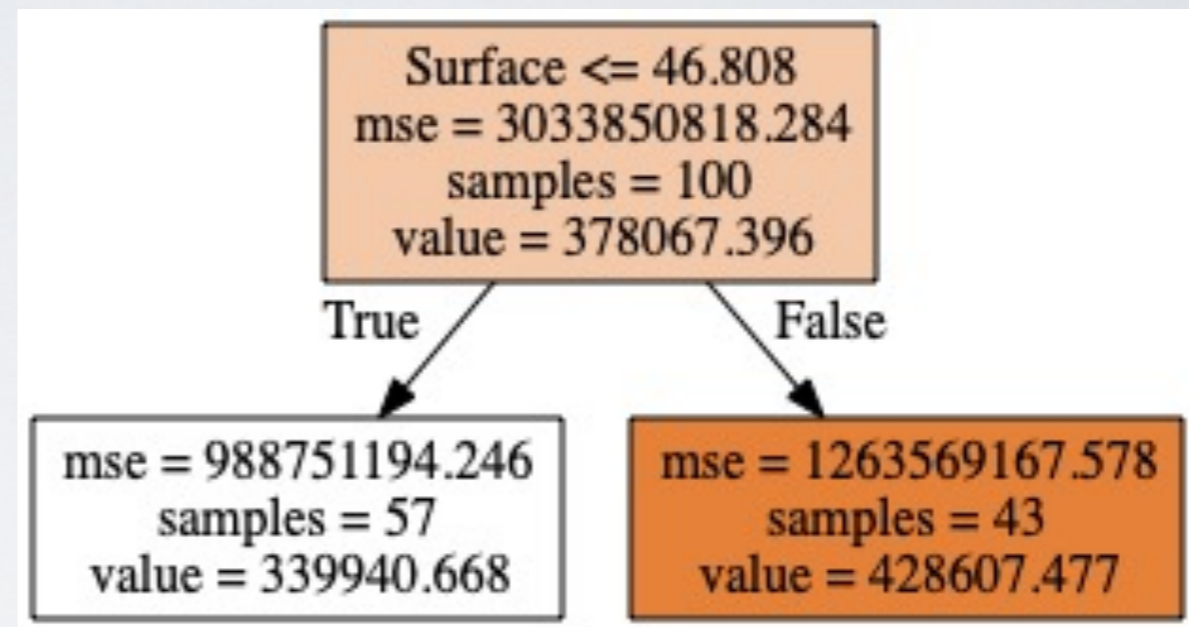
NON-LINEAR REGRESSION:  
DECISION TREE REGRESSION

# 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

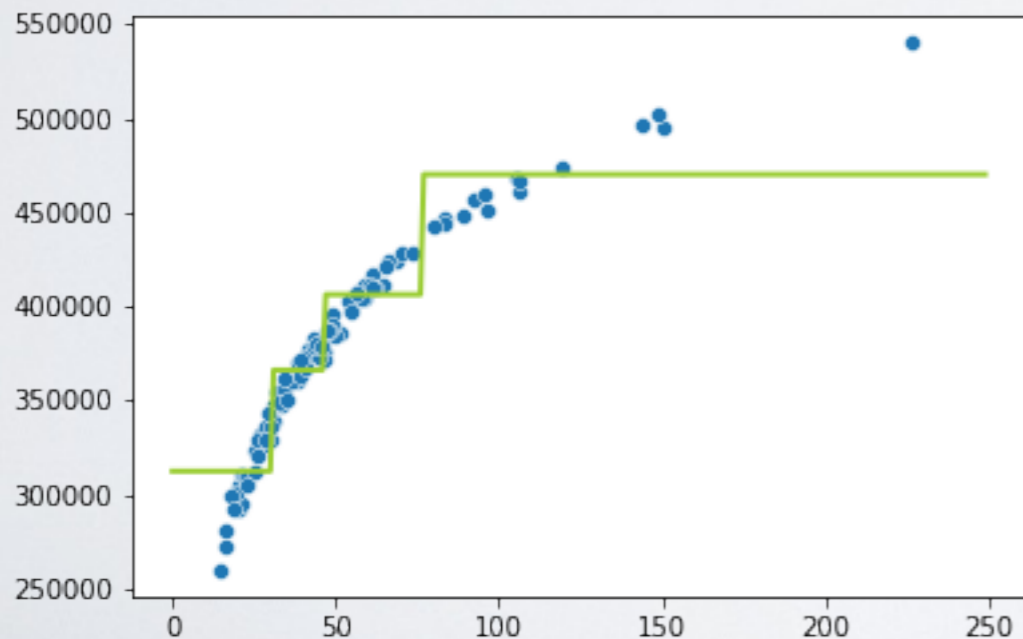
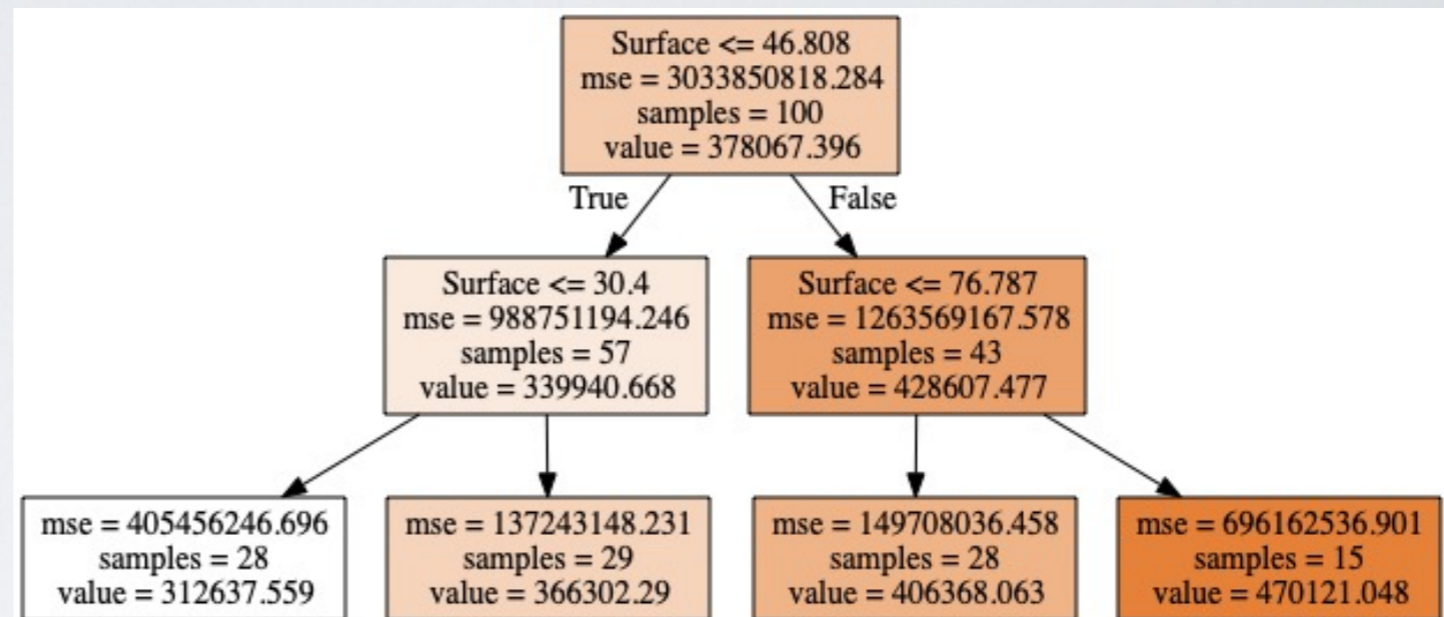


MSE 1106922922.7787206  
RMSE 33270.45119589935  
MAE 27836.40899704275  
R2 0.6351425995939648



# DECISION TREE

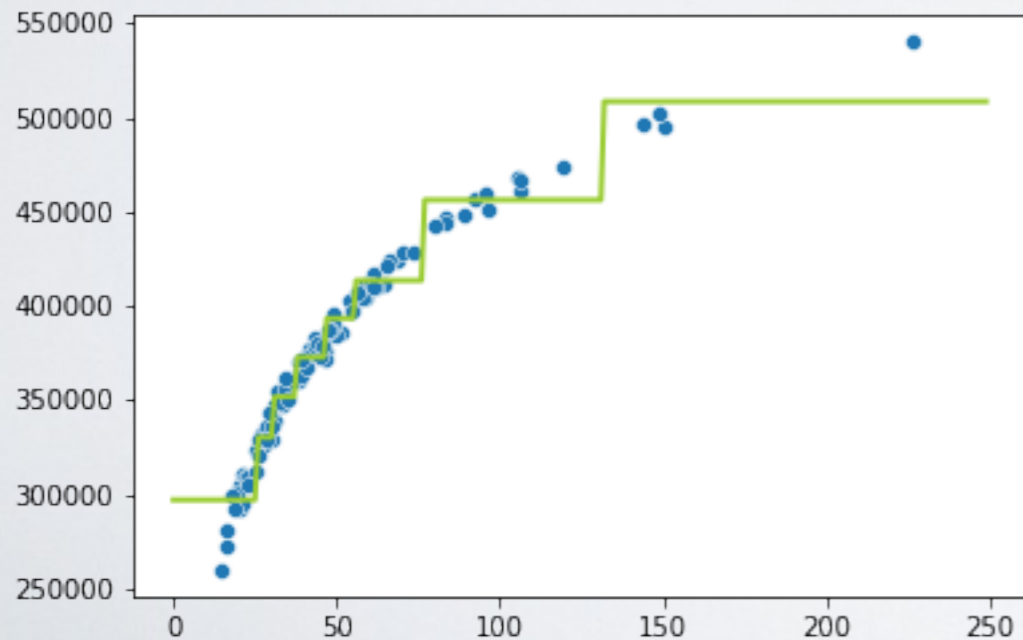
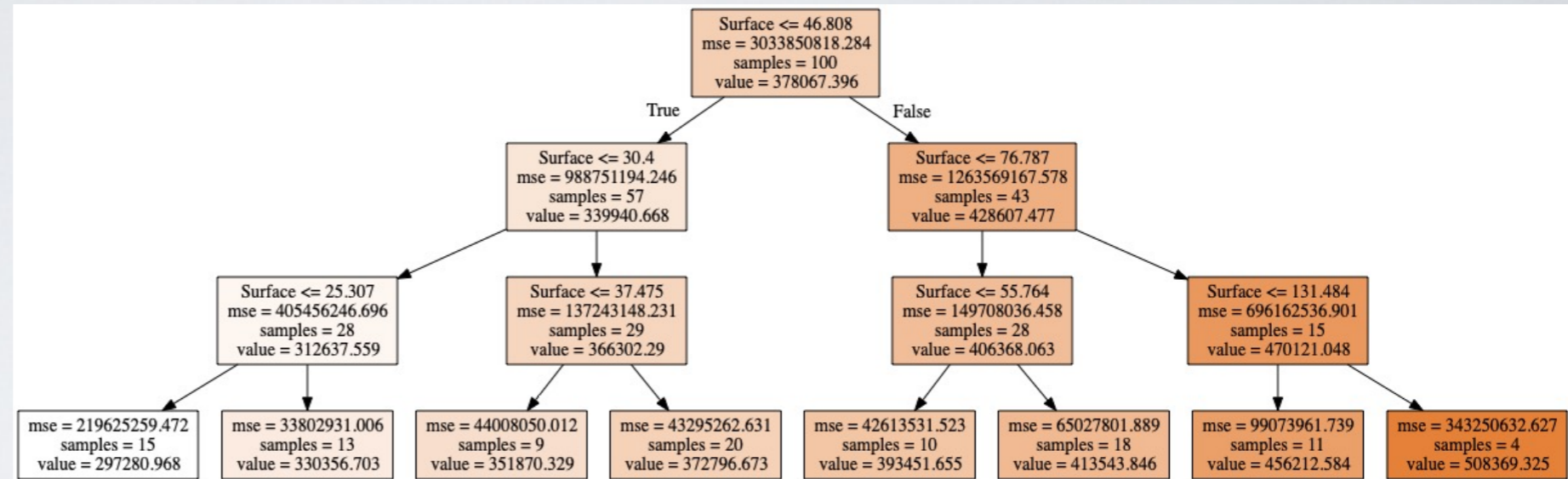
- Ex: Using
  - MSE as split criteria
  - 2 Level of splitting



MSE 299670892.805488  
RMSE 17311.00496232059  
MAE 13262.652619929546  
R2 0.9012242490634346

# DECISION TREE

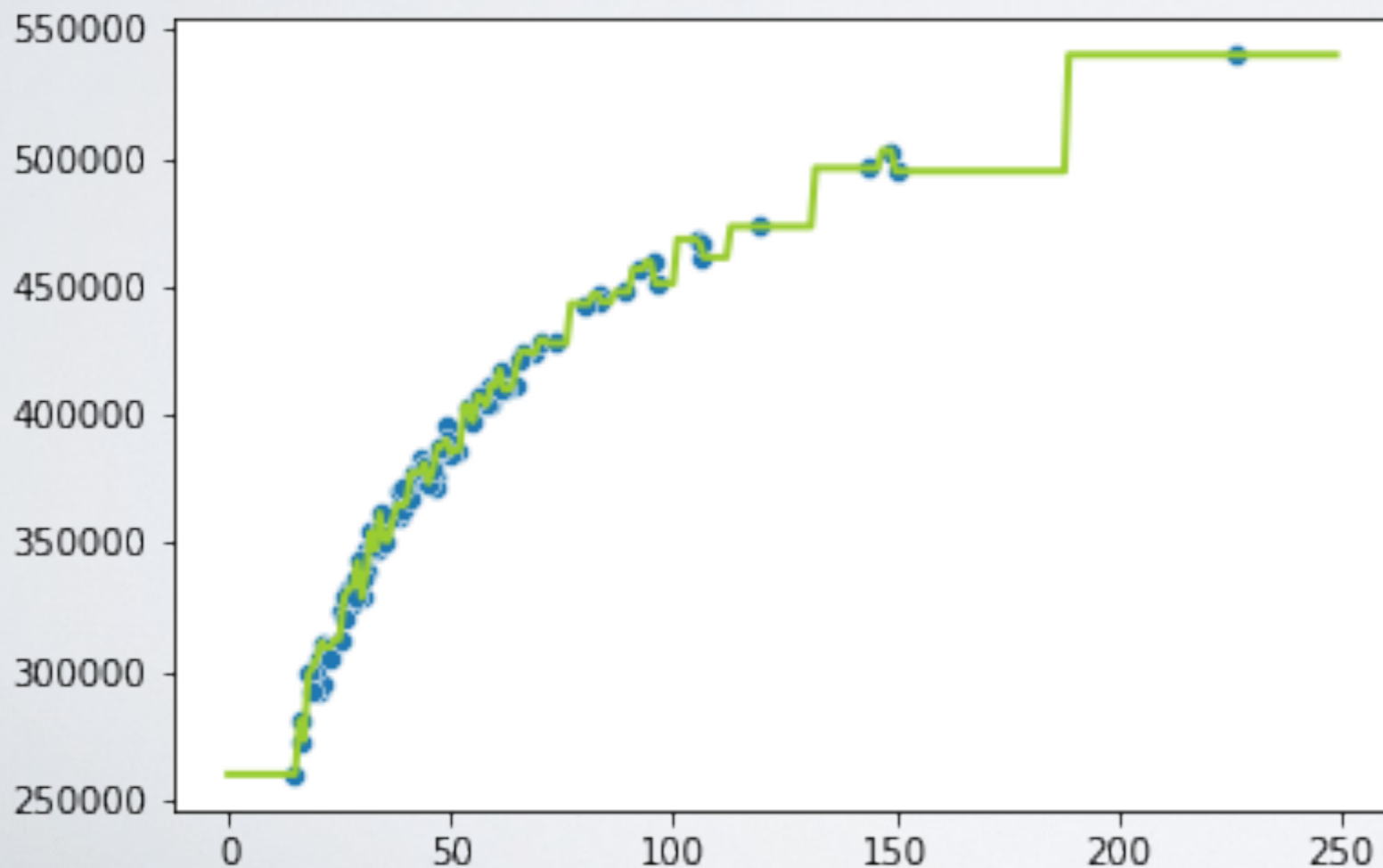
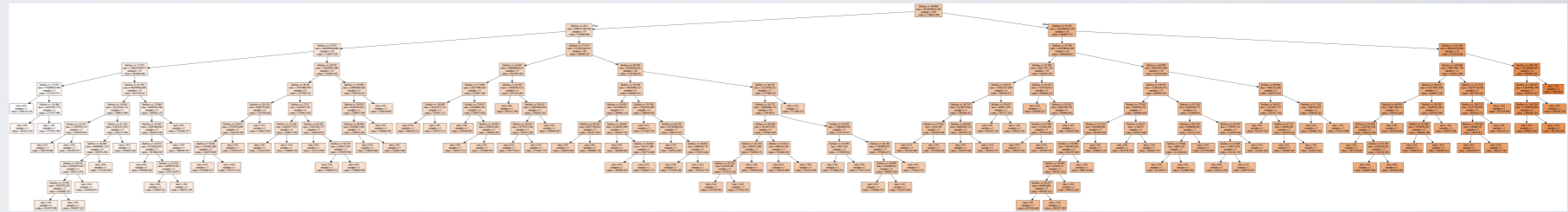
- Ex: Using
  - ▶ MSE as split criteria
  - ▶ 3 Level of splitting



MSE 90552465.56733872  
RMSE 9515.905924678886  
MAE 7434.910779663157  
R2 0.9701526307682573

# 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**

Scores on  
Train Set

MSE 0.0  
RMSE 0.0  
MAE 0.0  
R2 1.0

Decision Tree, **levels=5**

MSE 9675372.95170697  
RMSE 3110.5261535159884  
MAE 2364.5552169188454  
R2 0.9968108606746918

Scores on  
Test Set

MSE 60522590.58807978  
RMSE 7779.626635519199  
MAE 6427.594619486819  
R2 0.9689849224913336

MSE 47482936.48734139  
RMSE 6890.786347532579  
MAE 5748.307144423111  
R2 0.9756671526915104



# 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

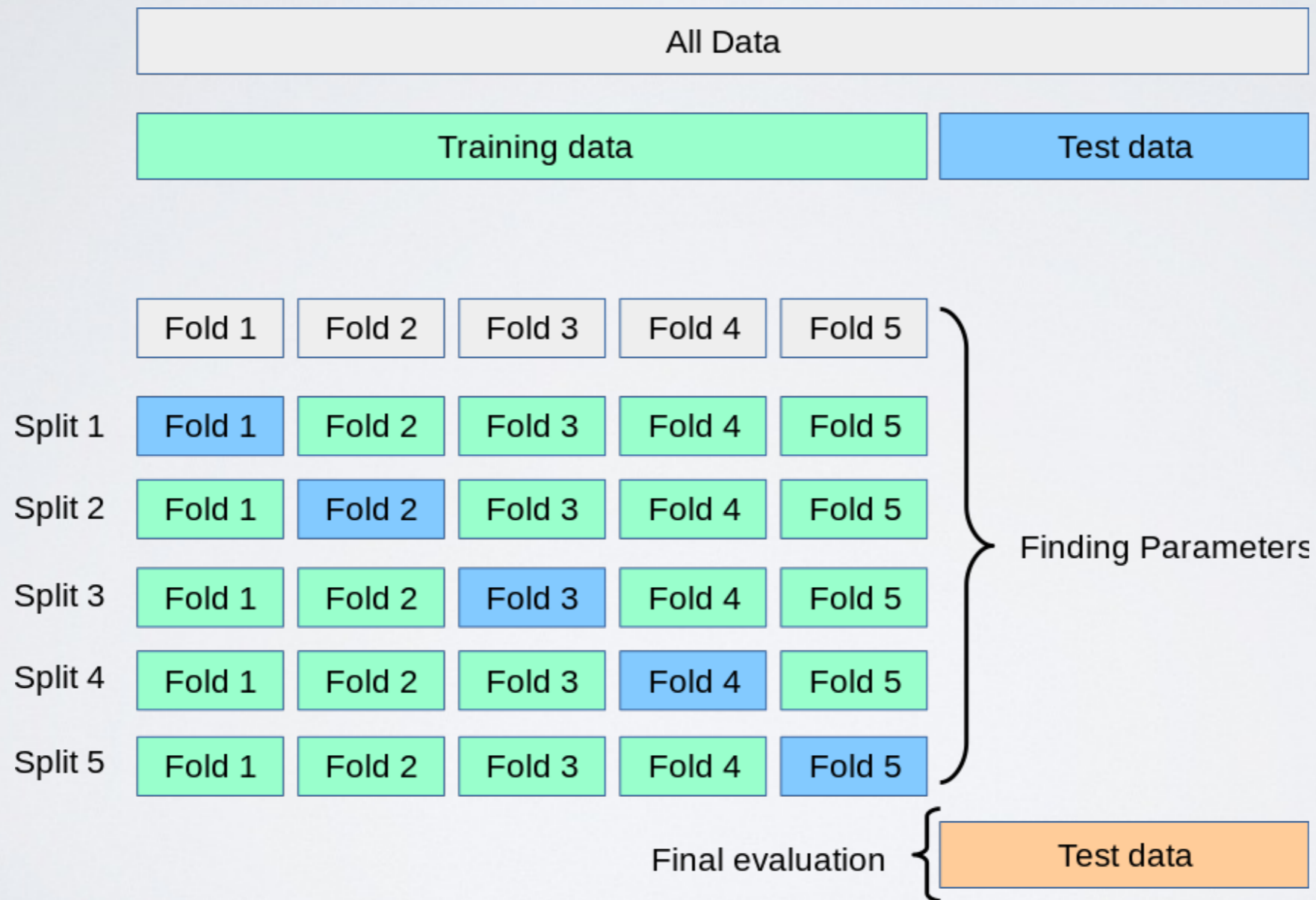
# TRAIN/TEST SPLIT

- 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

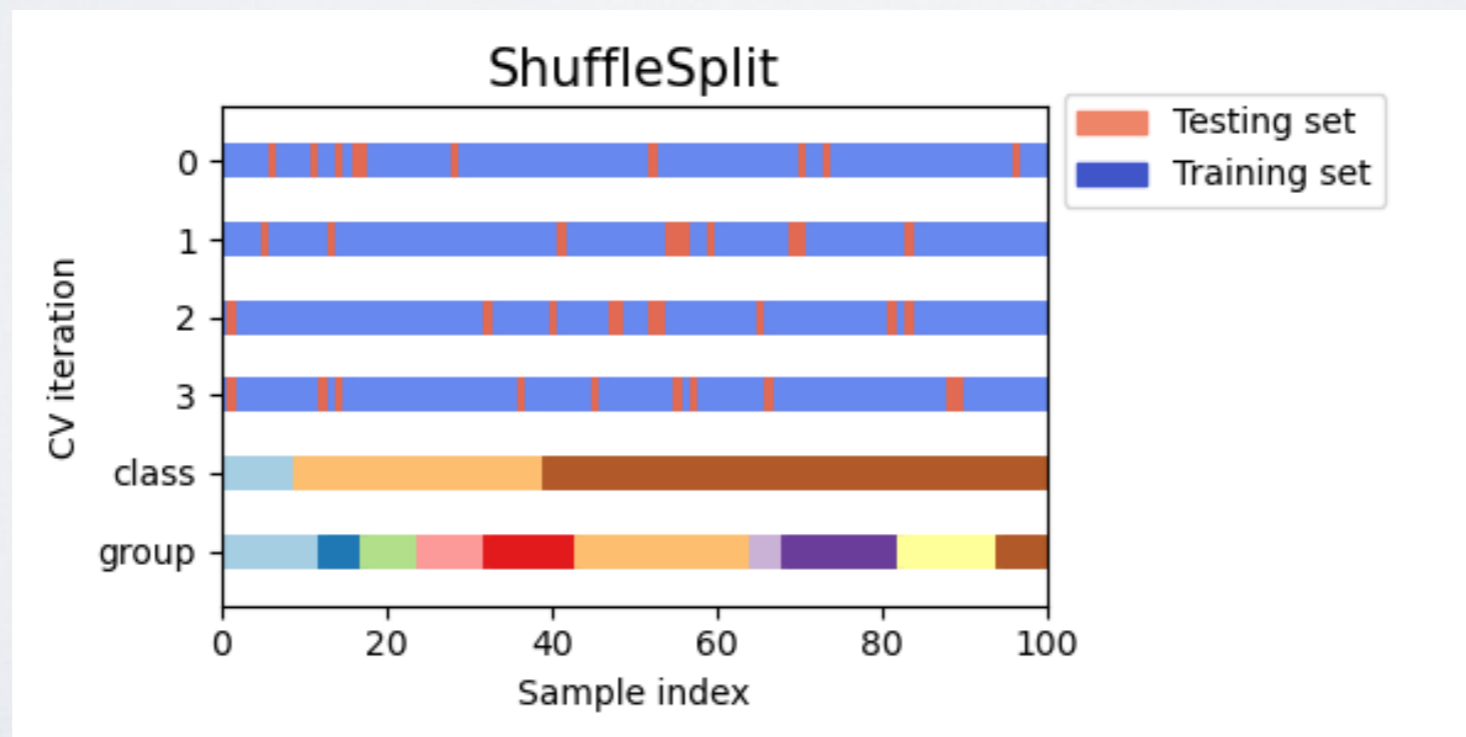
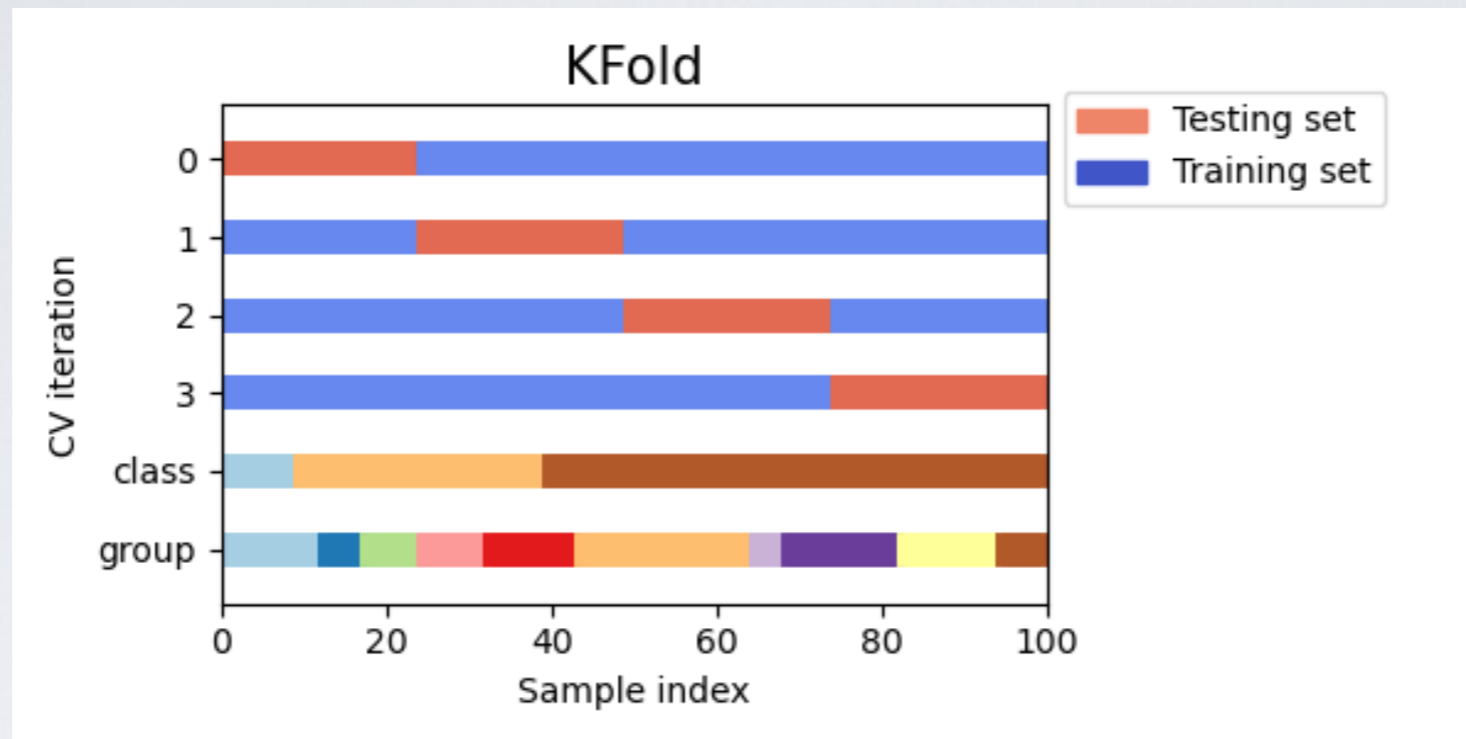
# CROSS VALIDATION

- 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

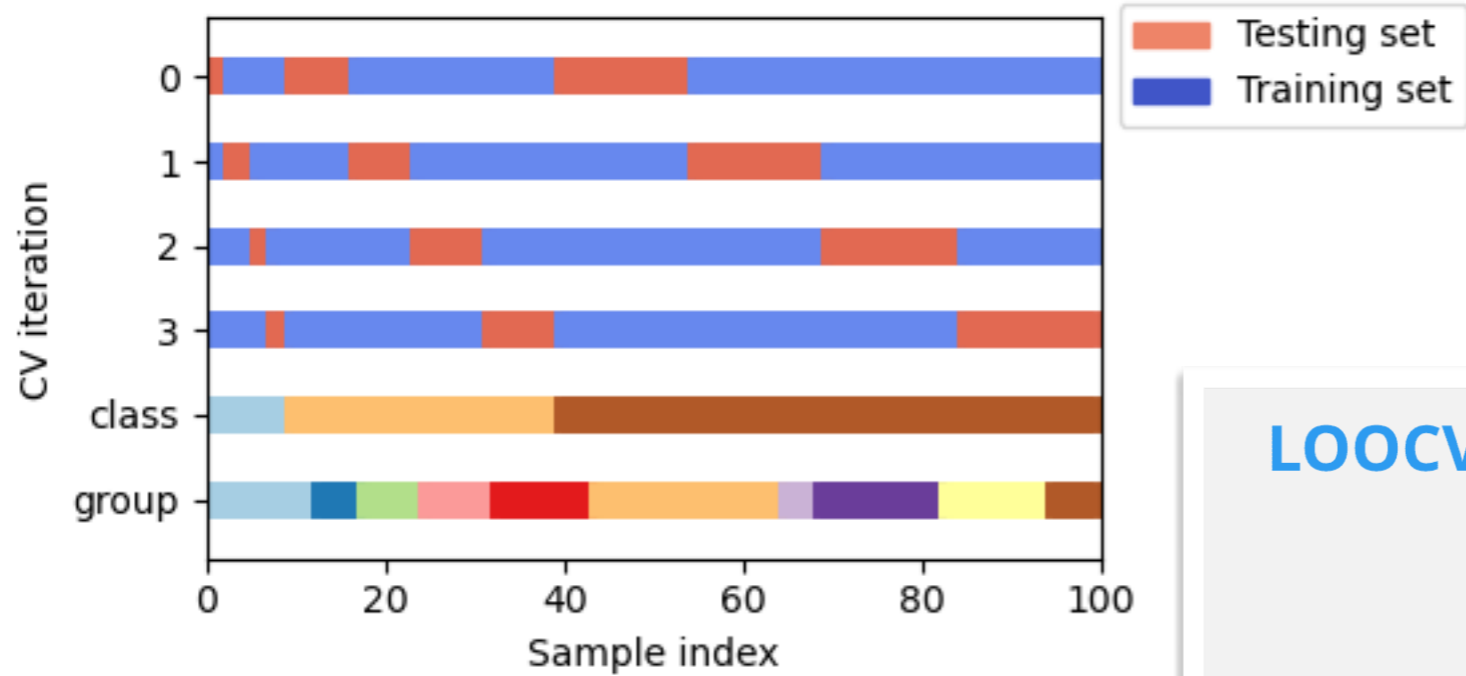


# CROSS VALIDATION

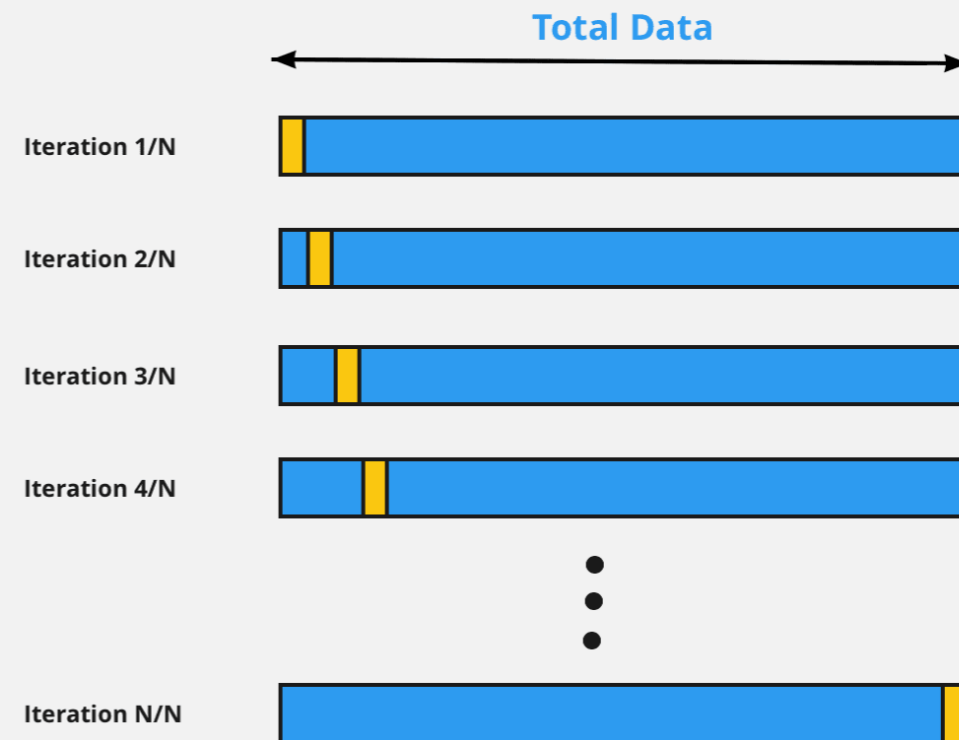


# CROSS VALIDATION

StratifiedKFold



## LOOCV: Leave One Out Cross Validation



# 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

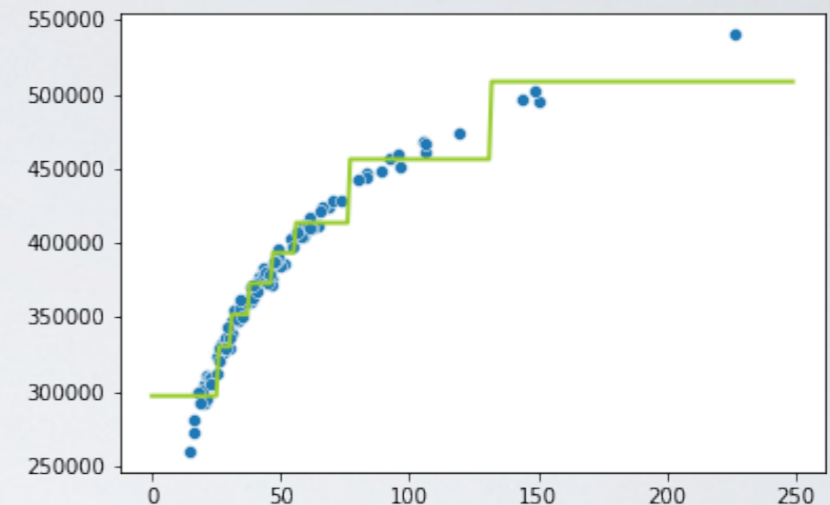
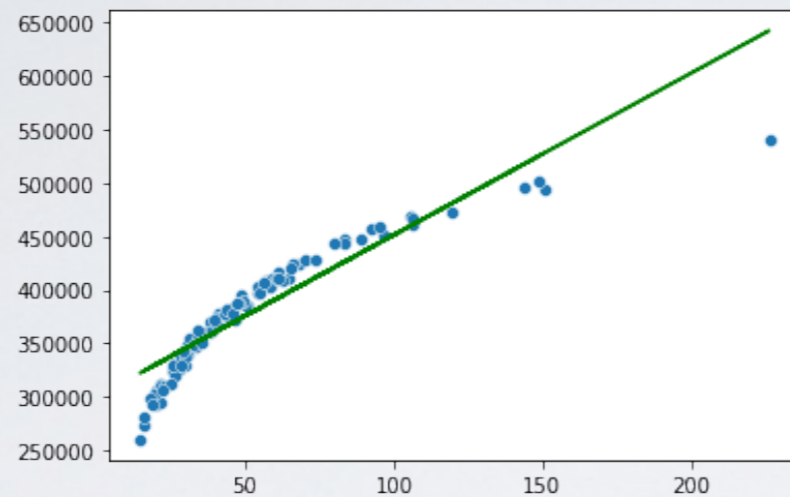
## BACK TO THE METHOD



# 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 0.8437196622358905

MSE 9675372.95170697  
RMSE 3110.5261535159884  
MAE 2364.5552169188454  
R2 0.9968108606746918

Test

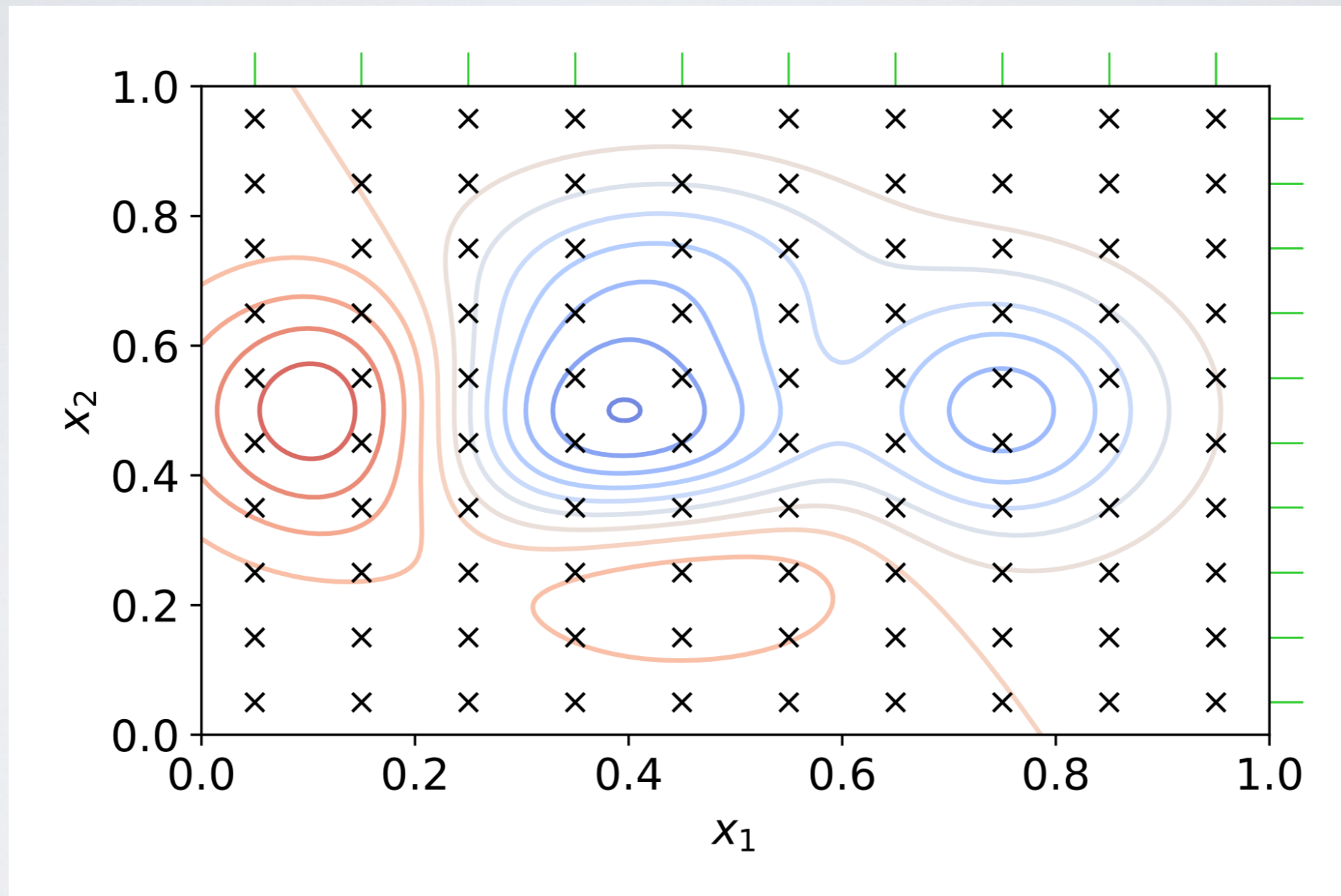
MSE 297361867.9984524  
RMSE 17244.18359907051  
MAE 14666.202886910516  
R2 0.8476155548782759

MSE 47482936.48734139  
RMSE 6890.786347532579  
MAE 5748.307144423111  
R2 0.9756671526915104

# 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

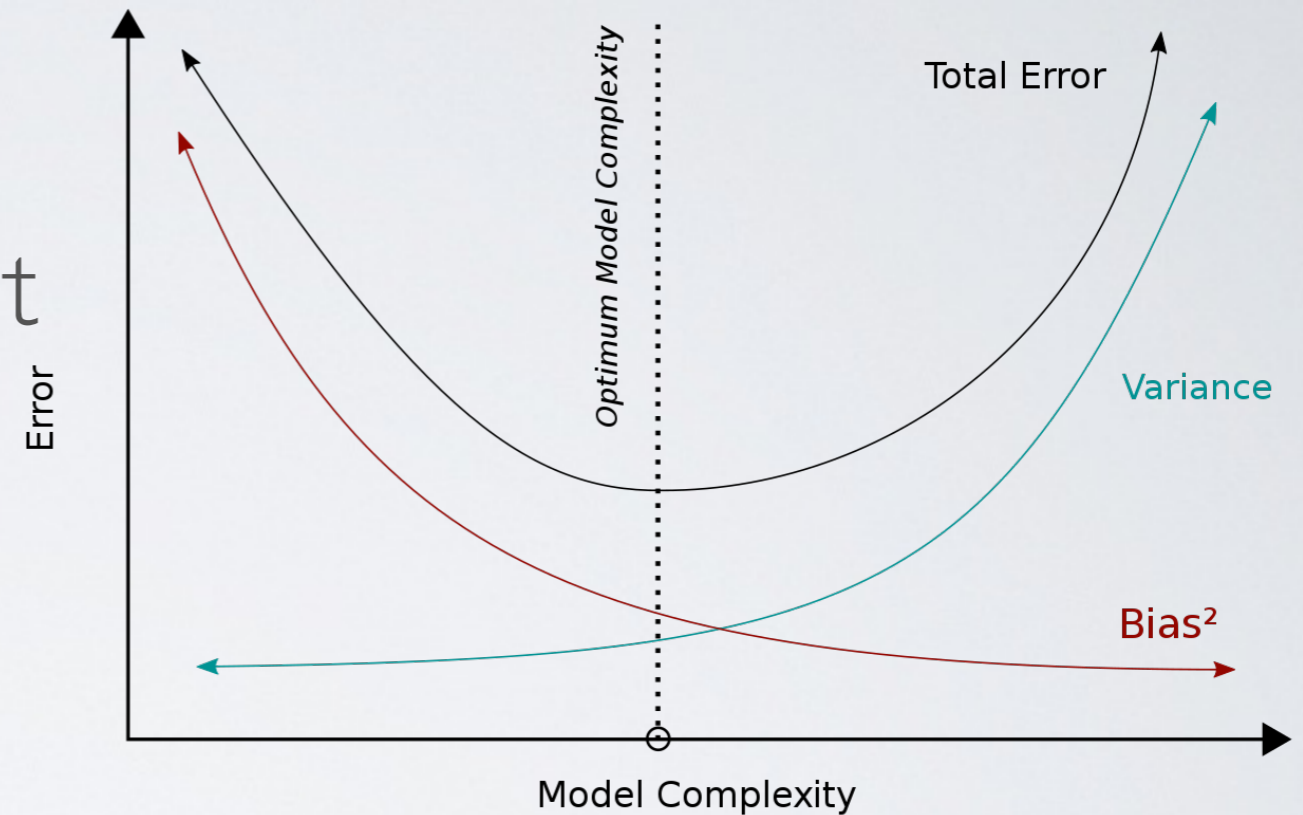
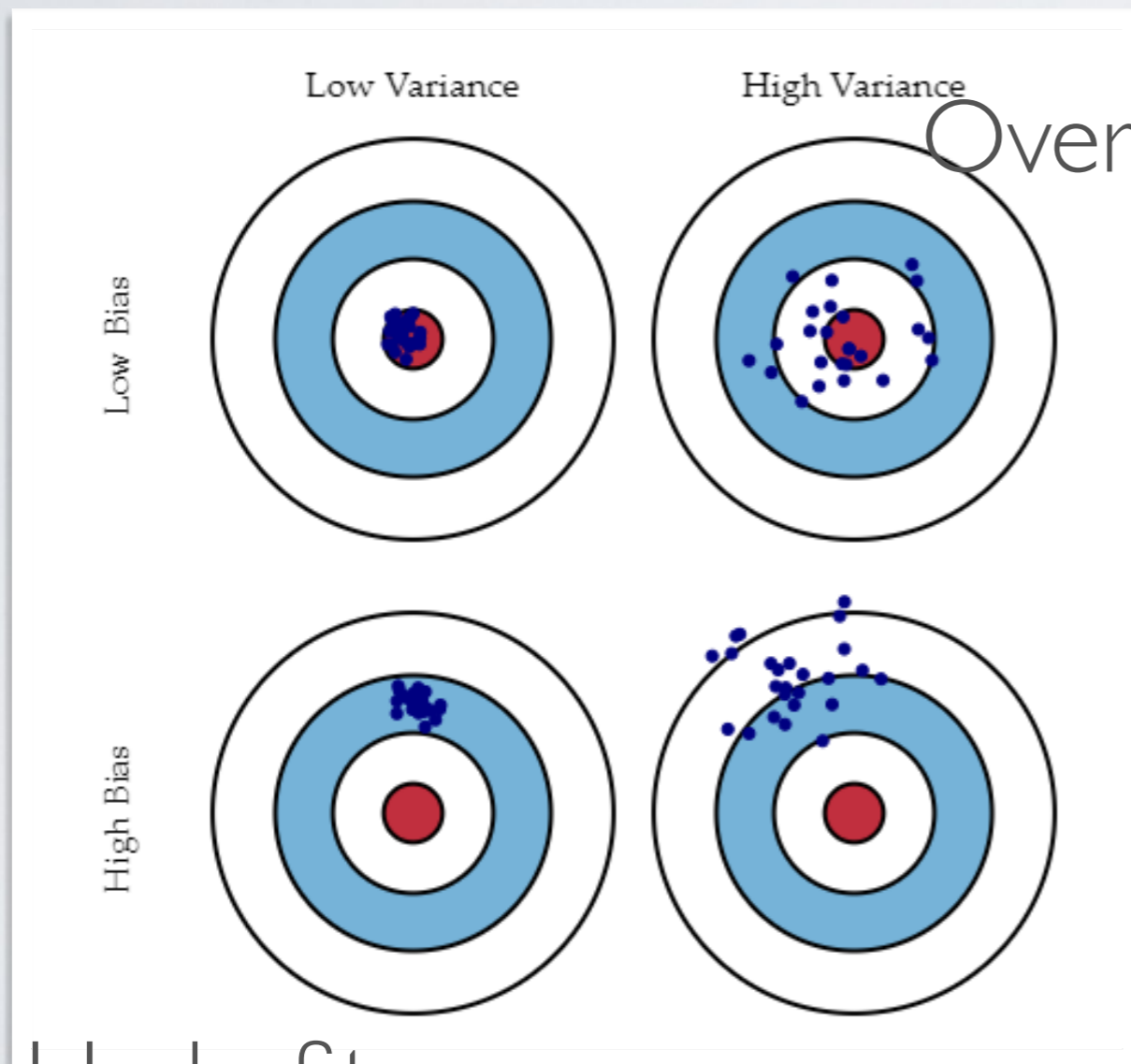


More clever methods exist: Bayesian optimization, etc.

# 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



$$\text{Bias}_D[\hat{f}(x; D)] = E_D[\hat{f}(x; D)] - f(x)$$

$$\text{Var}_D[\hat{f}(x; D)] = E_D[(E_D[\hat{f}(x; D)] - \hat{f}(x; D))^2]$$

D: subsets.

X: all elements in all subsets