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

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

- 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

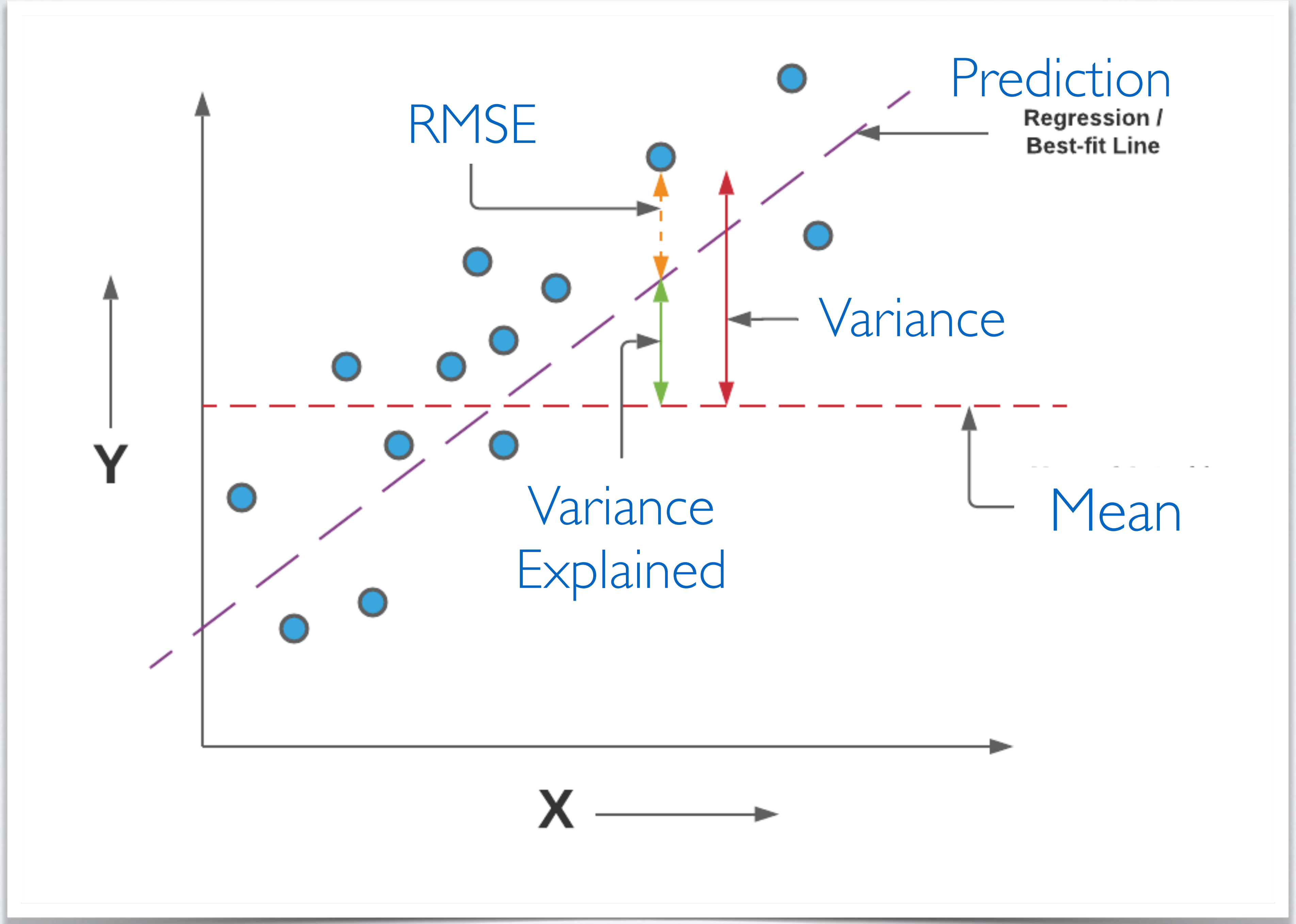
MEAN SQUARED ERROR

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



Prediction
Regression / Best-fit Line
RMSE
Variance
Variance Explained
Mean
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

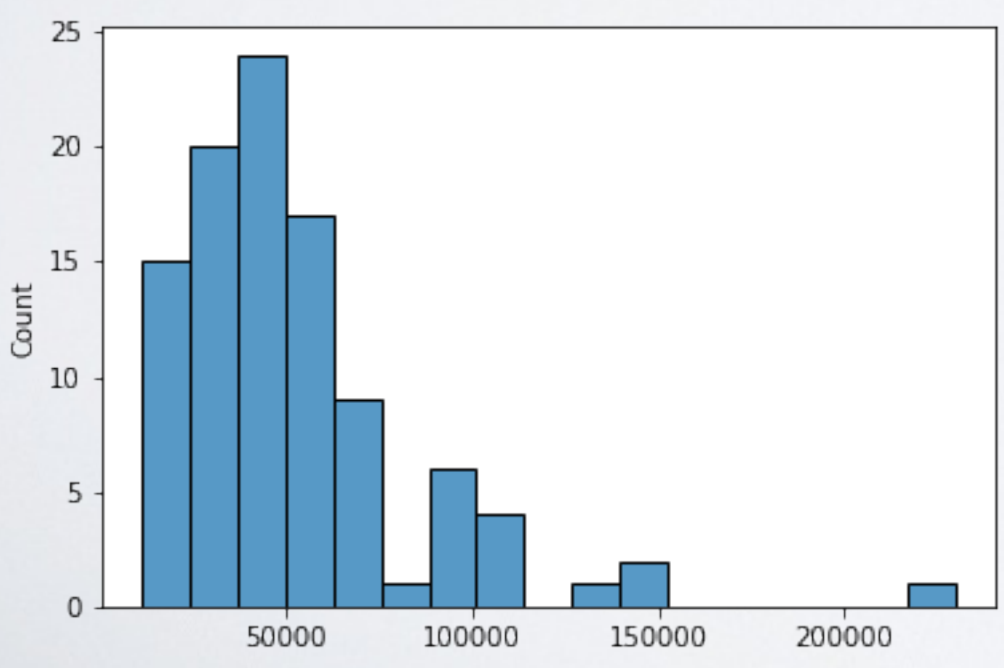
- More information can allow you to judge better. There is no "truth".

NAIVE/STATISTICAL PREDICTION

Baseline

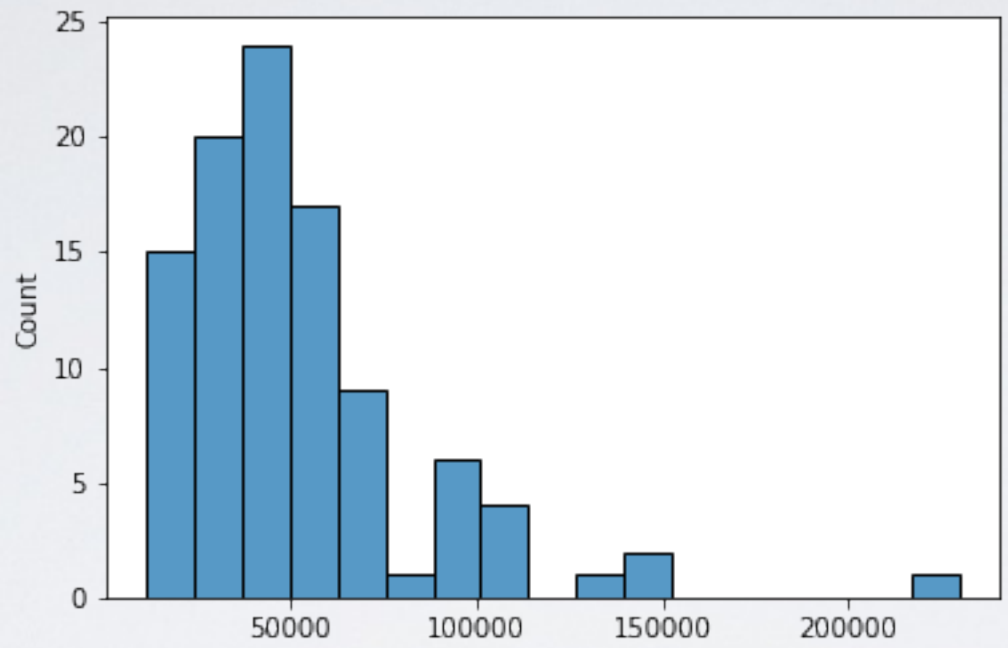
BASELINE

- 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

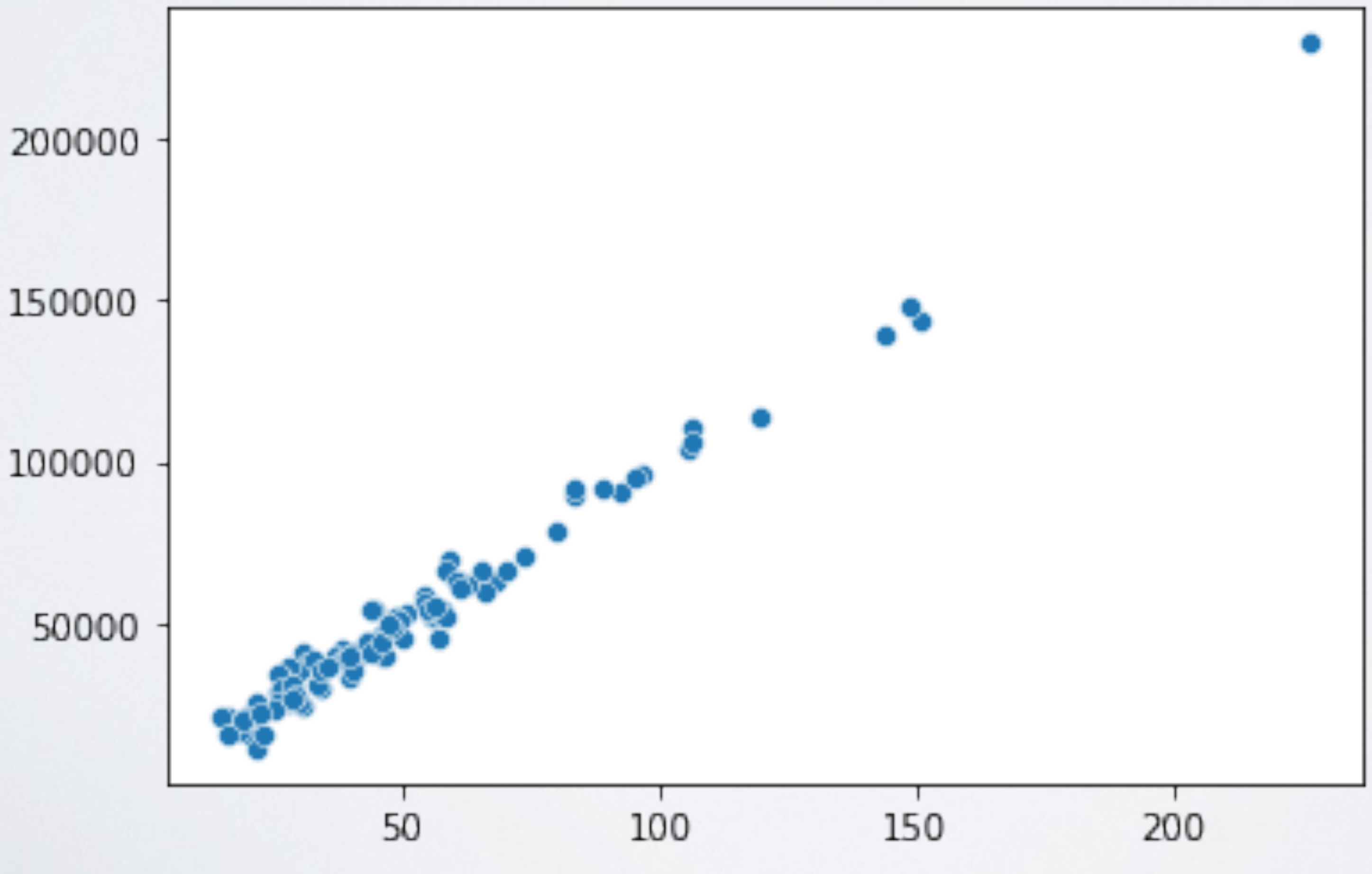
```
MSE 1179133659.4166086
RMSE 34338.51568452848
MAE 21658.66828240126
R2 -0.06675615376207489
```

MAE lower

LINEAR REGRESSION

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

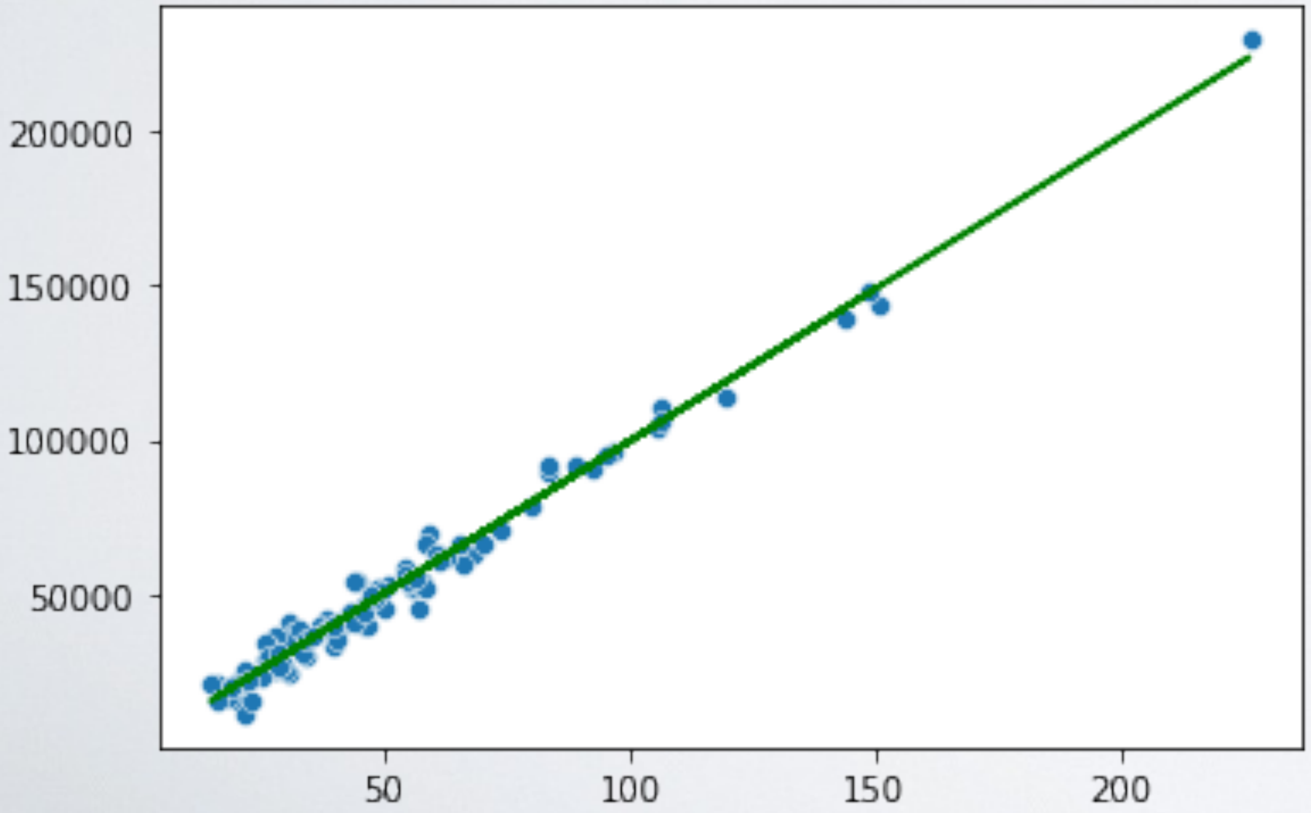
- We will use **linear regression** method, and more specifically **Ordinary Least Square**. First, with a single variable:

- 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 0.9813015148342528
```

LINEAR REGRESSION

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

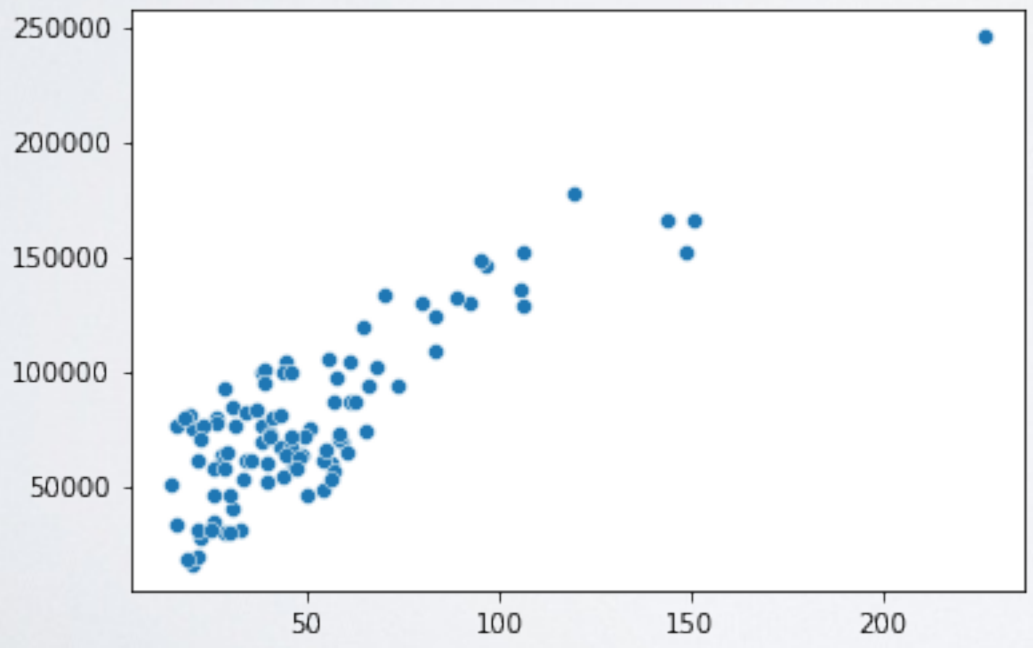
Using Mean	Using Median	Using Linear Regression
MSE 1105345073.7155044 RMSE 33246.73027104326 MAE 22740.967725747014 R2 0.0	MSE 1179133659.4166086 RMSE 34338.51568452848 MAE 21658.66828240126 R2 -0.06675615376207489	MSE 20668278.463901177 RMSE 4546.237836266508 MAE 3512.3861644882704 R2 0.981301514834252

LINEAR REGRESSION

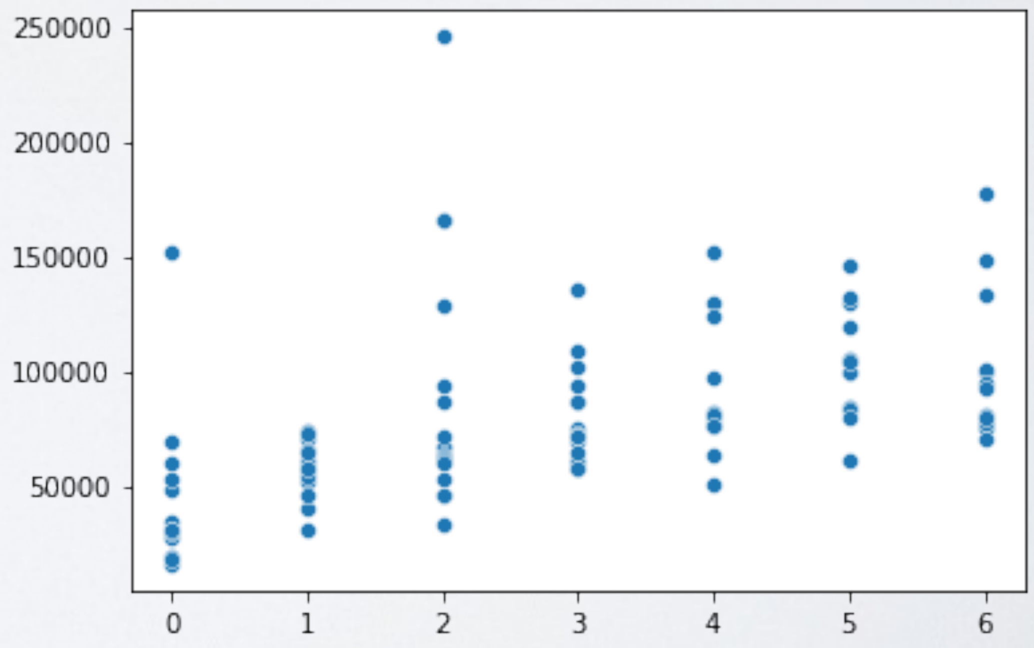
- Note: To generate the data, I used indeed a linear model, with parameters
  - β_0=~~987~~ 0
  - β_1=~~779~~ 1000

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

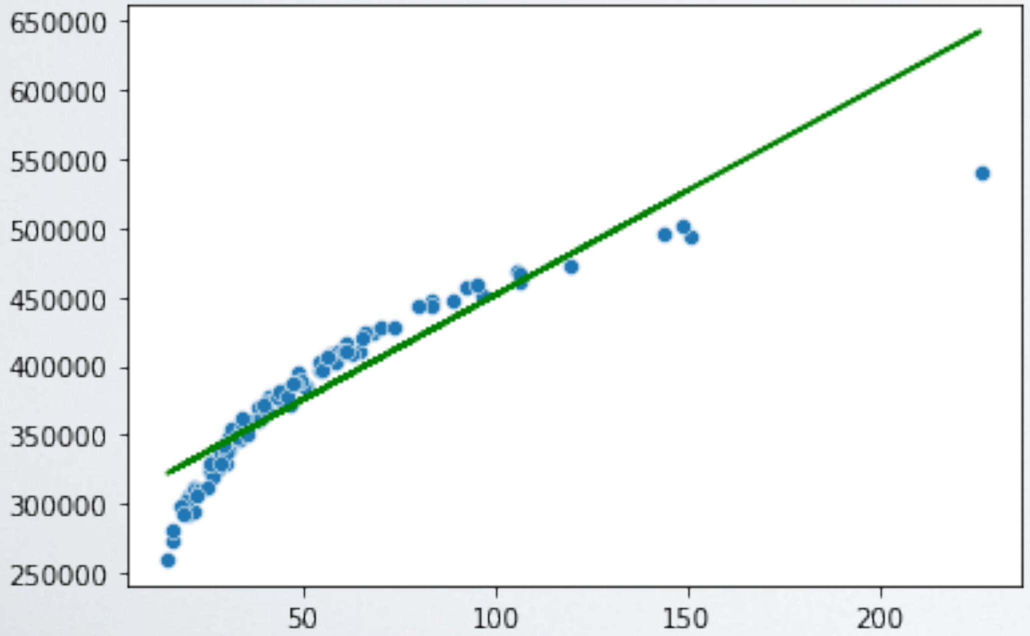
Surfaces only	Floor only	All features
MSE 388200345.3991482	MSE 785600976.607142	MSE 22157971.6387145
RMSE 19702.800445600322	RMSE 28028.57428780747	RMSE 4707.225471412486
MAE 16757.480694933285	MAE 22165.777484397917	MAE 3617.346073048316
R2 0.7329146952183824	R2 0.34222807880552575	R2 0.9847551176123155

Generative Parameters
- β_0= 0 β_1= 1 000, β_2 =10 000

Found Parameters
- β_0= 579 β_1= 994, β_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 0.9922842323758786
```

LINEAR REGRESSION

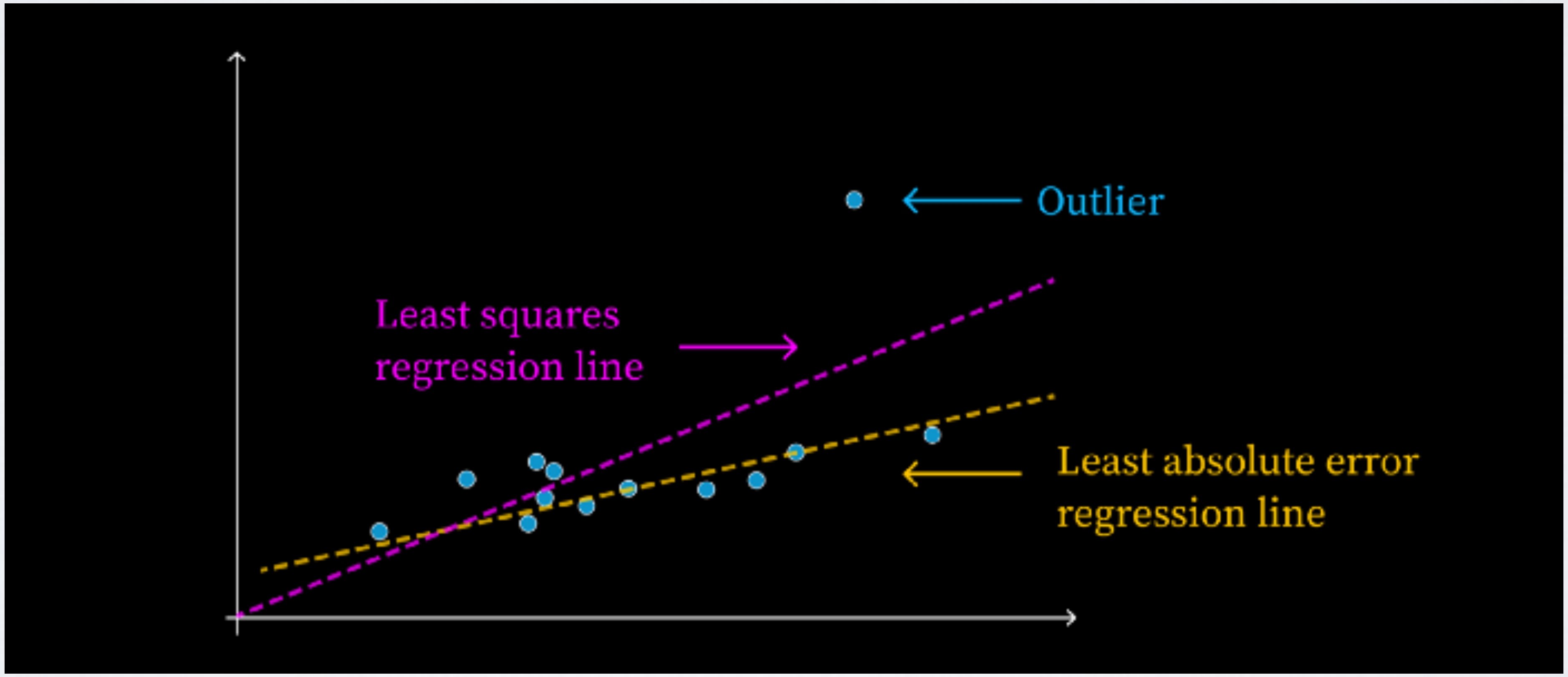
- Linear regression works if there are indeed linear relations
  - But there is no particular reason for relations to be linear
- In many scientific domains (e.g., epidemiology, biology, econometrics, etc.), linear regression is still widely used.
  - Why ?

OLS STRENGTH

- 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

OLS KNOWN WEAKNESS

- MSE is known to be sensitive to outliers



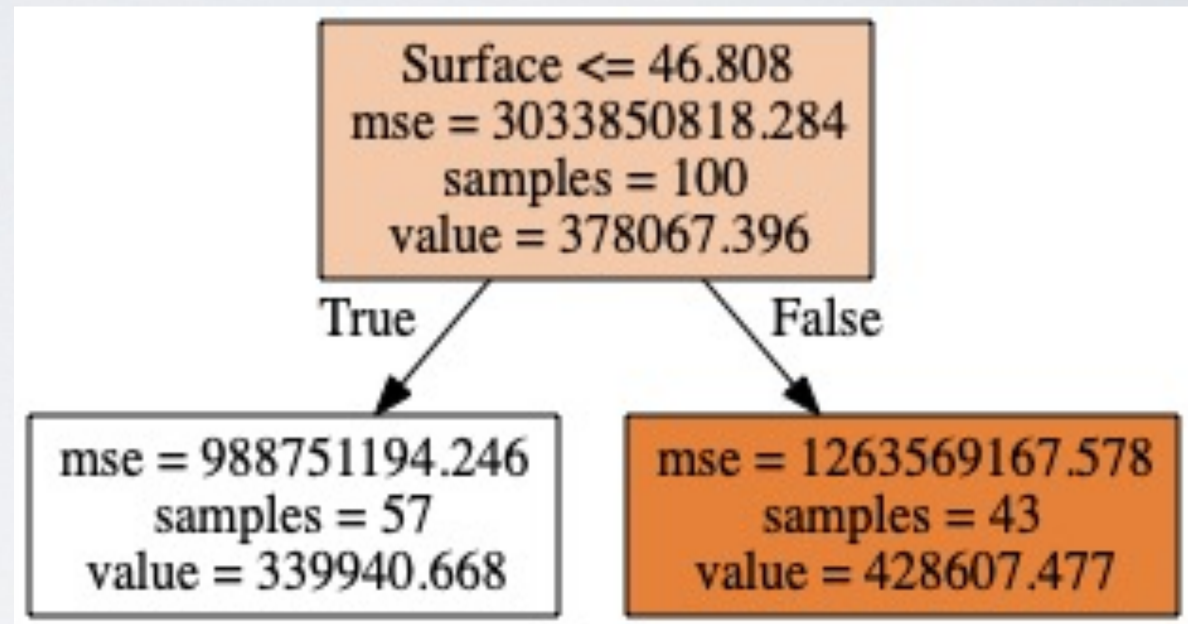
NON-LINEAR REGRESSION: DECISION TREE REGRESSION

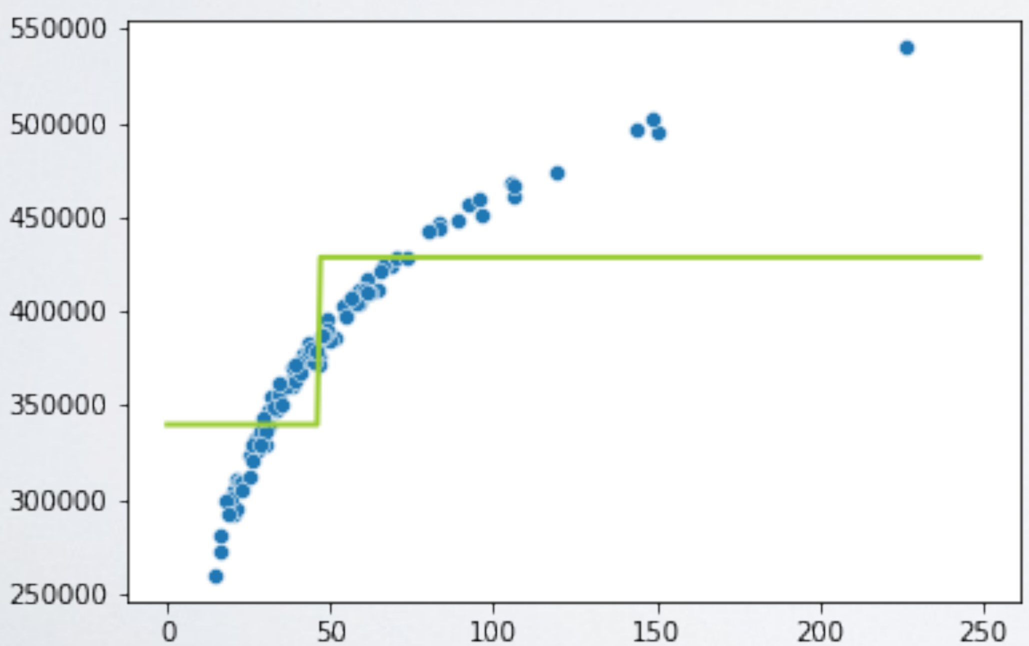
DECISION TREE

- 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

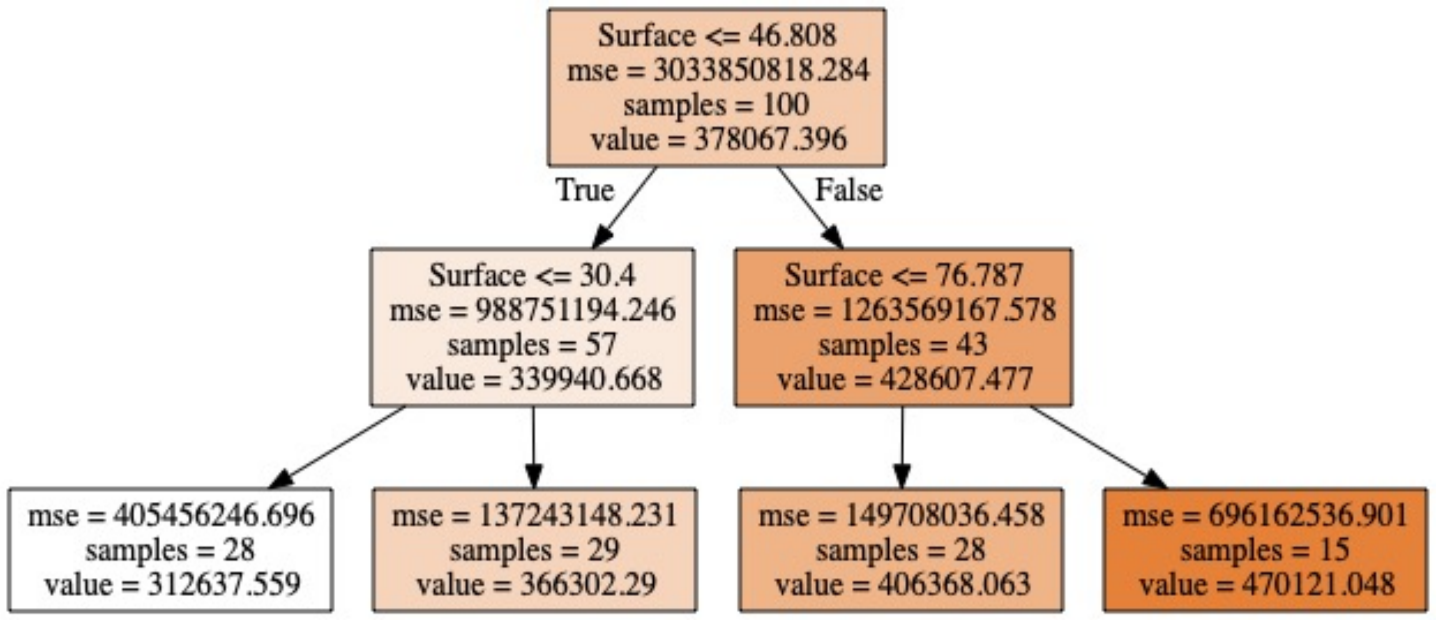


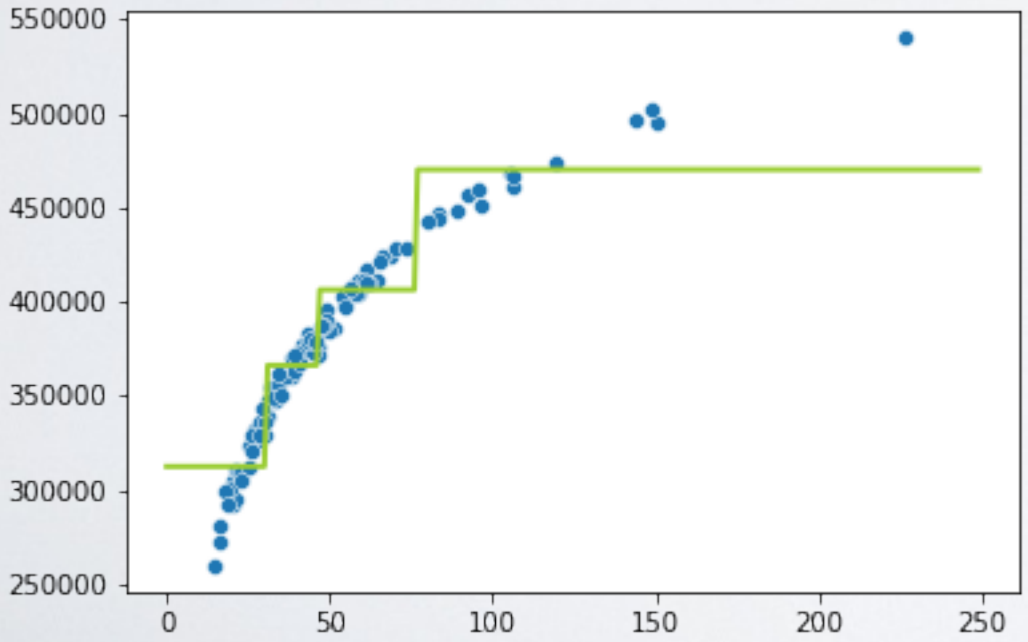



```
MSE 1106922922.7787206
RMSE 33270.45119589935
MAE 27836.40899704275
R2 0.6351425995939648
```

DECISION TREE

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

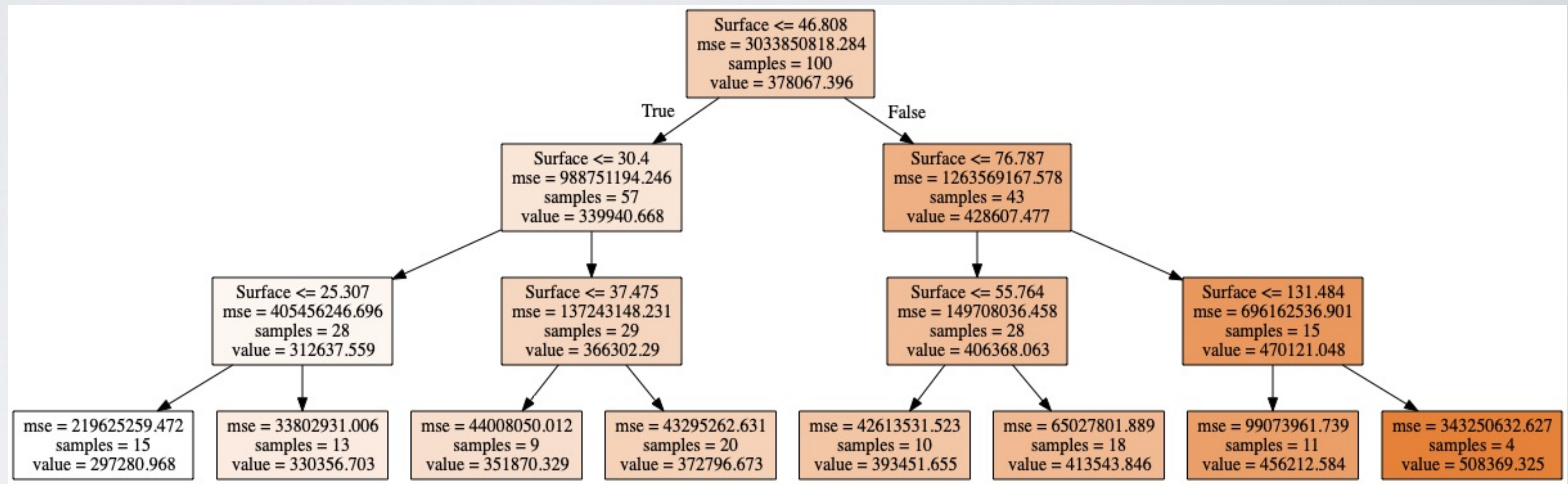


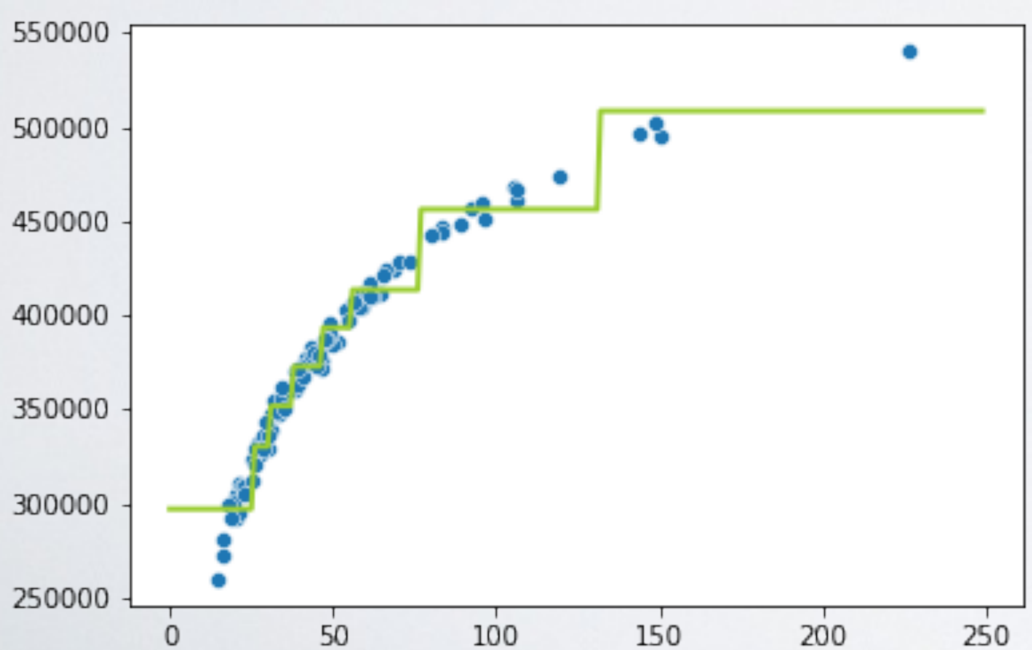



```
MSE 299670892.805488
RMSE 17311.00496232059
MAE 13262.652619929546
R2 0.9012242490634346
```

DECISION TREE

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

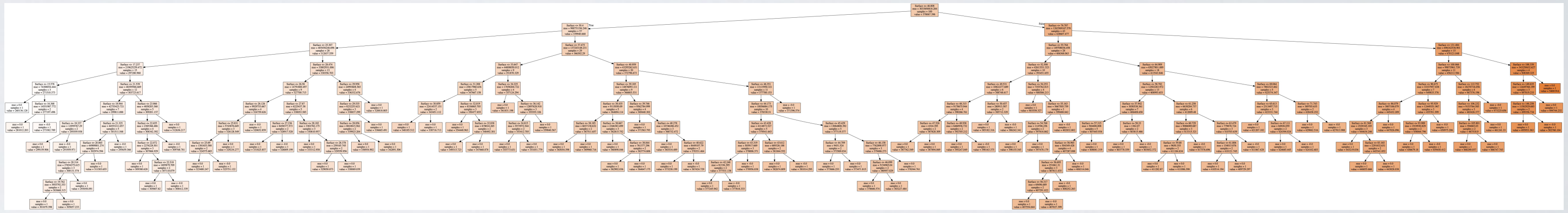


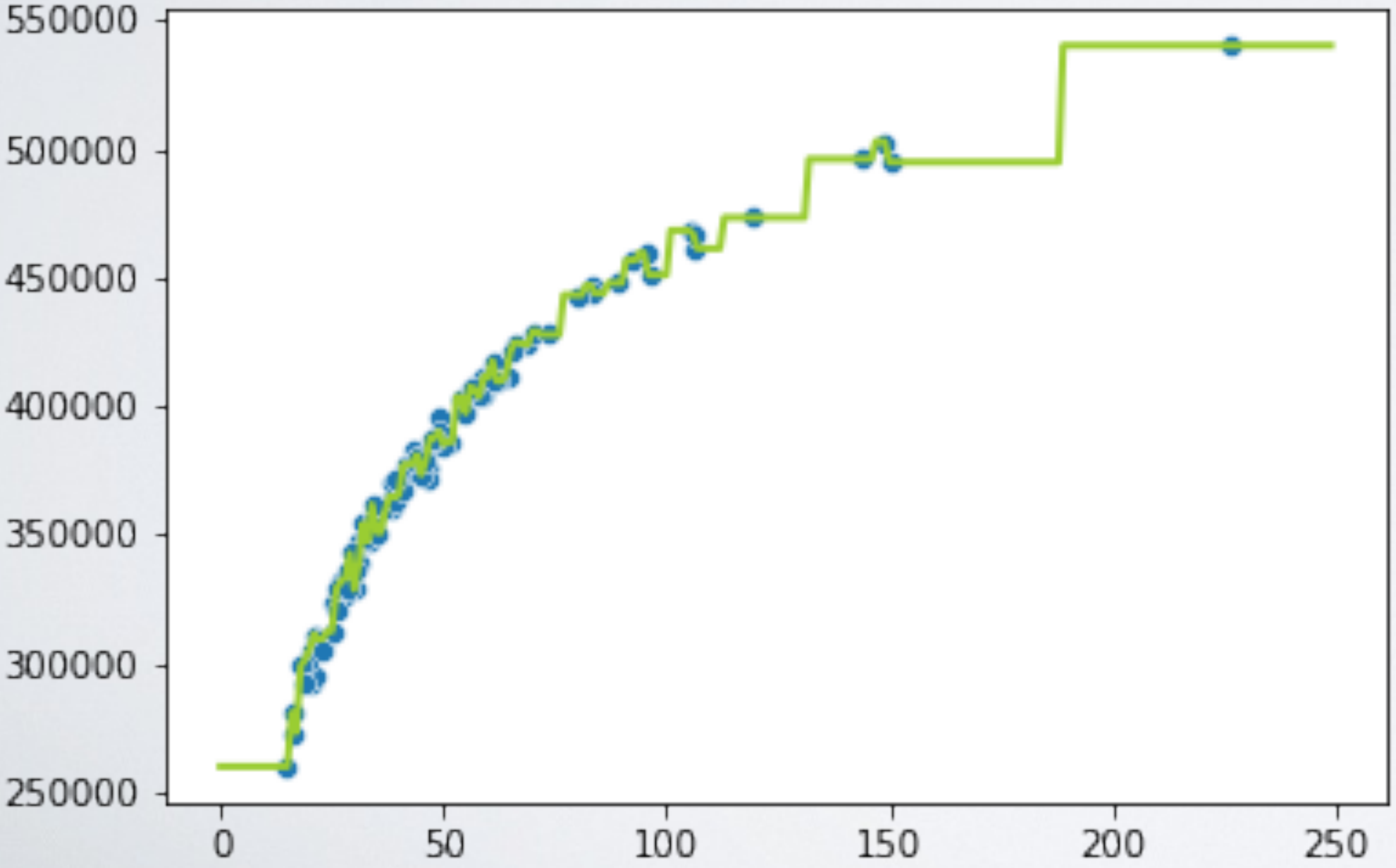



```
MSE 90552465.56733872
RMSE 9515.905924678886
MAE 7434.910779663157
R2 0.9701526307682573
```

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

AVOIDING OVERFIT

Train set

Do whatever you want :)

Test set

Use only once !

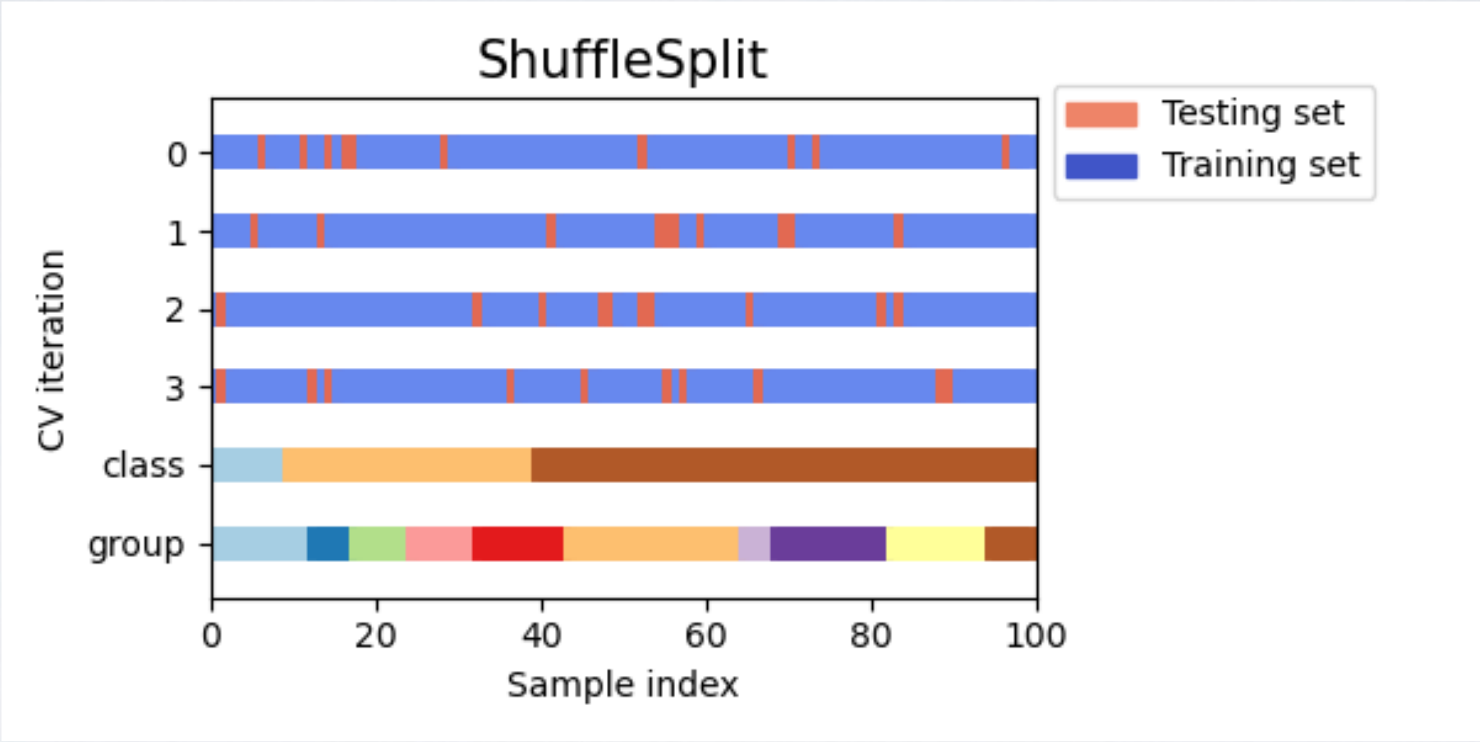
AVOIDING OVERFIT

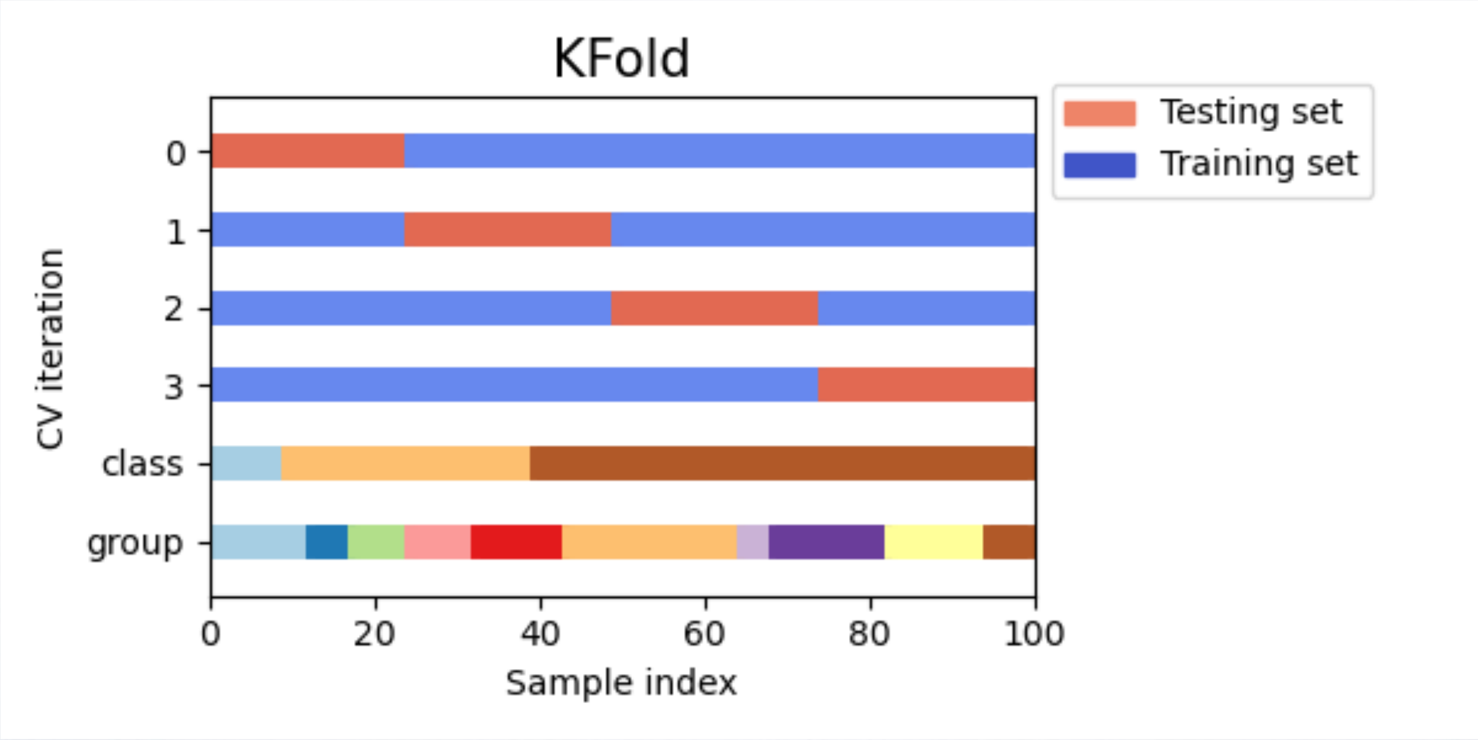
	Decision Tree, **levels=10**	Decision Tree, **levels=5**
Scores on Train Set	MSE 0.0 RMSE 0.0 MAE 0.0 R2 1.0	MSE 9675372.95170697 RMSE 3110.5261535159884 MAE 2364.5552169188454 R2 0.9968108606746918
Scores on Test Set	MSE 60522590.58807978 RMSE 7779.626635519199 MAE 6427.594619486819 R2 0.9689849224913336	MSE 47482936.48734139 RMSE 6890.786347532579 MAE 5748.307144423111 R2 0.9756671526915104

TRAIN/TEST SPLIT

- What size should your test set have?
  - No good answer. 66% Train, 33% Test is often a default choice
- Problem is if data is scarce
  - =>Cross validation

CROSS VALIDATION





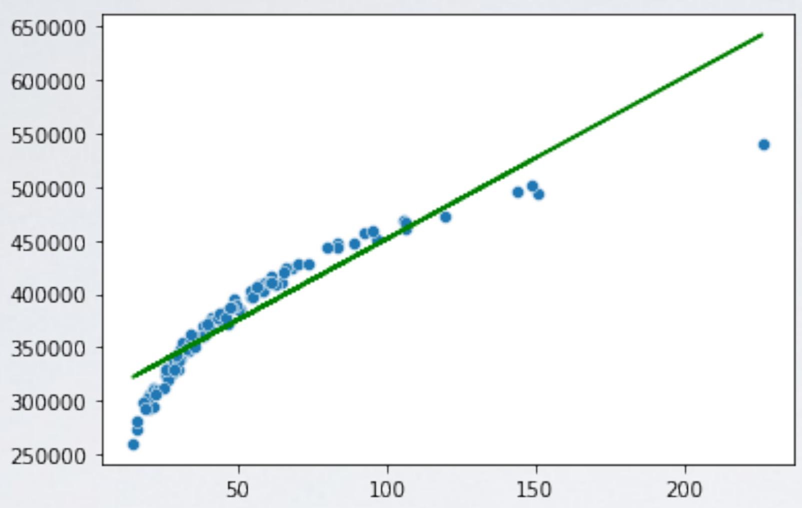
FIGHTING OVERFIT

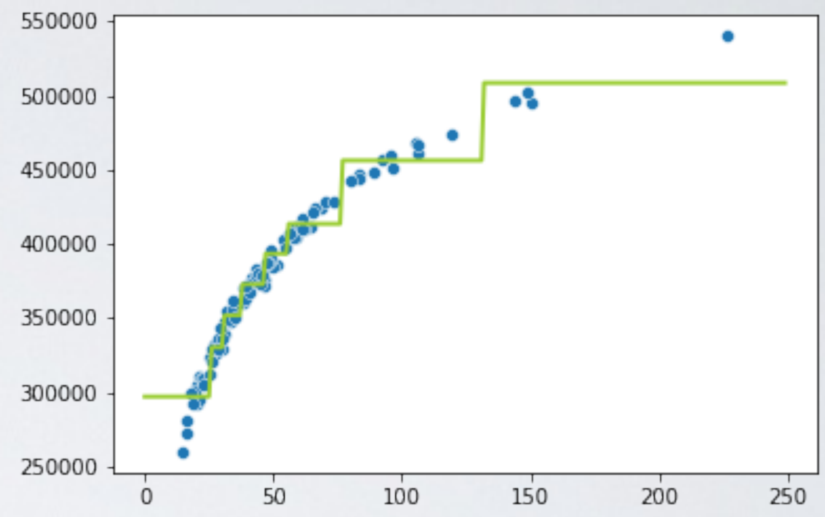
BACK TO THE METHOD

FIGHTING OVERFIT

- Implicit limit to overfit:
  - Because a method has a limited power of expression, it cannot overfit “too much”.
  - =>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

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 0.975667152691510 4

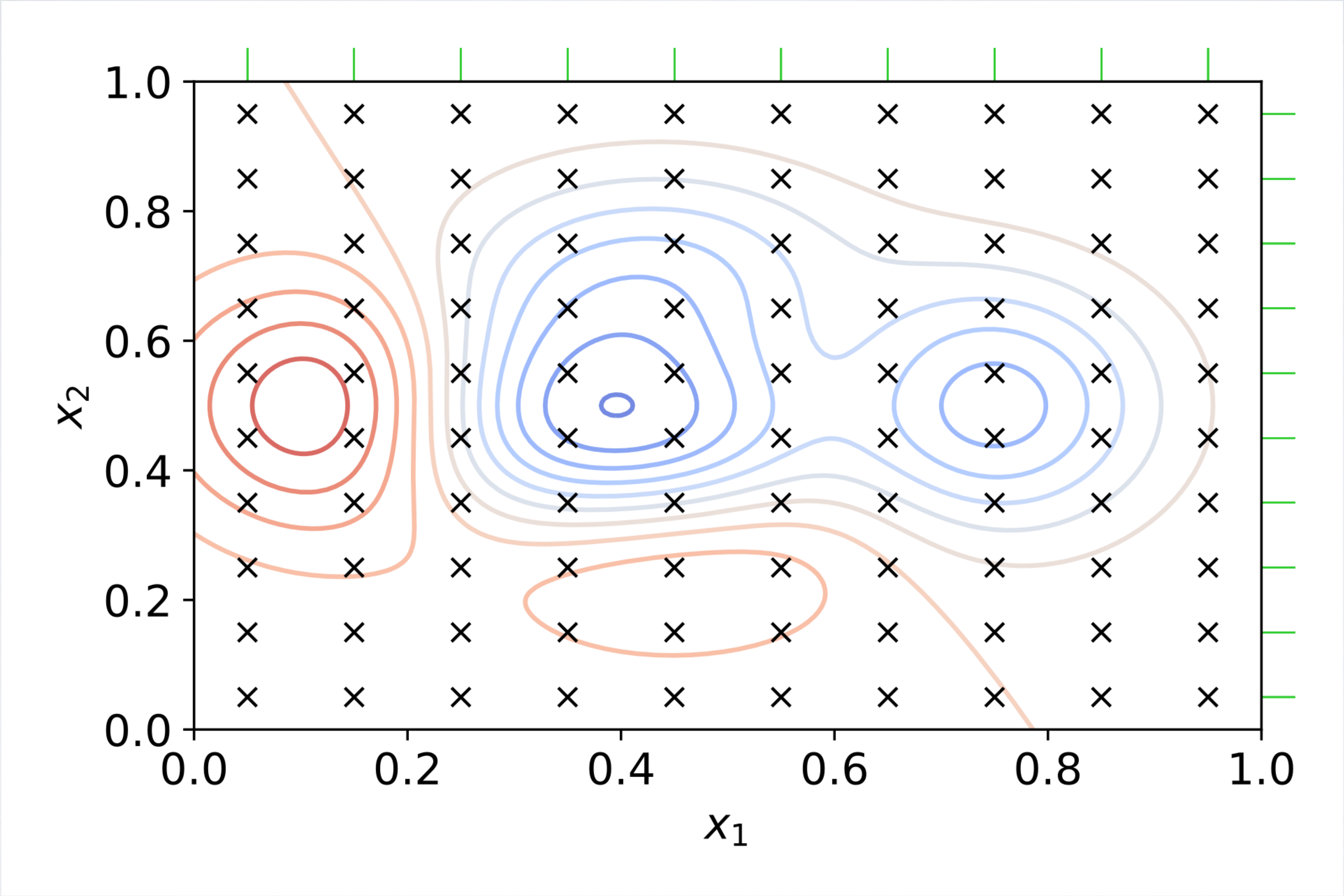
No visible overfit= underfit?

Still Some overfit

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…

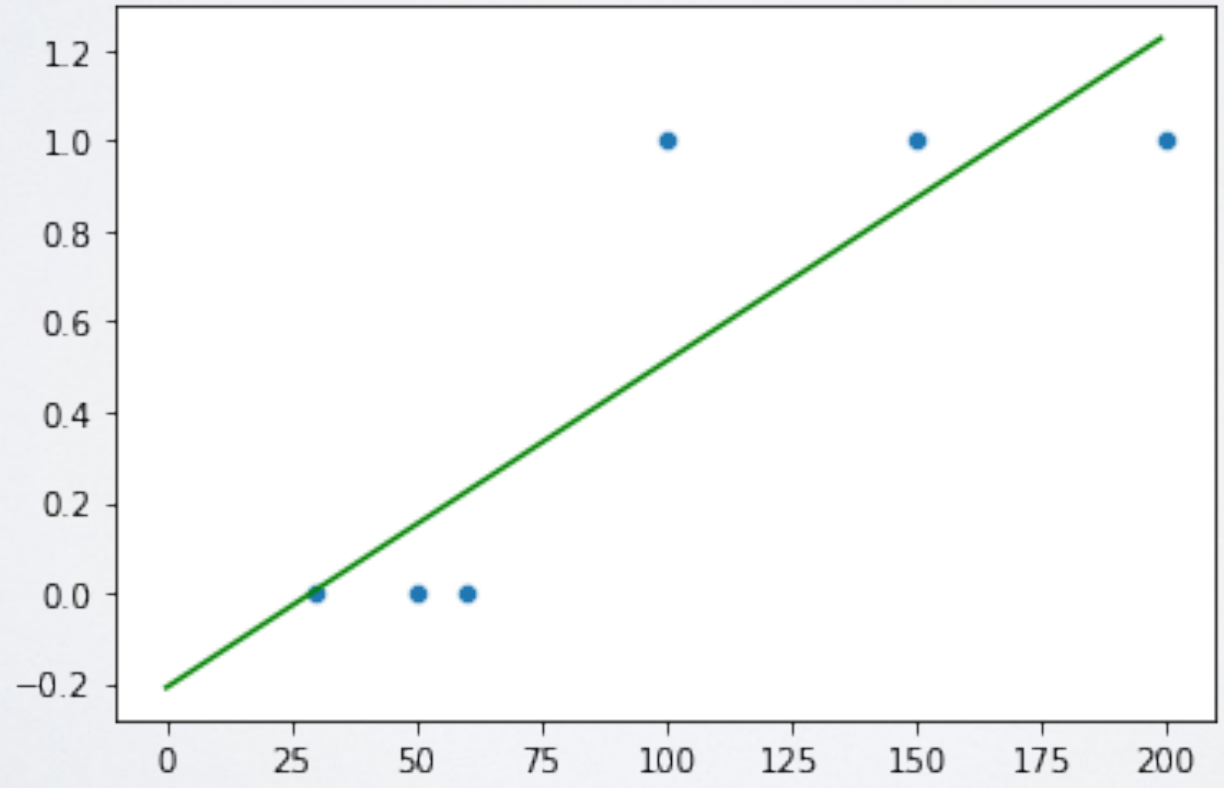
CLASSIFICATION

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

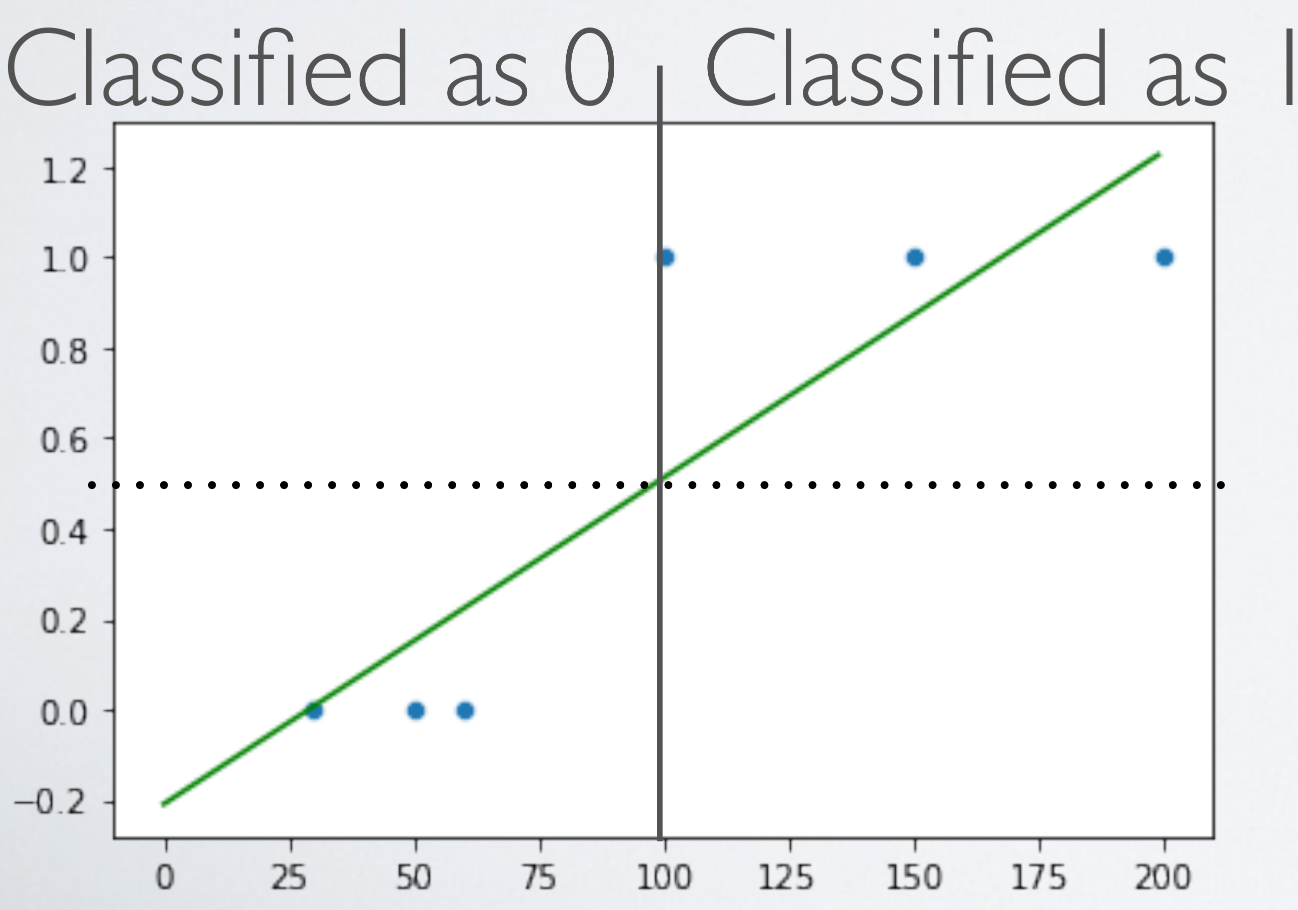
LINEAR CLASSIFICATION

- 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

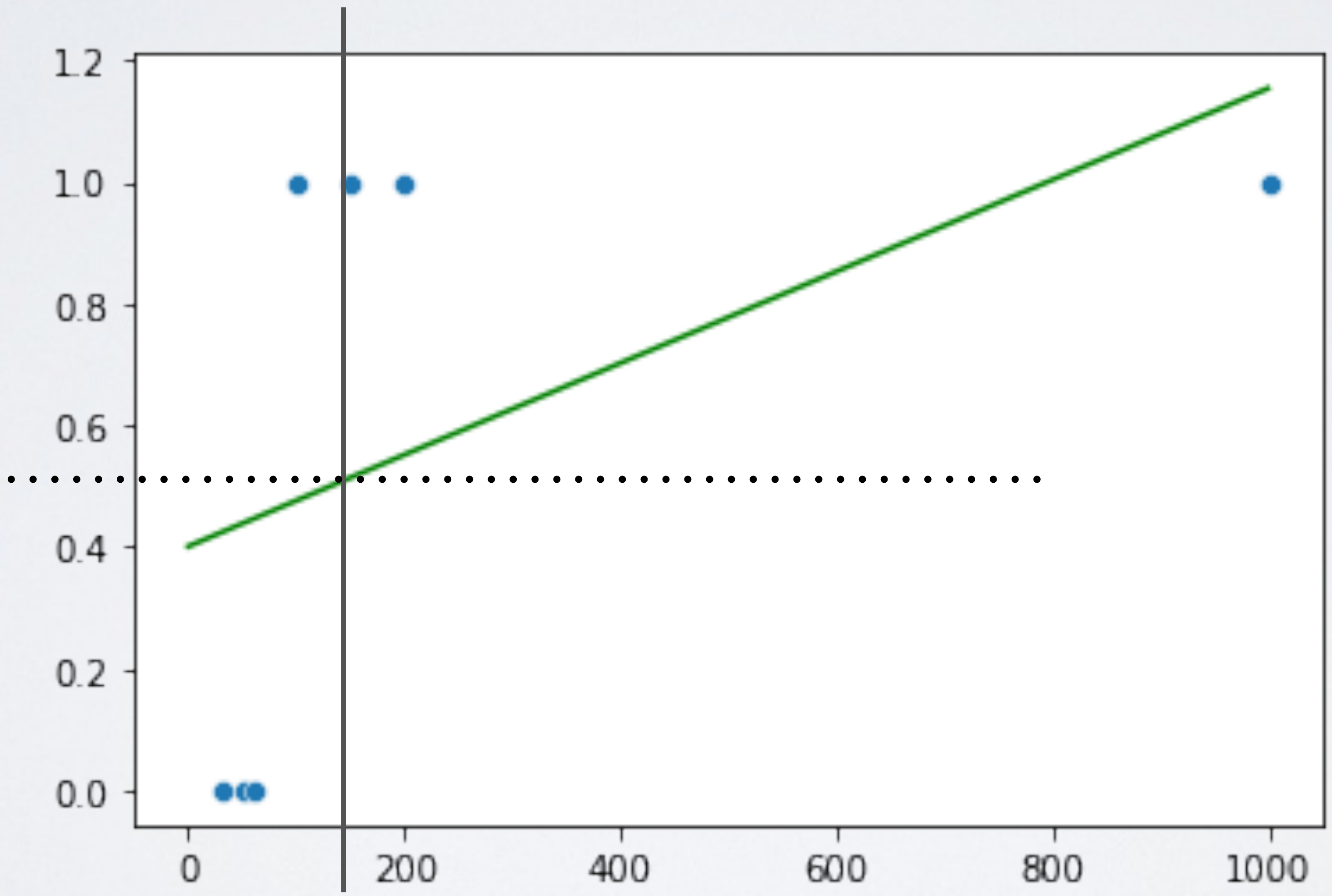
- We can easily adapt linear regression
- Imagine a 1 feature example:
  - We want to classify between apartments and houses
  - Our (unique) feature is dwelling surface




```
MSE 0.06361520558572538
RMSE 0.2522205494913636
MAE 0.20506852857512292
R2 0.7455391776570985
```

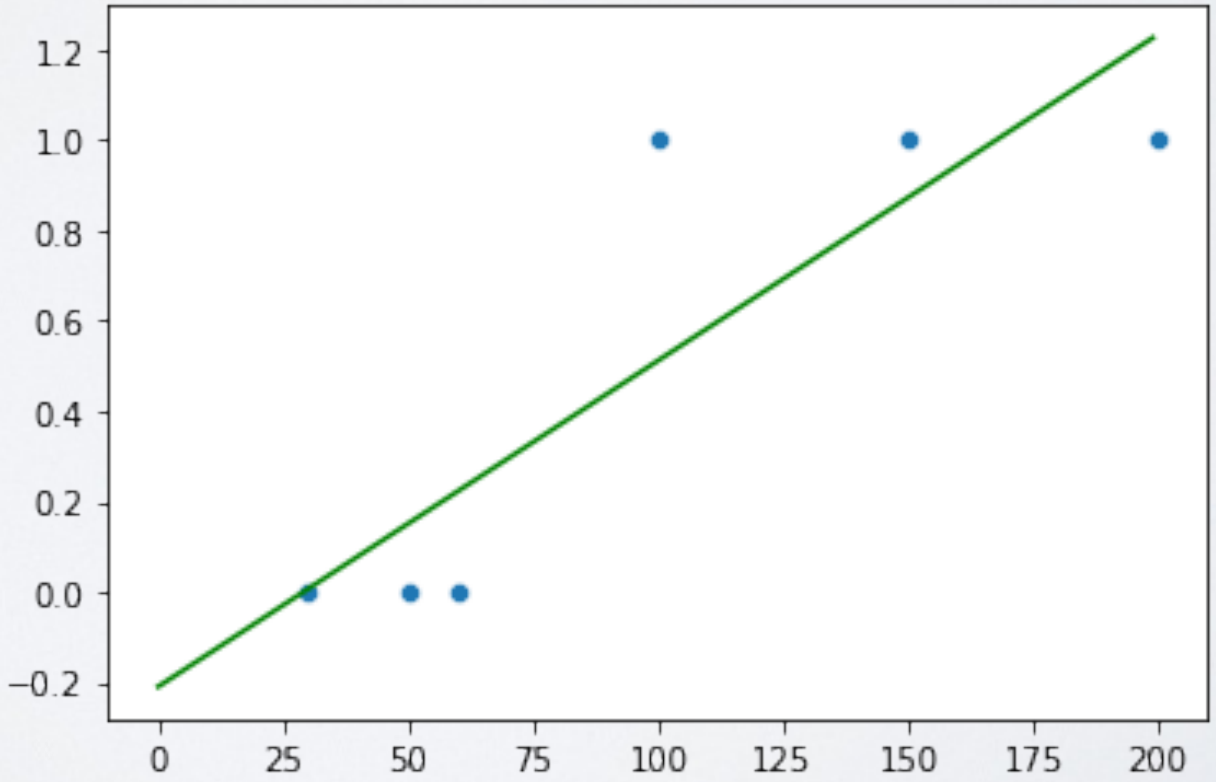
LINEAR CLASSIFICATION

- Weaknesses: Outliers

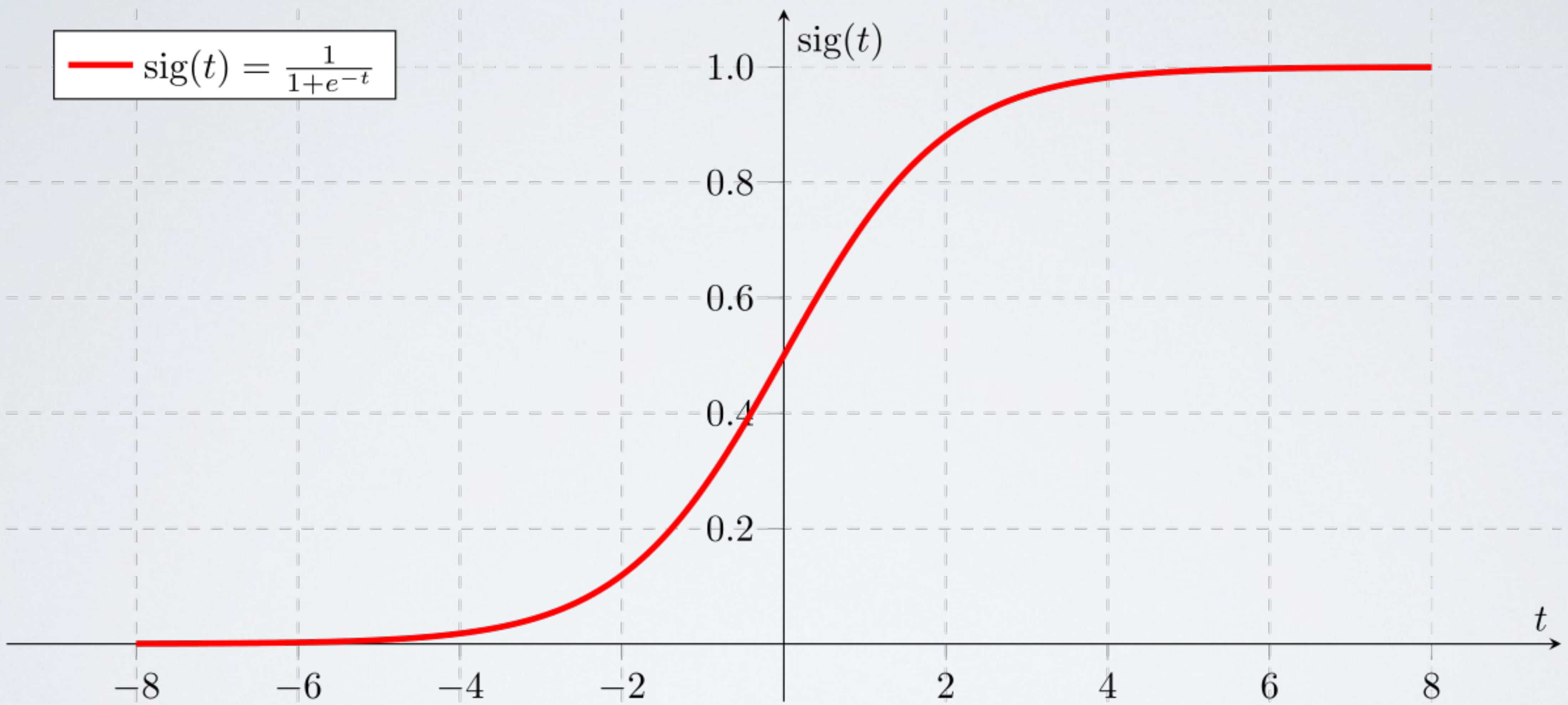


LINEAR CLASSIFICATION

- 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

After reformulation:

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

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

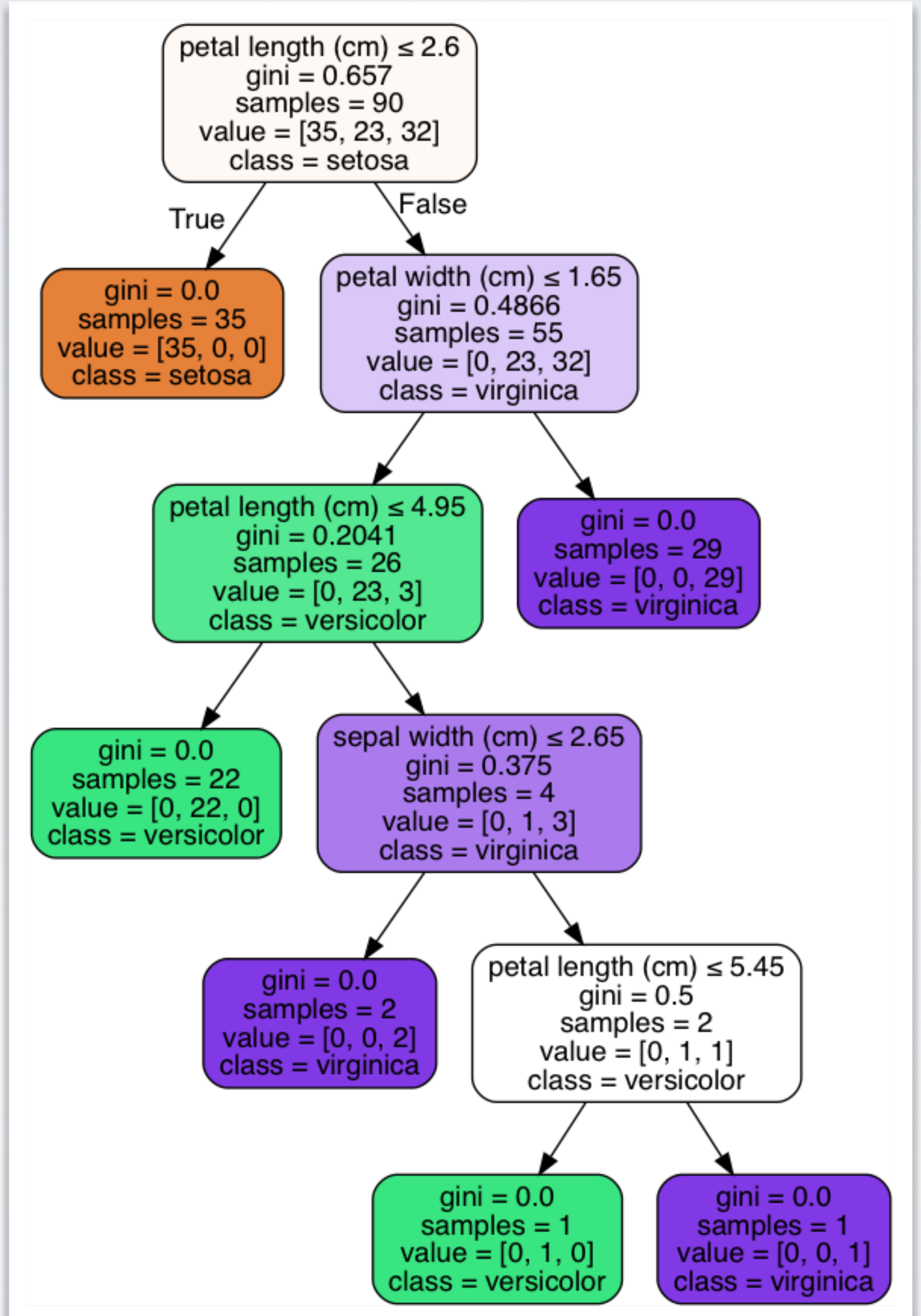
DECISION TREE

- 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

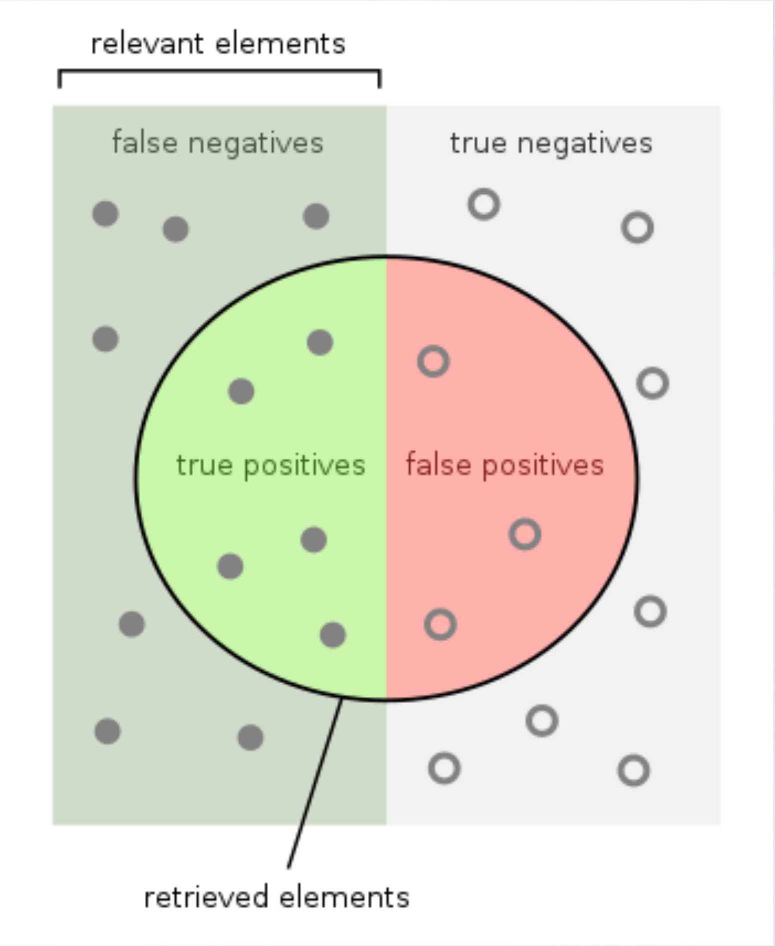
- Most common homogeneity/diversity/inequality/purity scores
  - p_i: fraction of items of class i
  - Gini Coefficient: $1 - \sum_j p_j^2$
    - If we classify by taking an element at random, probability to be wrong.
  - Entropy: $-\sum_j p_j \cdot log_2 p_j$
    - Interpretation: average # of bits required to encode the information of the class of each item

DECISION TREE



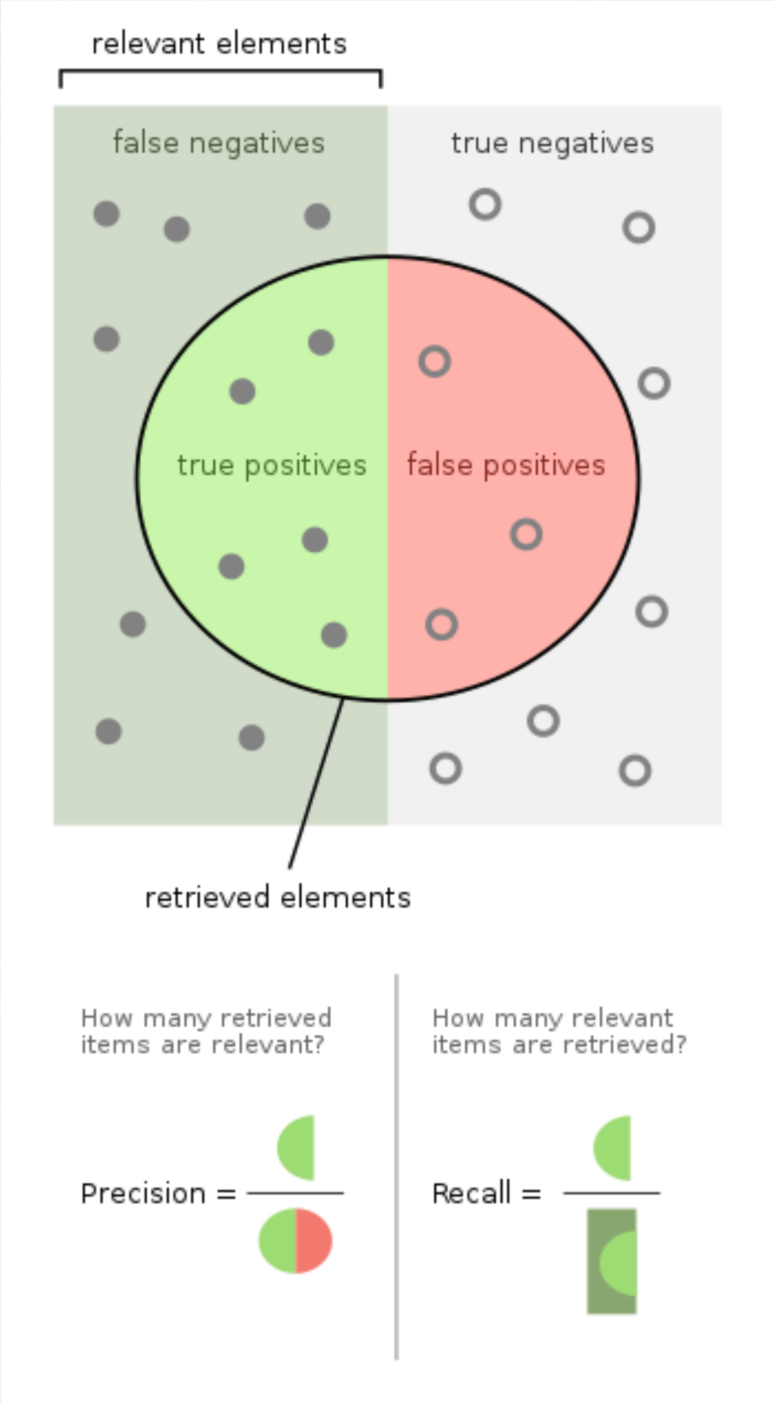
CLASSIFICATION: EVALUATION

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



CLASSIFICATION: EVALUATION

- Precision= $\dfrac{TP}{TP + FP}$
  - Among those predicted as True, fraction of really True
- Recall= $\dfrac{TP}{TP + FN}$
  - Among those really true, what fraction did we identity correctly



ACCURACY

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

- 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

- Will see in link prediction class