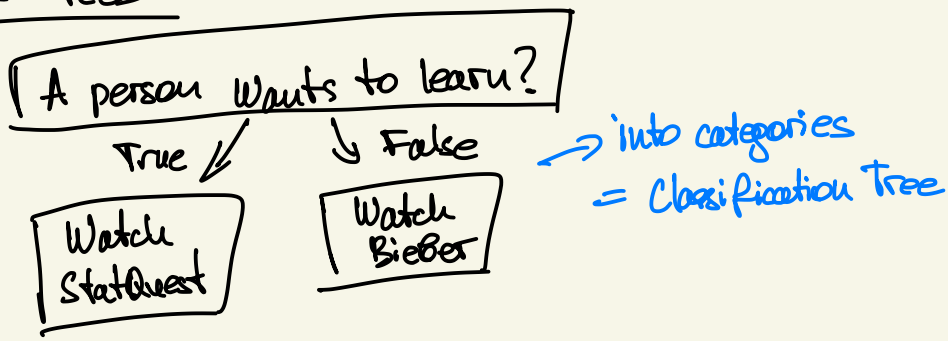
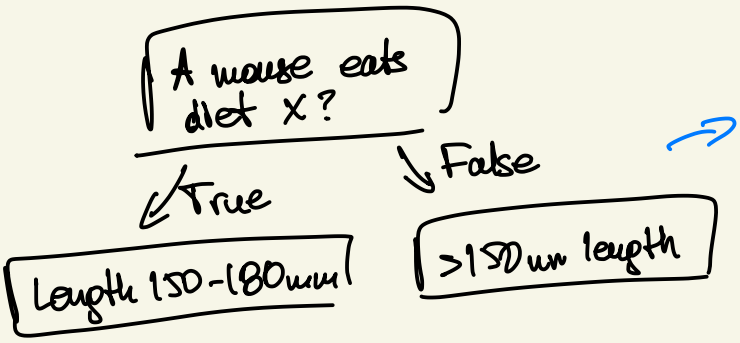
Trees and Forests
(StatQuest)

# Decision Trees

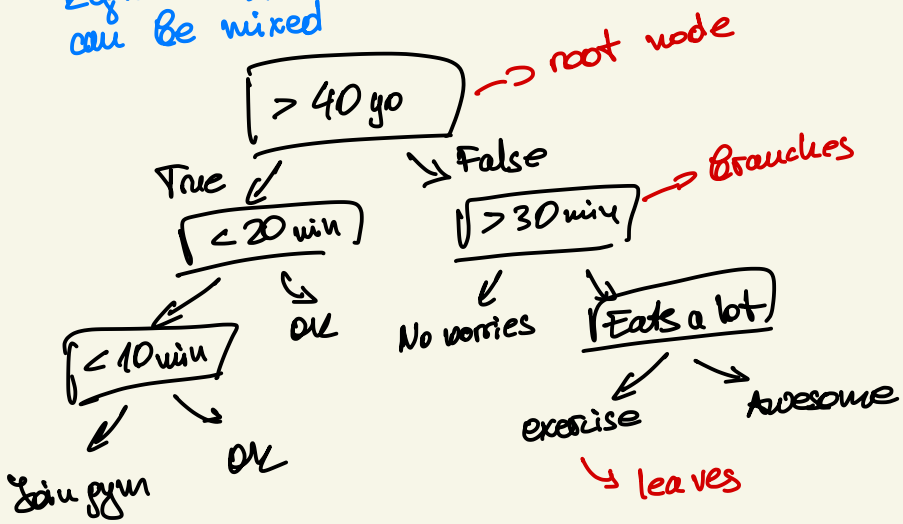



→ into categories
= Classification Tree




→ Regression tree (numeric values)

Regression & Classifications can be mixed




- Build tree from data
  How well each feature predict outcome? → Best fit is root
  ↳ Impure leaves

soda?

| Yes | No | Yes | No |
|---|---|---|---|
| 3 | 1 | 0 | 3 |

- Gini Impurity:
  $= 1 - (\text{prob of yes})^2 - (\text{prob No})^2$
  $= 1 - \left(\frac{1}{4}\right)^2 - \left(\frac{3}{4}\right)^2 = 0{,}375$ for a leaf

  weighted for both leaves:

  $$\frac{4}{(4+3)} 0.375 + \frac{3}{4+3} \cdot 0{,}444 = 0{,}405$$

  for a feature

- Gini Impurity for numeric value

sort:

7
9,5 > 12
15 > 18
26,5 > 35
⋮

for all
⇒ lowest impurity of all features ⇒ root

Age < 9,5

| Yes | No | Yes | No |
|---|---|---|---|
| 0 | 1 | 3 | 3 |

$$Imp_L = 1 - \left(\frac{0}{0+1}\right)^2 - \left(\frac{1}{0+1}\right)^2 = 0$$

$$Imp_R = 1 - \left(\frac{3}{6}\right)^2 - \left(\frac{3}{6}\right)^2 = 0,5$$

$$Imp_{TOTAL} = \frac{1}{1+6} \cdot 0 + \frac{6}{1+6} \cdot 0,5 = 0,429$$

- Nodes after root

reduce with node impure ← other features

Soda?

| Yes | No |
|---|---|
| 3 | 1 |

| Yes | No |
|---|---|
| 0 | 3 |

Popcorn?

| Yes | No |
|---|---|
| 1 | 1 |

| Yes | No |
|---|---|
| 2 | 0 |

$Imp_{Total} = 0,25$

Age < 12,5

| Yes | No |
|---|---|
| 0 | 1 |

| Yes | No |
|---|---|
| 3 | 0 |

$Imp_{TOTAL} = 0$ ⇒ use to split node

- Output of a leaf = category with most values

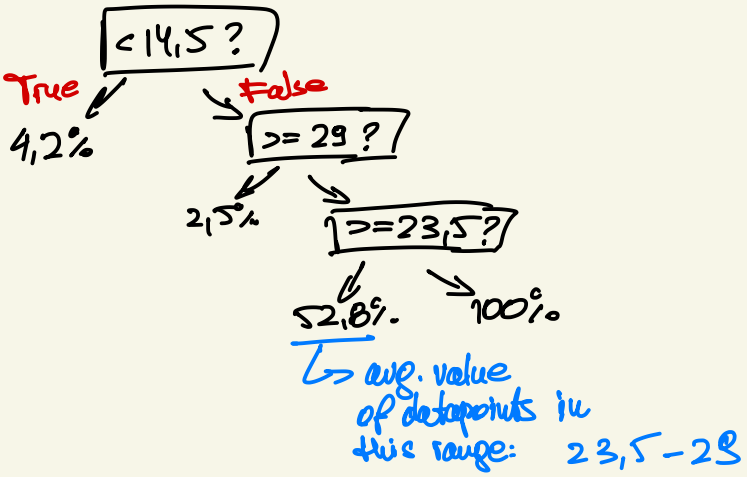
- Overfit if only 1 person gets to leaf

require >3 people per leaf to prevent overfit

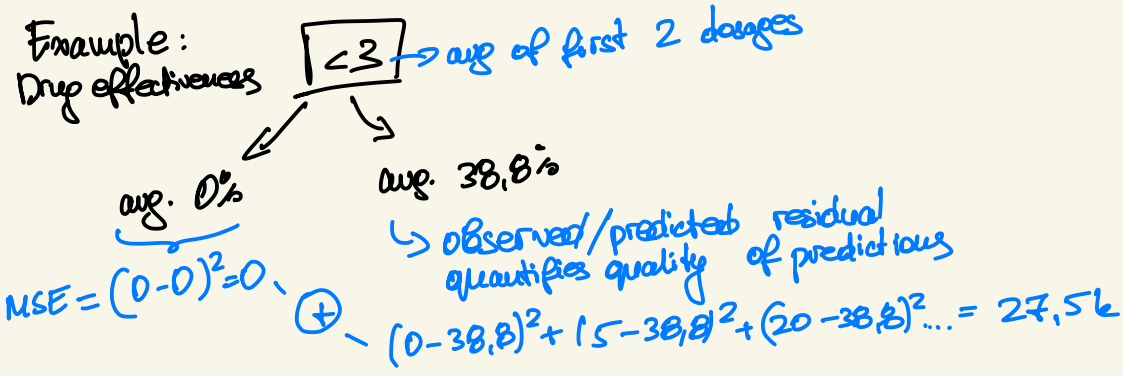
- Pruning for overfit
- Cross-validation

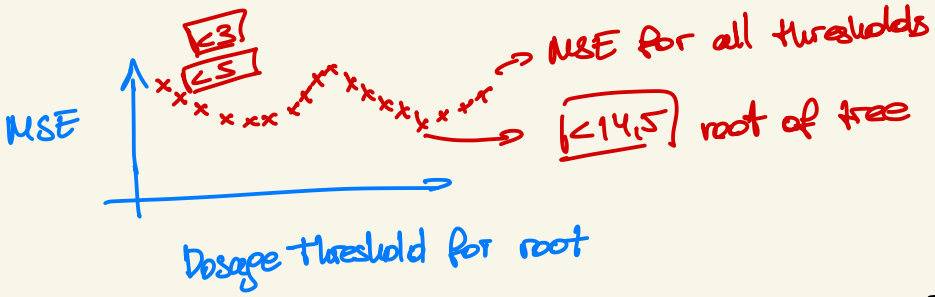
## Regression Trees

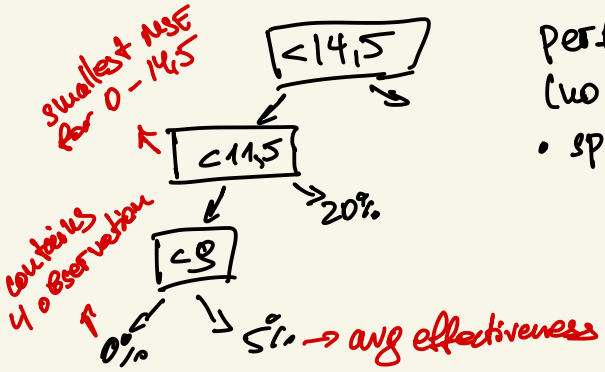
each leaf represents numeric value




Example: Drug effectiveness




perfect fit may overfit
(no Bias, but huge variance)
- split only when >20 observations

- Multiple Features
  - sort numeric values of feature and calculate MSE for root
  - choose lowest MSE
  - if categorical, set sex = Female? and calculate MSE
    ↳ no > or <
  - stop when observations < 20

## Prune Regression Trees

- many versions, here -> cost complexity pruning or weakest link pruning
- large residuals, fort better when testing (overfitting)
- remove leaves, decrease depth

- Calculate SSR for each tree (sum of squared residuals)

  Full tree: for each leaf: 320 + 75 + 148,8 + 0 = 543,8

  - 1 leaf: SSR = 5,5k
  - 2 leaves: SSR = 18,2k
  - 3 leaves: SSR = 28,8k

Tree Score = SSR + $\alpha \cdot T$

$\alpha$ = tuning parameter from cross validation
T = complexity penalty f(number of leaves)

$Tree_1 = 543,8 + 10k \cdot 4 = 40,5k$

$Tree_2 = 5,5k + 10k \cdot 3 = 35,5k$ -> pick lowest score

$Tree_3 = 18,2k + 10k \cdot 2 = 38,2k$

$Tree_4 = 28,8k + 10k = 38,8k$

- start with $\alpha = 0$ and increase $\alpha$ until removing leaves gives tree lower score

  $\alpha = 0$ = full size -> calculate SSR for testing for each new tree
  $\alpha = 10k$ = - 1 leaf
  $\alpha = 15k$ = - 2 leaves
  $\alpha = 22k$ = - 3 leaves

  with new train/test data, build new trees, calculate $\alpha$, calculate SSR 10 times (10 times cross validation)
  lowest SSR $\alpha$ is the final $\alpha$ value

  ↳ use $\alpha$ in the full data

## Cross Validation

- devide 25% - 75% test-train (4 Fold Cross Validation)
- use each 25% Block as testing Block
- keep track of how well method did with test data
  - ↳ Log Regression / SVM / k nearest
- Leave One Out Cross validation (tests each sample individually)
- In practice: 10 Fold
- tuning parameter (guessed) cross valid. can determine it

## Decision Trees Feature Selection / Missing Data

- predict missing values with regression
- select features with PCA

## Random Forest

- Training
- Original Dataset —(random samples; same samples more than once)→ Bootstrapped Dataset

- Create Decision Tree, first use random features (columns), not all, for root node
- after root, from the remaining all columns, randomly select subset of features and determine best fit for the node
- Build as usual the tree
- repeat whole process 100+ times

- Testing
  - run data through each generated tree
  - select class based on most votes

Bootstrapping + aggregate to make decision = Bagging

1/3 of original data is not in Bootstrapped dataset

↳ out of Bag dataset

check which trees label out-of-Bag sample falsely ⇒ out-of-Bag error

↳ $\frac{\text{falsely classified}}{\text{correctly classified}}$

- Compare hyperparameters
  - change number of vars used per step
  - $\sqrt{\#\text{vars}}$ start, then try more or less

# Random Forest Missing Data / Sample Clustering

- in original data: make guess based on label, other features
- after running data through a tree, keep similar (ended on same leaf) samples in proximity matrix

samples

| | 1 | 2 | 3 | 4 |
|---|---|---|---|---|
| 1 | | | | |
| 2 | | | | |
| 3 | | | (1) | |
| 4 | | | (1) | |

Add proximity matrices after all trees have been executed

- devide matrix values by number of trees 4

3/4 ended up on same leaf node

| Blocked Arteries |
|---|
| No |
| Yes |
| No |
| ? |

| | 1 | 2 | 3 | 4 |
|---|---|---|---|---|
| 1 | | | | |
| 2 | | | | |
| 3 | | | | |
| 4 | 0,1 | 0,1 | 0,8 | |

Both have 'no' => 0,8

$\frac{0,1}{0,1+0,1+0,8} = 0,1$ → weight for 'yes'

frequency of 'yes' = 1/3 of all occurances

$\frac{1}{3} \cdot 0,1 = 0,03$ weighted frequency

'No': $\frac{0,1+0,8}{1} \cdot \frac{2}{3} = 0,6$ weighted frequency

=> No is more likely for this sample

- repeat whole process until missing values converge → don't change when we recalculate ~6-7 times

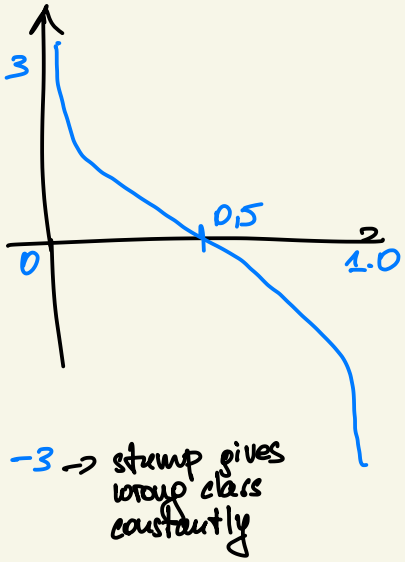
— Proximity matrices
- largest number in proximity = 100% close
- 1 - proximity = distance between samples
- heatmap, mds plot of distances

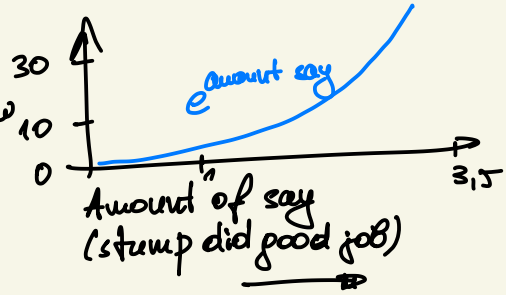
— Missing data in new samples
- 2 copies - with/without heart disease (label)
- guess missing features and perform the iterative process 4 until it converges
- run through random forrest and see which was labeled correctly most of the times
- the option with most corrects wins

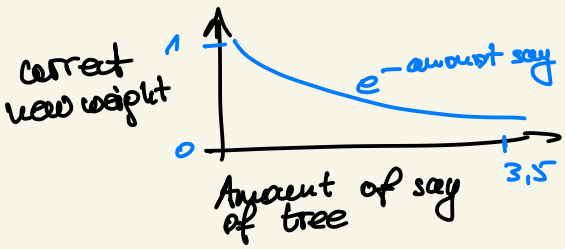
# AdaBoost

- in forest of AdaBoost trees have 1 node + 2 leaves (stump)
- stumps are weak learners
- some stumps have larger vote
- order is important, errors of prev. stump influence next stump

– Creation
- add feature „Sample Weight" (starts at $\frac{1}{\text{total samples}}$) = sample importance
- find which feature best classifies the samples and use it as root (using Gini Index)
- Determine stump Total Error (sum of weights of incorrectly classified samples)
  ↳ Total Error: 0–1
- Amount of stump's say = $\frac{1}{2}\log\left(\frac{1-\text{Total Error}}{\text{Total Error}}\right)$
- if Total Error $\neq$ 0/1, ⇒ add small $\varepsilon$
- Emphasize wrongly classified samples weights:
  new weight (incorrect) = sample weight · $e^{\text{amount of say}}$
  correct sample weight = sample weight · $e^{-\text{amount of say}}$
  normalize new sample weights to add to (1)
- use new sample weights for new stumps
- Generate random num 0–1 and use weights as distribution to select samples in new collection (more heavy weight samples)
- assign equal weights to all samples again and repeat whole process
- on classify – add amounts-of-say for same classification trees

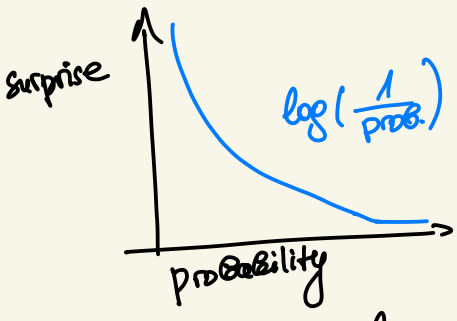



# Entropy

- low probability = high surprise
- high prob. = low surprise
- surprise = $\log\left(\frac{1}{P(X)}\right)$




- for 2 outputs: $\log_2$

Heads 80% $\Rightarrow \log_2\left(\frac{1}{0,8}\right) = 0,15$ surprise

Tails 10% $\Rightarrow \log_2\left(\frac{1}{0,1}\right) = 3,32$

- Total surprise = $\sum$ surprises

$(0,8 \cdot 100) \cdot 0,15 + (0,1 \cdot 100) \cdot 3,32 = 46,7$ for 100 coin flips

(0,8 · 100): 100 tosses heads; 0,15: surprise

$\Rightarrow 0,47$ surprise/coin flip = Entropy = E(surprise)

E(surprise) = $0,8 \cdot 0,15 + 0,1 \cdot 3,32 = 0,47$ = avg. surprise to expect

$$\text{Entropy} = \sum \log\left(\frac{1}{p(x)}\right) p(x) = -\sum p(x) \log(p(x))$$

$\log\left(\frac{1}{p(x)}\right)$: surprise; $p(x)$: probability of surprise

Example:

Area A: 6/7 orange chickens

$\Rightarrow$ Entropy = $6/7 \cdot \log_2\left(\frac{1}{6/7}\right) + 1/7 \cdot \log_2\left(\frac{1}{1/7}\right) = 0,59$

Area C: (equal number of chickens)

Entropy = $\frac{7}{14} \cdot \log_2\left(\frac{1}{7/14}\right) + \frac{7}{14} \cdot \log_2\left(\frac{1}{7/14}\right) = 1$

## Gradient Boost

- continuous values - Gradient Boost for Regression
- makes single leaf → first guess ist avg. value
- create tree with 8-32 leaves (Fixed size, multiple error-based trees)
- stop when additional trees fail to improve the fit

### - Building Model

- start with avg. weight leaf (everyone is 71.2 kg)
- second tree:
- Error = Observed weight - Predicted weight = Pseudo Residual column
- Use all features to build tree, predicting the residuals (restrict leaves, use avg of multiple samples)
- start with initial 71,2 prediction; run through the tree and sum them
- Prevent overfit with low bias/high variance: avg.weight + learning rate x tree

predicted = $71,2 + (0,1 \cdot 16,8) = 72,9$

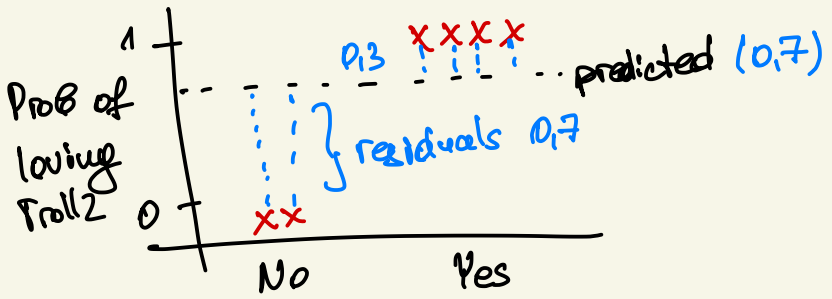
→ scales between 0-1 taking smaller steps

- recalculate residuals before building next tree
- Build tree 3 to predict new residuals (# leaves can be different)
- scale by 0,1 again, residuals, predict residuals on next run again

- make trees until residual improvement is too small
- predict by running through 1st leaf, and all following trees

## Gradient Boost for Classification

- start with single leaf: log(odds) → $\log\left(\frac{4}{2}\right) = 0{,}7$
- convert to probability: $\frac{e^{\log(odds)}}{1+e^{\log(odds)}} = 0{,}7$ 4 people love / 2 people don't
- calculate pseudo residuals (Residual = Observed − Predicted)

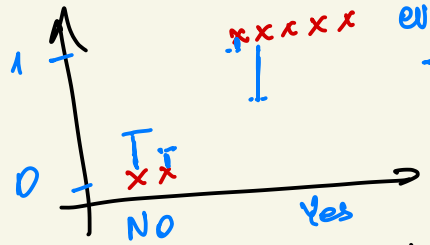



- Build a tree using all other features + limit number of leaves to 8-32
- when multiple values reside in a leaf: $\frac{\sum_i \text{residual}_i}{\sum_i \text{Previous Prob}_i \times (1 - \text{Prev Prob}_i)}$ → all residuals in the leaf

For 2 residuals in 1 leaf:

$$\frac{0{,}3 + (-0{,}7)}{0{,}7 \cdot (1-0{,}7) + 0{,}7(1-0{,}7)} = -1$$

- Scale new tree by a learning rate (≈ 0,1) $\frac{-0{,}7 \rightarrow \text{current leaf}}{\sum 0{,}7 \cdot (1-0{,}7) \rightarrow \text{prob from initial leaf}} = -3.3$ output for leaf
- $0{,}7 + 0{,}8 \cdot 1{,}4 = (1{,}8)$ adds prediction
  - initial leaf → 1st
  - ↳ prob. $= \frac{e^{1{,}8}}{1+e^{1{,}8}} = 0{,}9$ → save in a column
- calculate new residuals = Observed − Predicted

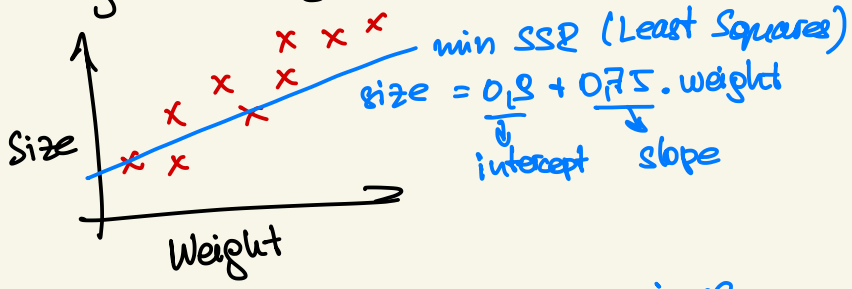


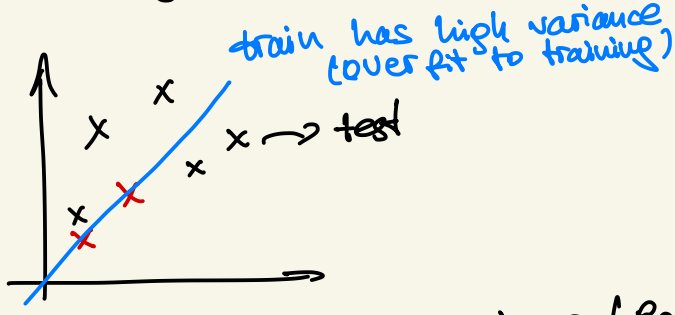

every person has own prediction → different residuals

- save residuals and build new tree, predict residuals
- take last predicted prob. from the chart for each sample
- when predicting: start at initial leaf and follow trees
- stop training when improvement diminish

# Regularization

## - Ridge (L2) Regression




introduce Bias to drop Variance (Better long-term predictions)

- Ridge regression minimizes $SSR + \lambda \cdot slope^2$ → penalty

Example

$size = 0,4 + 1,3 \cdot weight$ = perfect fit

$\Rightarrow SSR = 0$

$\lambda = 1$, $slope = 1,3$

$\Rightarrow 0 + 1 \cdot 1,3^2 = 1,69$ for SSR line

- minimize penalty to create Ridge Regression Line
- $\lambda$ determines how small the slope will be
- 10-fold cross validation to determine $\lambda$
- in log regression: sum of likelihoods + $\lambda \cdot slope^2$
- every parameter is scaled by $\lambda$
- More parameters:
  Size = y-intercept + slope 1 . Weight + slope 2 . Age
  ridge regression can solve for absent parameters with cross validation

## - Lasso (L1) Regression

- $SSR + \lambda \cdot |slope|$ $\lambda \in [0; +\infty)$ determined by cross-validation

size = y-intercept + slope . weight + diet difference . High Fat Diet

=> minimize: $SSR + \lambda \times (|slope| + |diet\ diff|)$

- lasso regression can shrink slope to 0 (ridge only approaches 0)
- eliminates variables without impact

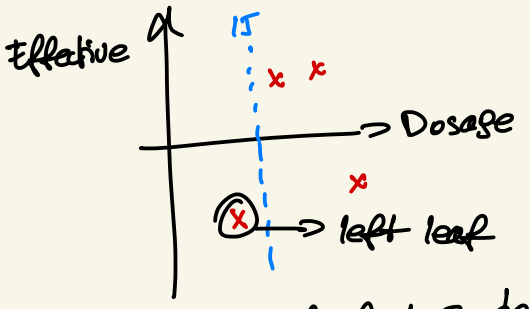
## - Elastic Net Regression

- combines Ridge and Lasso
- $SSR + \lambda_1 (|var_1| + |var_2| \dots) + \lambda_2 (var_1^2 + var_2^2 \dots)$

- use cross validation to find $\lambda_1 / \lambda_2$
- good if there is correlation between variables

## XGBoost Regression

- make initial prediction → 0,5
- calculate residuals (Observed − Predicted)
- fit XGBoost Tree to the Residuals
- start tree with single leaf (all residuals here)
- calculate Similarity Score = $\frac{(\sum \text{residuals})^2}{\#\text{residuals} + \lambda}$ → regularization param $= \frac{(-10{,}5 + 6{,}5 + 7{,}5 - 7{,}5)^2}{4+0}$ $= 4$




Dosage < 15

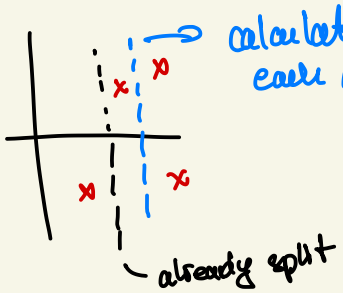
−10,5 | 6,5 | 7,5 | −7,5 → similarity = 14,08

similarity score = $\frac{(-10{,}5)^2}{1+0} = 110{,}25$ ($\lambda$)

- split on avg of first 2 dosages
- similar residuals = large score
- How much better leaves cluster similar residuals than root?

Gain = $\text{Left}_{\text{similarity}} + \text{Right}_{\text{similar}} - \text{Root}_{\text{similar}} = 120{,}33$

- shift threshold to between 2/3
  calculate similarity / Gain (for all remaining datapoints)
- choose threshold with largest Gain for the root




→ calculate similarity scores and gain for each available avg on the leaf

- repeat for other leaves (allow up to 6 levels)

### − Pruning XGBoost

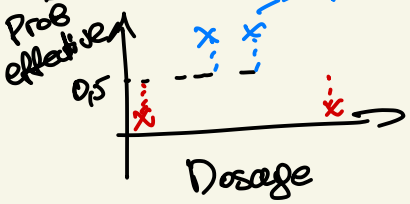
- pick number (130) = $\gamma$ (user defined for pruning)
- (lowest Branch Gain) − $\gamma$ if < 0 ⇒ remove Branch

  140,17 − 130 = 10,17 ⇒ Branch remains
- perform for upper Branches as long as last Branch was pruned

- Setting $\lambda$
  - set $\lambda$ to 1, recalculate similarity, similarities decrease
    decrease in similarity $\propto \frac{1}{\#\text{residuals}}$
  - larger $\lambda$ leads to smaller Gain, more pruning
  - if Gain is negative, pruning will be executed
  - Output value for a leaf = $\frac{\text{sum of Residuals}}{\#\text{residuals} + \lambda}$
  - when $\lambda > 0$, it reduces influence of observation on the output (sensitivity to individual observation)

- Predictions
  - start with initial prediction + Learning Rate · Output of tree
    ↳ $\varepsilon = 0{,}3$ default
    dosage $0{,}5 + 0{,}3 \cdot (-10{,}5) = -2{,}65$ effectiveness
  - takes small steps
  - Build new tree based on the new residuals
  - continue until residuals small, or max # trees is reached

XGBoost Classification




- make initial prediction (default = 0,5) 50% chance of effective
- fit tree to the residuals

$$\text{similarity score} = \frac{(\sum \text{Residual}_i)^2}{\sum [\text{Prev. Prob}_i \cdot (1 - \text{Prev Prob}_i)] + \lambda}$$ → regularization

- all residuals to the leaf, calculate similarity for the leaf
- try splitting values into 2 leaves, calculate similarity score for each leaf
- Calculate Gain = $\text{Left}_{\text{simil}} + \text{Right}_{\text{simil}} - \text{Root}_{\text{simil}}$
- if no other threshold gives us higher gain → chose this
- Build until reaching max level
- min number of residuals is determined by cover
  For classification: cover = $\sum [\text{PrevProb}_i \cdot (1 - \text{PrevProb}_i)] + \lambda$
  For regression: Cover = Number of Residuals

min value for cover = 1

- if Cover of the leaf < 1 => remove leaf

— Pruning
- pick $\gamma$
- if Gain - $\gamma$ < 0 => prune
- $\lambda > 0$ reduce the sensitivity to individual observations

— Convert into probability

$\frac{p}{1-p}$ = odds  $p = 0.5 \Rightarrow \log(\text{odds}) = 0$

$\log\left(\frac{p}{1-p}\right) = \log(\text{odds})$

- Output of a leaf  $\text{Output} = \frac{(\sum \text{Residual}_i)}{\sum[(\text{PrevProb}_i \cdot (1-\text{PrevProb}_i)] + \lambda}$

- Initial prediction: $\log\left(\frac{p}{1-p}\right)$
- Initial prediction + learning rate · Tree output value = $\log(\text{odds})$
  $= \Sigma$

  → $\text{Prob} = \frac{e^{\log(\text{odds})}}{1 + e^{\log(\text{odds})}}$