

# BI-VZD přednáška 3

Alexander Kovalenko

FIT ČVUT

28. 02. 2022

Authors: Karel Klouda, Daniel Vašata, Alexander Kovalenko

Please report any problems, suggestions etc. in [GitLab](#).

File version: 28. února 2022 09:26.

## Co bude v dnešní přednášce

- ensemble metody – základní myšlenka
- Bagging - náhodné lesy
- Boosting - AdaBoost

## Ensemble metody: základní myšlenka

- Základní myšlenka spočívá v tom, že namísto jednoho modelu (např. rozhodovacího stromu) použijeme **více modelů** a jejich predikce nějakým způsobem zkombinujeme do finálního rozhodnutí.

## Ensemble metody: základní myšlenka

- Základní myšlenka spočívá v tom, že namísto jednoho modelu (např. rozhodovacího stromu) použijeme **více modelů** a jejich predikce nějakým způsobem zkombinujeme do finálního rozhodnutí.
- My si ukážeme dva nejobvyklejší způsoby: **Bagging = bootstrap aggregating** a **Boosting**.

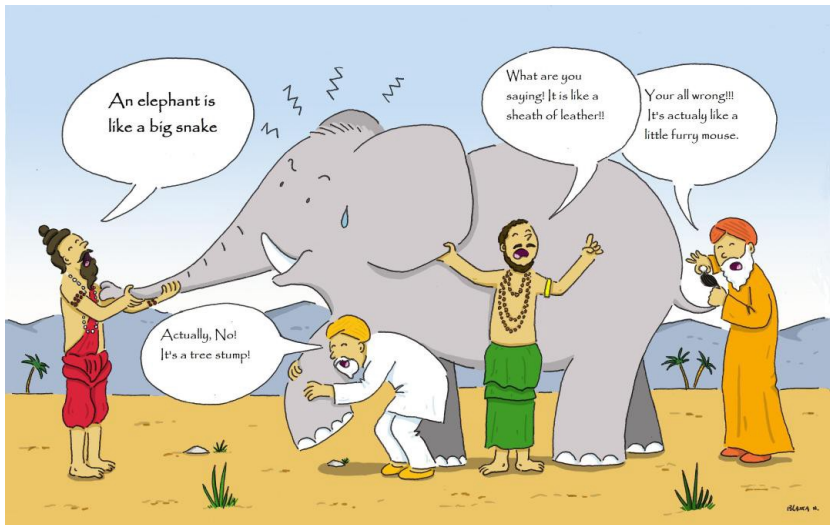
## Ensemble metody: základní myšlenka

- Základní myšlenka spočívá v tom, že namísto jednoho modelu (např. rozhodovacího stromu) použijeme **více modelů** a jejich predikce nějakým způsobem zkombinujeme do finálního rozhodnutí.
- My si ukážeme dva nejobvyklejší způsoby: **Bagging = bootstrap aggregating** a **Boosting**.
- Každý z těchto dvou přístupů si ilustrujeme na nejznámějších reprezentantech, které navíc spočívají ve skládání rozhodovacích stromů, které už známe z minulé přednášky.

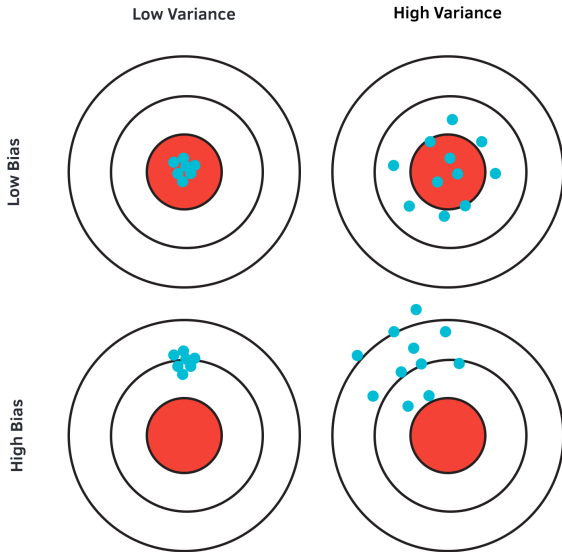
## Ensemble metody: základní myšlenka

- Základní myšlenka spočívá v tom, že namísto jednoho modelu (např. rozhodovacího stromu) použijeme **více modelů** a jejich predikce nějakým způsobem zkombinujeme do finálního rozhodnutí.
- My si ukážeme dva nejobvyklejší způsoby: **Bagging = bootstrap aggregating** a **Boosting**.
- Každý z těchto dvou přístupů si ilustrujeme na nejznámějších reprezentantech, které navíc spočívají ve skládání rozhodovacích stromů, které už známe z minulé přednášky.
- Tyto dva reprezentanti jsou **Náhodné lesy** (angl. **Random Forest**) a **AdaBoost**.

## Proč ensemble metoda funguje?

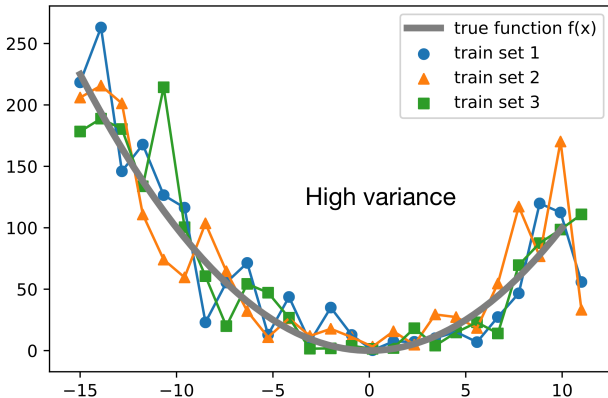


# Proč ensemble metoda funguje?





## Proč ensemble metoda funguje?



## Náhodný les pro klasifikaci: základní myšlenka

Pro jednoduchost předpokládejme, že máme binární klasifikační problém, tj. rozhodujeme jestli  $Y = 0$  nebo  $Y = 1$ .

1. Ze vstupního trénovacího datasetu  $\mathcal{D}$  vytvoříme  $n$  datasetů  $\mathcal{D}_1, \dots, \mathcal{D}_n$  stejně velkých jako  $\mathcal{D}$  pomocí metody **bootstrap**, neboli (středoškolsky) pomocí výběru *s opakováním*.

## Náhodný les pro klasifikaci: základní myšlenka

Pro jednoduchost předpokládejme, že máme binární klasifikační problém, tj. rozhodujeme jestli  $Y = 0$  nebo  $Y = 1$ .

1. Ze vstupního trénovacího datasetu  $\mathcal{D}$  vytvoříme  $n$  datasetů  $\mathcal{D}_1, \dots, \mathcal{D}_n$  stejně velkých jako  $\mathcal{D}$  pomocí metody **bootstrap**, neboli (středoškolsky) pomocí výběru *s opakováním*.
2. Na každém datasetu  $\mathcal{D}_i$  naučíme rozhodovací strom tak, jak jsme si předvedli v minulé přednášce. Může být málo hluboký, klidně hloubky (parametr `max_depth`) dva nebo tři (používá se i hloubka jedna). Označme tyto stromy  $T_1, \dots, T_n$ .

## Náhodný les pro klasifikaci: základní myšlenka

Pro jednoduchost předpokládejme, že máme binární klasifikační problém, tj. rozhodujeme jestli  $Y = 0$  nebo  $Y = 1$ .

1. Ze vstupního trénovacího datasetu  $\mathcal{D}$  vytvoříme  $n$  datasetů  $\mathcal{D}_1, \dots, \mathcal{D}_n$  stejně velkých jako  $\mathcal{D}$  pomocí metody **bootstrap**, neboli (středoškolsky) pomocí výběru *s opakováním*.
2. Na každém datasetu  $\mathcal{D}_i$  naučíme rozhodovací strom tak, jak jsme si předvedli v minulé přednášce. Může být málo hluboký, klidně hloubky (parametr `max_depth`) dva nebo tři (používá se i hloubka jedna). Označme tyto stromy  $T_1, \dots, T_n$ .
3. Každý datový bod (tj. řádek z tabulky s daty  $\mathcal{D}$ ) proženeme všemi stromy  $T_1, \dots, T_n$  a od každého z nich si uložíme rozhodnutí  $Y_1, \dots, Y_n$ .

## Náhodný les pro klasifikaci: základní myšlenka

Pro jednoduchost předpokládejme, že máme binární klasifikační problém, tj. rozhodujeme jestli  $Y = 0$  nebo  $Y = 1$ .

1. Ze vstupního trénovacího datasetu  $\mathcal{D}$  vytvoříme  $n$  datasetů  $\mathcal{D}_1, \dots, \mathcal{D}_n$  stejně velkých jako  $\mathcal{D}$  pomocí metody **bootstrap**, neboli (středoškolsky) pomocí výběru *s opakováním*.
2. Na každém datasetu  $\mathcal{D}_i$  naučíme rozhodovací strom tak, jak jsme si předvedli v minulé přednášce. Může být málo hluboký, klidně hloubky (parametr `max_depth`) dva nebo tři (používá se i hloubka jedna). Označme tyto stromy  $T_1, \dots, T_n$ .
3. Každý datový bod (tj. řádek z tabulky s daty  $\mathcal{D}$ ) proženeme všemi stromy  $T_1, \dots, T_n$  a od každého z nich si uložíme rozhodnutí  $Y_1, \dots, Y_n$ .
4. Všechny tyto stromy  $T_1, \dots, T_n$  tvoří **náhodný les** a jeho finální rozhodnutí o hodnotě  $Y$  je dané většinovým rozhodnutím stromů, je-li např. v množině  $\{Y_1, \dots, Y_n\}$  více jedniček než nul, je predikce náhodného lesa  $Y = 1$ .

## Bootstrap

Ukážeme si, jak funguje *bootstrap* na jednoduchém příkladu a našem datasetu:

id	rýmička	pohlaví	> 39°C	vstal(a)?	věk
1	ano	muž	ne	ne	65
2	ne	žena	ano	ano	34
3	ne	muž	ne	ano	72
4	ano	žena	ano	ne	20
5	ano	muž	ne	ano	45

## Bootstrap

Ukážeme si, jak funguje *bootstrap* na jednoduchém příkladu a našem datasetu:

id	rýmička	pohlaví	> 39°C	vstal(a)?	věk
1	ano	muž	ne	ne	65
2	ne	žena	ano	ano	34
3	ne	muž	ne	ano	72
4	ano	žena	ano	ne	20
5	ano	muž	ne	ano	45

Chceme-li vytvořit „bootstrapem“ dataset velikosti pět, pětkrát si **náhodně vybereme řádek s tabulky** s tím, že se řádky v našem výběru **mohou opakovat**. Vybereme-li např. řádky s id 1,4,3,3,1, dostaneme dataset

id	rýmička	pohlaví	> 39°C	vstal(a)?	věk
1	ano	muž	ne	ne	65
2	ne	žena	ano	ano	34
3	ne	muž	ne	ano	72
1	ano	muž	ne	ne	65
3	ne	muž	ne	ano	72

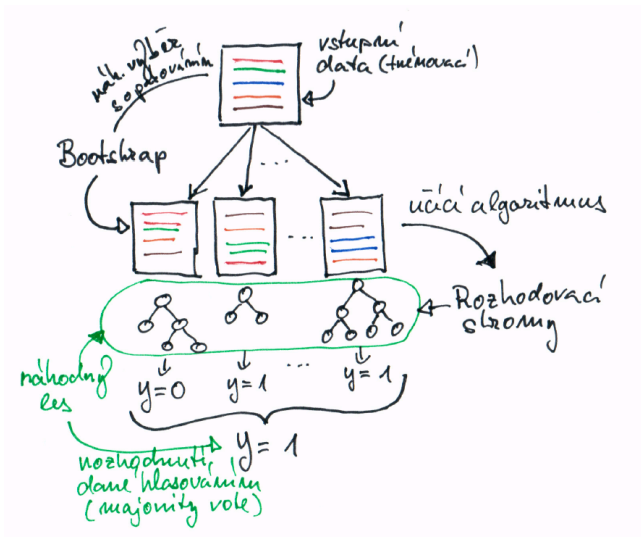
## Bootstrap

Ted' musíme vytvořit rozhodovací strom, který vezme v úvahu pouze omezenou podmnožinu proměnných. Příznaky mohou být vybrány náhodně. Přičemž u velkých datastetu počet příznaků v stromu je **druhá odmocnina** počtu příznaků v původním datasetu plus-minus autobus pro každý krok.

id	rýmička	pohlaví	> 39°C	vstal(a)?	věk
1	ano	muž	ne	ne	65
2	ne	žena	ano	ano	34
3	ne	muž	ne	ano	72
1	ano	muž	ne	ne	65
3	ne	muž	ne	ano	72



# Náhodný les pro klasifikaci: základní myšlenka na obrázku



## Poznámky (1/2)

- Základními parametry metod `RandomForestClassifier` a `RandomForestRegressor` v `sklearn` jsou:

## Poznámky (1/2)

- Základními parametry metod `RandomForestClassifier` a `RandomForestRegressor` v `sklearn` jsou:
  - ▶ `n_estimators`: určuje počet stromů v náhodném lese,

## Poznámky (1/2)

- Základními parametry metod `RandomForestClassifier` a `RandomForestRegressor` v `sklearn` jsou:
  - ▶ `n_estimators`: určuje počet stromů v náhodném lese,
  - ▶ `max_depth`: určuje maximální hloubku stromů v lese (je lepší použít nízkou hodnotu).

## Poznámky (1/2)

- Základními parametry metod `RandomForestClassifier` a `RandomForestRegressor` v `sklearn` jsou:
  - ▶ `n_estimators`: určuje počet stromů v náhodném lese,
  - ▶ `max_depth`: určuje maximální hloubku stromů v lese (je lepší použít nízkou hodnotu).
- Tyto metody mají také mnoho shodných parametrů s analogickými metodami pro jednoduché rozhodovací stromy; tyto parametry se přenáší na jednotlivé stromy v náhodném lese.

## Poznámky (1/2)

- Základními parametry metod `RandomForestClassifier` a `RandomForestRegressor` v `sklearn` jsou:
  - ▶ `n_estimators`: určuje počet stromů v náhodném lese,
  - ▶ `max_depth`: určuje maximální hloubku stromů v lese (je lepší použít nízkou hodnotu).
- Tyto metody mají také mnoho shodných parametrů s analogickými metodami pro jednoduché rozhodovací stromy; tyto parametry se přenáší na jednotlivé stromy v náhodném lese.
- V případě regresního problému (spojité  $Y$ ) se postupuje opět analogicky jako u regresních rozhodovacích stromů: predikce náhodného lesa se bere jako průměr z predikcí jednotlivých stromů lesa.

## Poznámky (2/2)

- U ensemble metod, které skládají rozhodnutí z více modelů, je důležité, aby jednotlivé modely nebyly stejné ale naopak co nejpestřejší.

## Poznámky (2/2)

- U ensemble metod, které skládají rozhodnutí z více modelů, je důležité, aby jednotlivé modely nebyly stejné ale naopak co nejpestřejší.
- Toho se většinou docílí nějakým druhem „randomizace“; v případě náhodných lesů je tento náhodný prvek daný bootstrap metodou pro generování jednotlivých trénovacích datasetů.



## Poznámky (2/2)

- U ensemble metod, které skládají rozhodnutí z více modelů, je důležité, aby jednotlivé modely nebyly stejné ale naopak co nejpestřejší.
- Toho se většinou docílí nějakým druhem „randomizace“; v případě náhodných lesů je tento náhodný prvek daný bootstrap metodou pro generování jednotlivých trénovacích datasetů.
- Jak jsme si řekli, rozhodovací stromy jsou velice citlivé na změny v trénovacích datech, a tak stačí odlišnost datasetů daná bootstrapem, abychom získávali velice odlišné rozhodovací stromy.

## Poznámky (2/2)

- U ensemble metod, které skládají rozhodnutí z více modelů, je důležité, aby jednotlivé modely nebyly stejné ale naopak co nejpestřejší.
- Toho se většinou docílí nějakým druhem „randomizace“; v případě náhodných lesů je tento náhodný prvek daný bootstrap metodou pro generování jednotlivých trénovacích datasetů.
- Jak jsme si řekli, rozhodovací stromy jsou velice citlivé na změny v trénovacích datech, a tak stačí odlišnost datasetů daná bootstrapem, abychom získávali velice odlišné rozhodovací stromy.
- Přestože mohou být jednotlivé stromy sami o sobě slabé modely (podmodelům v ensemble metodách se říká *weak learners*), jejich kolektivní rozhodování dává až překvapivě dobré výsledky.

## Poznámky (2/2)

- U ensemble metod, které skládají rozhodnutí z více modelů, je důležité, aby jednotlivé modely nebyly stejné ale naopak co nejpestřejší.
- Toho se většinou docílí nějakým druhem „randomizace“; v případě náhodných lesů je tento náhodný prvek daný bootstrap metodou pro generování jednotlivých trénovacích datasetů.
- Jak jsme si řekli, rozhodovací stromy jsou velice citlivé na změny v trénovacích datech, a tak stačí odlišnost datasetů daná bootstrapem, abychom získávali velice odlišné rozhodovací stromy.
- Přestože mohou být jednotlivé stromy sami o sobě slabé modely (podmodelům v ensemble metodách se říká *weak learners*), jejich kolektivní rozhodování dává až překvapivě dobré výsledky.
- Náhodné lesy jsou na rozdíl od rozhodovacích stromů velice robustní a poměrně odolné vůči přeučení. Bohužel ale ztrácejí jejich jednoduchost a snadnou interpretovatelnost.

## Rozhodovací stromy s `sample_weight` (1/2)

- Abychom mohli vysvětlit, jak funguje algoritmus AdaBoost, potřebujeme pochopit použití vážených dat u rozhodovacích stromů.

Rozhodovací stromy s `sample_weight` (1/2)

- Abychom mohli vysvětlit, jak funguje algoritmus AdaBoost, potřebujeme pochopit použití vážených dat u rozhodovacích stromů.
- V řeči `sklearn` se jedná o použití parametru `sample_weight` v metodě `.fit(X,y,sample_weight)` tříd `DecisionTreeClassifier` a `DecisionTreeRegressor`.

Rozhodovací stromy s `sample_weight` (1/2)

- Abychom mohli vysvětlit, jak funguje algoritmus AdaBoost, potřebujeme pochopit použití vážených dat u rozhodovacích stromů.
- V řeči `sklearn` se jedná o použití parametru `sample_weight` v metodě `.fit(X,y,sample_weight)` tříd `DecisionTreeClassifier` a `DecisionTreeRegressor`.
- Pokud máme v trénovacích datech  $N$  dat (řádků v tabulce, poloangl. „samplů“), je proměnná `sample_weight` pole  $[w_0, w_1, \dots, w_{N-1}]$  vah jednotlivých bodů, obvykle **znormovaných na jedničku**, tj.

$$\sum_{i=0}^{N-1} w_i = 1.$$

Rozhodovací stromy s `sample_weight` (2/2)

- Při učení stromu se váhy projeví v kroku, ve kterém se počítá informační zisk (resp. Gini index)

$$H(\mathcal{D}) - t_L H(\mathcal{D}_L) - t_R H(\mathcal{D}_R)$$

$$\text{kde } t_L = \frac{\#\mathcal{D}_L}{\#\mathcal{D}} \text{ a } t_R = \frac{\#\mathcal{D}_R}{\#\mathcal{D}}.$$

**Výsledkem použití vah je to, že strom se učí tak, aby správně predikoval zejména datové body s vyšší vahou. (Toto si rozmyslete!)**

Rozhodovací stromy s `sample_weight` (2/2)

- Při učení stromu se váhy projeví v kroku, ve kterém se počítá informační zisk (resp. Gini index)

$$H(\mathcal{D}) - t_L H(\mathcal{D}_L) - t_R H(\mathcal{D}_R)$$

kde  $t_L = \frac{\#\mathcal{D}_L}{\#\mathcal{D}}$  a  $t_R = \frac{\#\mathcal{D}_R}{\#\mathcal{D}}$ .

- Konkrétně se přepočítá hodnota  $t_L$  a  $t_R$ , tedy velikosti množin dat vzniklých rozdělením  $\mathcal{D}$ : Hodnota  $t_L$  je rovna součtu vah bodů, které spadají do  $\mathcal{D}_L$ , poděleném sumou vah všech bodů z  $\mathcal{D}$ . Analogicky pro  $t_R$ .

**Výsledkem použití vah je to, že strom se učí tak, aby správně predikoval zejména datové body s vyšší vahou. (Toto si rozmyslete!)**



Rozhodovací stromy s `sample_weight` (2/2)

- Při učení stromu se váhy projeví v kroku, ve kterém se počítá informační zisk (resp. Gini index)

$$H(\mathcal{D}) - t_L H(\mathcal{D}_L) - t_R H(\mathcal{D}_R)$$

$$\text{kde } t_L = \frac{\#\mathcal{D}_L}{\#\mathcal{D}} \text{ a } t_R = \frac{\#\mathcal{D}_R}{\#\mathcal{D}}.$$

- Konkrétně se přepočítá hodnota  $t_L$  a  $t_R$ , tedy velikosti množin dat vzniklých rozdělením  $\mathcal{D}$ : Hodnota  $t_L$  je rovna součtu vah bodů, které spadají do  $\mathcal{D}_L$ , poděleném sumou vah všech bodů z  $\mathcal{D}$ . Analogicky pro  $t_R$ .
- Jsou-li např. v  $\mathcal{D}_L$  body z řádků s indexy 3, 10, 15, 25 a v  $\mathcal{D}$  body s indexy 3, 10, 15, 17, 21, 25, je hodnota

$$t_L = \frac{w_3 + w_{10} + w_{15} + w_{25}}{w_3 + w_{10} + w_{15} + w_{17} + w_{21} + w_{25}}$$

**Výsledkem použití vah je to, že strom se učí tak, aby správně predikoval zejména datové body s vyšší vahou. (Toto si rozmyslete!)**

## AdaBoost (Adaptive Boosting): základní myšlenka

- Stejně jako při Baggingu konstruueme více modelů (opět uvažujeme rozhodovací stromy, i když AdaBoost může používat i jiné typy modelů) a finální rozhodnutí je (váženou) kompozicí rozhodnutí jednotlivých modelů.

## AdaBoost (Adaptive Boosting): základní myšlenka

- Stejně jako při Baggingu konstruueme více modelů (opět uvažujeme rozhodovací stromy, i když AdaBoost může používat i jiné typy modelů) a finální rozhodnutí je (váženou) kompozicí rozhodnutí jednotlivých modelů.
- Na rozdíl od Baggingu ale při Boostingu nejsou tyto modely nezávislé, ale jsou seřazené a každý další je ovlivněn těmi předchozími.

## AdaBoost (Adaptive Boosting): základní myšlenka

- Stejně jako při Baggingu konstruueme více modelů (opět uvažujeme rozhodovací stromy, i když AdaBoost může používat i jiné typy modelů) a finální rozhodnutí je (váženou) kompozicí rozhodnutí jednotlivých modelů.
- Na rozdíl od Baggingu ale při Boostingu nejsou tyto modely nezávislé, ale jsou seřazené a každý další je ovlivněn těmi předchozími.
- Tento vliv je při AdaBoostu realizován pomocí vah datových bodů: **Při konstrukci  $n$ tého stromu je zvýšena váha těm bodům, které předchozí  $(n - 1)$ tý strom klasifikoval špatně.**

## AdaBoost (Adaptive Boosting): základní myšlenka

- Stejně jako při Baggingu konstruuujeme více modelů (opět uvažujeme rozhodovací stromy, i když AdaBoost může používat i jiné typy modelů) a finální rozhodnutí je (váženou) kompozicí rozhodnutí jednotlivých modelů.
- Na rozdíl od Baggingu ale při Boostingu nejsou tyto modely nezávislé, ale jsou seřazené a každý další je ovlivněn těmi předchozími.
- Tento vliv je při AdaBoostu realizován pomocí vah datových bodů: **Při konstrukci  $n$ tého stromu je zvýšena váha těm bodům, které předchozí  $(n - 1)$ tý strom klasifikoval špatně.**
- Takovýmito změnami vah je zajištěno, že se další model soustředí více na ty datové body, se kterými si předchozí modely neporadily.

## AdaBoost (Adaptive Boosting): základní myšlenka

- Stejně jako při Baggingu konstruueme více modelů (opět uvažujeme rozhodovací stromy, i když AdaBoost může používat i jiné typy modelů) a finální rozhodnutí je (váženou) kompozicí rozhodnutí jednotlivých modelů.
- Na rozdíl od Baggingu ale při Boostingu nejsou tyto modely nezávislé, ale jsou seřazené a každý další je ovlivněn těmi předchozími.
- Tento vliv je při AdaBoostu realizován pomocí vah datových bodů: **Při konstrukci  $n$ tého stromu je zvýšena váha těm bodům, které předchozí  $(n - 1)$ tý strom klasifikoval špatně.**
- Takovýmito změnami vah je zajištěno, že se další model soustředí více na ty datové body, se kterými si předchozí modely neporadily.
- Přibližně takto funguje Boosting obecně, my si dále ukážeme konkrétní implementaci této myšlenky známou jako algoritmus AdaBoost [Freund, Schapire (1997)].

## AdaBoost (Adaptive Boosting): popis algoritmu (1/3)

- Na začátku máme dataset  $\mathcal{D}$  s  $N$  datovými body.
- Pro jednoduchost opět uvažujeme binární klasifikaci. Existují ale i modifikace algoritmu pro více než dvě třídy a i pro regresi.
- Počet zkonstruovaných stromů je zadán uživatelem v parametru `n_estimators`.

AdaBoost:

1. Nastavme váhy rovnoměrně, tedy  $w_i = \frac{1}{N}$  a položme  $m = 1$ .

## AdaBoost (Adaptive Boosting): popis algoritmu (1/3)

- Na začátku máme dataset  $\mathcal{D}$  s  $N$  datovými body.
- Pro jednoduchost opět uvažujeme binární klasifikaci. Existují ale i modifikace algoritmu pro více než dvě třídy a i pro regresi.
- Počet zkonstruovaných stromů je zadán uživatelem v parametru `n_estimators`.

AdaBoost:

1. Nastavme váhy rovnoměrně, tedy  $w_i = \frac{1}{N}$  a položme  $m = 1$ .
2. Pokud  $m \leq \text{n\_estimators}$ , naučme strom  $T^{(m)}$  na datech  $\mathcal{D}$  s váhami  $w_i$ .



## AdaBoost (Adaptive Boosting): popis algoritmu (1/3)

- Na začátku máme dataset  $\mathcal{D}$  s  $N$  datovými body.
- Pro jednoduchost opět uvažujeme binární klasifikaci. Existují ale i modifikace algoritmu pro více než dvě třídy a i pro regresi.
- Počet zkonstruovaných stromů je zadán uživatelem v parametru `n_estimators`.

AdaBoost:

1. Nastavme váhy rovnoměrně, tedy  $w_i = \frac{1}{N}$  a položme  $m = 1$ .
2. Pokud  $m \leq \text{n\_estimators}$ , naučme strom  $T^{(m)}$  na datech  $\mathcal{D}$  s váhami  $w_i$ .
3. Do proměnné  $e^{(m)}$  uložíme součet vah těch bodů z  $\mathcal{D}$ , které jsou špatně klasifikované stromem  $T^{(m)}$ .

## AdaBoost (Adaptive Boosting): popis algoritmu (1/3)

- Na začátku máme dataset  $\mathcal{D}$  s  $N$  datovými body.
- Pro jednoduchost opět uvažujeme binární klasifikaci. Existují ale i modifikace algoritmu pro více než dvě třídy a i pro regresi.
- Počet zkonstruovaných stromů je zadán uživatelem v parametru `n_estimators`.

AdaBoost:

1. Nastavme váhy rovnoměrně, tedy  $w_i = \frac{1}{N}$  a položme  $m = 1$ .
2. Pokud  $m \leq \text{n\_estimators}$ , naučme strom  $T^{(m)}$  na datech  $\mathcal{D}$  s váhami  $w_i$ .
3. Do proměnné  $e^{(m)}$  uložíme součet vah těch bodů z  $\mathcal{D}$ , které jsou špatně klasifikované stromem  $T^{(m)}$ .
4. Pokud je  $e^{(m)} = 0$  nebo  $e^{(m)} \geq \frac{1}{2}$ , skončíme.

## AdaBoost (Adaptive Boosting): popis algoritmu (2/3)

5. Pro stromem  $T^{(m)}$  špatně klasifikované body nastavme nové váhy

$$w_i \leftarrow \frac{1 - e^{(m)}}{e^{(m)}} w_i.$$

Díky předpokladu  $0 < e^{(m)} < 1/2$  je faktor  $\frac{1 - e^{(m)}}{e^{(m)}}$  větší než jedna a váhy těchto bodů se zvyšují!

**Výsledkem algoritmu** je tedy až  $n\_estimators$  rozhodovacích stromů  $T^{(1)}, T^{(2)}, \dots$

## AdaBoost (Adaptive Boosting): popis algoritmu (2/3)

5. Pro stromem  $T^{(m)}$  špatně klasifikované body nastavme nové váhy

$$w_i \leftarrow \frac{1 - e^{(m)}}{e^{(m)}} w_i.$$

Díky předpokladu  $0 < e^{(m)} < 1/2$  je faktor  $\frac{1 - e^{(m)}}{e^{(m)}}$  větší než jedna a váhy těchto bodů se zvyšují!

6. Znormalizujeme váhy tak, aby jejich součet byl jedna.

**Výsledkem algoritmu** je tedy až  $n\_estimators$  rozhodovacích stromů  $T^{(1)}, T^{(2)}, \dots$

## AdaBoost (Adaptive Boosting): popis algoritmu (2/3)

5. Pro stromem  $T^{(m)}$  špatně klasifikované body nastavme nové váhy

$$w_i \leftarrow \frac{1 - e^{(m)}}{e^{(m)}} w_i.$$

Díky předpokladu  $0 < e^{(m)} < 1/2$  je faktor  $\frac{1 - e^{(m)}}{e^{(m)}}$  větší než jedna a váhy těchto bodů se zvyšují!

6. Znormalizujeme váhy tak, aby jejich součet byl jedna.
7. Zvětšíme  $m$  o jedna a vraťme se do bodu 2.

**Výsledkem algoritmu** je tedy až  $n\_estimators$  rozhodovacích stromů  $T^{(1)}, T^{(2)}, \dots$

## AdaBoost (Adaptive Boosting): popis algoritmu (3/3)

Abychom určili rozhodnutí tohoto modelu pro nějaký datový bod  $x$ , postupujeme následovně:

1. Každému stromu  $T^{(m)}$  přiřadíme váhu

$$w_T^{(m)} = \text{learning\_rate} \cdot \log \frac{1 - e^{(m)}}{e^{(m)}},$$

kde  $e^{(m)}$  je číslo z kroku 4 výše a `learning_rate` je parametr zadáný uživatelem.

## AdaBoost (Adaptive Boosting): popis algoritmu (3/3)

Abychom určili rozhodnutí tohoto modelu pro nějaký datový bod  $x$ , postupujeme následovně:

1. Každému stromu  $T^{(m)}$  přiřadíme váhu

$$w_T^{(m)} = \text{learning\_rate} \cdot \log \frac{1 - e^{(m)}}{e^{(m)}},$$

kde  $e^{(m)}$  je číslo z kroku 4 výše a `learning_rate` je parametr zadaný uživatelem.

2. Sečti váhy  $w_T^{(m)}$  všech stromů, které pro  $x$  predikují  $Y = 1$  a to samé udělej pro stromy predikující  $Y = 0$ .

## AdaBoost (Adaptive Boosting): popis algoritmu (3/3)

Abychom určili rozhodnutí tohoto modelu pro nějaký datový bod  $x$ , postupujeme následovně:

1. Každému stromu  $T^{(m)}$  přiřadíme váhu

$$w_T^{(m)} = \text{learning\_rate} \cdot \log \frac{1 - e^{(m)}}{e^{(m)}},$$

kde  $e^{(m)}$  je číslo z kroku 4 výše a `learning_rate` je parametr zadaný uživatelem.

2. Sečti váhy  $w_T^{(m)}$  všech stromů, které pro  $x$  predikují  $Y = 1$  a to samé udělej pro stromy predikující  $Y = 0$ .
3. Rozhodni se pro tu z možností, pro kterou je součet vah vyšší.



## AdaBoost (Adaptive Boosting): poznámky

- Verze algoritmu pro „více než binární“ klasifikaci se nazývá *AdaBoost-SAMME* [Zhu, Rosset, Zou, Hastie (2006)]. Tato verze je implementována v `sklearn`.

## AdaBoost (Adaptive Boosting): poznámky

- Verze algoritmu pro „více než binární“ klasifikaci se nazývá *AdaBoost-SAMME* [Zhu, Rosset, Zou, Hastie (2006)]. Tato verze je implementována v `sklearn`.
- Varianta pro regresi se zase nazývá *AdaBoost.R2* [Drucker (1997)].

## AdaBoost (Adaptive Boosting): poznámky

- Verze algoritmu pro „více než binární“ klasifikaci se nazývá *AdaBoost-SAMME* [Zhu, Rosset, Zou, Hastie (2006)]. Tato verze je implementována v `sklearn`.
- Varianta pro regresi se zase nazývá *AdaBoost.R2* [Drucker (1997)].
- AdaBoost nemusí nutně používat rozhodovací stromy; je možné použít jakýkoli model, který umí pracovat s parametrem `sample_weight`.

## AdaBoost (Adaptive Boosting): poznámky

- Verze algoritmu pro „více než binární“ klasifikaci se nazývá *AdaBoost-SAMME* [Zhu, Rosset, Zou, Hastie (2006)]. Tato verze je implementována v `sklearn`.
- Varianta pro regresi se zase nazývá *AdaBoost.R2* [Drucker (1997)].
- AdaBoost nemusí nutně používat rozhodovací stromy; je možné použít jakýkoli model, který umí pracovat s parametrem `sample_weight`.
- V `sklearn` implementaci jsou rozhodovací stromy výchozí volba (parametr `base_estimator`) (stromy hloubky 1, pokud `base_estimator=None`).

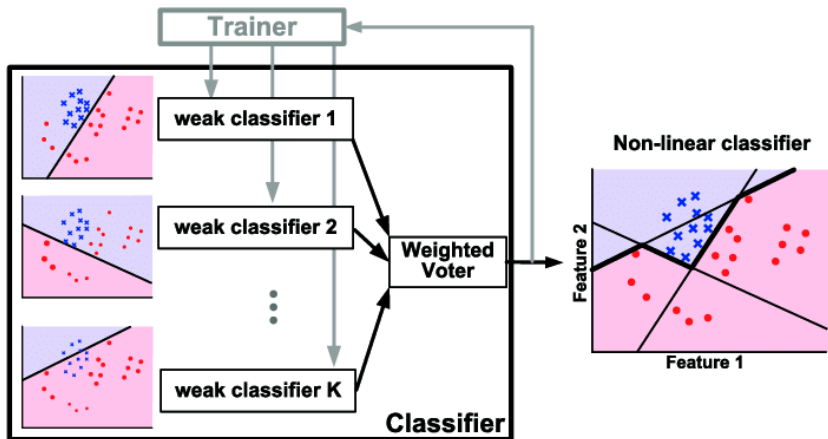
## AdaBoost (Adaptive Boosting): poznámky

- Verze algoritmu pro „více než binární“ klasifikaci se nazývá *AdaBoost-SAMME* [Zhu, Rosset, Zou, Hastie (2006)]. Tato verze je implementována v `sklearn`.
- Varianta pro regresi se zase nazývá *AdaBoost.R2* [Drucker (1997)].
- AdaBoost nemusí nutně používat rozhodovací stromy; je možné použít jakýkoli model, který umí pracovat s parametrem `sample_weight`.
- V `sklearn` implementaci jsou rozhodovací stromy výchozí volba (parametr `base_estimator`) (stromy hloubky 1, pokud `base_estimator=None`).
- Parametr `learning_rate` je obvykle zaváděn u většiny ensemble metod spadajících do kategorie Boostingu.

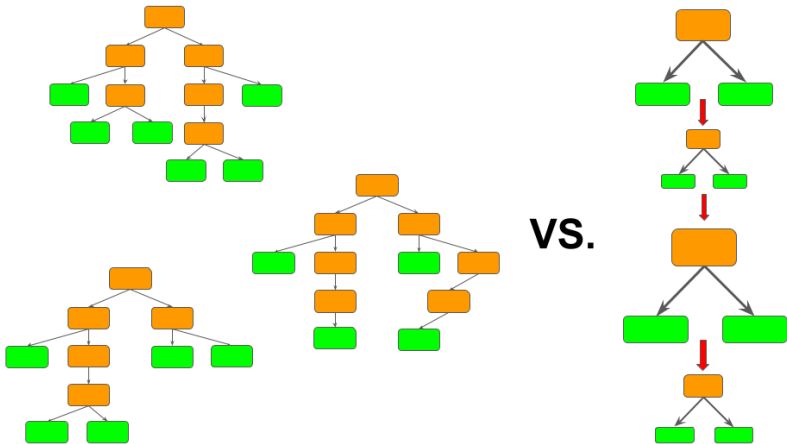
## AdaBoost (Adaptive Boosting): poznámky

- Verze algoritmu pro „více než binární“ klasifikaci se nazývá *AdaBoost-SAMME* [Zhu, Rosset, Zou, Hastie (2006)]. Tato verze je implementována v `sklearn`.
- Varianta pro regresi se zase nazývá *AdaBoost.R2* [Drucker (1997)].
- AdaBoost nemusí nutně používat rozhodovací stromy; je možné použít jakýkoli model, který umí pracovat s parametrem `sample_weight`.
- V `sklearn` implementaci jsou rozhodovací stromy výchozí volba (parametr `base_estimator`) (stromy hloubky 1, pokud `base_estimator=None`).
- Parametr `learning_rate` je obvykle zaváděn u většiny ensemble metod spadajících do kategorie Boostingu.
- Jedná se o tzv. **regularizaci**: čím je `learning_rate` nižší, tím je model odolnější vůči přeučení. Nevýhodou je, že je pak obvykle nutné zvýšit počet stromů (parametr `n_estimators`).

## Random Forest vs. AdaBoost



# Random Forest vs. AdaBoost





## Více o ensemble metodách:

- [click!](#)