BI-ML1.21 přednáška 2

Karel Klouda

FIT ČVUT

5. 10. 2023

Autoři: Karel Klouda, Daniel Vašata.
Problémy, návrhy apod. hlaste v GitLabu.
Verze souboru: 2. listopadu 2023 06:28.

Co bude v dnešní přednášce

- základní informace, úvodní příklad
- problém klasifikace
- jak fungují rozhodovací stromy
- jak se budují rozhodovací stromy
- hyperparametry a testovací, trénovací a validační data
- jak na spojité příznaky
- jak na spojitou vysvětlovanou proměnnou

Příklad: prenatální (1/4)

- S prvním *datovým modelem* jste se pravděpodobně setkali ještě v prenatálním období Vašeho života.
- Při odhadování porodní váhy plodu pomocí ultrazvuku (tzv. SONO) se používá mj. následující model:

Model pro porodní váhu (Shephard et al.)

$$\log_{10}(eFW) = -1.749 + 0.017 \cdot BPD + 0.005 \cdot AC - \frac{2.646}{1000} \cdot (BPD \cdot AC),$$

eFW = odhadovaná hmotnost při narození, BPD = biparietal diameter (příčný průměr hlavy), AC = abdominal circumference (obvod břicha).

- Pro nás to bude **lineární regresní model** (5. přednáška), kde je
  - porodní váha eFW tzv. **vysvětlovaná proměnná** (angl. **target variable**),
  - BPD, AC a $BPD \cdot AC$ jsou pak tři tzv. **příznaky** (anlg. a často i česky **features**),
  - čísla $1.749, 0.017$ atd. jsou takzvané **regresní koeficienty** (příp. váhy), obecně **parametry modelu**.

Příklad: prenatální (2/4)

Model pro porodní váhu (Shephard et al.)

$$\log_{10}(eFW) = -1.749 + 0.017 \cdot BPD + 0.005 \cdot AC - \frac{2.646}{1000} \cdot (BPD \cdot AC),$$

- **Jak se tento model používá?**
  1. Na ultrazvuku se změříte veličiny BPD a AC,
  2. získaná čísla dosadíte do pravé strany vzorce,
  3. výsledek je odhad hodnoty $\log_{10}(eFW)$ a tedy i kýžené porodní váhy eFW.

- **Kde se vzala ta divná čísla jako 2.646 apod.?**
  To se právě budeme učit v tomto kurzu: Jsou to parametry zvoleného modelu a ty se vždy nějakým způsobem odhadují na základě dostupných dat, to je tzv. **učení modelu**.

Příklad: prenatální (3/4)

Jak asi pan Shephard et al. došli právě k tomuto modelu:

1. Z nějakého důvodu věřili, že příznaky BPD a AC jsou pro porodní váhu důležité. Nejspíše ale zkoušeli i jiné, ale ty buď nepřinášely velké zlepšení nebo se daly BPD a AC plně nahradit.

2. Pomocí různých testovacích procedur si jako model zvolili lineární regresní model s třemi příznaky a čtyřmi koeficienty:

$$\log_{10}(eFW) = w_0 + w_1 \cdot BPD + w_2 \cdot AC + w_3 \cdot (BPD \cdot AC).$$

3. Předchozí fázi, kdy hledáme „tvar“ modelu, se říká **ladění hyperparametrů** (angl. **hyperparameter tuning**).

4. Koeficienty w_i pak odhadli z dat o již narozených dětech, u kterých měli přesnou porodní váhu i prenatálně naměřené hodnoty BPD a AC.

Příklad: prenatální (4/4)

Zmíněná data, na kterých se model učil (a testoval), mohla vypadat nějak takto:

`kid_id`	FW	BCD	AC
1	číslo	číslo	číslo
2	číslo	číslo	číslo
⋮	⋮	⋮	⋮

Pro zajímavost (detaily v 5. přednášce):

- Označme sloupec pod FW jako vektor $\boldsymbol{Y}$
- a vytvořme matici $\mathbf{X}$ o čtyřech sloupcích, kde v každém řádku je vektor $(1, BCD, AC, BCD \cdot AC)$.
- Nejpoužívanější metoda pro výpočet koeficientů w_i je **metoda nejmenších čtverců**, ta nám říká, že

$$(w_0, w_1, w_2, w_3)^T = (\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T\boldsymbol{Y}.$$

- Tento vzorec jsme získali vyřešením jistého optimalizačního problému (a.k.a. *hledání extrémů*), což je velice častý případ: **učení modelu = optimalizace**!

Supervizované vs. nesupervizované učení

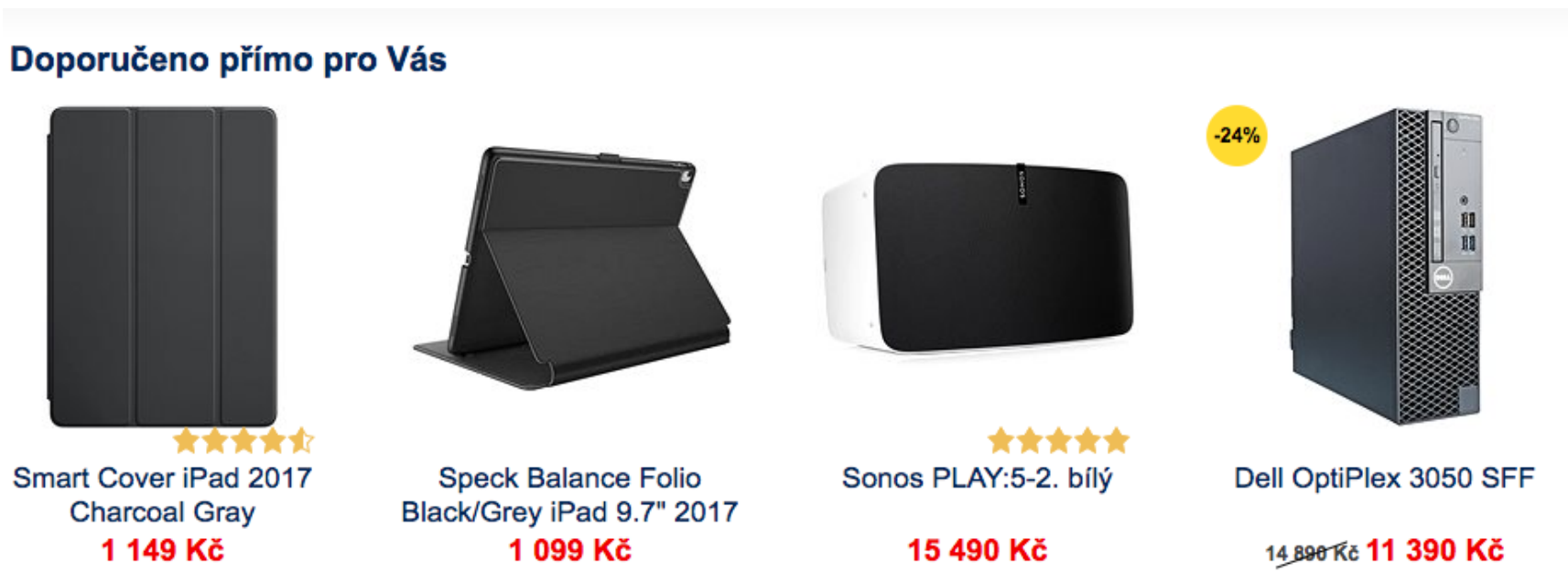
- Předchozí *prenatální* příklad je typickou ukázkou **supervizovaného učení** (učení s učitelem, angl. **supervized learning**).
- Tím „učitelem“ jsou zde známé hodnoty porodních vah u dětí, což je veličina, kterou se snažíme pomocí modelu *predikovat* resp. pochopit, na čem závisí.
- Někdy takovou veličinu ale ani nemáme a prostě se v datech pokoušíme nějak vyznat a najít jejich skrytou strukturu.
- Takovým problémům se říká **nesupervizované učení** (učení bez učitele) a typickým příkladem je **clusterování** dat (téma 11. přednášky).

Příklady nesupervizovaného učení

- Problém clusterování je velice obvyklý v praxi.
- Pokud máte například e-shop (nebo banku, nebo telefonního operátora), chcete se vyznat ve svých zákaznících, o kterých máte nasbíraná různá data (tzv. *customer segmentation*).
- Můžete tak hledat např. podmnožinu „nejlepších“ zákazníků, kterým má cenu věnovat speciální péči. Nebo naopak skupinu, která potřebuje k polepšení pomoci nějakou reklamní akcí (cílení reklamy je velký byznys).
- Do nesupervizovaného učení také (obvykle) spadá i **detekce anomálií** (angl. **anomaly detection**).
- Např. banka se snaží najít podezřelé transakce (fraud detection, ochrana proti zneužití karty, atp.).

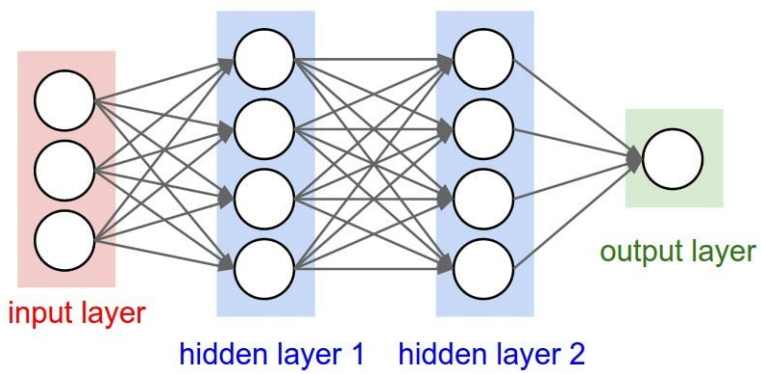
Další příklady: doporučovací systémy

- Dalším příkladem problému řešeného pomocí zkoumání dat je tzv. **doporučování** (angl. **recommendation**).
- Například: vlastníte-li e-shop (příp. internetový časopis, iTunes, Netflix atp.), snažíte se na základě dat o zákaznících a zejména zákazníkovi, který právě prohlíží Vaše stránky, odhadnout, co by si tak mohl ještě chtít koupit (přečíst, podívat, poslechnout) a to mu ukázat.



Další příklady: data bez jasných příznaků

- Data ale vždy nemusí mít formu tabulky. Může se jednat o obrázky, videa, časové řady, dlouhé texty atp., ze kterých je těžké získat pro modely příznaky.
- V takovém případě si musíte dát práci a nějaké příznaky z dat vydolovat (tzv. **feature extraction**).
- Nebo použijete algoritmy a metody, které si příznaky vytvářejí samy automaticky.
- Mezi takové metody patří (čím dál populárnější a mocnější) umělé **neuronové sítě** (angl. artificial **neural networks**, ANN)
- O ANN budeme mluvit v v BI-ML2. Používají se k všemožným úkolům (překlady, detekce objektů v obrázku videu, hraní GO, clusterování, detekci anomálií, . . .).



Co je problém klasifikace

- Supervizované učení: Snažíme se zjistit, jak vysvětlovanou proměnnou Y ovlivňují příznaky $X_0, X_1, \ldots, X_{p-1}$, hledáme tedy nějaký funkční vztah tak, aby „co nejvíce platilo“

$$Y \approx f(X_0, X_1, \ldots, X_{p-1}).$$

- Funkce f nemusí být nutně podobná funkcím, které znáte z analýzy. Např. v této přednášce to bude strom ϟ.
- Tvar hledané funkce často ovlivňuje to, jakých hodnot může nabývat vysvětlovaná proměnná Y:
  - Může-li nabývat jen několik málo hodnot, mluvíme o problému **klasifikace** (angl. **classification**). Sem spadá např. určení, jestli pacient má/nemá nemoc, jaké písmeno je (ručně) napsáno na obrázku, atp.
  - Může-li nabývat tolika hodnot, že je rozumnější ji považovat za *spojitou*, mluvíme o problému **regrese** (angl. **regression**).
- **Rozhodovací stromy** (angl. **decision trees**) lze použít pro oba typy problému: my začneme tím klasifikačním.

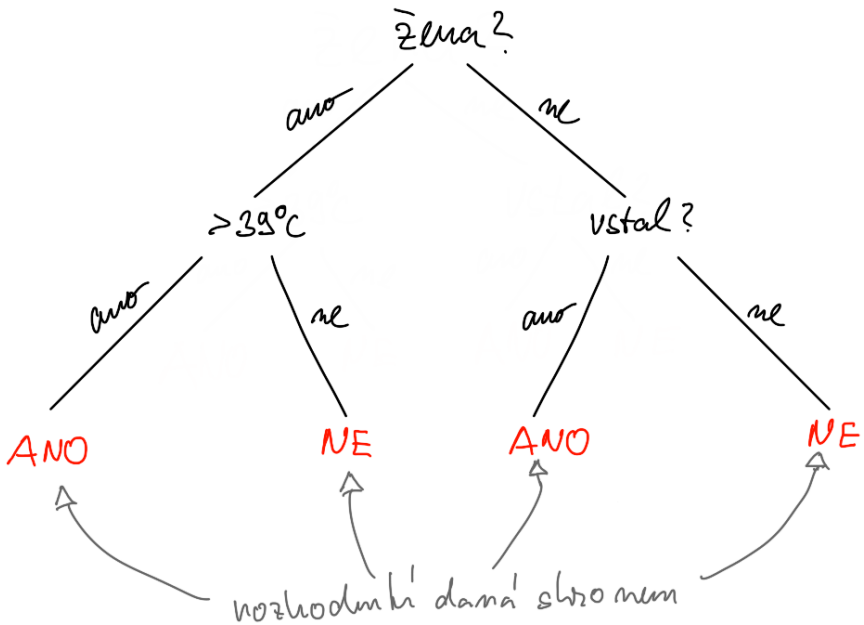
Ukázka použití rozhodovacího stromu (1/6)

- Velmi často je klasifikační problém **binární**, kdy proměnná Y může mít jen dvě hodnoty.
- My si použití stromu ukážeme na (vymyšlených) datech a problému určování, jestli pacient má či nemá závažnou nemoc známou jako „rýmička“.
- Příznaky budou pro jednoduchost také binární: Pohlaví (žena/muž), horečka ($> 39°\mathrm{C}/\leq 39°\mathrm{C}$) a to, jestli daný člověk zvládl/nezvládl vstát z postele.
- Ukážeme si dva rozhodovací stromy a porovnáme si, jak je který z nich dobrým modelem následujících dat:

rýmička	pohlaví	$> 39°\mathrm{C}$	vstal(a)?
ano	muž	ne	ne
ne	žena	ano	ano
ne	muž	ne	ano
ano	žena	ano	ne

Ukázka použití rozhodovacího stromu (2/6)

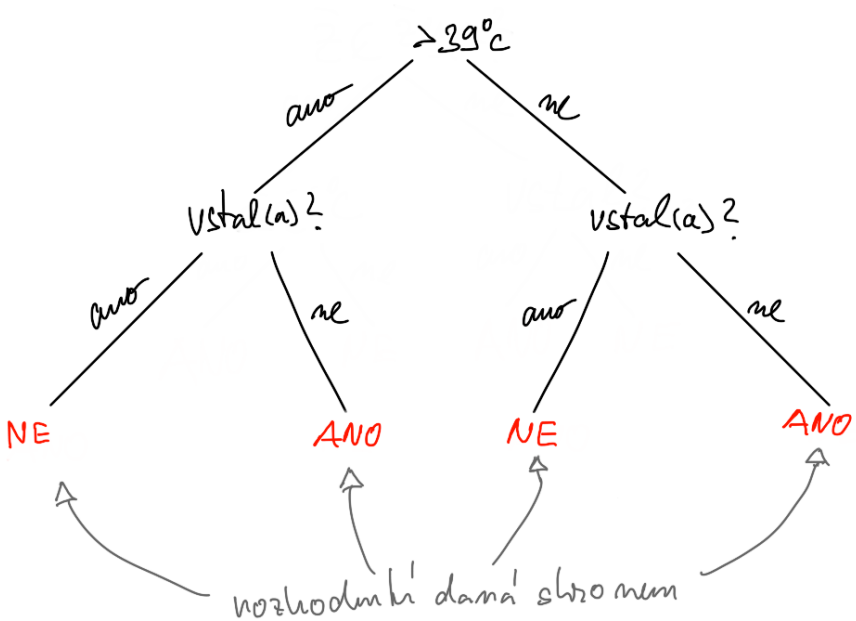
Strom 1:



rýmička	pohlaví	$> 39°\mathrm{C}$	vstal(a)?	co říká strom
ano	muž	ne	ne	ne
ne	žena	ano	ano	ano
ne	muž	ne	ano	ano
ano	žena	ano	ne	ano

Ukázka použití rozhodovacího stromu (3/6)

Strom 2:



rýmička	pohlaví	$> 39°\mathrm{C}$	vstal(a)?	co říká strom
ano	muž	ne	ne	ano
ne	žena	ano	ano	ne
ne	muž	ne	ano	ne
ano	žena	ano	ne	ano

Ukázka použití rozhodovacího stromu (4/6)

- Strom 1 dává špatný výsledek pro tři řádky, strom 2 dává správné výsledky pro všechny řádky.
- Strom 2 je tedy, zdá se, lepší. Je však toto dostatečné zdůvodnění?
- My ve skutečnosti chceme vědět, jak často se strom trefí **pro všechna možná data**.
- Což je trochu smůla, protože všechna možná data nikdy nemáme. Máme většinou jen „jednu tabulku“ dat a ta nám musí stačit jak pro vytvoření (neboli naučení) stromu, tak i pro ověření toho, jak je dobrý.
- Jak se to dělá, si ukážeme později.

Ukázka použití rozhodovacího stromu (5/6)

- Strom 1 i strom 2 jsou stromy hloubky 2 a mají tedy 4 listy.
- Kdybychom tedy vytvořili strom hloubky 3, měl by 8 listů. Přesně tolik je ale taky možných kombinací hodnot tří příznaků! **To už není model, ale index!**
- Strom hloubky tři by se tedy mýlil pouze v případě, že by hodnoty všech příznaků byly stejné, ale hodnota vysvětlované proměnné by byla jiná (např. kdyby jeden pacient byl chlapík s angínou):

rýmička	pohlaví	$> 39°$C	vstal(a)?
⋮	⋮	⋮	⋮
ano	muž	ano	ne
ne	muž	ano	ne
⋮	⋮	⋮	⋮

- V takovém případě by se mýlil libovolný strom a dokonce i jakákoli funkce příznaků (viz definici funkce).

Ukázka použití rozhodovacího stromu (6/6)

- Skutečným model rýmičky je, jak známo, toto: rýmička nastává právě když

$$(\text{žena} \ \wedge (> 39^\circ) \wedge \ \text{nevstala}) \vee (\text{muž} \ \wedge ((> 39^\circ) \vee \ \text{nevstal}))$$

- Takový model ale není postižitelný stromem hloubky dva! Vzpomeňme BI-MLO/BI-DML a minimalizaci formulí v disjunktivním normálním tvaru ...

Konstrukce stromu: základní úkoly

Postupně si ukážeme, jak se řeší následující problémy:

- Máme-li data s příznaky i s hodnotami vysvětlované proměnné, jak zkonstruuji rozhodovací strom, který data co nejlépe modeluje?
- Jak poznám, že můj vytvořený strom není jen dobrým modelem dat, která mám, ale bude dobře fungovat i pro data jiná?
- Jak poznám, jakou mám zvolit hloubku stromu příp. jiné jeho parametry?
- Jak si poradit s nebinárními, nebo dokonce se spojitými parametry?

Konstrukce stromu: formulace úlohy

- Na vstupu máme N řádkovou tabulku s hodnotami pro binární vysvětlovanou proměnnou a p binárních příznaků $X_0, X_1, \ldots, X_{p-1}$.
- Cílem je vytvořit strom zadané hloubky k, který správně přiřadí hodnotu Y co nejvíce řádkům z tabulky.
- Jak to vyřešit? Vyzkoušejme všechny stromy a pro každý změříme podíl správně určených a je hotovo!
- Nebo je to snad problém?
- Stromů hloubky 1 je p, stromů hloubky 2 je $p \cdot (p-1)^2$, stromů hloubky 3 je „fakt hodně“, ...
- Konstrukce hrubou silou je neprůchozí kvůli počtu možných stromů, ve skutečnosti je konstrukce optimálního stromu **NP-úplný** problém (viz [Hyafil, Rivest, (1976)]).

Konstrukce stromu: hladový ID3 algoritmus

- Pro konstrukci stromů se používá **hladový algoritmus** označovaný jako **ID3** (resp. jeho rafinovanější verze **C4.5** a **C5** vše od **Johna Rosse Quinlana**).
- Tyto algoritmy pro danou množinu dat vybírají jeden (ze zatím nepoužitých) příznaků, který rozdělí data na dvě části tak, že vzniklé rozdělení maximalizuje vybrané kritérium (viz dále).
- Daná množina dat je tak rozdělena na dvě části a na každou zvlášť je pak aplikován stejný postup, jehož výsledkem jsou další dva příznaky, použité jako kritérium a rozdělení na čtyři podmnožiny dat.
- Takto se postupuje, dokud nenastane nějaké zastavovací kritérium (maximální hloubka stromu, na listech stromu už je málo dat, atp.).
- Jak zvolit to kritérium? Používají se dvě, o obou si řekneme později.

Konstrukce stromu: ukázková data

Zkusme zkonstruovat strom pro následující binární data s třemi příznaky:

id	Y	X_0	X_1	X_2
0	1	1	0	0
1	1	0	1	1
2	1	1	0	0
3	1	1	1	1
4	0	0	0	1
5	0	0	1	0
6	0	0	0	1
7	0	1	1	0

? Jaký příznak použít jako první k rozdělení dat?

- Příznak X_0 rozdělí data na dvě části[1] $\{1_0, 1_2, 1_3, 0_7\}$ a $\{1_1, 0_4, 0_5, 0_6\}$.
- Příznak X_1 rozdělí data na dvě části $\{1_1, 1_3, 0_5, 0_7\}$ a $\{1_0, 1_2, 0_4, 0_6\}$.
- Příznak X_2 rozdělí data na dvě části $\{1_1, 1_3, 0_4, 0_6\}$ a $\{1_0, 1_2, 0_5, 0_7\}$.

[1] Uvádíme hodnotu Y a jako dolní index `id` přísl. řádku.

Konstrukce stromu: kritérium volby příznaku

- Příznak X_0 rozdělí data na dvě části $\{1_0, 1_2, 1_3, 0_7\}$ a $\{1_1, 0_4, 0_5, 0_6\}$.
- Příznak X_1 rozdělí data na dvě části $\{1_1, 1_3, 0_5, 0_7\}$ a $\{1_0, 1_2, 0_4, 0_6\}$.
- Příznak X_2 rozdělí data na dvě části $\{1_1, 1_3, 0_4, 0_6\}$ a $\{1_0, 1_2, 0_5, 0_7\}$.

? Který příznak je lepší a jak to změřit?

- Na vstupu jsou data $\{1_0, 1_1, 1_2, 1_3, 0_4, 0_5, 0_6, 0_7\}$, kde jsou hodnoty 0 a 1 zastoupeny rovnoměrně.
- Ideální by byl příznak, který data rozdělí na dvě skupiny, jednu se samými 1 a druhou s 0. Takový ale nemáme k dispozici.
- Příznaky X_1 a X_2 jsou ale opačný extrém: Data rozdělí na dva kusy, kde jsou opět 0 a 1 zastoupeny přesně napůl.
- Příznak X_0 pak představuje zřejmě nejlepší volbu, neb data alespoň trochu více „uspořádá“: z rovnoměrného zastoupení k poměru 1 ku 3.
- Jak tuto uspořádanost resp. neuspořádanost měřit?

Míra **ne**uspořádanosti

- Máme množinu $\mathcal{D}$ nul a jedniček (nebo i více hodnot) a chceme nějak změřit, jak moc je uspořádaná.
- Co by taková míra měla splňovat? Označme p_0 a p_1 poměry počtu 0 resp. 1 v množině (tj. $p_0 + p_1 = 1$).
  1. Míra by měla být nezáporná (z technických důvodů).
  2. Pokud jsou v množině např. samé nuly (tj. $p_0 = 1$), měla by být neuspořádanost nulová.
  3. Měla by být maximální, pokud jsou počty nul a jedniček stejné, tj. když $p_0 = p_1 = \frac{1}{2}$.
  4. Měla by to být rostoucí funkce p_0 na intervalu $[0, \frac{1}{2}]$ a klesající na intervalu $[\frac{1}{2}, 1]$.
- Taková funkce **měřící neuspořádanost** existuje a říká se jí **Entropie**:

$$H(\mathcal{D}) = -p_0 \log p_0 - p_1 \log p_1 = -p_0 \log p_0 - (1 - p_0) \log(1 - p_0).$$

- Pro nebinární případ s k různými hodnotami je to analogické:

$$H(\mathcal{D}) = -\sum_{i=0}^{k-1} p_i \log p_i.$$

Entropie: poznámky

- Formálněji se budete entropii věnovat v předmětu *NI-VSM: Vybrané statistické metody*, kde si zavedete entropii *náhodné veličiny*. Nám ale stačí předchozí intuitivní zavedení[2].
- Ve vzorci pro entropii budeme používat dvojkový logaritmus. V takovém případě se jednotce entropie říká **bit**.
- Entropie množiny našich vstupních dat $\mathcal{D} = \{1_0, 1_1, 1_2, 1_3, 0_4, 0_5, 0_6, 0_7\}$ je rovna 1, neboť $p_0 = \frac{1}{2}$, a tedy

$$H(\mathcal{D}) = -\frac{1}{2}\log\frac{1}{2} - \left(1-\frac{1}{2}\right)\log\left(1-\frac{1}{2}\right) = -2\frac{1}{2}\log\frac{1}{2} = -\log\frac{1}{2} = 1.$$

- Entropie množiny dat $\mathcal{D}_1 = \{1_0, 1_1, 1_3, 0_7\}$ je rovna ($p_0 = \frac{1}{4}$)

$$H(\mathcal{D}_1) = -\frac{1}{4}\log\frac{1}{4} - \left(1-\frac{1}{4}\right)\log\left(1-\frac{1}{4}\right) = 0.811278124459133 28\ldots$$

[2] To, co jsme zavedli, není přesně řečeno entropie, ale její odhad na základě dat. Skutečné pravděpodobnosti p_i totiž vlastně neznáme a používáme jen jejich odhad.

Konstrukce stromu: informační zisk

Vraťme se k otázce, který příznak použít pro rozdělení množiny dat $\mathcal{D} = \{1_0, 1_1, 1_2, 1_3, 0_4, 0_5, 0_6, 0_7\}$:

- Příznak X_0: $\mathcal{D}_1 = \{1_0, 1_2, 1_3, 0_7\}$ a $\mathcal{D}_0 = \{1_1, 0_4, 0_5, 0_6\}$.
- Příznak X_1: $\mathcal{D}_1 = \{1_1, 1_3, 0_5, 0_7\}$ a $\mathcal{D}_0 = \{1_0, 1_2, 0_4, 0_6\}$.
- Příznak X_2: $\mathcal{D}_1 = \{1_1, 1_3, 0_4, 0_6\}$ a $\mathcal{D}_0 = \{1_0, 1_2, 0_5, 0_7\}$.

Chceme vybrat příznak, který rozdělením dat nejvíce sníží neuspořádanost! Toto snížení se určuje tzv. **informačním ziskem** (angl. **information gain**), který je definován jaké „entropie $\mathcal{D}$ mínus vážený součet entropií $\mathcal{D}_0$ a $\mathcal{D}_1$“.

Formálně:

$$IG(\mathcal{D}, X_i) = H(\mathcal{D}) - t_0 H(\mathcal{D}_0) - t_1 H(\mathcal{D}_1)$$

kde $\mathcal{D}_0$ a $\mathcal{D}_1$ jsou podmnožiny dat $\mathcal{D}$, pro které $X_i = 0$ resp. $X_i = 1$, a t_i je podíl počtu prvků v $\mathcal{D}_i$ a $\mathcal{D}$, neboli $t_i = \frac{\#\mathcal{D}_i}{\#\mathcal{D}}$.

Konstrukce stromu pro ukázková data (1/4)

Spočítejme informační zisk pro naše tři příznaky:

- Příznak X_0: $\mathcal{D}_1 = \{1_0, 1_2, 1_3, 0_7\}$ a $\mathcal{D}_0 = \{1_1, 0_4, 0_5, 0_6\}$.
- Příznak X_1: $\mathcal{D}_1 = \{1_1, 1_3, 0_5, 0_7\}$ a $\mathcal{D}_0 = \{1_0, 1_2, 0_4, 0_6\}$.
- Příznak X_2: $\mathcal{D}_1 = \{1_1, 1_3, 0_4, 0_6\}$ a $\mathcal{D}_0 = \{1_0, 1_2, 0_5, 0_7\}$.
- Už víme, že $H(\mathcal{D}) = 1$.
- Pro X_0 dostáváme $H(\mathcal{D}_0) = H(\mathcal{D}_1) = 0.811278124459133 8$ a tedy

$$IG(\mathcal{D}, X_0) = H(\mathcal{D}) - \frac{1}{2}H(\mathcal{D}_0) - \frac{1}{2}H(\mathcal{D}_1) = 1 - 0.811 = 0.189.$$

- Pro X_1 s X_2 postupujeme podobně. Pro ně platí $H(\mathcal{D}_0) = H(\mathcal{D}_1) = 1$ a tedy

$$IG(\mathcal{D}, X_1) = IG(\mathcal{D}, X_2) = H(\mathcal{D}) - \frac{1}{2}H(\mathcal{D}_0) - \frac{1}{2}H(\mathcal{D}_1) = 1 - 1 = 0.$$

Vítězem hladového souboje je tedy příznak X_0.

Konstrukce stromu pro ukázková data (2/4)

- Příznak X_0: $\mathcal{D}_1 = \{1_0, 1_2, 1_3, 0_7\}$ a $\mathcal{D}_0 = \{1_1, 0_4, 0_5, 0_6\}$.
- Nyní aplikujeme stejný postup dvakrát: na $\mathcal{D}_1$ a na $\mathcal{D}_0$. Nyní už ale nemá smysl dělit data podle X_0, takže budeme zkoušet jen X_1 a X_2.
- Data $\mathcal{D}_1$ příznak X_1 rozdělí na $\mathcal{D}_{11} = \{1_3, 0_7\}$ a $\mathcal{D}_{10} = \{1_0, 1_2\}$, informační zisk je tedy

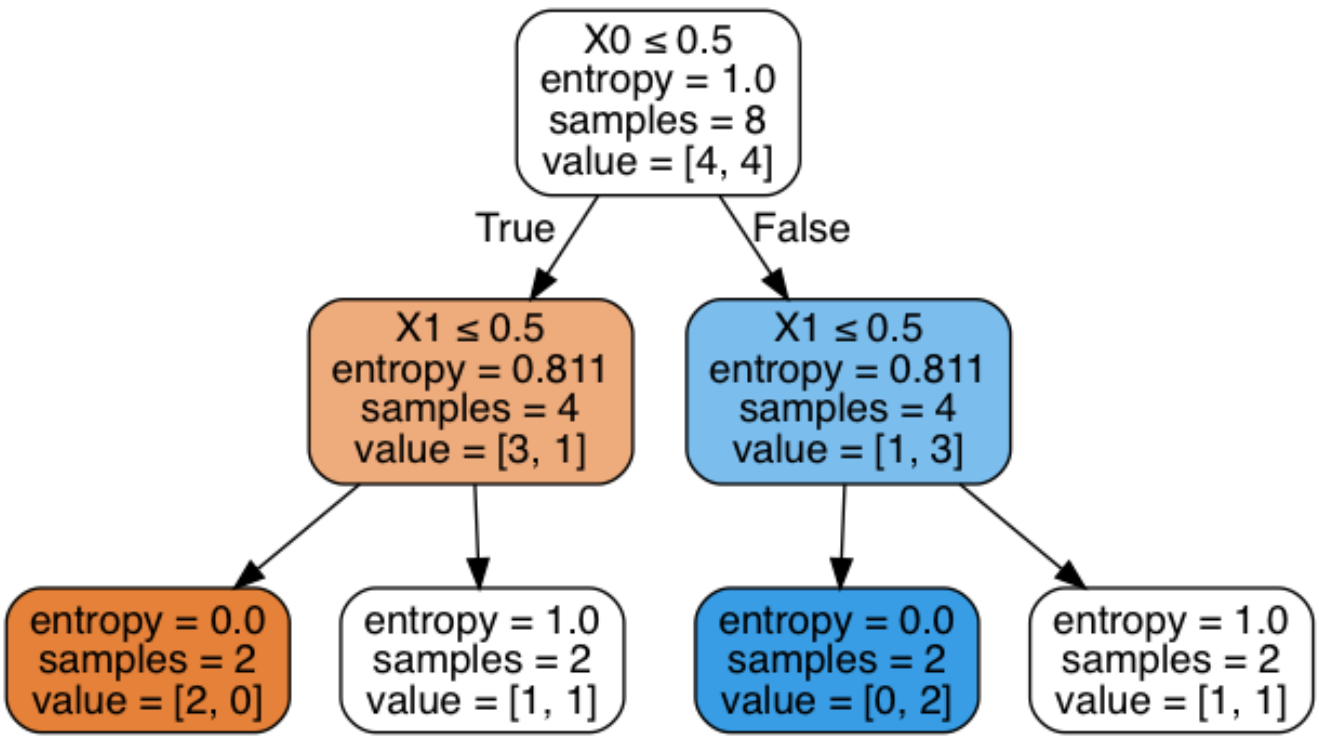
$$IG(\mathcal{D}_1, X_1) = H(\mathcal{D}_1) - \frac{1}{2}H(\mathcal{D}_{11}) - \frac{1}{2}H(\mathcal{D}_{10}) = 0.811 - \frac{1}{2}1 - \frac{1}{2}0 = 0.311\,.$$

- Pro příznak X_2 vyjde informační zisk nižší (zhruba 0.123, spočítejte si), takže hladový souboj vyhrává příznak X_1.

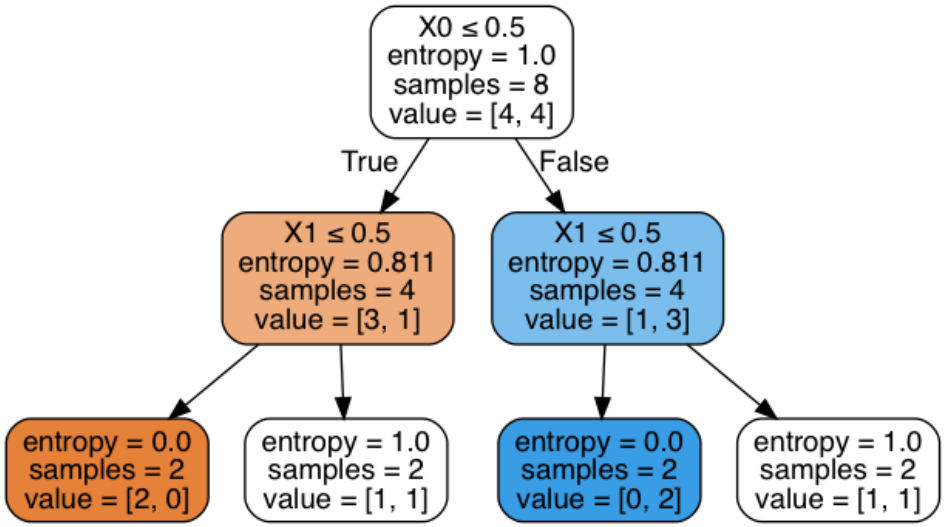
Konstrukce stromu pro ukázková data (3/4)

Takto přesně postupuje i implementace konstruování rozhodovacích stromů v knihovně `sklearn`:

```
from sklearn.tree import DecisionTreeClassifier
dt = DecisionTreeClassifier(criterion='entropy', max_depth=2)
dt.fit(X,Y)
```



Konstrukce stromu pro ukázková data (4/4)



- Strom jsme zkonstruovali, ale ještě jsme k jeho jednotlivým listům nepřiřadili rozhodnutí, jestli pro daný list má být výsledek $Y = 1$ nebo $Y = 0$.

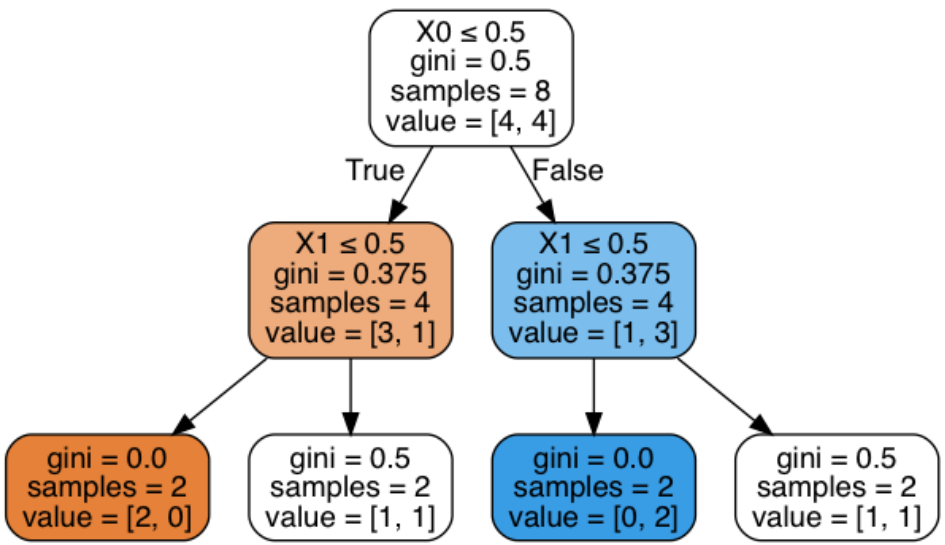
- To se dělá prostým hlasováním daty, které do daného listu „propadnou“: převažují-li nuly (první list zleva), je rozhodnutí $Y = 0$, převažují-li jedničky (třetí list), je to $Y = 1$ a při shodě (druhý a čtvrtý) přiřadíme rozhodnutí náhodně.

Gini index

- Namísto entropie se také používá **Gini index** (angl. **Gini impurity**), pro množinu $\mathcal{D}$ s k různými hodnotami:

$$GI(\mathcal{D}) = 1 - \sum_{0}^{k-1} p_i^2 = \sum_{0}^{k-1} p_i(1 - p_i).$$

- Gini index má podobné vlastnosti jako Entropie. Je to jakási míra toho, že nově přidaný prvek bude špatně klasifikován.
- Jinak ale vše funguje stejně, jen se nahradí $H(\mathcal{D})$ výrazem $GI(\mathcal{D})$.
- Pro naše data to při použití Gini indexu dopadne stejně (blbě):



Hladový $\neq$ optimální

- Získaný rozhodovací strom nebyl bezchybným modelem, neboť vždy u dvou dat rozhodl špatně.
- Jak jsme viděli dříve, je to někdy nevyhnutelná situace, řešitelná pouze tím, že se použije strom s větší hloubkou.
- V tomto případě to tak ale není, neboť existuje strom hloubky dva, který daných osm dat modeluje perfektně.
- **Tento optimální strom ale není dosažitelný hladovým algoritmem!** Přitom je celkem očividně daný podmínkou $Y = 1 \Leftrightarrow (X_1 = X_2)$.

id	Y	X_0	X_1	X_2
0	1	1	0	0
1	1	0	1	1
2	1	1	0	0
3	1	1	1	1
4	0	0	0	1
5	0	0	1	0
6	0	0	0	1
7	0	1	1	0

Konstrukce rozhodovacího stromu: shrnutí

- Konstrukce optimálního stromu je NP-úplný problém, a proto se v praxi používají hladové strategie (algoritmy ID3, C4.5 a C5 od Johna Rosse Quinlana), které ale často najdou *suboptimální* řešení.

- Hladový algoritmus funguje rekurzivně: Pro danou množinu dat najde příznak, který tuto množinu rozdělí tak, aby bylo dosaženo maximálního možného *informačního zisku* (příp. Gini indexu). Stejný postup se pak opakovaně aplikuje na podmnožiny vzniklé tímto rozdělením.

- Takto se data dělí na menší a menší podmnožiny, dokud nenastane **ukončovací podmínka**, kterou si uživatel zvolil (max. hloubka stromu, minimální počet dat v množině, minimální nutná hodnota informačního zisku, atp., viz cvičení).

Jak je můj strom dobrý?

Nyní se dostáváme k druhé otázce: **Jak poznám, že můj vytvořený strom není jen dobrým modelem dat, která mám, ale bude dobře fungovat i pro data jiná?**

- Chceme-li rozhodovat, jak je model dobrý, potřebujeme nějakou objektivní vyčíslitelnou míru jeho kvality.
- Volba této míry je důležitou součástí celého procesu hledání modelu. Míry se obvykle velmi liší pro problémy klasifikace a regrese.
- My se nyní věnujeme klasifikaci a vystačíme si s přirozenou mírou **klasifikační přesnosti** (angl. **classification accuracy**), která prostě měří poměr správně klasifikovaných, tedy je rovna číslu (příp. procentu)

$$\frac{\text{počet správně klasifikovaných dat}}{\text{počet všech dat}}.$$

- Např. rozhodovací strom zkonstruovaný algoritmem ID3 pro množinu osmi dat se pletl ve dvou případech, a tedy jeho přesnost byla $\frac{6}{8} = 0.75 = 75\ \%$.

Jak je můj strom dobrý? Jakože opravdu?

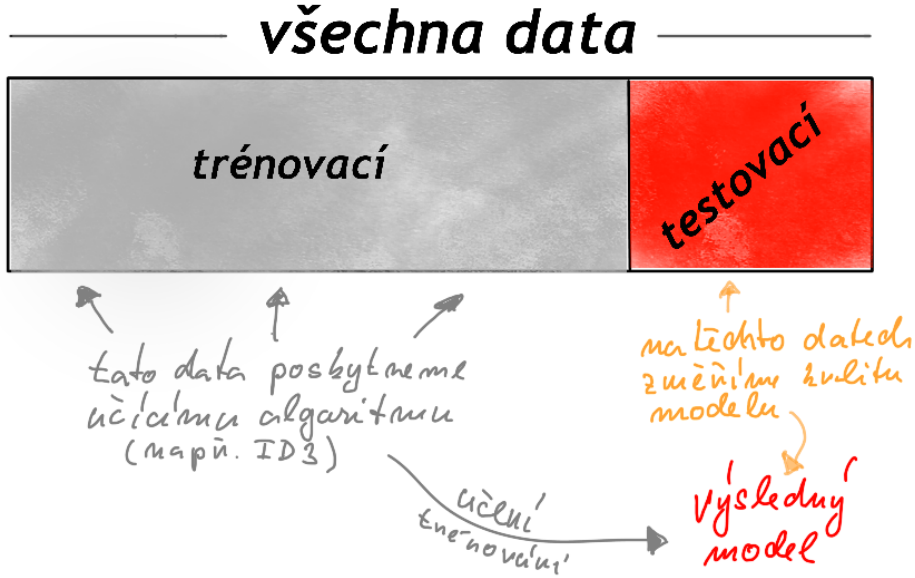
- Mohlo by se zdát, že máme vyřešeno: prostě zkonstruujeme strom, který má pro naše data nejvyšší přesnost.
- S tímto přístupem bychom ale narazili, neb by vedl k hlubokým stromům, které mají na listech třeba jen jeden datový bod.
- Platí totiž, že čím je strom hlubší, tím má lepší přesnost!
- My vlastně nechceme maximalizovat přesnost na našich datech, ale **na všech možných datech**, která ovšem nemáme.

(?) Jak to vyřešit?

- Uděláme to tak, že naše data rozdělíme na dva kusy: na jednom (tom větším), model naučíme (např. C4.5 algoritmem) a na tom druhém kusu změříme přesnost. **Tak dostaneme spolehlivější odhad toho, jak se bude náš model chovat pro data, na kterých se neučil.**

Trénovací a testovací data

- Těm dvěma kusům dat se obvykle říká **trénovací** a **testovací data** (angl. **train** a **test set**).
- Chybovost modelu (pro nás nepřesnost = 1 - přesnost) na těchto množinách dat se pak adekvátně říká **trénovací** a **testovací chyba** (angl. **train** a **test error**).



Přeučení modelu (1/2)

- V případě stromů zjevně platí, že čím hlubší strom učíme, tím dostaneme menší trénovací chybu. Spolehlivější měrou kvality je však testovací chyba, neb tou trénovací si lžeme do kapsy: Měříme, jak dobře jsme *přizpůsobili* strom dostupným datům, nikoli jak dobře jsme odhadli skrytý model za těmito daty schovaným (pokud tedy na takový model věříte).

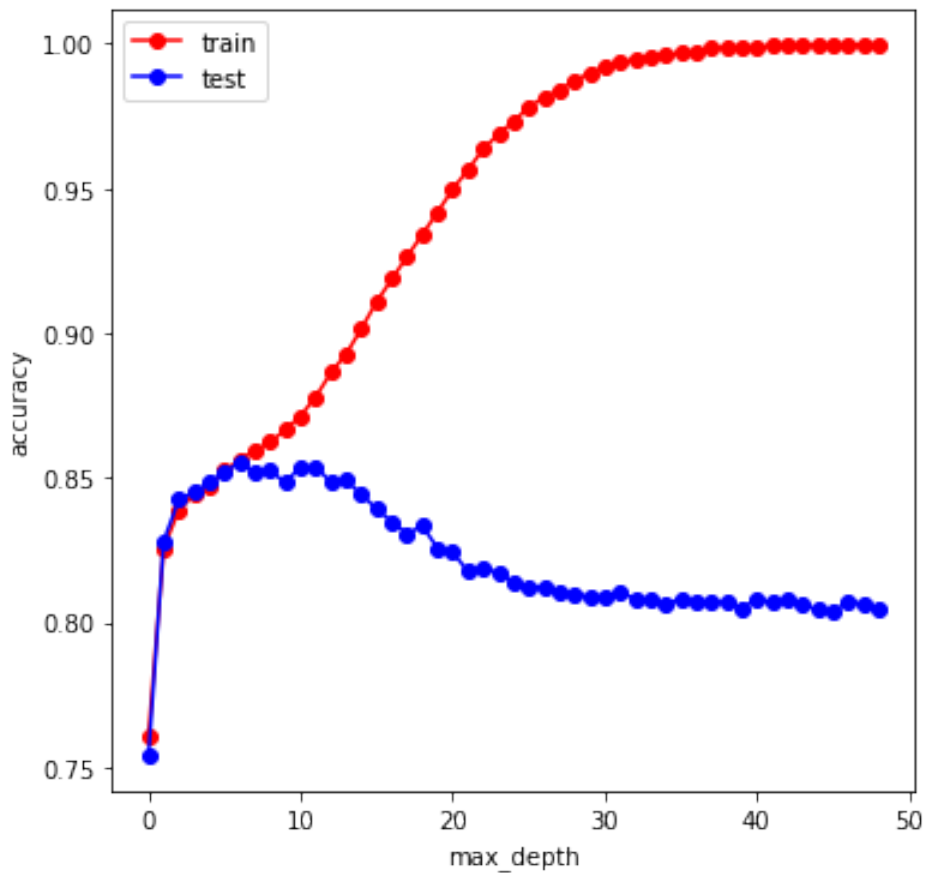
- Tomuto přílišnému přizpůsobení trénovacím datům se říká **přeučení** modelu (angl. **overfitting**).

- Obvykle platí, že čím složitější model (v našem příp. čím hlubší strom), tím nižší trénovací chyba.

- Testovací chyba se však chová jinak: Nejdříve se zvětšováním složitosti modelu klesá, ale v jistý okamžik začne růst. Najít tento bod zlomu je úkolem celého procesu budování modelu.

Přeučení modelu (2/2)

- Následující obrázek ukazuje typický vývoj trénovací a testovací chyby v závislosti na hloubce stromu (`max_depth`).
- Z tohoto obrázku je vidět, že nejrozumnější volba parametru `max_depth` je 6.



Ladění hyperparametrů (1/3)

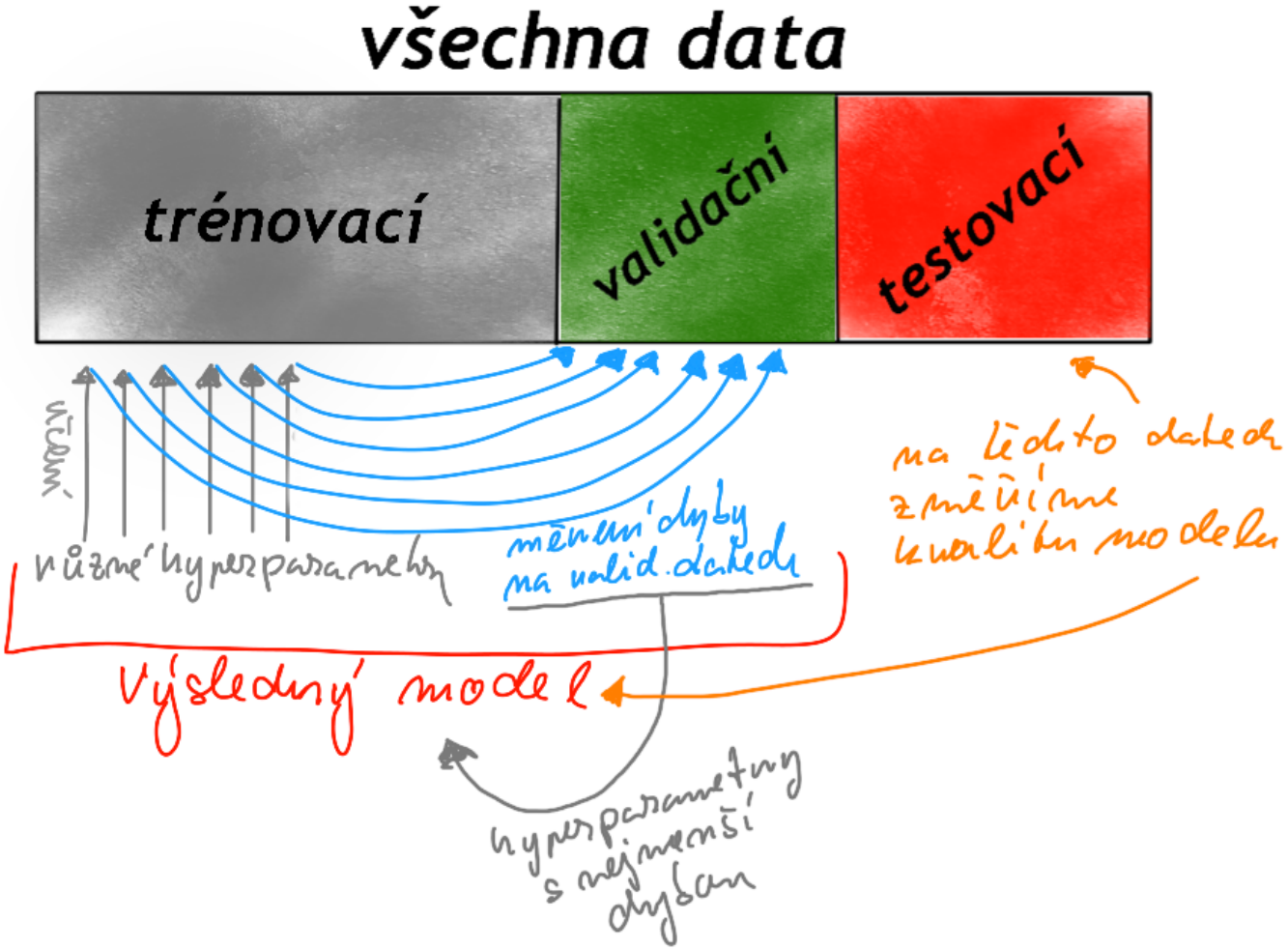
Teď se dostáváme ke třetí otázce: **Jak poznám, jakou mám zvolit hloubku stromu příp. jiné jeho parametry?**

- Můžeme postupovat takto:
  1. Data si rozdělíme na trénovací a testovací.
  2. Pro různé hodnoty hloubky stromu (parametr `max_depth`) naučíme rozhodovací strom na trénovacích datech
  3. a pro každou hloubku stromu změříme přesnost (classification accuracy) na testovacích datech.
  4. Vyberme tu hloubku, která dává nejmenší testovací chybu.
  5. Tuto hodnotu vezmeme také jako odhad chyby na reálných datech, a tedy jako objektivní míru kvality modelu.
- Je někde problém? Ano, je!
- Výsledná testovací chyba bude zpravidla příliš optimistickým odhadem skutečnosti! Výsledný model (konkrétně parametr hloubka stromu) byl vybrán na základě těchto dat a model je tedy těmto datům přizpůsobený.
- Takto bychom porušili zásadu, že **při učení modelu nesaháme na testovací data**, pokud má být testovací chyba rozumným odhadem skutečné chyby modelu.

Ladění hyperparametrů (2/3)

- Jak se s tímto problémem vypořádat?
- Obvykle se to dělá tak, že se data rozdělí ne na dvě, ale na tři podmnožiny: trénovací, **validační** (angl. **validation**) a testovací:
  1. Pro různé hodnoty hloubky stromu `max_depth` naučíme rozhodovací strom
  2. a změříme jeho chybu (přesnost) na validačních datech.
  3. Jako optimální hodnotu vybereme tu s nejmenší chybou (tj. s nejvyšší přesností).
  4. Chyba na testovacích datech, které dosud ležely ladem, je pak rozumným odhadem chyby modelu.
- Parametrům, jako je `max_depth`, které určují *tvar* nebo *komplexitu* modelu, se říká **hyperparametry modelu**.
- Určování těchto parametrů pomocí validačních dat se říká **ladění** (angl. **tuning**).
- Tento parametr nemusí být jeden, ale může jich být více. Pokud jsou spojité, může být výpočetně složité jich otestovat *reprezentativní* množství a je třeba dělat kompromisy.

Ladění hyperparametrů (3/3)



Ladění hyperparametrů: poznámky (1/2)

- Pro představu, jaké hyperparametry jsou dostupné pro rozhodovací stromy v knihovně `sklearn`:

```
Init signature: DecisionTreeClassifier(criterion='gini', splitter='best', max_de
pth=None, min_samples_split=2, min_samples_leaf=1, min_weight_fraction_leaf=0.0,
 max_features=None, random_state=None, max_leaf_nodes=None, min_impurity_decreas
e=0.0, min_impurity_split=None, class_weight=None, presort=False)
```

- Význam těchto parametrů si vysvětlíte (a ukážete) na cvičení.
- Dělení dat na trénovací/validační/testovací podmnožiny je dobré dělat náhodně. Tj. nevybrat první půlku dat jako trénovací, další čtvrtinu jako validační a zbytek vzít jako testovací.
- V pořadí dat by mohla být nějaká zákonitost, která by znamenala, že trénovací a testovací data nebudou reprezentativní.
- Proto se postupuje tak, že se data vybírají náhodně. V `sklearn` je na to metoda:

```
from sklearn.model_selection import train_test_split
Xtrain, Xtest, ytrain, ytest = train_test_split(X, y, test_size=0.25, random_state=33)
```

Ladění hyperparametrů: poznámky (2/2)

- Neexistuje žádný optimální poměr velikosti trénovací/validační/testovací množiny dat.
- Obvyklé poměry jsou takové, že 20 % dat vezmeme jako testovací množinu a ze zbytku vezmeme 20 % jako validační množinu. Co zbude, jsou trénovací data.
- Místo 20 % lze použít 25 %, nebo 30 %.
- Lze použít i sofistikovanější strategie, např. měnit velikost validační množiny a koukat se, co to dělá.
- Často používanou metodou je také **cross-validace**, o které budeme mluvit později.

Vícehodnotové příznaky: one-hot encoding

- Mnohé implementace rozhodovacích stromů (vč. té v `sklearn`) umí konstruovat pouze binární stromy.
- Neumí si tak řádně poradit s diskrétními (**kategorickými**) příznaky, které mají více než dvě různé hodnoty.
- Tento problém lze vyřešit pomocí **one-hot encoding**, kdy se jeden kategoriální příznak s n různými hodnotami nahradí n binárními **dummy variables**.
- Např. v našem příkladu s rýmičkou bychom mohli mít tři možnosti: vstal(a), nevstal(a), spadl(a) z postele. Abychom toto převedli do binárních příznaků pomoci dummy příznaků, museli bychom zavést tři nové binární příznaky *vstal(a) / nevstal(a) / spadl(a)*:

původní příznak	$\rightarrow$ *dummy příznaky* $\rightarrow$	vstal	nevstal	spadl
vstal	$\longrightarrow$	1	0	0
nevstal	$\longrightarrow$	0	1	0
spadl	$\longrightarrow$	0	0	1

Nominální a ordinální příznaky

- Metoda *one-hot encoding* má své nevýhody:
  - Zvyšuje výrazně počet příznaků (dimenzi datasetu), což speciálně u rozhodovacích stromů může výrazně podporovat přeučení.
  - Z datasetu sice nemizí žádná informace, ale vzhledem k tomu, že algoritmus pro konstrukci stromů není optimální (jen „hladový“), může se zvýšení počtu příznaků projevit na kvalitě modelu.
  - Není vhodná pro všechny druhy kategoriálních příznaků, zejm. ne pro ty, kde je mezi jednotlivými kategoriemi nějaké pořadí.

- Kategoriální příznaky jsou v zásadě dvou druhů:

  Nominální Mezi kategoriemi není žádné pořadí, např. *místo narození*, *fakulta*, *pohlaví*, atp.

  Ordinální Mezi kategoriemi je nějaké přirozené uspořádání, např. *vzdělání (základní* $<$ *střední* $<$ *s maturitou* $<$ *univerzitní)* atp.

- Metodu *one-hot encoding* je tedy lepší používat pouze pro nominální příznaky. Ordinální je lepší nechat tak jak jsou, `sklearn` s nimi může pracovat jako se spojitými příznaky, což dává větší smysl (viz dále).

Spojité příznaky

- V datech ale většinou nemáme pouze kategoriální příznaky. Často se v nich objevují příznaky jako *věk*, *cena*, *teplota*, *tlak* atp., které je lepší chápat jako příznaky **spojité** (angl. **continuous**).

- V rozhodovacích stromech mohou k větvení sloužit i tyto příznaky. Pravidlo pro (binární) větvení může být např. `věk` < 30, které opět množinu dat rozdělí na dvě části.

- Oproti binárním příznakům má smysl, aby se spojité proměnné objevily ve stromu vícekrát. Rozdělíme-li data pravidlem `věk` < 30, dává smysl „starší“ část dat dělit znovu pravidlem `věk` < 60. U pravidla s binárním příznakem `pohlaví == žena` toto smysl nedávalo.

Spojité příznaky a algoritmus

- Algoritmus pro binární příznak $X \in \{0, 1\}$ fungoval tak, že spočítal *informační zisk* rozdělení dat $\mathcal{D}$ podle pravidla $X == 0$ a buď toto dělení použil, nebo si vybral jiný příznak s vyšším informačním ziskem.

- Spojitý příznak X s hodnotami např. z intervalu $[0, 100]$ musí fungovat jinak, neboť pravidel tvaru $X < d$ existuje vlastně nekonečno (pro všechna možná $0 < d \leq 100$).

- V základním nastavení algoritmus funguje tak, že ale vlastně všechna dělení ve tvaru $X < d$ vyzkouší a vybere to s nejvyšším informačním ziskem. Funguje to takto:
  1. Projdi všechny možné hodnoty příznaku X v právě dělené množině dat $\mathcal{D}$ a setřiď je podle velikosti: $x_0 < x_1 < \cdots < x_\ell$.
  2. Vyzkoušej rozdělení dat podle pravidla $X < x_i$ pro všechna $i = 1, 2, \ldots, \ell$ a pro každé takové rozdělení $\mathcal{D}$ spočítej informační zisk.
  3. Jako nejlepší pravidlo dělení podle příznaku X vezmi to s největším spočítaným informačním ziskem.

Spojité příznaky a algoritmus: příklad

Představme si, že máme data $\mathcal{D}$ s osmi řádky s příznakem `věk`, který má pro řádky 0 až 7 tyto hodnoty:

$$12_0, 35_1, 15_2, 65_3, 35_4, 15_5, 20_6, 20_7,$$

přičemž binární vysvětlovaná proměnná Y má pro tyto řádky hodnoty

$$0_0, 1_1, 0_2, 0_3, 1_4, 0_5, 1_6, 1_7.$$

Entropie $\mathcal{D}$ je tedy maximální, tj. $H(\mathcal{D}) = 1$.
Dle předchozího postupujeme tak, že možné hodnoty X seřadíme:
$12, 15, 20, 35, 65$, a vyzkoušíme všechna následující rozdělení.

pravidlo	$\mathcal{D}_L$	$\mathcal{D}_R$	informační zisk
$X < 15$	0_0	$1_1, 0_2, 0_3, 1_4, 0_5, 1_6, 1_7$	0.14
$X < 20$	$0_0, 0_2, 0_5$	$1_1, 0_3, 1_4, 1_6, 1_7$	0.55
$X < 35$	$0_0, 0_2, 0_5, 1_6, 1_7$	$1_1, 0_3, 1_4$	0.05
$X < 65$	$0_0, 1_1, 0_2, 1_4, 0_5, 1_6, 1_7$	0_3	0.14

Pokud se tedy nenajde jiný příznak dávající lepší informační zisk, použije algoritmus pro rozdělení dat $\mathcal{D}$ pravidlo $X < 20$.

Poznámky

- Pokud má spojitý příznak příliš mnoho hodnot, obvykle se postupuje tak, že se uvažuje pravidlo $X < x_i$ například pouze pro každé desáté x_i. Příp. se náhodně vyzkouší daný počet hodnot.
- Implementace v `sklearn` se chová vlastně ke všem příznakům jako ke spojitým.
- Dovede si tak celkem dobře poradit s ordinálními příznaky.
- Naopak k nominálním se chová poněkud nešťastně.
- V případě rovnosti informačního zisku si `sklearn.tree.DecisionTreeClassifier` vybírá náhodně. Algoritmus se tak může chovat nedeterministicky, přestože k tomu není žádný viditelný důvod.

Rozhodovací stromy pro regresi

Pokud bychom chtěli stejně jako s diskrétní pracovat se **spojitou vysvětlovanou proměnnou**, musíme vyřešit dva problémy:

- Jak bude naučený strom určovat *predikovanou* hodnotu vysvětlované proměnné?
  - V případě diskrétní proměnné se rozhodovalo hlasováním v rámci listu: Skončila-li cesta stromem pro datový bod v listu, ve kterém z trénovacích dat skončilo nejvíce bodů s hodnotou $Y = x$, rozhodnutí daného listu bylo $Y = x$.
  - V případě spojité proměnné Y (např. věk, teplota, cena, doba trvání, ...) se může snadno stát, že každý z trénovacích bodů má jinou hodnotu.

- Jaké kritérium použijeme v hladovém algoritmu pro výběr nejlepšího příznaku k dělení?
  - V případě diskrétní proměnné jsme používali *entropii* resp. *gini index*.
  - Ty jsou ale v případě spojité proměnné nepoužitelné (rozmyslete si, jak by to vypadalo, zejm. jaké hodnoty by měla čísla p_i).
  - Pro spojitou vysvětlovanou proměnnou budeme muset najít jiné kritérium.

Jak strom určuje své rozhodnutí

- Předpokládejme, že už máme naučený strom a chceme najít predikovanou hodnotu Y pro nějaký datový bod s příznaky $X_0, X_1, \ldots, X_{p-1}$.
- Pomocí rozhodovacích pravidel v daném stromu a hodnot příznaků $X_0, X_1, \ldots, X_{p-1}$ se dostaneme do listu, kde při učení „skončilo“ šest datových bodů z trénovacího množiny dat.
- Předpokládejme, že vysvětlovaná proměnná Y pro těchto šest bodů měla hodnoty
$$\{10, 15, 20, 25, 30, 35\}.$$
- Jaké má být tedy rozhodnutí stromu pro body, které skončí v tomto listu?
- Obvykle se bere **průměr hodnot z listu**, v našem případě tedy 22.5.

Co použít místo entropie?

- V případě klasifikace se stromy rozhodovaly pomocí „hlasování“ v rámci jednotlivých listů. Minimalizace entropie měla zaručit, aby toto hlasování mělo co nejjednoznačnějšího vítěze.
- V případě regrese se strom rozhoduje tak, že v rámci listu spočítá průměr.
- Naší snahou by tedy mělo být, aby **hodnoty vysvětlované v rámci listu byly co nejblíže střední hodnoty**. Jak toto měřit?
- Známou mírou „odchylky od střední hodnoty“ je **MSE = mean squared error**, což je skoro stejná veličina jako (výběrový) **rozptyl** (angl. **sample variance**).

Co použít místo entropie? MSE!

- Představme si, že máme data s hodnotami vysvětlované proměnné

$$\{Y_0 = 10, Y_1 = 15, Y_2 = 20, Y_3 = 25, Y_4 = 30, Y_5 = 35\}.$$

- Průměr (tj. odhad střední hodnoty) této množiny je $\overline{Y} = 22.5$.
- Odhad MSE je číslo $\mathrm{MSE}(\mathbf{Y}) = (Y_0, Y_1, \ldots, Y_{N-1})$, což není nic jiného, než aritmetický průměr čtverců vzdáleností od průměru:

$$\mathrm{MSE}(\mathbf{Y}) = \frac{1}{N} \sum_{j=0}^{N-1} (Y_j - \overline{Y})^2$$

pro naše data tedy

$$\mathrm{MSE}((10, 15, 20, 25, 30, 35)) = \frac{1}{6}((10 - 22.5)^2 + (15 - 22.5)^2 +$$
$$+(20 - 22.5)^2 + (25 - 22.5)^2 + (30 - 22.5)^2 + (35 - 22.5)^2) = 72.9.$$

Algoritmus CART

- Hladovému algoritmu, ve kterém se vrcholy rozhodovacího stromu volí tak, aby se minimalizovalo MSE, se říká **CART = Classification and Regression Trees**.

- Funguje stejně jako algoritmus ID3 představený dříve: kvalitu rozdělení množiny $\mathcal{D}$ na podmnožiny $\mathcal{D}_L$ a $\mathcal{D}_R$ ale namísto entropie

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

používá MSE

$$\mathrm{MSE}(\mathcal{D}) - t_L \mathrm{MSE}(\mathcal{D}_L) - t_R \mathrm{MSE}(\mathcal{D}_R),$$

kde $t_L = \frac{\#\mathcal{D}_L}{\#\mathcal{D}}$ a $t_R = \frac{\#\mathcal{D}_R}{\#\mathcal{D}}$ a $\mathrm{MSE}(\mathcal{D})$ je MSE spočítané pro hodnoty vysvětlované proměnné Y pro všechny body z $\mathcal{D}$.

Poznámky

- Místo MSE lze používat také MAE = Mean Absolute Error:

$$\mathrm{MAE}(\mathbf{Y}) = \frac{1}{N} \sum_{i=0}^{N-1} |Y_i - \overline{Y}|,$$

  který místo čtverce vzdálenosti používá absolutní hodnotu.

- Rozhodovací stromy mají mnoho výhod:
  - Nenáročnost na přípravu dat: poradí si s kategoriálními i spojitými příznaky, s chybějícími hodnotami, atp.
  - Jsou jednoduché a srozumitelné a učení je relativně rychlé.
  - Jsou dobře interpretovatelné a jejich rozhodnutí lze snadno rozklíčovat.

- Ale mají samozřejmě i nevýhody:
  - Jsou *nerobustní*: i drobná změna v trénovacích datech může znamenat zásadní změnu struktury výsledného stromu.
  - Většina implementací podporuje pouze binární stromy.
  - Najít optimální strom je NP-úplný problém.
  - Je snadné rozhodovací stromy přeučit.