

Kapitola 1

Úvod do strojového učení

V této kapitole se seznámíte se základními principy strojového učení (anglicky „Machine learning“, „ML“). Pochopíte, co znamená pojem model, jaké existují typy učení a jakým způsobem probíhá obecný proces tvorby a používání modelu. Kapitola bude sloužit jako úvodní teoretický základ pro všechny následující metody a algoritmy, které se budou probírat v dalších částech kurzu.

Cíle kapitoly (po přečtení této kapitoly budete schopni)

- C1.1. Vysvětlit, co je to *model* strojového učení a jak probíhá proces jeho tvorby.
- C1.2. Rozlišit základní typy učení: s učitelem, bez učitele a zpětnovazební učení, včetně jejich typických příkladů.
- C1.3. Pochopit rozdíl mezi klasifikací, regresí a shlukováním.
- C1.4. Popsat obecný proces použití strojového učení – od získání dat přes jejich analýzu, přípravu, rozdělení a trénink modelu až po jeho vyhodnocení a použití.
- C1.5. Porozumět tomu, kdy má smysl použít metody strojového učení a kdy je vhodné hledat jednodušší.
- C1.6. Seznámit se s klíčovými metrikami pro hodnocení klasifikačních a regresních modelů a uvědomit si, že volba vhodné metriky závisí na konkrétní úloze a na tom, jaké typy chyb jsou v daném kontextu přijatelné.
- C1.7. Popsat význam pojmů trénovací, validační a testovací data a jejich roli v hodnocení modelu.
- C1.8. Seznámit se s problémy „overfitting“ a „underfitting“.

Klíčové myšlenky (po přečtení této kapitoly byste si měli pamatovat)

- KM1.1. Strojové učení je nástroj, nikoli cíl. Než dojde k vytvoření modelu, je nutné si položit otázku, zda je tento postup skutečně nutný a přínosný.
- KM1.2. Model je matematické zjednodušení reality. Učí se ze zkušeností (dat), aby mohl dělat predikce nebo rozhodnutí.
- KM1.3. Existují různé typy učení. Učení s učitelem (tzv. „supervised learning“) se učí z dat, která obsahují vstupy i známé výstupy (čili vstupní data a jejich odpovídající výsledky). Učení bez učitele (tzv. „unsupervised learning“) se snaží najít strukturu v datech, u kterých jejich výstupy (jejich odpovídající výsledky) nejsou známé. Zpětnovazební učení (tzv. „reinforcement learning“) je odlišný přístup, kdy se model učí prostřednictvím zpětné vazby v podobě odměn a trestů při interakci s okolním prostředím.
- KM1.4. Klasifikace, regrese a shlukování představují základní úlohy, které si modely kladou.

- KM1.5. Správné rozdělení dat a validace jsou nezbytné pro to, aby model nebyl „přeučný“ (anglicky „overfitting“) ani příliš jednoduchý a nepřesný (anglicky „underfitting“), a aby dobře fungoval i na nových, dosud neviděných datech.
- KM1.6. Pro hodnocení ML modelů je důležité využívat vhodné metriky, které poskytují různé pohledy na jejich kvalitu a pomáhají posoudit, zda model splňuje požadavky konkrétní úlohy.
-

Nové příkazy *(po přečtení kapitoly se seznámím s těmito novými příkazy)*

```
make_classification, train_test_split, DummyClassifier, LogisticRegression,  
accuracy_score, precision_score, recall_score, f1_score, roc_auc_score,  
average_precision_score, confusion_matrix, roc_curve, precision_recall_curve,  
mean_squared_error, mean_absolute_error, r2_score
```



TURINGINSTITUT

1.1 Co je model ve strojovém učení?

Model si můžeme představit jako pravidlo nebo funkci, která převádí vstupy na výstup. Vstupem mohou být číselné či kategorické veličiny, text, obraz nebo zvuk; výstupem pak např. pravděpodobnost, že e-mail je spam, odhad ceny nemovitosti či přiřazení pozorování do určité skupiny. Učení modelu znamená nalézt takové parametry této funkce, které minimalizují chybu vůči „pravdě“ obsažené v datech. Při učení je zásadní rozlišovat mezi **parametry** (hodnoty, které se odhadují z dat) a **hyperparametry** (nastavení zvolené člověkem, které ovlivňuje složitost a chování modelu). Samotné „naučení“ ještě nezaručuje užitečnost – klíčová je **generalizace**, tedy schopnost modelu poskytovat dobré výsledky i na nových, dosud neviděných datech. Přeučený model si „pamatuje“ tréninkové situace až příliš, zatímco nedostatečně naučený model je tak jednoduchý, že nezachytí ani základní vzorce. V praxi se proto hledá rovnováha mezi složitostí a zobecněním.

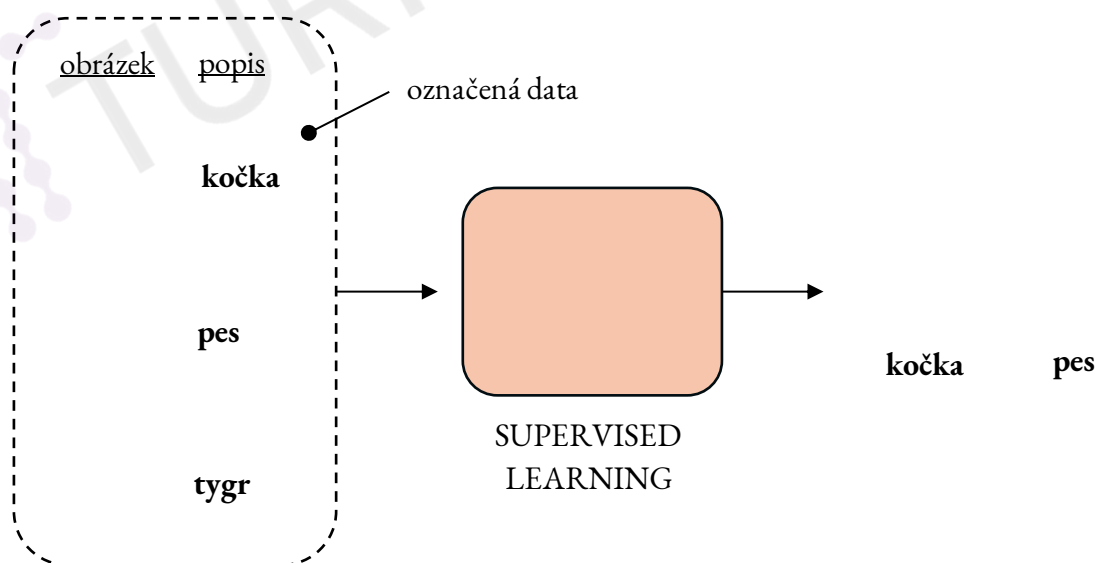
1.2 Typy učení

Učení s učitelem („supervised learning“)

V této situaci jsou k dispozici páry (vstup, správný výstup). Model se učí funkci, která vstupy „mapuje“ na výstupy co nejpresněji. V rámci učení s učitelem rozlišujeme klasifikaci a regresi.

- klasifikace: přiřazuje data do předem daných tříd (např. „schválit/neschválit“, „je spam/není spam“),
- regrese: předpovídá spojitou číselnou hodnotu (např. „očekávaná spotřeba paliva“).

Důležitou roli hraje volba metrik: u klasifikace se typicky sleduje „accuracy“, „precision/recall“ či F1-score, u regrese pak průměrná absolutní chyba či například střední kvadratická chyba. Správná metrika se vybírá podle účelu – například u detekce podvodů je často důležitější zachytit co nejvíce skutečných podvodů (vyznačuje se vysokým „recall“), a to i za cenu většího množství falešných poplachů.



Obrázek 1: Zjednodušený princip modelu strojového učení: učení s učitelem.

Na obrázku č. 1 je vyobrazen zjednodušený trénovací proces modelu strojového učení s učitelem, obrázky představují vstupní data, jejich popisy představují jim odpovídající správné výstupy (obrázek

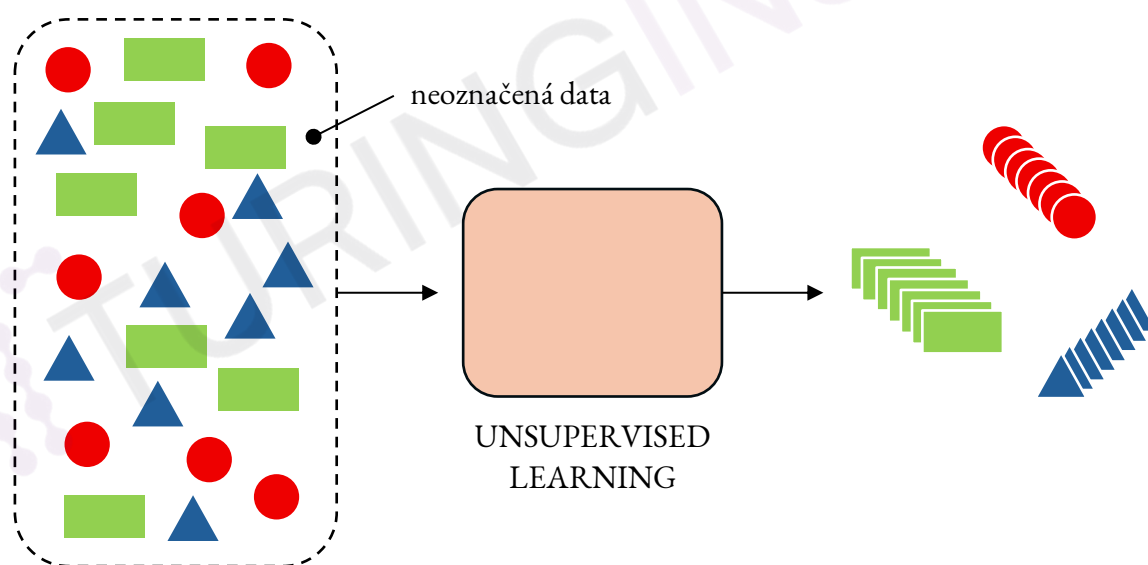
s kočkou = kočka, obrázek se psem = pes apod.). Při trénování takto správně označená (obrázky označené svými popisy/správnými výstupy) data prochází modelem strojového učení a ten se díky nim, prostřednictvím úprav svých vnitřních hyperparametrů, učí přiřazovat správné výstupy svým odpovídajícím vstupům. Po úspěšném natrénování by měl model být schopen správně kvalifikovat obrázek, který doposud neviděl a který obsahuje zvíře, které se model při trénování učil přiřazovat. Když však prochází natrénovaným modelem obrázek se zvířetem (v tomto případě například kůň), které model během trénování neviděl, výsledkem může být chybná klasifikace nebo přiřazení k nejbližší známé třídě, protože model nemá zkušenost s touto novou kategorií.

Učení bez učitele („unsupervised learning“)

Pokud jsou dostupná pouze vstupní data bez „správných odpovědí“, snaží se model odhalit strukturu skrytou v datech. Mezi základní úlohy modelu patří:

- shlukování („clustering“): seskupuje podobná data dohromady – například na základě chování zákazníků lze odhalit skupiny s podobnými potřebami;
- snížení dimenze: nalezení menšího počtu proměnných, které vystihují podstatu dat stejně dobře jako původní velké množství parametrů.

U učení bez učitele neexistuje jedno „správné“ řešení, kvalita se posuzuje pomocí určitých vnitřních ukazatelů (např. silueta ve shlukování) a především podle užitečnosti při jeho nasazení na nějaký praktický úkol.

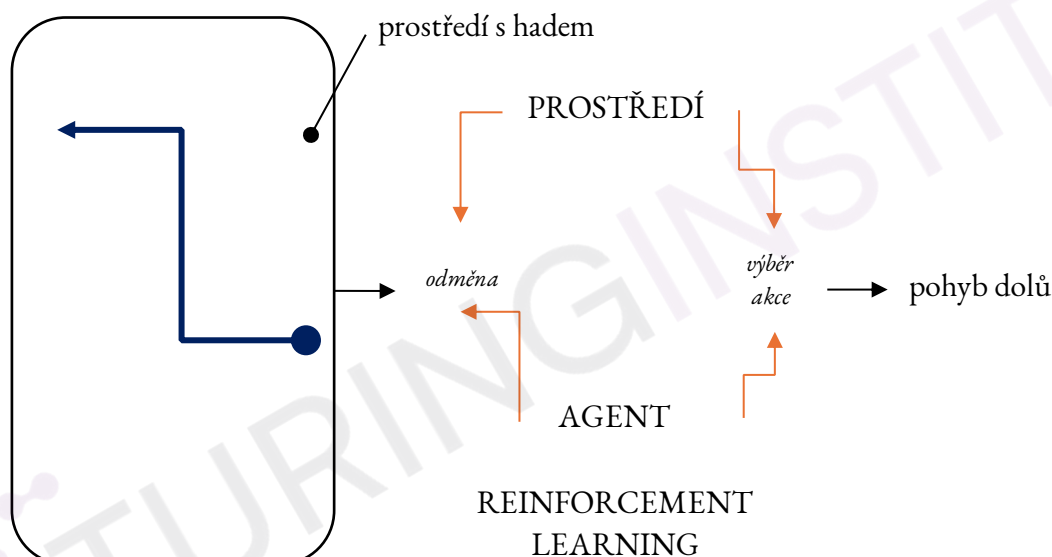


Na obrázku č. 2 je znázorněn princip učení bez učitele („unsupervised learning“). Vstupní data jsou v tomto případě neoznačená – nemají přiřazené žádné kategorie ani popisy, které by určovaly správný výstup (čili které by je definovaly). Model strojového učení proto pracuje pouze s jejich vlastnostmi a vzájemnými podobnostmi. Úkolem algoritmu je (v tomto případě algoritmu shlukování) nalézt v datech strukturu, tedy seskupit je do skupin (shluků) nebo odhalit skryté souvislosti. Na schématu je vidět, že různorodé vstupy (různé tvary a barvy) procházejí modelem, který je dokáže rozdělit do přirozeně vzniklých skupin podle jejich podobnosti. Model přitom nezná předem, že například červené kruhy tvoří jednu skupinu a modré trojúhelníky jinou – tuto informaci si „sám objeví“.

Výhodou učení bez učitele je, že lze pracovat i s daty, pro která nejsou k dispozici žádné popisy, což je častý případ v praxi. Nevýhodou je naopak to, že výsledky nemusí vždy odpovídat lidským představám o „správném“ rozdělení, protože algoritmus se řídí čistě podobnostmi a strukturou skrytou v datech.

Zpětnovazební učení („reinforcement learning“, „RL“)

Zpětnovazební učení se od předchozích přístupů liší povahou zpětné vazby. Model zde vystupuje jako tzv. „agent“, který jedná ve svém okolním prostředí. V každém kroku vidí stav tohoto okolí, zvolí nějakou akci (svou činnost: „to, co v tuto chvíli agent provede“) a za ní získá odměnu – jeho cílem je naučit se strategii volby akcí, která maximalizuje dlouhodobý součet jeho budoucích odměn. Typicky je tento model založen na tzv. „Markovovém rozhodovacím procesu“. Model zpětnovazebního učení řeší problémy, ve kterých je důležitá posloupnost rozhodnutí – řízení robota, řízení provozu nebo například pohyb hada ve známé hře „Snake“. Na rozdíl od učení s učitelem nebývá k dispozici „správný výstup“ pro každý stav – agent se musí učit z následků svých vlastních akcí (např. „had se musí naučit z následků svých vlastních pohybů“).



Na obrázku č. 3 je vyobrazen princip zpětnovazebního učení („reinforcement learning“). V tomto přístupu se pracuje s neoznačenými daty, která agent zpracovává prostřednictvím interakcí s prostředím. Agent na základě svého stavu volí určitou akci, kterou působí na prostředí. Prostor následně reaguje – změní svůj stav a poskytne agentovi zpětnou vazbu ve formě odměny nebo trestu. Díky této zpětné vazbě se agent postupně učí, jaké chování je výhodné, a jaké naopak ne. Cílem je nalézt takovou strategii výběru akcí, která povede k maximalizaci celkové dlouhodobé odměny. Na schématu je tento proces znázorněn jako cyklus mezi agentem a prostředím, a právě tento cyklus umožňuje průběžné zlepšování rozhodování. Tento princip je základem pro řešení úloh, u kterých je třeba se přizpůsobovat měnícím se okolním podmínkám – typickým příkladem jsou systémy řízení robotů, počítačové hry (například hra „snake“) nebo optimalizace rozhodovacích procesů.

1.3. Obecný proces použití strojového učení

Obecný postup práce se strojovým učením je založen na sledování několika navazujících kroků, které umožňují postupně přejít od prvotní myšlenky k praktickému využití. Tento proces je strukturován

tak, aby zajišťoval systematicčnost, opakovatelnost a možnost ověření dosažených výsledků. Díky tomu je možné uplatnit strojové učení v různých oblastech a zároveň zachovat potřebnou míru spolehlivosti i srozumitelnosti celého postupu.

1. Definice problému:

Je třeba přesně formulovat otázku: „Co chci modelovat nebo předpovídat? Jak se výsledek využije?“.

2. Shromáždění dat:

Bez kvalitních dat nelze postavit kvalitní model. Data musí být reprezentativní a dostatečně rozsáhlá.

3. Explorační analýza dat (anglicky „exploratory data analysis“, „EDA“)

Zkoumání dat – jejich rozdělení, vypořádání se s chybějícími hodnotami, analýza extrémů dat, lokalizace vztahů mezi proměnnými apod.

4. Rozhodnutí, zda je strojové učení potřebné:

Pokud lze model snadno vyřešit pravidlem (např. „všichni lidé pod 18 let nemohou volit“), není potřeba složitý model strojového učení.

5. Příprava dat:

Čištění, odstraňování chyb, vyčlenění kategorií, transformace dat do správných datových typů, změna velikosti dat apod.

6. Rozdělení dat:

Data se typicky rozdělují na tři části:

- trénovací sada (používá se na trénování modelu),
- validační sada (slouží k ladění parametrů a porovnávání modelů),
- testovací sada (slouží ke zhodnocení finální podoby modelu).

7. Trénink modelu:

Během tréninku se algoritmus zvoleného modelu učí na základě trénovací sady dat.

8. Vyhodnocení:

Použití vhodných metrik pro vyhodnocení kvality natrénovaného modelu – accuracy, precision, F1-score apod.

9. Nasazení modelu a jeho sledování:

Proces nasazení modelu na reálný problém a jeho sledování (z důvodu možnosti změny charakteru vstupních dat).

1.4 Kdy má smysl použít strojové učení?

Strojové učení se uplatňuje zejména tehdy, když klasické postupy nestačí nebo by byly příliš složité. Typickým příkladem je situace, kdy v datech existují skryté souvislosti, které se dají jen obtížně popsat pevnými pravidly (např. „pokud je venku 25 °C, zákazník si určitě koupí zmrzlinu“ – takto jednoduché vztahy obvykle v reálných datech neexistují). Algoritmus je pak schopen tyto složitější vzory rozpoznat a využít je k předpovědi nebo k rozhodování.

Velký přínos má strojové učení i při práci s rozsáhlými datovými soubory (např. miliony záznamů o zákaznících či tisíce měření ze senzorů), kde by ruční vyhodnocení nebylo možné nebo by trvalo nepřiměřeně dlouho. Díky automatizaci je možné rychle zpracovat velké objemy informací a dosáhnout výsledků, které by jinak nebyly dostupné.

Důležitým kritériem je také význam výsledků. Pokud nasazení modelu dokáže přinést úsporu času (automatické třídění e-mailů), snížení nákladů (predikce poruch strojů, aby se předešlo nákladným opravám) nebo zvýšení bezpečnosti (systémy pro detekci podvodů v bankovníctví), má jeho využití velký smysl.

Naopak v případech, kdy lze problém jednoduše vyřešit pomocí jednoznačných pravidel (např. „všichni lidé mladší 18 let nemohou volit“) nebo základních výpočtů (např. výpočet průměru známek ve třídě), bývá efektivnější zvolit právě tento jednodušší přístup. Nasazení strojového učení je vhodné pouze tam, kde jeho složitost a nároky odpovídají skutečnému přínosu.

1.5 Metriky kvality ML modelu

Kvalitu modelu strojového učení nelze posuzovat jen na základě toho, že model dokáže vytvořit výstup, ale především podle toho, jak přesně a spolehlivě řeší zadaný problém. Abychom mohli výkon modelu objektivně vyjádřit, využívají se různé metriky, které převádějí výsledky jeho práce do číselné podoby. Tyto ukazatele umožňují porovnávat modely mezi sebou, odhalovat jejich silné i slabé stránky a zvolit ten, který je pro konkrétní úlohu nejvhodnější. Správná volba metriky přitom závisí na typu problému – jiná kritéria se uplatňují u klasifikace, jiná u regrese či jiných úloh. Základní metriky jsou popsány níže (názvy jsou uvedeny v angličtině, neboť takto jsou standartně používány v modelech a jejich překlad by nebyl účelný).

Vysvětlení důležitých pojmů:

Pojem	Definice	Analogie
True Positive (TP)	Model správně označí pozitivní případ jako pozitivní.	Pacient nemoc má a model to správně pozná.
True Negative (TN)	Model správně označí negativní případ jako negativní.	Pacient nemoc nemá a model správně řekne, že je zdravý .
False Positive (FP)	Model chybně označí negativní případ jako pozitivní („falešný poplach“).	Pacient je zdravý , ale model tvrdí, že má nemoc – zbytečné vyšetření.
False Negative (FN)	Model chybně označí pozitivní případ jako negativní („přehlédnutí“).	Pacient nemoc má , ale model tvrdí, že je zdravý – nemoc zůstane skrytá.

Klasifikace

- a. „accuracy“ – podíl všech správně určených tříd (pozitivních i negativních) k celkovému počtu vzorků:

$$accuracy = \frac{TP + TN}{TP + TN + FP + FN}$$

kde TP ... True Positive, TN ... True Negative, FP ... False Positive, FN ... False Negative

Vysvětlení:

„Accuracy“ se často používá jako první měřítko kvality modelu. Vyjadřuje, v kolika případech se model trefil správně – tedy kolik procent všech předpovědí bylo správných. Na první pohled to zní jednoduše: čím vyšší číslo, tím lepší model. Problém ale nastává, když jsou naše data nevyvážená – tedy když je některých typů případů hodně a jiných jen velmi málo. Představme si příklad: máme **tisíc** bankovních plateb a jen **deset** z nich je podvodných. Pokud model předpoví, že žádná platba **není podvod**, bude mít „accuracy“ 990 z 1000, tedy **99 %**. Na papíře to vypadá jako skvělý výsledek, ale ve skutečnosti to tak však není, protože model nedokázal najít ani jeden skutečný podvod.

- b. „precision“ – určuje, jaký podíl případů, které model označil jako pozitivní, je skutečně pozitivní:

$$precision = \frac{TP}{TP + FP}$$

kde TP ... True Positive, FP ... False Positive

Vysvětlení:

„Precision“ říká, jak velká část případů, které model označil jako „pozitivní“, byla skutečně správná. Jinými slovy – když model vydá „poplach“, jak moc se na tento poplach dá spolehnout. Tato metrika je důležitá, protože model může někdy označovat hodně věcí jako pozitivní, ale přitom se často mýlit. Na první pohled může model působit úspěšně, protože zachytí spoustu pozitivních případů, ale pokud je mezi nimi i velké množství falešných poplachů, takový model není prakticky použitelný. Představme si příklad: lékařský test označí 100 lidí jako nemocné, ale skutečně nemocných je pouze 20. „Precision“ je tedy 20 %. To znamená, že 80 % poplachů bylo falešných a tato metrika je zachytila. Model s takovou hodnotou „precision“ by nebyl spolehlivý, protože většinu zdravých lidí by označil jako nemocné.

- c. „recall“ – vyjadřuje, jaký podíl všech skutečně pozitivních případů byl správně rozpoznán:

$$recall = \frac{TP}{TP + FN}$$

kde TP ... True Positive, FN ... False Negative

Vysvětlení:

„Recall“ říká, jak velkou část všech skutečně pozitivních případů model dokázal správně odhalit. Jinými slovy – když existují pozitivní případy (např. nemocní pacienti), kolik z nich model opravdu zachytil. Problém, který „recall“ odhaluje, nastává tehdy, když model sice většinou správně označuje negativní případy, ale velkou část pozitivních přehlíží. To znamená že model nechává „utéct“ mnoho důležitých případů, které by měl najít. Mějme příklad: máme 100 nemocných pacientů a test správně označí jen 50 z nich a druhých 50 označí jako zdravé. „Recall“ je tedy 50 %. Takový model by byl velmi nebezpečný, protože polovina nemocných by zůstala bez diagnózy.

- d. „F1-score“ – harmonický průměr mezi „precision“ a „recall“, který je vhodný, pokud je třeba vyvážit obě metriky:

$$F1 = 2 \cdot \frac{precision \cdot recall}{precision + recall}$$

Vysvětlení:

„F1-score“ je kombinací „precision“ a „recall“. Ukazuje, jak si model vede celkově, pokud potřebujeme brát v úvahu obě metriky zároveň. Problém, který „F1-score“ pomáhá řešit, nastává tehdy, když má model jednu z těchto metrik vysokou a druhou nízkou. V takovém případě by použití jen jedné z metrik mohlo vést k mylnému dojmu, že model funguje dobře. Tato metrika ale ukáže, že výkon modelu je ve skutečnosti slabší, protože je nevyvážený. Představme si příklad: model dokáže zachytit téměř všechny nemocné pacienty (má tedy vysokou hodnotu metriky „recall“), ale zároveň označuje mnoho zdravých lidí jako nemocné (má tedy nízkou hodnotu metriky „precision“) – či přesně naopak, tedy dává málo falešných poplachů (má vysokou „precision“), ale většinu nemocných přehlédne (má nízký „recall“). V obou případech „F1-score“ ukáže, že model není spolehlivý.

- e. ROC-AUC – udává schopnost modelu odlišit pozitivní a negativní případy napříč všemi prahy rozhodování, hodnota se pohybuje mezi 0 a 1, kde 1 značí dokonalou klasifikaci; matematicky se jedná o plochu pod křivkou True Positive Rate vs. False Positive Rate.

Vysvětlení:

Modely často nedávají jen odpovědi „ano“ či „ne“, ale i nějaké skóre – číslo, které ukazuje, jak moc si model myslí, že daný případ patří do pozitivní skupiny. U modelu, který by filtroval spamové e-maily, by to mohlo být například „s pravděpodobností 0.8 (80 %) je tento e-mail spam“. Abychom z takového čísla dostali jednoznačné rozhodnutí, musíme si zvolit práh (neboli „threshold“). Řekneme tedy, že pokud je skóre vyšší než 0.5, považujeme e-mail za spam, ale pokud je nižší, spam to není. Kdybychom práh nastavili na 0.9, model by označil za spam jen ty e-maily, u kterých si je opravdu jistý. Naopak s prahem nastaveným na 0.3 by model označil jako spam skoro vše, co vypadá alespoň trochu podezřele. Problém, který zde ale nastává, je, že různá nastavení prahu modelu vedou k různému chování modelu:

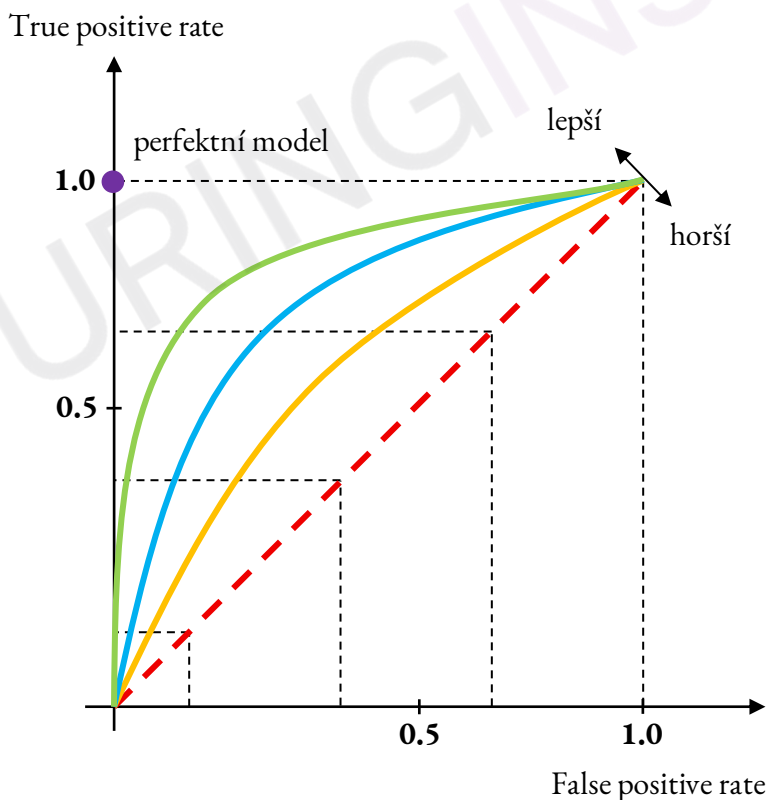
- **vysoký práh** – málo falešných poplachů, ale také větší riziko, že některé spamy uniknou,

- **nízký práh** – většina spamů bude odhalena, ale přibude falešných označení (označení normálního e-mailu jako spamu).

“ROC” křivka právě ukazuje, jak se model chová, když tento práh měníme. Na jedné ose je podíl správně zachycených pozitivních případů („true positive rate“, zvaný také „recall“), na druhé je podíl falešných poplachů („false positive rate“). „AUC“ („area under curve“) představuje plochu pod touto křivkou a platí, že čím větší plocha, tím lepší model – model totiž dokáže najít vysoký podíl skutečně pozitivních případů, a přitom si udrží nízký počet falešných poplachů (viz zelená křivka na grafu níže). Naopak, pokud má model ROC křivku blízko úhlopříčky (viz červená čárkovaná čára na grafu níže), plocha pod ní je malá, což znamená, že se model chová podobně jako náhodný tip. Platí tedy:

- hodnota 0.5 (červená tečkovaná čára na grafu níže, plocha se rovná polovině čtverce 1.0×1.0) znamená, že model je na úrovni **házení mincí** – nedokáže rozlišit mezi pozitivními a negativními případy,
- hodnota 1 (fialová tečka na grafu níže, plocha se rovná celému čtverci 1.0×1.0) znamená, že model dokáže **perfektně odhalit** pozitivní a negativní případy bez ohledu na to, kde nastavíme práh.

Každý bod na jednotlivé křivce odpovídá jednomu konkrétnímu nastavení prahu – celá křivka tedy ukazuje výkon modelu pro všechny možné varianty prahu.



Osy grafu mají tento význam:

- „**true positive rate**“ (také nazvaný „recall“) udává, jak velkou část skutečně pozitivních případů model správně zachytí, čím vyšší hodnota, tím méně případů, které jsou důležité, model přehlédl,
- „**false positive rate**“ udává, jak velkou část skutečně negativních případů model chybně označil jako pozitivní, čím nižší hodnota, tím méně falešných poplachů model vydal.

Níže je uvedena tabulka č. 1, která přehledně ukazuje rozsah hodnot klasifikačních metrik a jejich základní interpretaci. Slouží jako pomůcka pro rychlou a základní orientaci při posuzování kvality modelu.

Tabulka 1: Tabulka typických hodnot metrik klasifikace a jejich vysvětlení.

Metrika	Typické hodnoty	Interpretace
Accuracy	0–1	<div><div>Hodnoty blízke 1 (100 %)</div><div>velmi dobré</div></div> <p>U nevyvážených dat však může být zavádějící (např. 95 % accuracy v případě, když model vždy tipuje většinu, znamená, že sice správně určil většinu případů, ale nikdy nerozpoznal tu menšinovou třídu). Proto sama o sobě nemusí být dostatečná.</p>
Precision	0–1	<div><div>Blízko 1</div><div>velmi dobré (model má málo falešných poplachů)</div></div> <div><div>Blízko 0</div><div>špatné (většina pozitivních predikcí je omyl)</div></div> <p>Vysoká precision je důležitá tam, kde je problém falešně označit něco jako pozitivní (např. zdravého pacienta jako nemocného).</p>
Recall	0–1	<div><div>Blízko 1</div><div>velmi dobré (model zachytí téměř všechny pozitivní případy)</div></div> <div><div>Blízko 0</div><div>špatné (většinu pozitivních případů přehlédne)</div></div> <p>Důležité tam, kde nesmí nic uniknout (např. diagnostika nemocí).</p>

F1-score	0–1	Blízko 1 velmi dobré , vyvážený model
		Blízko 0 špatné , model selhává
Používá se jako kompromisní měřítko mezi precision a recall.		
ROC-AUC	0.5 náhodný tip, 1 perfektní tip, <0.5 horší než náhoda	>0.9 výborný model
		0.8–0.9 dobrý+
		0.7–0.8 příjemný
		<0.7 slabý
		0.5 náhodný tip (model nic neumí)
		<0.5 model je horší než náhoda (což znamená, že by šlo výsledky jen převrátit a dostat lepší výkon)

Jednoduchý příklad s použitím knihovny scikit-learn:

```
from sklearn.metrics import accuracy_score, precision_score, recall_score,
f1_score, roc_auc_score

# Příklad skutečných a predikovaných tříd
y_pravdivé = [0, 1, 1, 0, 1, 0, 1]
y_predikovane = [0, 1, 0, 0, 1, 0, 1]
y_pravdepodobnosti = [0.1, 0.9, 0.4, 0.2, 0.8, 0.3, 0.7] # pravděpodobnosti
pro ROC-AUC

print("Accuracy:", accuracy_score(y_pravdivé, y_predikovane))
print("Precision:", precision_score(y_pravdivé, y_predikovane))
print("Recall:", recall_score(y_pravdivé, y_predikovane))
print("F1 score:", f1_score(y_pravdivé, y_predikovane))
print("ROC-AUC:", roc_auc_score(y_pravdivé, y_pravdepodobnosti))
```

Regrese

- a. MSE („Mean Squared Error“ – průměrná kvadratická chyba) – představuje průměr kvadrátů rozdílů mezi predikovanými a skutečnými hodnotami, velké chyby jsou výrazně penalizovány:

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

kde: n ... celkový počet vzorků, y_i ... skutečná hodnota pro i -tý vzorek, $\hat{y}_i$... predikovaná hodnota pro i -tý vzorek.

Vysvětlení:

MSE udává průměr velikostí chyb mezi skutečnými a předpovězenými hodnotami. Každá chyba se přitom umocní, takže velké odchylky mají na výslednou hodnotu mnohem větší vliv než malé. Představme si příklad: model předpovídá ceny bytů. U většiny bytů je odhad docela blízko – například skutečná cena je 3 000 000 Kč a model odhadne 2 990 000 Kč (chyba 10 tisíc). Jenže u jednoho bytu se model spletl výrazně – skutečná cena je 3 000 000 Kč, ale model odhadne 3 200 000 Kč (chyba 200 tisíc). Právě tato velká odchylka způsobí, že hodnota MSE se výrazně zvětší, i když ostatní chyby byly malé.

- b. RMSE („Root Mean Squared Error“ – odmocněná kvadratická chyba) – představuje odmocninu z MSE a ukazuje průměrnou chybu ve stejných jednotkách jako výstupní veličina:

$$RMSE = \sqrt{\frac{1}{n} \sum_{i=1}^n (y_i - \hat{y}_i)^2},$$

kde: n ... celkový počet vzorků, y_i ... skutečná hodnota pro i -tý vzorek, $\hat{y}_i$... predikovaná hodnota pro i -tý vzorek.

Vysvětlení:

RMSE vychází ze stejného principu jako MSE, jen na závěr vezme odmocninu, aby se výsledek vrátil zpět do původních jednotek (např. v korunách). Představme si příklad: pokud vezmeme stejnou situaci s odhady cen bytů, kde většina chyb je kolem 10 tisíc a jedna chyba je 200 tisíc, MSE vyjde jako poměrně velké číslo, které se špatně interpretuje (je obtížně pochopitelné). RMSE však poskytne výsledek v korunách – například 15 tisíc. Můžeme tedy přímo říct, že se model v průměru mylí zhruba o 15 tisíc korun. Díky tomu je RMSE pro praxi srozumitelnější než MSE.

- c. MAE („Mean Absolute Error“ – průměrná absolutní chyba) – představuje průměr absolutních rozdílů mezi predikovanými a skutečnými hodnotami, jedná se o méně citlivý ukazatel na extrémní odchylky než například MSE:

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

kde: n ... celkový počet vzorků, y_i ... skutečná hodnota pro i -tý vzorek, $\hat{y}_i$... predikovaná hodnota pro i -tý vzorek.

Vysvětlení:

MAE počítá průměr všech chyb, ale každou bere stejně – velké odchylky nemají větší váhu než malé, jako tomu bylo u MSE. Mějme příklad: pokud model opět předpovídá ceny bytů a u čtyř bytů se splete o 5 tisíc, 10 tisíc a 20 tisíc korun, pak MAE vyjde 12 500 Kč. To znamená, že průměrná odchylka modelu od skutečných cen je 12.5 tisíce korun. I kdyby se u jednoho bytu spletl opravdu hodně, třeba o 200 tisíc korun, tento jeden případ výslednou hodnotu MAE tolik nezvětší jako u MSE, proto se MAE hodí tam, kde chceme všechny chyby posuzovat stejně.

- d. R^2 (koeficient determinace) – ukazuje, jak dobře model vystihuje vztah mezi vstupy a skutečnými výstupy, matematicky hodnota vyjadřuje, jak velký podíl rozptylu (variability) skutečných hodnot dokáže model vysvětlit svou predikcí:

$$R^2 = 1 - \frac{\sum_{i=1}^n (y_i - \hat{y}_i)^2}{\sum_{i=1}^n (y_i - \bar{y})^2},$$

kde: n ... celkový počet vzorků, y_i ... skutečná hodnota pro i -tý vzorek, $\hat{y}_i$... predikovaná hodnota pro i -tý vzorek, $\bar{y}$... průměr skutečných hodnot.

Vysvětlení:

R^2 říká, jak dobře model vystihuje rozdíly mezi skutečnými hodnotami. Hodnota 1 znamená, že model vše předpověděl dokonale. Hodnota 0 znamená, že model není lepší než velmi jednoduchý přístup: kdybychom u všech bytů odhadli jen stejnou hodnotu – například průměrnou cenu v dané oblasti, ve které se byty nacházejí. Záporné hodnoty této metriky ukazují, že si model vede dokonce hůř než toto jednoduché „hádání průměru“. Mějme příklad: uvažujme data obsahující ceny bytů, u kterých je průměrná cena 3 000 000 Kč. Pokud bychom pro každý byt řekli: „odhad je 3 miliony“, dostali bychom model s $R^2 = 0$. Skutečnost je ale taková, že některé byty stojí méně (menší rozloha, horší lokalita) a jiné více (větší byty, lepší poloha). Pokud náš model vezme tyto faktory v úvahu a dostáhne $R^2 = 0.8$, znamená to, že dokáže vysvětlit 80 % rozdílů v cenách bytů. Zbýlých 20 % už nevystihne, protože závisí na faktorech, které model vůbec nezná – třeba stav budovy, výhled z okna nebo konkrétní patro.

Níže je uvedena tabulka č. 2, která přehledně ukazuje rozsah hodnot regresních metrik a jejich základní interpretaci. Slouží jako pomůcka pro rychlou a základní orientaci při posuzování kvality modelu.

Metrika	Interpretace	
MSE	Nízké hodnoty	velmi dobré Hodnota blížící se nule znamená malé chyby.
	Vysoké hodnoty	špatné I malé odchylky se kvadraticky zvětšují.
RMSE	Nízké hodnoty	velmi dobré
	Vysoké hodnoty	špatné
MAE	Nízké hodnoty	velmi dobré Udává průměrnou chybu bez kvadratického zvýraznění extrémů. Pokud je MAE blízko 0, model předpovídá velmi přesně.
	Vysoké hodnoty	špatné
R ²	Hodnoty blízke 1	velmi dobré Model velmi dobře vysvětluje data, predikce jsou přesné.
	Hodnoty blízke 0	špatné Model nepřináší žádné zlepšení oproti jednoduchému odhadu průměrné hodnoty.

	Hodnoty pod 0	velice špatné Model předpovídá hůře než tento jednoduchý odhad.
--	---------------	---

Tabulka 2: Tabulka typických hodnot metrik regrese a jejich vysvětlení.

Jednoduchý příklad s použitím knihovny scikit-learn:

```
from sklearn.metrics import mean_squared_error, mean_absolute_error, r2_score
import numpy as np

# Příklad skutečných a predikovaných hodnot
y_pravdivé = [3.0, -0.5, 2.0, 7.0]
y_predikované = [2.5, 0.0, 2.1, 7.8]

print("MSE:", mean_squared_error(y_pravdivé, y_predikované))
print("RMSE:", np.sqrt(mean_squared_error(y_pravdivé, y_predikované)))
print("MAE:", mean_absolute_error(y_pravdivé, y_predikované))
print("R2:", r2_score(y_pravdivé, y_predikované))
```

Metriky prakticky – využití knihovny scikit-learn

Tato kapitola ukazuje praktické využití metrik v knihovně scikit-learn při práci s klasifikačními modely. Na příkladu logistické regrese si ukážeme, jak lze vedle prosté „accuracy“ sledovat i další ukazatele, jako jsou „precision“, „recall“, „F1-score“ nebo „ROC-AUC“. Díky tomu můžeme nejen posoudit, zda náš model překonává jednoduchý „baseline“ model (většinou třídu), ale také naladit vhodný rozhodovací práh a vizualizovat chování modelu pomocí ROC křivky.

A. import potřebných knihoven

```
import numpy as np
import matplotlib.pyplot as plt

from sklearn.datasets import make_classification
from sklearn.model_selection import train_test_split
from sklearn.dummy import DummyClassifier
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import (
    accuracy_score, balanced_accuracy_score, precision_score, recall_score,
    f1_score, roc_auc_score, average_precision_score, confusion_matrix,
    roc_curve, precision_recall_curve
)
```

B. příprava dat

```
X, y = make_classification(
    n_samples=6000, n_features=20, n_informative=6, n_redundant=4,
    class_sep=1.2, weights=[0.9, 0.1], flip_y=0.01, random_state=42
)

# Rozdělení: trénink (60 %) / validace (20 %) / test (20 %)
X_trenovaci, X_docasne, y_trenovaci, y_docasne = train_test_split(
    X, y, test_size=0.4, stratify=y, random_state=42
)
```

```
X_validacni, X_testovaci, y_validacni, y_testovaci = train_test_split(
    X_docasne, y_docasne, test_size=0.5, stratify=y_docasne, random_state=42
)
```

C. tvorba většinového klasifikátoru – takový klasifikátor (nazvaný jako „baseline“), který vždy predikuje třídu, která se v daném souboru objevuje nejčastěji – funguje zde pro srovnání

```
baseline = DummyClassifier(strategy="most_frequent")
baseline.fit(X_trenovaci, y_trenovaci)
y_predikce_vetsinova = baseline.predict(X_testovaci)

print("=== BASELINE (většinová třída) - TEST ===")
print("Accuracy:          ", accuracy_score(y_testovaci,
y_predikce_vetsinova))
print("Precision:          ", precision_score(y_testovaci,
y_predikce_vetsinova, zero_division=0))
print("Recall:             ", recall_score(y_testovaci,
y_predikce_vetsinova))
print("F1 skóre:           ", f1_score(y_testovaci,
y_predikce_vetsinova))
print("Matice záměn:\n", confusion_matrix(y_testovaci, y_predikce_vetsinova))
```

D. tvorba ML klasifikátoru prostřednictvím logistické regrese

```
model = LogisticRegression(max_iter=1000, class_weight="balanced",
solver="liblinear", random_state=42)
model.fit(X_trenovaci, y_trenovaci)

# Pravděpodobnosti pro validaci a test
y_pravdepodobnosti_validacni = model.predict_proba(X_validacni)[: , 1]
y_pravdepodobnosti_testovaci = model.predict_proba(X_testovaci)[: , 1]
```

E. ladění prahu podle F1 metriky na validační sadě dat

```
prahy = np.linspace(0.05, 0.95, 91)
f1_hodnoty = []
for aktualni_prah in prahy:
    y_predikce_validacni = (y_pravdepodobnosti_validacni >=
aktualni_prah).astype(int)
    f1_hodnoty.append(f1_score(y_validacni, y_predikce_validacni))

index_nejlepsiho_prahu = int(np.argmax(f1_hodnoty))
prah_nejlepsi = prahy[index_nejlepsiho_prahu]
print(f"\nNejlepší práh podle F1 na VALIDACI: {prah_nejlepsi:.3f}
(F1={f1_hodnoty[index_nejlepsiho_prahu]:.3f})")
```

F. Vyhodnocení na testovacích datech – výchozí práh 0.5

```
y_predikce_vychozi_prah = (y_pravdepodobnosti_testovaci >= 0.5).astype(int)
print("\n=== LOGISTICKÁ REGRESE - TEST (výchozí práh 0.5) ===")
print("Accuracy:          ", accuracy_score(y_testovaci,
y_predikce_vychozi_prah))
print("Precision:          ", precision_score(y_testovaci,
y_predikce_vychozi_prah, zero_division=0))
print("Recall:             ", recall_score(y_testovaci,
y_predikce_vychozi_prah))
```

```
print("F1-score: ", f1_score(y_testovaci,
y_predikce_vychazi_prah))
print("ROC-AUC (z pravděpodobností): ", roc_auc_score(y_testovaci,
y_pravdepodobnosti_testovaci))
print("Matice záměn:\n", confusion_matrix(y_testovaci,
y_predikce_vychazi_prah))
```

G. Vyhodnocení na testovacích datech – naladěný práh

```
y_predikce_naladeny_prah = (y_pravdepodobnosti_testovaci >=
prah_nejlepsi).astype(int)
print("\n=== LOGISTICKÁ REGRESE - TEST (naladěný práh) ===")
print("Accuracy: ", accuracy_score(y_testovaci,
y_predikce_naladeny_prah))
print("Precision: ", precision_score(y_testovaci,
y_predikce_naladeny_prah, zero_division=0))
print("Recall: ", recall_score(y_testovaci,
y_predikce_naladeny_prah))
print("F1-score: ", f1_score(y_testovaci,
y_predikce_naladeny_prah))
print("ROC-AUC (z pravděpodobností): ", roc_auc_score(y_testovaci,
y_pravdepodobnosti_testovaci))
print("Matice záměn:\n", confusion_matrix(y_testovaci,
y_predikce_naladeny_prah))
```

H. Vykreslení grafů ROC-AUC metriky

```
false_positive_rate, true_positive_rate, _ = roc_curve(y_testovaci,
y_pravdepodobnosti_testovaci)

plt.figure()
plt.plot(false_positive_rate, true_positive_rate,
         label=f"ROC (AUC={roc_auc_score(y_testovaci,
y_pravdepodobnosti_testovaci):.3f})")
plt.plot([0, 1], [0, 1], "--", label="náhoda")
plt.xlabel("False Positive Rate")
plt.ylabel("True Positive Rate (Recall)")
plt.title("ROC křivka (TEST)")
plt.legend()
plt.show()
```

Co to je matice záměn (neboli „confusion matrix“)?

Matice záměn je tabulka, která popisuje chování klasifikačního modelu při rozpoznávání jednotlivých tříd. Na rozdíl od prosté „accuracy“ neříká jen to, kolik předpovědí bylo správně, ale také jakých chyb se model dopustil. Používá se především u binární klasifikace, což je situace, kdy se každý případ zařazuje do jedno ze dvou tříd, například „zdravý“, nebo „nemocný“ či „spam“, nebo „nespam“. Taková matice má čtyři základní buňky a vypadá následovně:

$$\text{matice záměn} = \begin{bmatrix} TP & FP & FN & TN \end{bmatrix}$$

a konkrétní matice záměn tedy může vypadat například takto:

$$\text{matice záměn} = \begin{bmatrix} 1027 & 48 & 41 & 84 \end{bmatrix}.$$

První řádek ukazuje, že 1027 negativních případů bylo správně rozpoznáno a 48 jich bylo chybně označeno jako pozitivní. Druhý řádek znamená, že 41 pozitivních případů model přehlédl a 84 správně odhalil. Matice záměn je velmi užitečná, protože nám umožňuje vidět, jakým způsobem model chybí. Pokud je FN vysoký, znamená to, že model přehlíží mnoho pozitivních případů, což může být například v lékařské diagnostice velmi nebezpečné. Pokud je naopak vysoký počet FP, model často vydává falešné poplachy, což může být problém třeba u filtrování e-mailů, kde by legitimní zprávy mohly končit ve složce spamu.

1.7 Vysvětlení dalších důležitých pojmů

Trénovací data

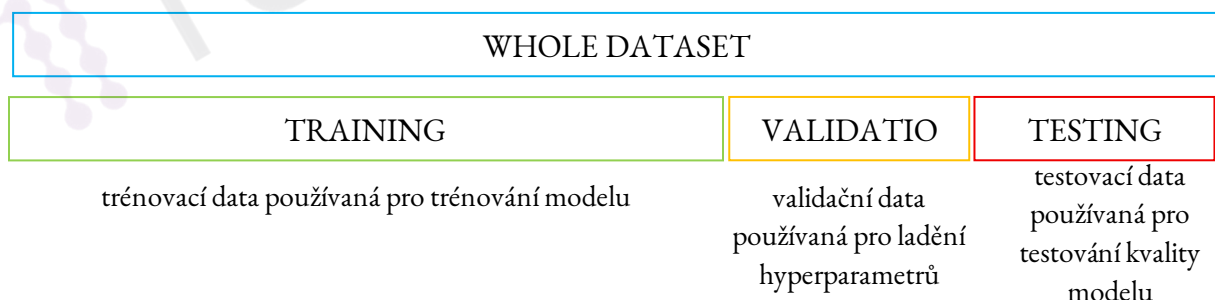
Trénovací data slouží k samotnému učení/trénování modelu. V případě rozhodovacích stromů je model na základě trénovacích dat budován tak, že hledá optimální rozdělení uzlů podle zvoleného kritéria (Giniho index, entropie aj.). Rozhodovací strom při trénování postupně vytváří větvení, která co nejlépe oddělují jednotlivé třídy nebo predikují cílovou hodnotu.

Validační data

Validační data se používají k ladění hyperparametrů modelu, tedy nastavení hodnot, které určují strukturu a chování vybraného ML modelu, ale nejsou přímo odhadovány ze vstupních dat. U rozhodovacích stromů se takto určuje například maximální hloubka (*max_depth*), minimální počet vzorků v listu/posledním uzlu (*min_samples_leaf*) aj. Výkon (jak je model přesný) modelu je během jeho trénování na trénovací sadě dat vyhodnocena pomocí validační sady. Podle výsledků tohoto výkonu se poté parametry rozhodovacího stromu upravují tak, aby se dosáhlo nejlepší rovnováhy mezi složitostí stromu a jeho schopností zobecnění na dosud neviděná data.

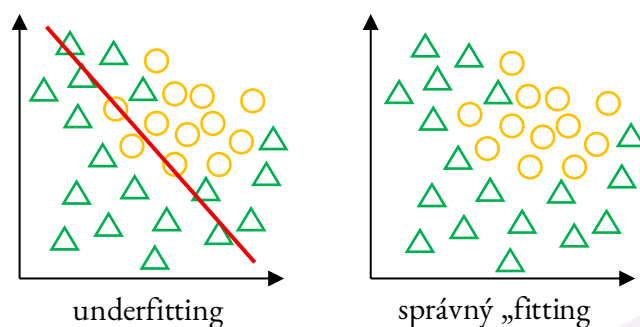
Testovací data

Testovací data slouží k finálnímu ověření výkonu modelu po jeho natrénování a vyladění. Tato data model nikdy během trénovací ani validační fáze „neviděl“, takže poskytují realistický odhad toho, jak bude model fungovat na zcela nových neviděných datech. U rozhodovacího stromu se v testovací sadě měří například „accuracy“, tzv. F1-score nebo jiné metriky vhodné pro daný úkol.



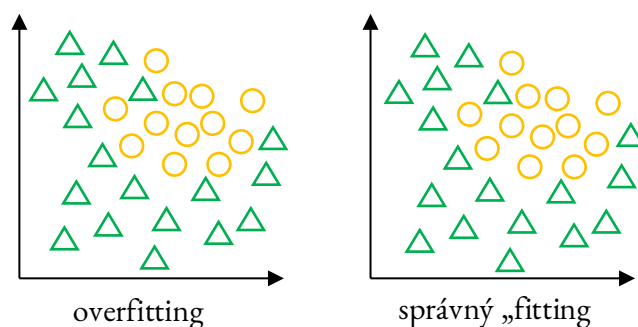
Nedostatečné naučení modelu („underfitting“)

Pojem „underfitting“ označuje situaci, kdy je model příliš jednoduchý na to, aby dokázal zachytit i základní vzory objevující se v datech. V důsledku toho dosahuje nízké přesnosti nejen na testovacích datech, ale i na datech trénovacích. V kontextu rozhodovacích stromů se „underfitting“ projevuje například příliš malou hloubkou stromu, kdy je k dispozici pouze velmi omezený počet rozhodovacích uzlů (listů, vysvětleno později v tomto kurzu). Rozhodovací strom tak může vyhodnocovat pouze jednoduché otázky, což vede k tomu, že model není schopen správně řešit složitější případy a jeho schopnost predikce je omezená. Na obrázku níže je „underfitting“ vyjádřen graficky na klasifikačním problému a porovnán se správným „fittingem“.



Přeučení modelu („overfitting“)

Pojem „overfitting“ označuje stav, kdy je model naopak příliš složitý a naučí se nejen užitečné vzory objevující se v datech, ale i náhodné odchylky a šum, které jsou specifické pro trénovací data. Takový model dosahuje velmi vysoké přesnosti na trénovacích datech, často blízké 100 %, avšak jeho výkon na nových doposud neviděných datech je podstatně horší. U rozhodovacích stromů se „overfitting“ projevuje tím, že strom roste do velké hloubky a vytváří větvení, které dokonale oddělí trénovací případy, často až na jednotlivé samotné datové body. Výsledkem jsou extrémně specifická pravidla, která nejsou obecně platná a ztrácí schopnost predikce pro nové situace. Na obrázku níže je „overfitting“ vyjádřen graficky na klasifikačním problému a porovnán se správným „fittingem“.



Hyperparametry („hyperparameters“)

Pojem „hyperparametry“ modelu strojového učení označuje hodnoty, které určují chování a nastavení modelu strojového učení ještě před samotným trénováním. Na rozdíl od parametrů modelu, které se model učí automaticky během tréninku (například váhy v neuronové síti), jsou hyperparametry voleny uživatelem nebo metodami ladění (myšleno jako metody ladění trénování modelu strojového učení). Patří sem například maximální hloubka rozhodovacího stromu, počet

skrytých vrstev v neuronové síti nebo rychlost učení (tzv. „learning rate“) u optimalizačních algoritmů. Výběr vhodných hyperparametrů má zásadní vliv na přesnost výsledného modelu. Pokud jsou nastaveny nevhodné, může dojít buď k nedostatečnému naučení modelu („underfitting“, vysvětleno již dříve v této kapitole), nebo naopak k přílišnému přizpůsobení se trénovacím datům („overfitting“, vysvětleno již dříve v této kapitole). V praxi se proto často používají techniky systematického vyhledávání, které uživateli pomáhají najít takovou kombinaci hyperparametrů, která vede k nejlepším výsledkům.

