Praktické aplikace strojového učení

Jan Mittner

Pokyny k online schůzce

1. Používejte nejlépe desktopovou aplikaci Teams.

2. Pokud nekladete dotaz nebo se neúčastníte diskuze, mějte prosím vypnutý mikrofon.

3. Pokud jen trochu můžete, mějte puštěnou kameru. Vás výraz tváře pomůže vyučujícímu :)

4. Jak můžete položit dotaz:
   a. Zapněte si mikrofon a rovnou se zeptejte.
      nebo:
   b. Napište dotaz do chatu. nebo:
   c. Použijte tlačítko zvednout ruku.
      (Po vyvolání ruku sundejte)

Agenda

- cíl kurzu
- požadavky na ukončení
- potřebné vstupní znalosti
- obsah kurzu
- úvod do strojového učení

Proč jste si kurz zapsali a co od něj očekáváte?

Cíl kurzu

- vysvětlit principy strojového učení
- představit nejpoužívanější algoritmy a modely
- ukázat typické úlohy řešené pomocí strojového učení
- naučit se pracovat s nástroji pro praktické aplikace strojového učení
- vyzkoušet si realizaci projektu strojového učení od začátku do konce

Požadavky na ukončení

- semestrální práce 50 bodů
- závěrečný test 50 bodů

- nutno získat alespoň polovinu bodů jak z testu tak za semestrální práci
- na cvičeních možné dostat bonusové body za realizaci vybraných úkolů

- pro úspěšné absolvování alespoň 60 bodů celkem

Potřebné vstupní znalosti a zázemí

Aby mělo smysl kurz vůbec absolvovat:

- základy programování (alespoň v rozsahu povinných kurzů FIS VŠE)
- základy matematiky (zejména témata jako funkce, vektory, matice, ...)
- základy statistiky

Velmi pomůže také:

- znalost Pythonu (alespoň z hlediska syntaxe a základních knihoven - viz dále)
- mít k dispozici výkonný počítač s GPU Nvidia (nebo cloudové služby)

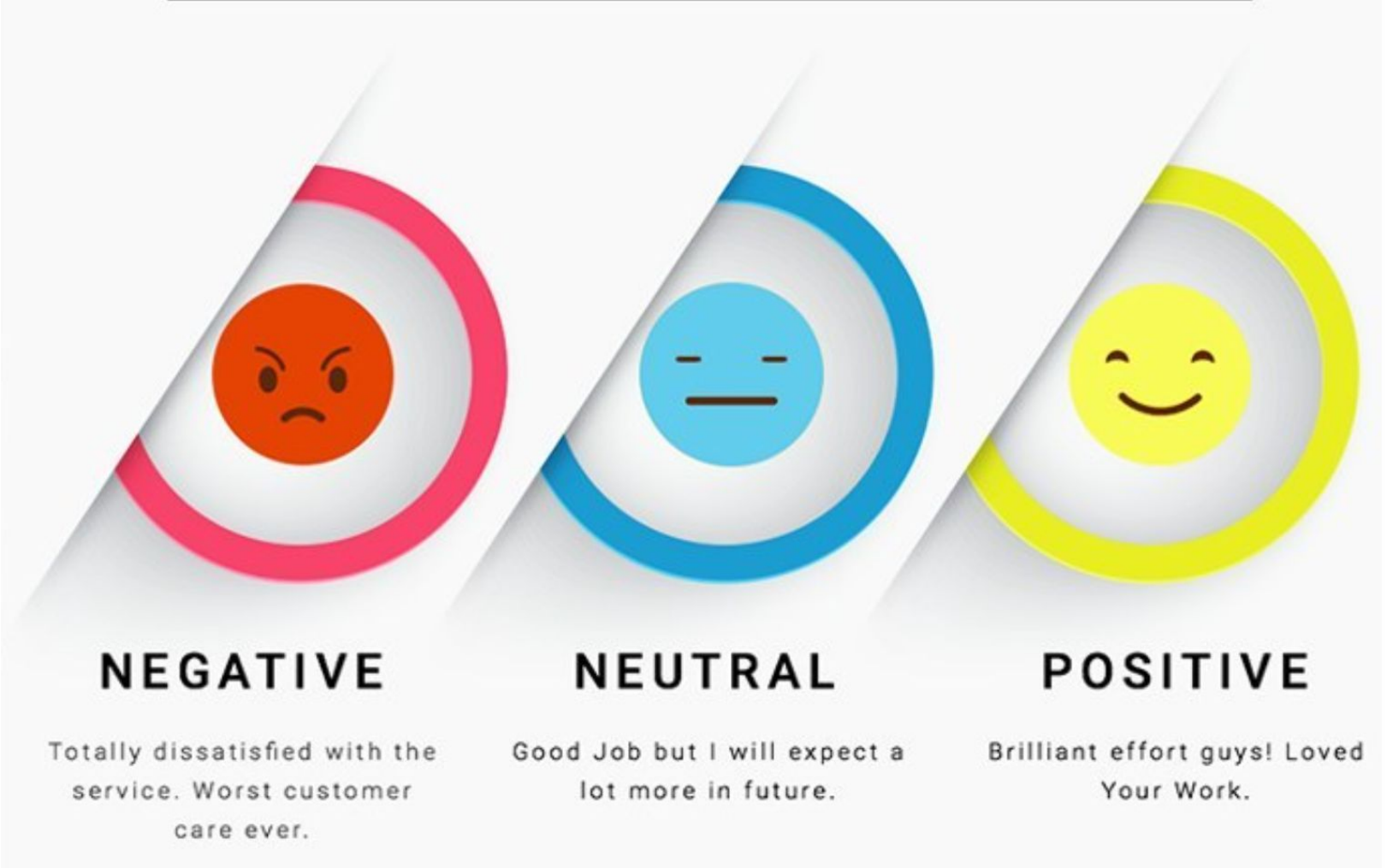
Obsah kurzu

1. Úvod do strojového učení
   - hlavní pojmy, historie, současnost
   - algoritmy, typy úloh, výzvy a problémy
2. Základní praktiky strojového učení
   - doporučený pracovní postup
   - projekt ML od začátku do konce
3. Klasifikace entit
   - binární a multiclass klasifikace
   - prakticky: rozpoznávání psaných číslic
4. Trénování modelů
   - Gradient Descent, regresní modely
   - prakticky: Titanic dataset
5. Pokročilé metody klasifikace
   - Support Vector Machines
   - Rozhodovací stromy, Random Forest
6. Unsupervised Learning. Doporučovací systémy
   - shlukování, redukce dimenzionality, detekce anomalit
   - doporučovací systémy
7. Neuronové sítě a Deep Learning
   - principy a klíčové pojmy, framework Keras
   - binární a multiclass klasifikace, regrese
8. Deep Learning pro počítačové vidění
   - konvoluční neuronové sítě
   - prakticky: rozpoznávání objektů v obrazu
9. Deep Learning pro texty a sekvenční data
   - word embeddings, rekurentní neuronové sítě
   - chatbot
10. Pokročilé techniky Deep Learningu
    - modely s více vstupy a více výstupy
    - TensorBoard
11. Generativní Deep Learning
    - generování textu, přenos stylu obrazů
    - variační autoenkodéry, GAN
12. Reinforcement Learning
    - principy a hlavní přístupy RL
    - Deep Q-Learning, knihovna TF-Agents

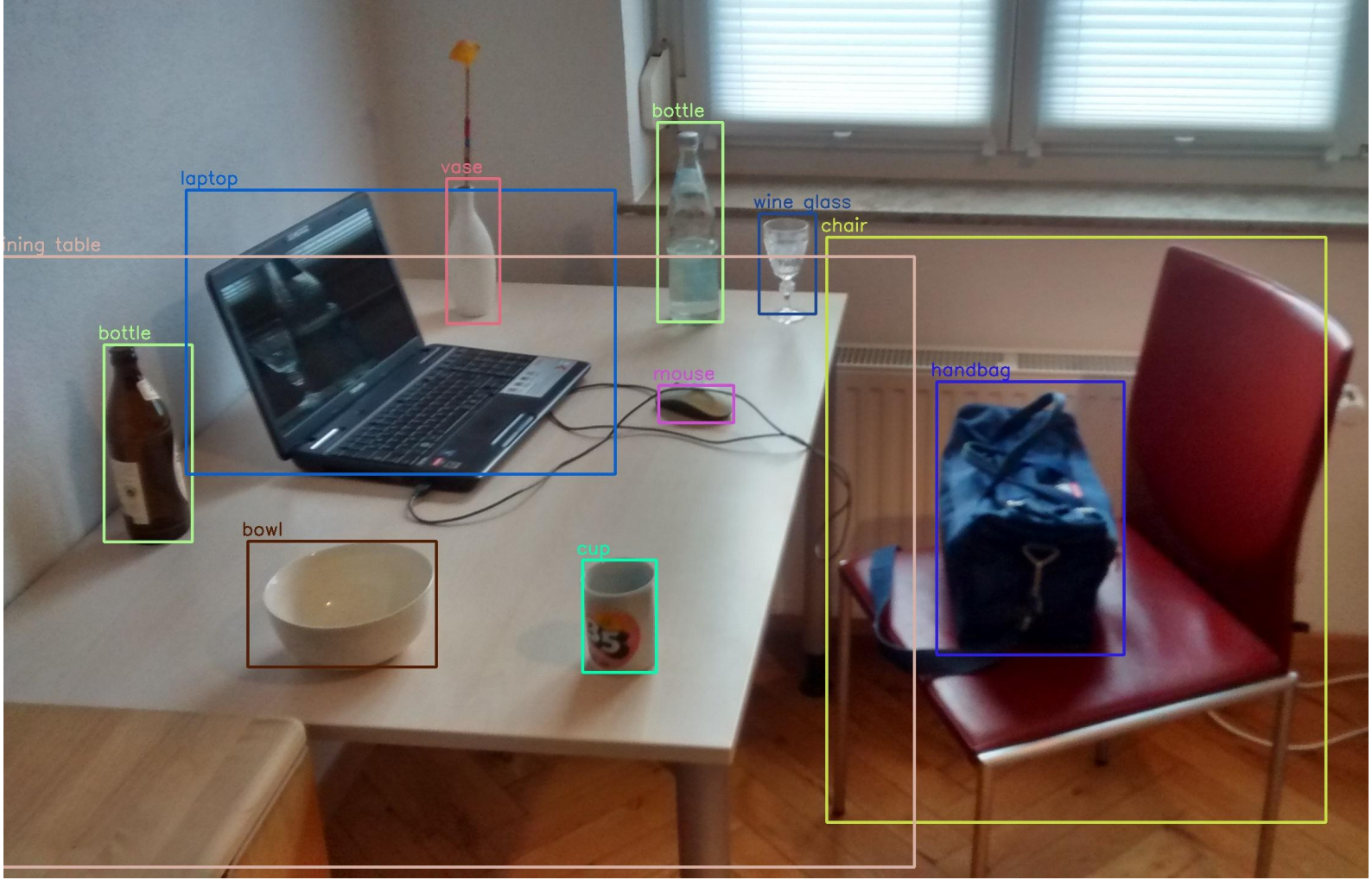
Co si tedy prakticky vyzkoušíme?

- rozpoznávání číslic z psaného textu
- predikce cen nemovitostí v dané lokalitě
- segmentace a určení podobnosti textu
- analýza sentimentu textu
- určení témat obsažených v textu
- doporučování filmů ke zhlédnutí
- rozpoznávání objektů v obrazu
- předpověď teploty z časové řady meteo údajů
- prediktivní psaní textu
- vytvoření chatbota
- neuronový přenos stylu obrazů
- vytvoření agenta, který hraje staré Atari hry lépe než člověk

Zdroj: https://en.wikipedia.org/wiki/MNIST_database

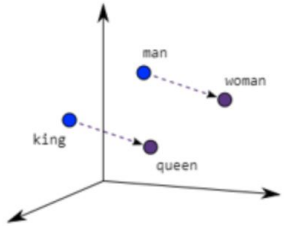


Zdroj: https://www.kdnuggets.com/2018/03/5-things-sentiment-analysis-classification.html

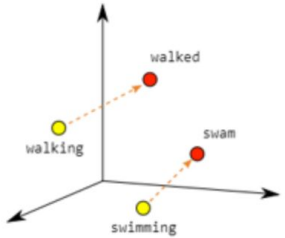


Zdroj: https://en.wikipedia.org/wiki/Object_detection

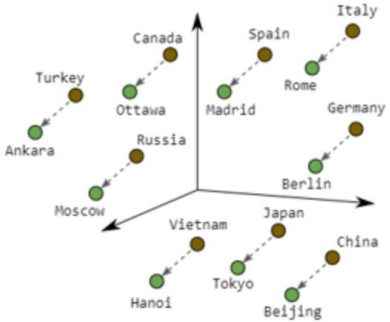
Word2Vec



Male-Female

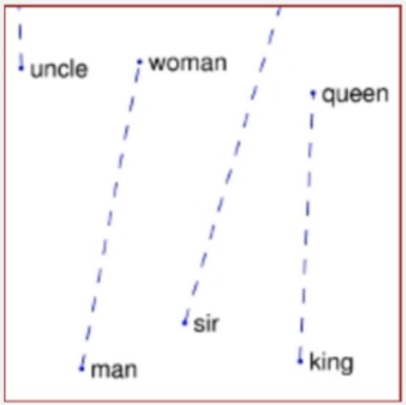


Verb Tense

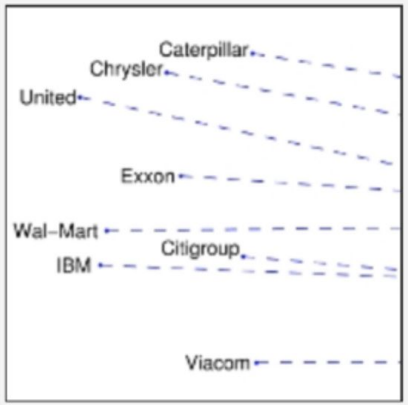


Country-Capital

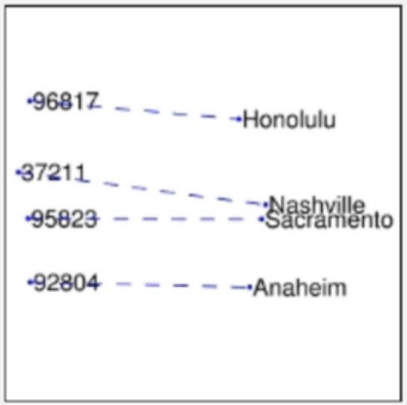
GloVe



man - woman



company - ceo



city - zip code

comparative - superlative

Zdroj: https://towardsdatascience.com/word-embeddings-for-nlp-5b72991e01d4

1 Upload photo

The first picture defines the scene you would like to have painted.

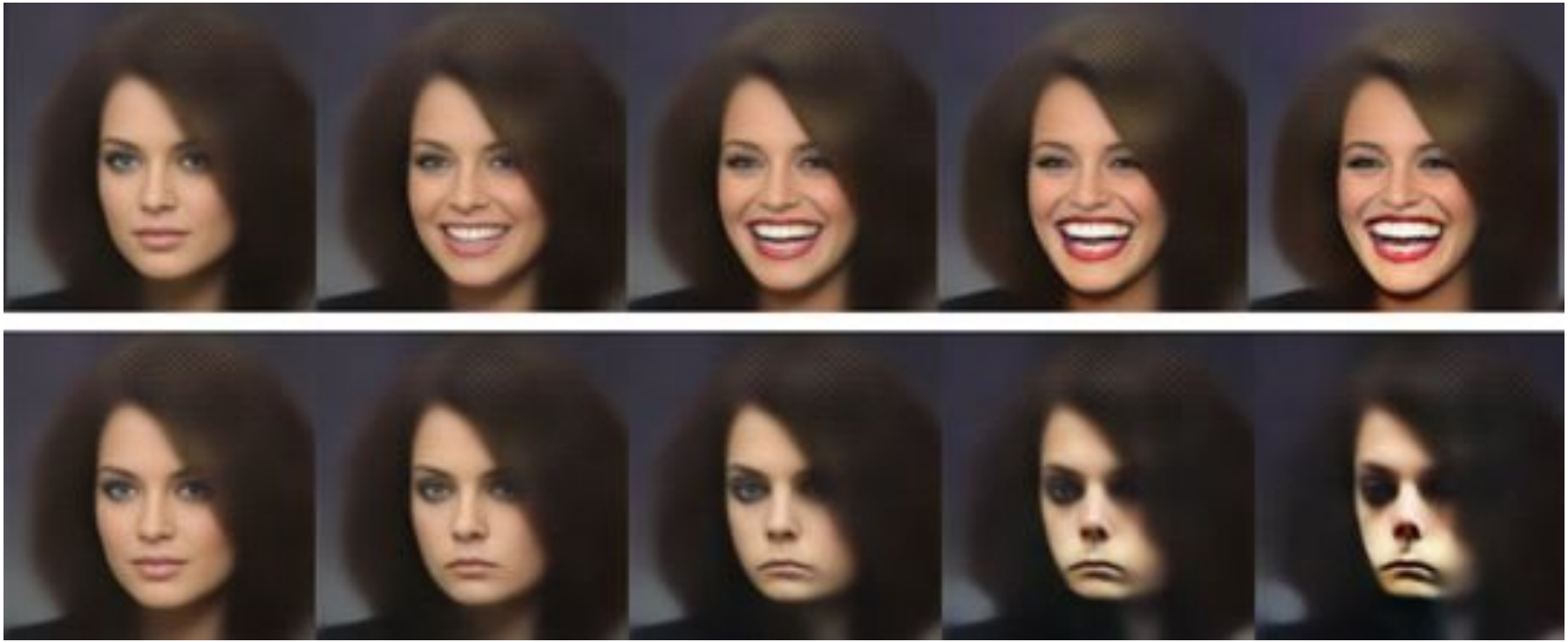
2 Choose style

Choose among predefined styles or upload your own style image.

3 Submit

Our servers paint the image for you. You get an email when it's done.

Zdroj: https://jvns.ca/blog/2017/02/12/neural-style/



TEXT PROMPT an armchair in the shape of an avocado. an armchair imitating an avocado.

AI-GENERATED IMAGES

Zdroj: https://openai.com/blog/dall-e/

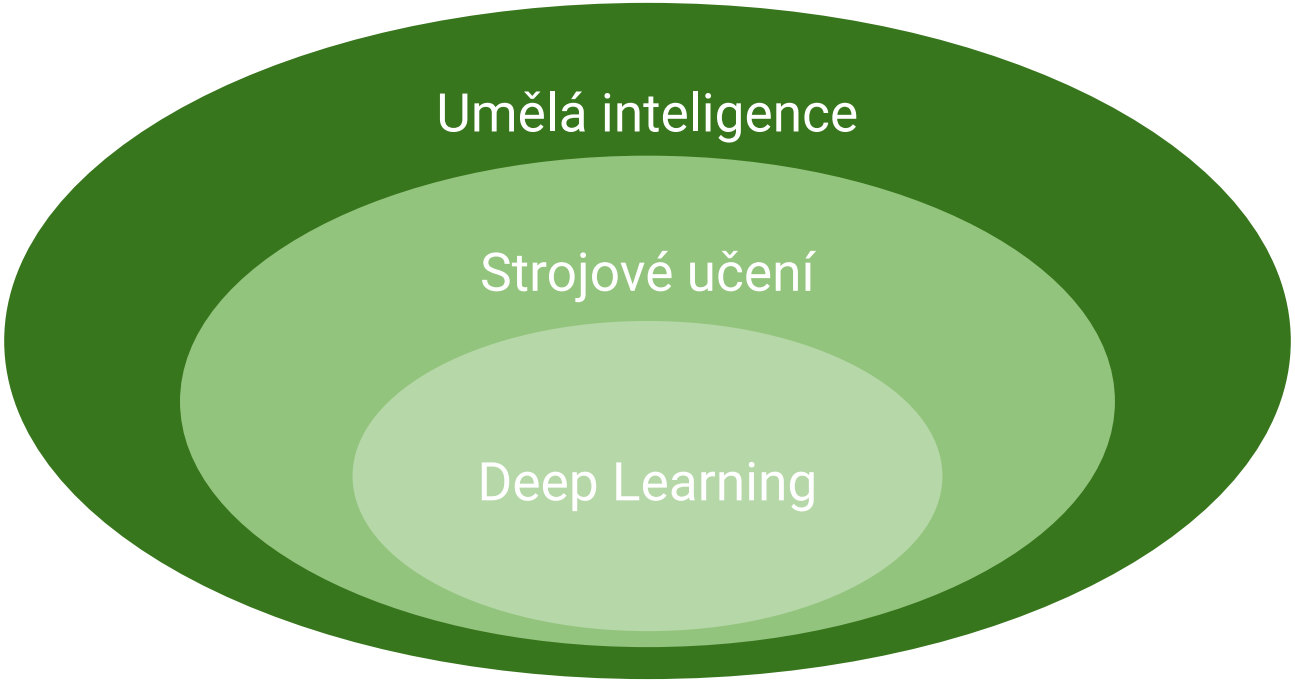
1. Úvod do strojového učení

Strojové učení (Machine Learning)

- inženýrská věda (a umění) zabývající se schopností počítačů učit se na základě dat

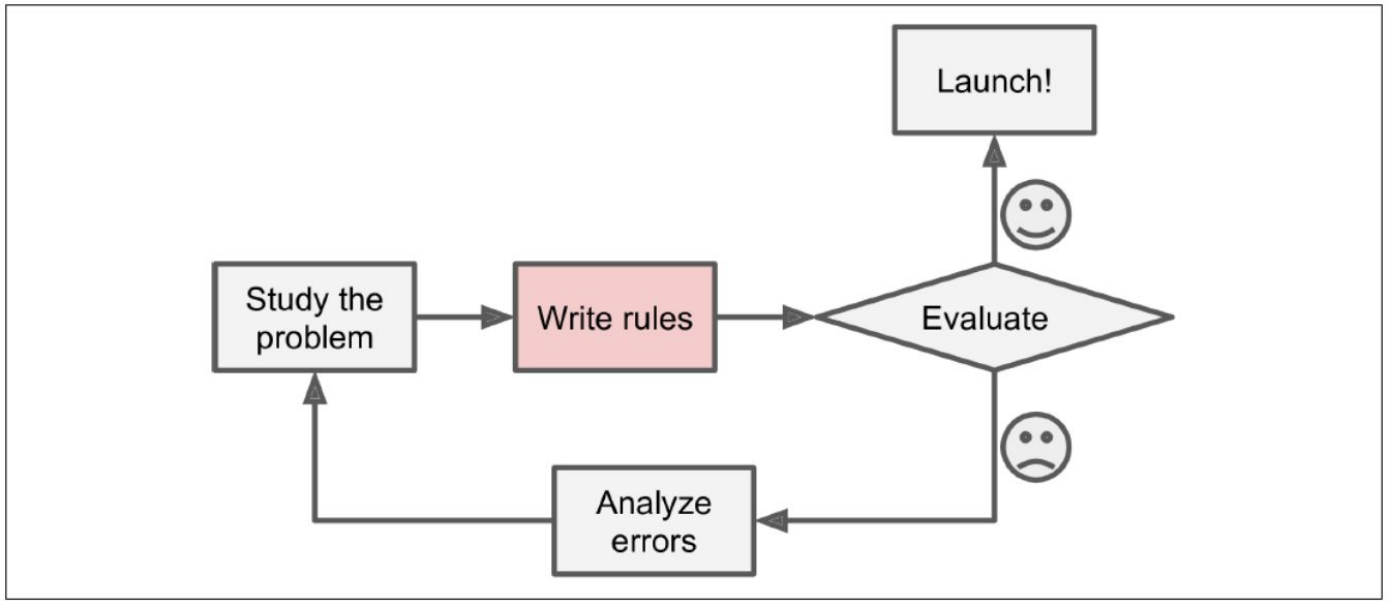
- příklad: SPAM filtr jako program strojového učení, který se naučí rozpoznávat spam na základě příkladů spamových e-mailů (označených lidmi) a běžných e-mailů
- příklady, podle kterých se systém učí = trénovací množina (training set)
    - jeden trénovací příklad = training instance / sample

Umělá inteligence vs. strojové učení vs. Deep Learning



Odlišnost strojového učení od klasických přístupů

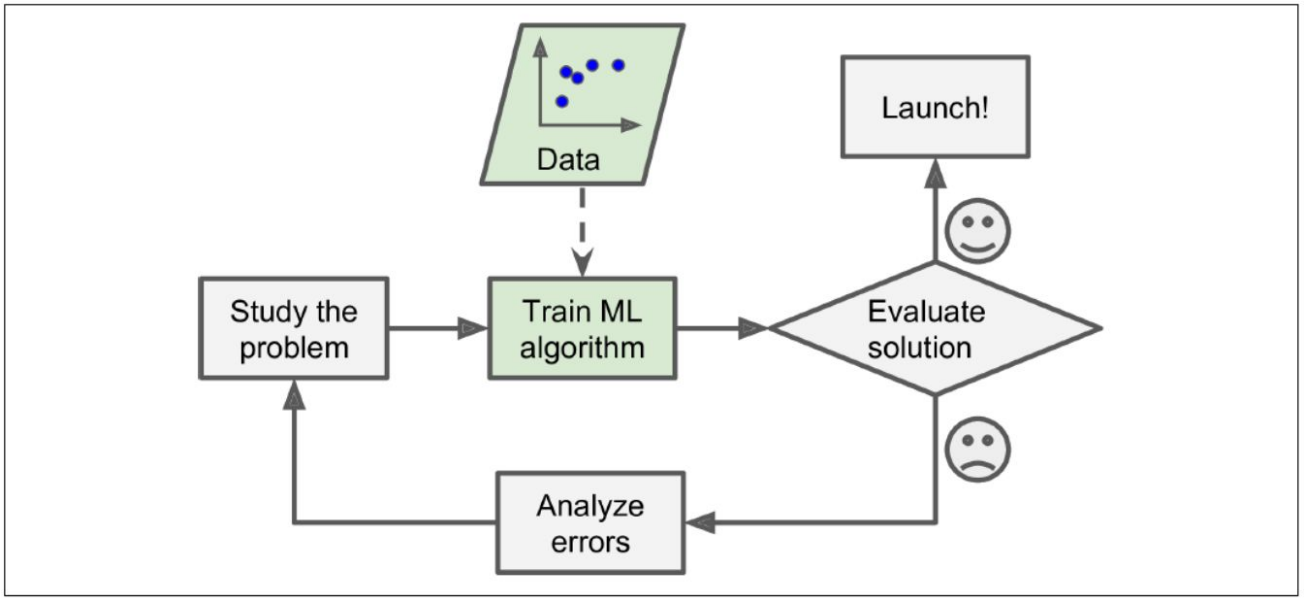
- klasický přístup k řešení problému pomocí softwaru



Zdroj: (Géron 2019)

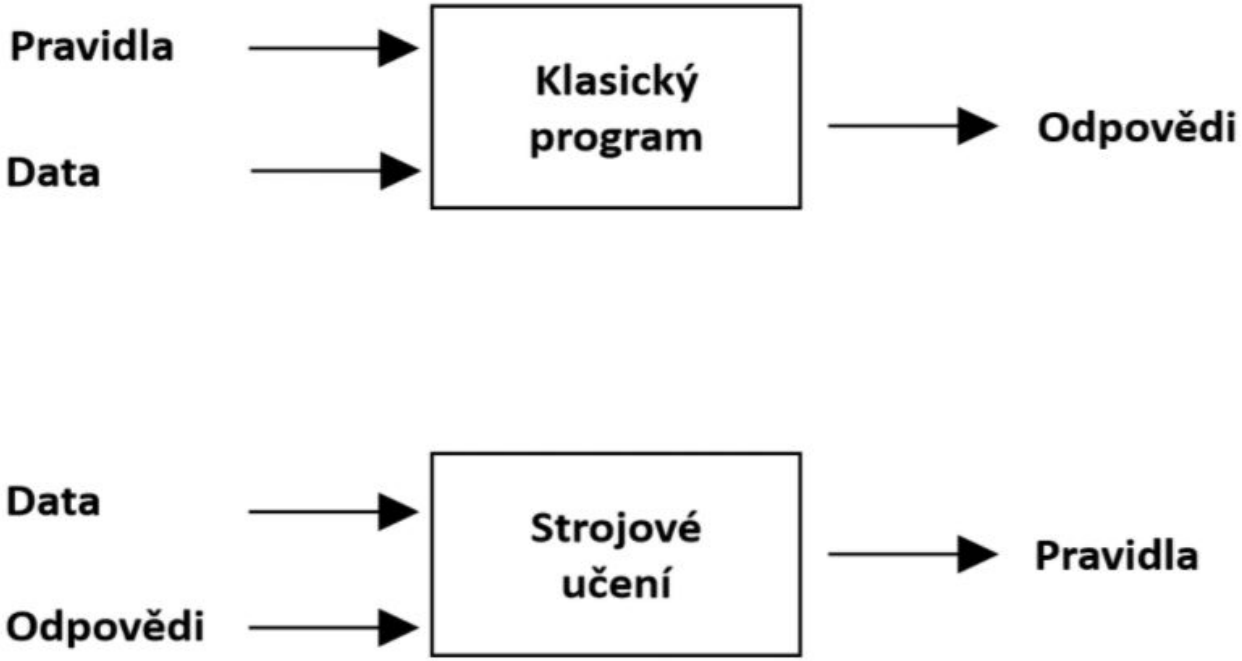
Odlišnost strojového učení od klasických přístupů

- přístup pomocí strojového učení



Zdroj: (Géron 2019)

Odlišnost strojového učení od klasických přístupů



Zdroj: (Chollet 2019)

Typy strojového učení

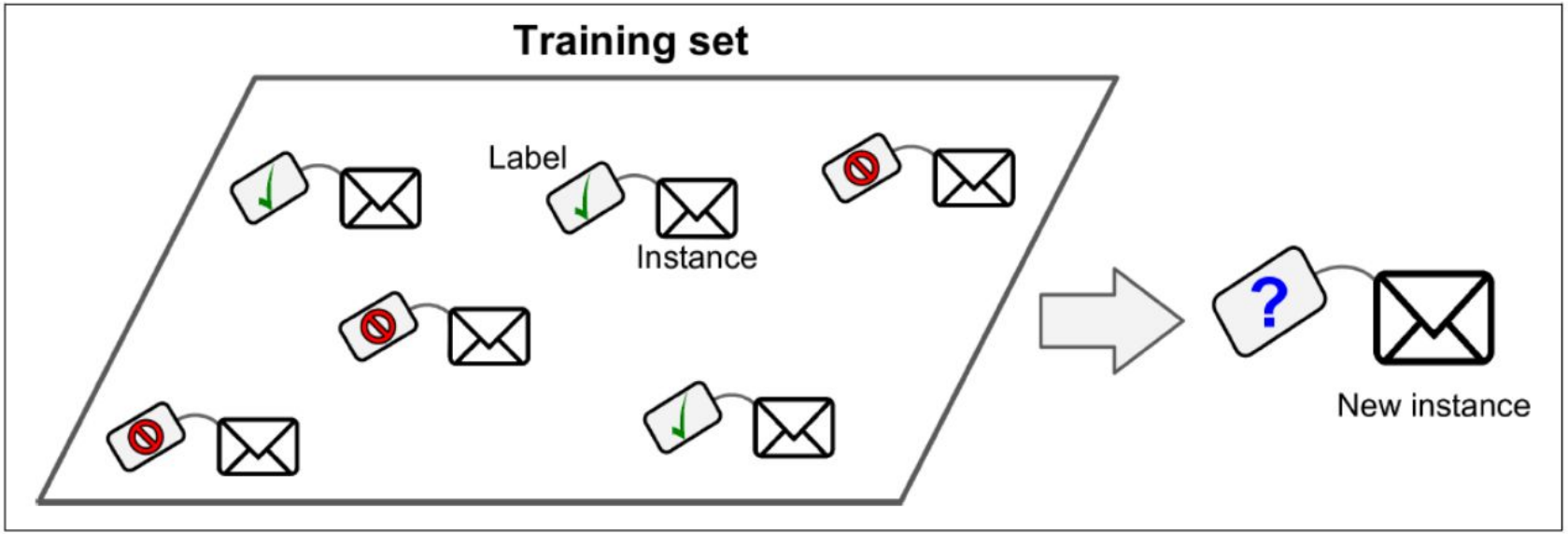
Podle čeho lze strojové učení kategorizovat

- zda jsou modely trénovány **s učitelem** (lidský dohled) nebo bez něj
  - učení s učitelem (supervised learning)
  - učení bez učitele (unsupervised learning)
  - částečně kontrolované učení (semi-supervised learning)
  - samořízené učení (self-supervised learning)
  - reinforcement learning
- zda se mohou modely trénovat **inkrementálně za běhu** či pouze dávkově
  - online learning
  - batch learning
- zda algoritmy fungují na bázi **prediktivních modelů**, které rozpoznávají vzory v datech, nebo zda pouze porovnávají nová data oproti známým datům
  - model-based learning
  - instance-based learning
- zda jsou modely mělké (shallow) nebo hluboké (vícevrstevné, **Deep Learning**)
- např. moderní SPAM filtr na bázi hluboké neuronové sítě: online, model-based, supervised, Deep Learning

Supervised vs. Unsupervised Learning

Supervised Learning

- učení s učitelem: trénovací data, která slouží jako vstupy pro algoritmus strojového učení, **obsahují** zároveň **požadované výstupy (labels)**



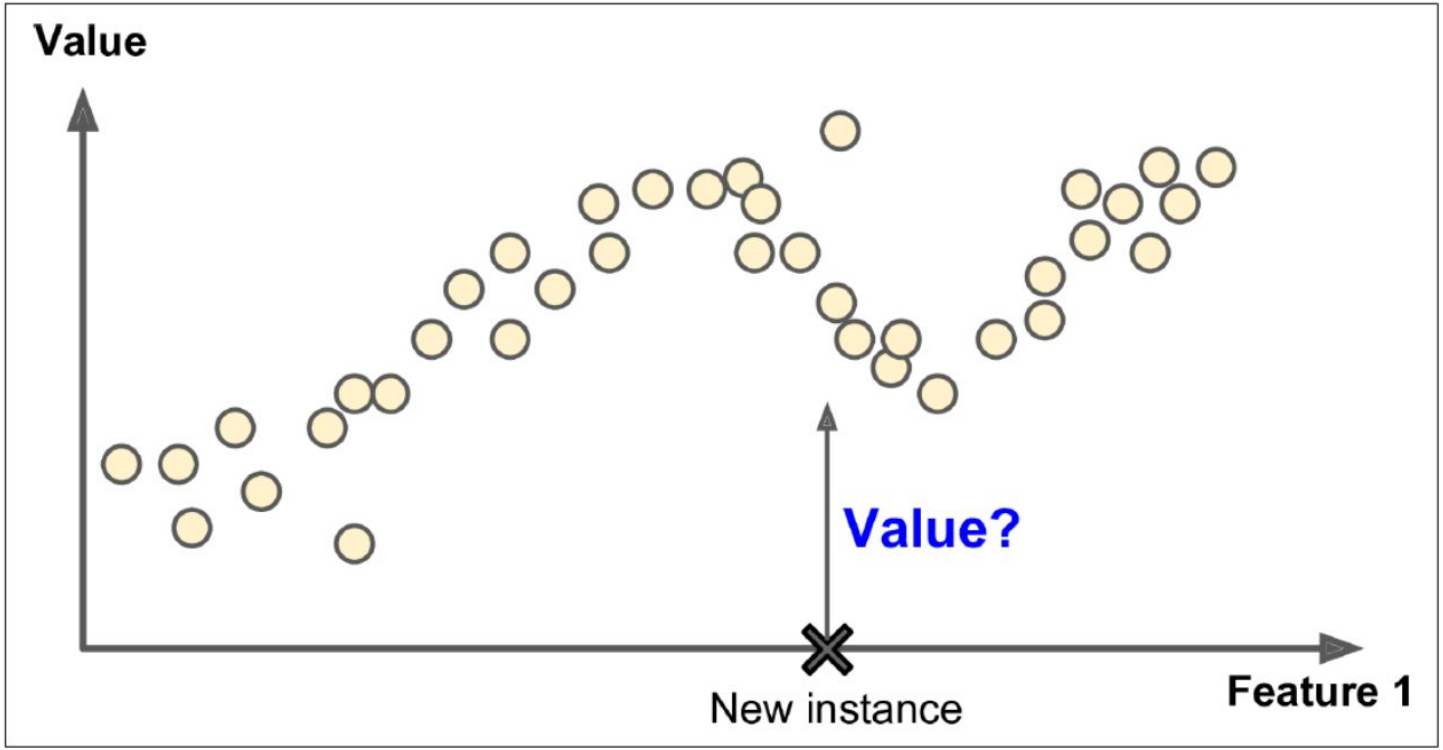
Zdroj: (Géron 2019)

Supervised Learning

- typická úloha učení s učitelem je **klasifikace**
  - např. SPAM filtr: trénování se provádí pomocí velkého množství příkladů e-mailů společně s jejich třídou (spam vs ne-spam)
    - úkolem algoritmu je naučit se správně klasifikovat nové e-maily

- další typická úloha učení s učitelem je predikce cílové číselné hodnoty = **regrese**
  - např. predikce ceny automobilu na základě jeho vlastností (značka, model, stáří, nájezd, výbava = prediktory)
  - trénování se provádí pomocí velkého množství příkladů automobilů obsahujících jak prediktory, tak cílové hodnoty (ceny)

Supervised Learning - regrese



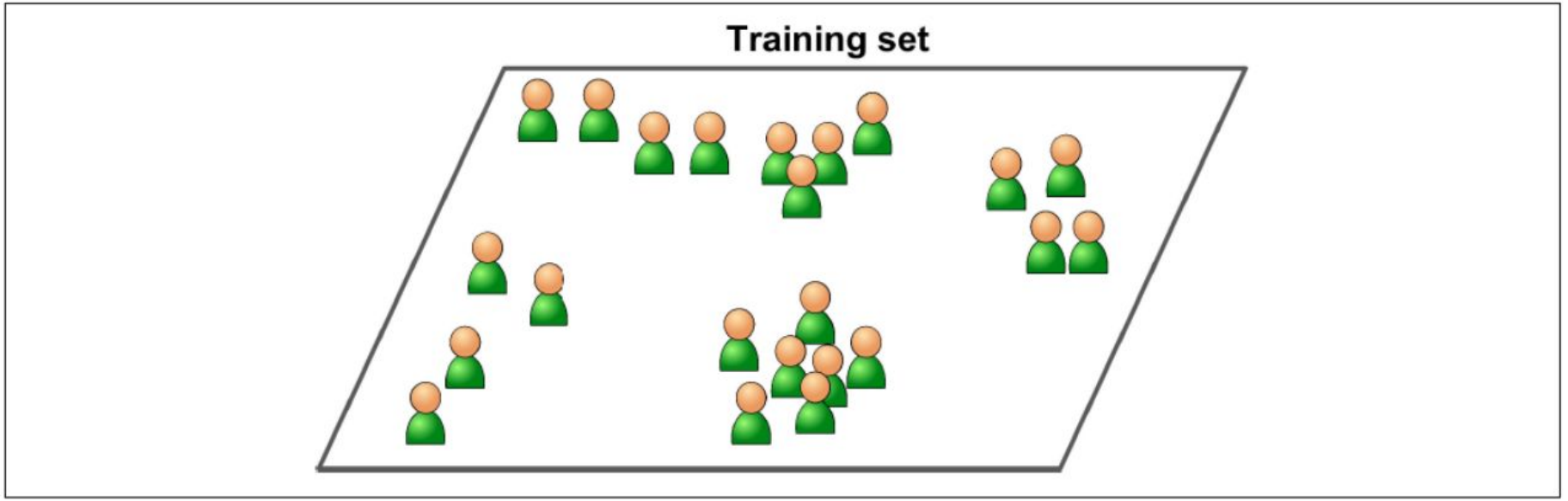
Zdroj: (Géron 2019)

Nejdůležitější algoritmy Supervised Learning

- k-nejbližších sousedů (k-Nearest Neighbors)
- lineární regrese (Linear Regression)
- logistická regrese (Logistic Regression)
- metoda podpůrných vektorů (Support Vector Machines)
- rozhodovací stromy a náhodné lesy (Decision Tree, Random Forest)
- neuronové sítě (Neural Network)

Unsupervised Learning

- při učení bez učitele, jak název napovídá, trénovací data neobsahují labely



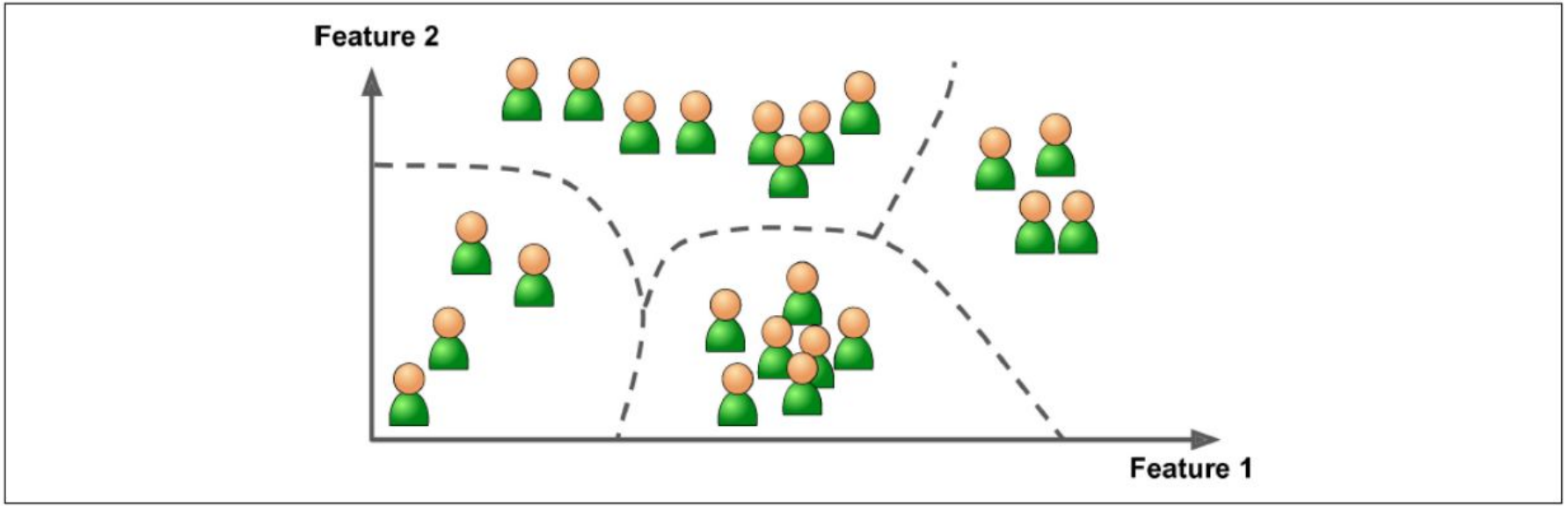
Zdroj: (Géron 2019)

Nejdůležitější algoritmy Unsupervised Learning

- shlukování (clustering)
  - K-Means
  - DBSCAN
  - hierarchická shluková analýza (HCA, Hierarchical Cluster Analysis)
- detekce anomálií (anomaly detection, novelty detection)
  - One-class SVM
  - Isolation Forest
- vizualizace a redukce dimenzionality
  - Principal Component Analysis (PCA)
  - Kernel PCA
  - Locally/Linear Embedding (LLE}
  - t-distributed Stochastic Neighbor Embedding (t-SNE)

Shlukování (Clustering)

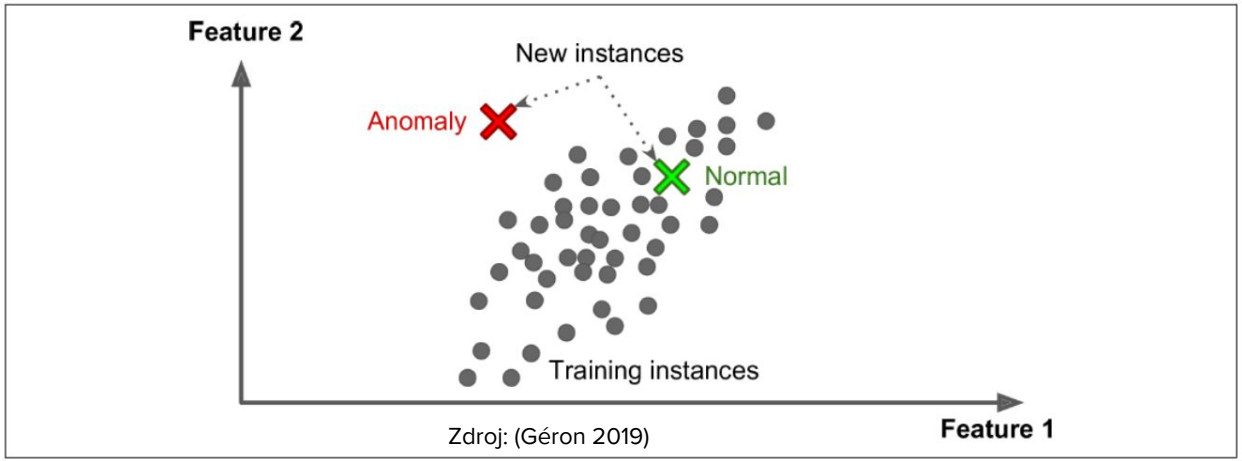
- příklad: segmentace návštěvníků webu podle jejich chování a preferencí
  - využitelné např. pro cílenou reklamu



Zdroj: (Géron 2019)

Detekce anomálií

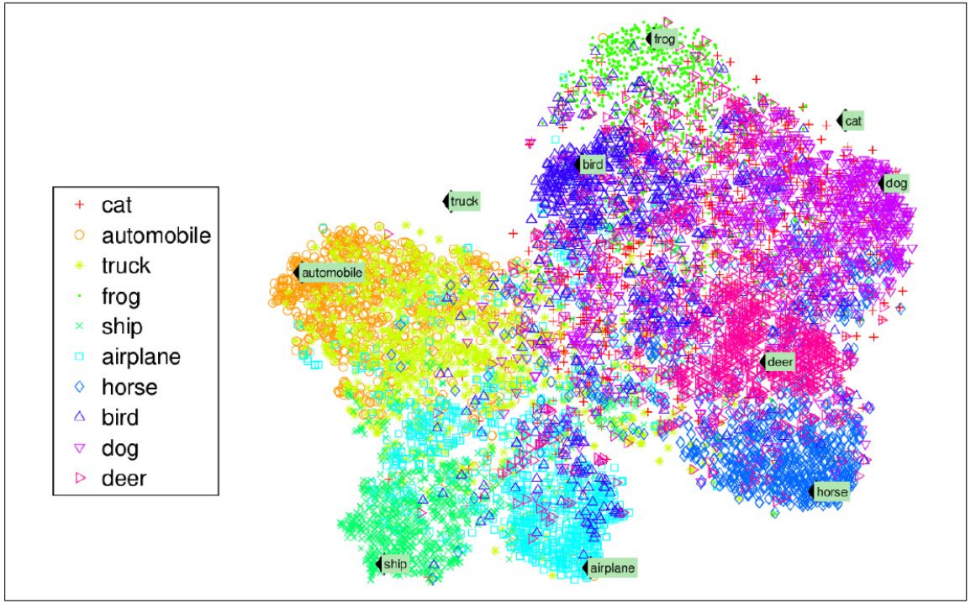
- příklady:
  - detekce nestandardních operací s kreditní kartou pro zabránění jejího zneužití
  - zachycení defektů ve výrobě
  - odstranění odlehlých hodnot (outliers) z datasetu před jeho předáním dalšímu algoritmu
- systému jsou ukázány normální instance během učení
  - při klasifikaci nové instance dokáže systém říct, zda se jedná o normální instanci nebo anomálii



Zdroj: (Géron 2019)

Vizualizace a redukce dimenzionality

- vstupem je velké množství multidimenzionálních dat (mnoho různých vlastností), výstupem 2D nebo 3D reprezentace dat, která může být jednoduše zobrazena a zkoumána
- cílem je zachovat maximum struktury v datech (např. odlišné segmenty ve vstupních datech by se neměly překrývat ani ve vizualizaci)



Zdroj: (Géron 2019)

Semi-supervised Learning

- některé algoritmy mohou pracovat s částečně označkovanými trénovacími daty
- typicky v poměru malého množství dat s labely a velkého množství dat bez nich
- příklad: rozpoznávání obličejů na fotkách
  - algoritmu se řekne, že na fotografii je určitý člověk a na základě toho ho dokáže rozpoznat i na dalších odlišných fotografiích

Self-supervised Learning

- “samořízené učení” - řízené učení bez lidí v učebním cyklu
- autoenkodéry
  - cílem je naučit se reprezentaci sady dat, obvykle za účelem snížení dimenze
    - např. word embeddings (podrobně později)
- predikce dalšího slova v psaném textu (na základě předchozích slov)
  - OpenAI GPT-2 a GPT-3 modely
- predikce dalšího snímku ve videu (na základě předchozích snímků)

Reinforcement Learning (“posilované učení”)

- Reinforcement Learning představuje zcela odlišný přístup od předchozích příkladů
- učící se systém (agent) pozoruje prostředí, ve kterém se nachází, volí a vykonává různé akce a na základě toho získává pozitivní nebo negativní odměny
- cílem je, aby se systém naučil jaká je nejlepší strategie (policy) pro maximalizaci odměny v průběhu času
  - výstupem tedy je správná volba akcí v různých situacích
- příklady:
  - někteří roboti využívají Reinforcement Learning k tomu, aby se naučili chodit
  - DeepMind AlphaGo porazil světového šampiona ve hře Go - strategii se naučil analyzováním mnoha her a následným hraním velkého množství her proti sobě

Batch vs. Online Learning

Batch Learning

- systém není schopen učit se inkrementálně, ale pro trénování potřebuje všechna dostupná data v jedné dávce (batch)
- trénování ze všech dat najednou typicky trvá dlouho a provádí se offline (mimo produkci)
- systém se nejdříve natrénuje a následně je spuštěn na produkci, aniž by se dále trénoval
- pokud chceme, aby se systém natrénoval i na nových datech, musíme ho trénovat znovu na celém datasetu (tedy nová i stará data)

Online Learning

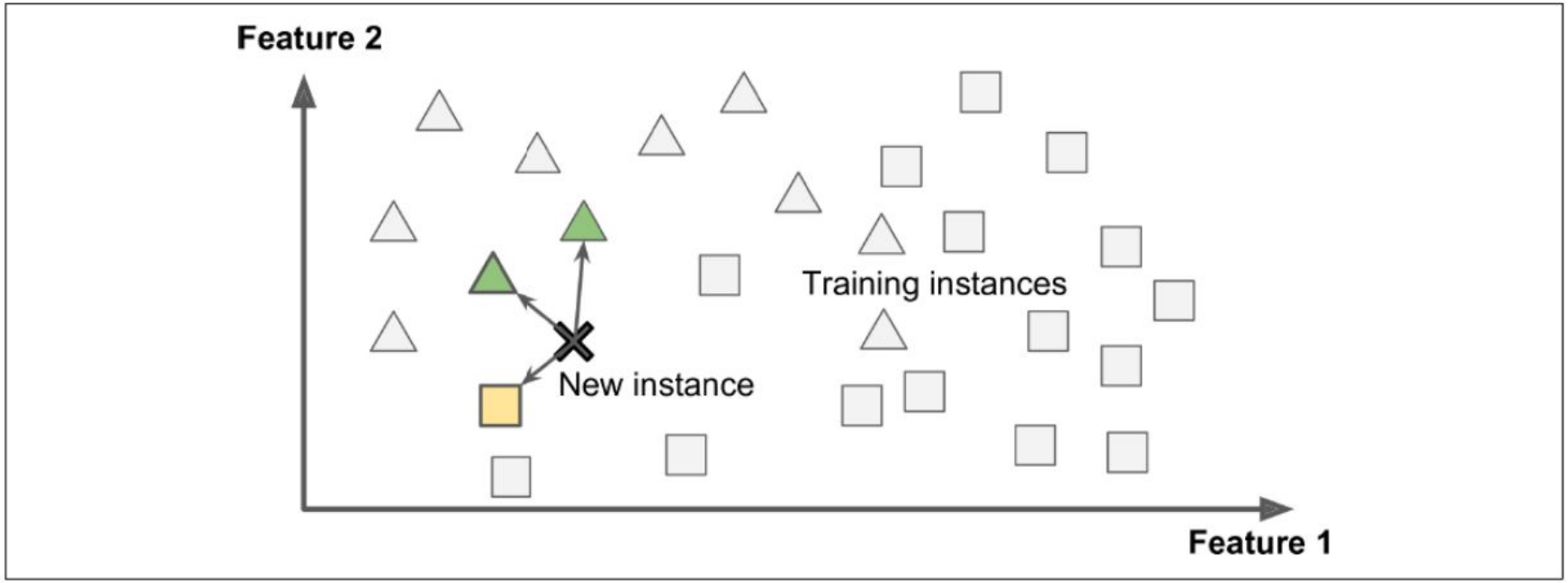
- systém je trénován inkrementálně na nově příchozích datech (buď po jednom nebo v mini dávkách)
- trénování v takto malých krocích je rychlé a levné, systém se tak může učit za běhu
- online learning je vhodný pro systémy, které kontinuálně přijímají nová data a potřebují se rychle adaptovat na změny
- není nutné uchovávát všechna historická data (pokud nechceme mít možnost model natrénovat od začátku)
- důležitý parametr online learning systémů: learning rate (jak rychle se model adaptuje na nová data)
    - vysoká rychlost učení: rychlá adaptace na nová data, zapomínání starých dat
    - nízká rychlost učení: větší setrvačnost, menší náchylnost na šum v nových datech
- pokud se do systému dostanou chybná data, negativně to ovlivní jeho výkon
    - nutno model monitorovat a průběžně zálohovat, aby se dalo vrátit k poslednímu dobrému stavu

Instance based vs. Model based

Instance based

- systém si zapamatuje trénovací data a generalizaci pak provádí porovnáváním nových příkladů vůči trénovacím
  - pro porovnávání se používá míra podobnosti (similarity measure)
- příklad:
  - SPAM filtr bude jako spam označovat ty e-maily, které jsou velmi podobné již označeným spamovým e-mailům
  - jako míra podobnosti dvou e-mailů lze v jednoduchém případě použít např. počet slov, která mají oba e-maily shodné
    - pokud má e-mail mnoho slov shodných se známým spamovým e-mailem, pak bude také označet jako spam
    - (později si ukážeme mnohem lepší způsoby porovnávání textových dokumentů)

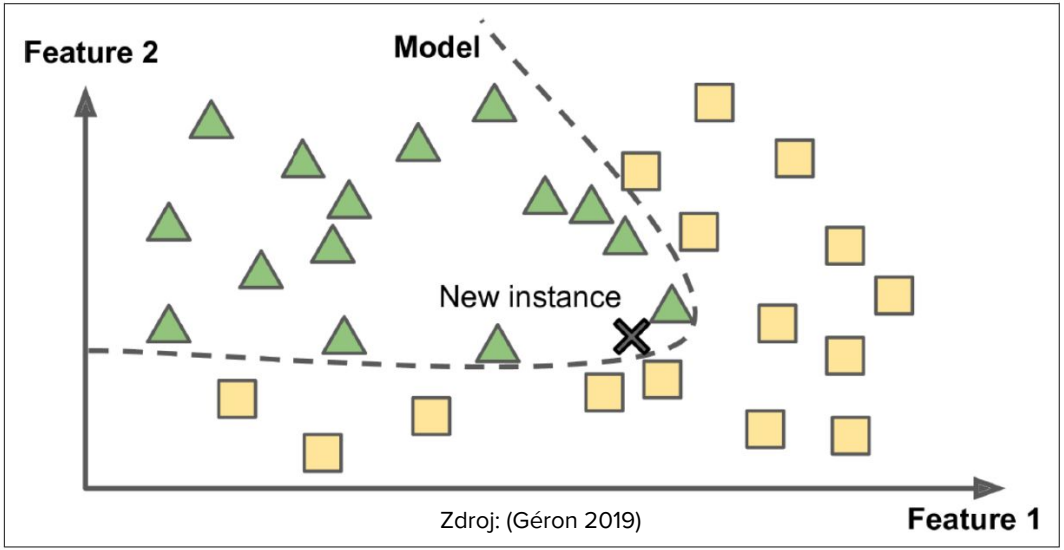
Instance based



Zdroj: (Géron 2019)

Model based

- jiný (v našem kurzu častější) způsob představuje vytvoření modelu na základě trénovacích dat
- vytvořený model se využije pro predikci výstupu (třídy, hodnoty) na základě příznaků (features) nového příkladu



Zdroj: (Géron 2019)

Mělké modely vs. Deep Learning

Deep Learning (“hluboké učení”)

- podskupina strojového učení, kde modely jsou tvořeny z více na sebe navazujících vrstev
- každá následující vrstva se učí smysluplnější reprezentace z dat (“destilace informací”)
- příklad: CNN (konvoluční neuronová síť)
  - první vrstva pracuje na úrovni pixelů a zpracovává informace jako jsou např. hrany v obrázku
  - další vrstvy pracují postupně se stále více abstraktními informacemi (textury, ..., objekty), přičemž jako vstup používají výstup předchozí vrstvy
- hloubka nepředstavuje “hlubší porozumění” problému, ale myšlenku postupných vrstev reprezentací
- Deep Learning má dosud průlomové výsledky například v těchto oblastech:
  - klasifikace obrazu (na úrovni člověka)
  - rozpoznávání řeči (na úrovni člověka)
  - přepis ručně psaného textu
  - výrazné zdokonalení strojového překladu (Google Translate - LSTM rekurentní neuronová síť)
  - autonomní řízení vozidel (na úrovni člověka)
  - vylepšené cílení reklam a vyhledávacích výsledků
  - nadlidská schopnost hraní různých her
  - generování textu, Q&A (téměř na úrovni člověka - OpenAI GPT-3 - Transformer)

Hlavní výzvy strojového učení

Výzvy strojového učení

- pokroku ve strojovém učení je dosahováno pomocí tří faktorů:
  - hardware
  - data
  - algoritmy

- v letech 1990–2000 hlavní překážkou data a hardware
  - od té doby masivní pokrok:
    - vzestup internetu
    - nové vysoce výkonné grafické čipy

Hardware

- v rozmezí 1990-2010 cca 5000x zrychlení procesorů
  - i na notebooku lze provozovat malé modely využívající Deep Learning, které ještě nedávno nebyly vůbec zpracovatelné
- typické modely pro computer vision nebo speech recognition však vyžadují výrazně vyšší výkon než mají běžné počítače
- NVIDIA investovala do vývoje grafických procesorových jednotek (GPU) pro videohry
  - GPU jsou masivně paralelní čipy (jednoúčelové superpočítače), které excelují v určitém typu výpočtů
- hluboké neuronové sítě vyžadují zejména velké množství malých násobení matic
  - vysoce paralelizovatelné
  - => vhodná operace pro GPU
  - např. NVIDIA Titan X zvládá 6,6 TFLOPS = 6,6 bilionů float32 operací / s = cca 350x víc než běžný ntb
- natrénovat CNN model na ImageNet datasetu (základ computer vision) trvá pouhé jednotky dní na moderním GPU

Data

- rozvoj internetu a sociálních sítí výrazně přispěl k pokroku v mnoha oblastech strojového učení
  - uživatelsky generované obrazové značky na Flickru + videa na YouTube
    - zásadní přínos pro počítačové vidění
  - Wikipedia
    - klíčový dataset pro zpracování přirozeného jazyka
  - ...
- dataset ImageNet
  - několik milionů obrázků, každý z nich ručně zařazený do jedné z 1000 kategorií
  - každoroční soutěž o nejlepší model počítačového vidění trénovaný na tomto datasetu
    - zásadní přínos pro rozvoj celé oblasti
- mnoho datasetů volně k dispozici online (ukážeme si později kde)
- s daty se však pojí i řada problému a výzev k řešení (viz dále)

Nedostatečné množství trénovacích dat

- většina algoritmů strojového učení potřebuje velké množství trénovacích dat
  - i pro jednoduché problémy jsou nutné řádově tisíce příkladů
  - pro komplexní problémy jako rozpoznávání obrazu nebo řeči pak i miliony příkladů
    - lze však použít předtrénované modely, které se jen doladí pro naši potřebu
    - pak mohou stačit i řádově tisíce obrázků pro klasifikaci
- pro řešení komplexních problémů je velké (resp. obrovské) množství dat často důležitější než samotný algoritmus
  - např. model GPT-3 natrénován na cca 500 mld. textových tokenů (z toho Wikipedie má 3)
    - cena potřebného výpočetního výkonu na jedno trénování odhadována na 4.6 M USD
- při řešení nestandardních úloh jsou však běžné malé a středně velké datasety
  - získat větší množství trénovacích dat může být obtížné nebo velmi nákladné
  - v takovém případě mají vhodné algoritmy zásadní význam

Nedostatečně reprezentativní trénovací data

- pokud má systém dobře generalizovat, musejí být trénovací data reprezentativní i pro nové příklady, na které ho chceme použít
  - pokud použijeme pro trénování málo reprezentativní dataset, nebude model schopen dělat dobré predikce
- problém zejména v případě malých datasetů
  - sampling noise - nereprezentativní data jako důsledek náhody při malém výběru
- bez problému nejsou ani velké datasety
  - sampling bias - zkreslení, pokud je použita nevhodná metoda výběru

Nízká kvalita dat

- pokud jsou v datech chyby, šum či odlehlé hodnoty (outliers), bude pro systém složitější najít v nich vzory, které jsou nezbytné pro jeho fungování
- příprava trénovacích dat (čištění dat) stojí mnoho času, ale pro správné fungování modelu může být tento krok zásadní

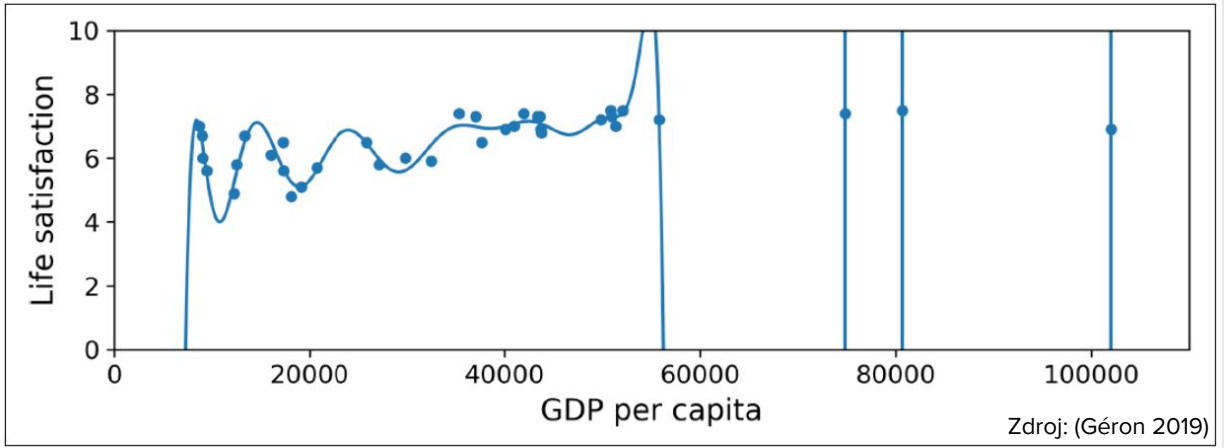
- základní způsoby přípravy a čištění dat
  - zjevná odlehlá pozorování (outliers) jednoduše vyřadit
  - chybějící příznaky (features) u příkladů (např. u některých zákazníků nám bude chybět jejich věk) - možná řešení:
    - zcela ignorovat tento příznak
    - zcela ignorovat příklady s chybějícími příznaky
    - doplnit chybějící hodnoty (např. průměr nebo medián)
    - natrénovat dva modely (jeden pro data s chybějícím příznakem)

Nerelevantní příznaky

- pro správně fungující model je nutné mít dostatek relevantních příznaků a minimum nerelevantních
- **konstrukce příznaků** (feature engineering) je zcela klíčová činnost pro modely strojového učení (zejména pro ty “mělké”)
  - cílem této činnosti je tedy sestavit kvalitní sadu příznaků v trénovacích datech
- hlavní činnosti v rámci konstrukce příznaků:
  - výběr příznaků (feature selection)
    - ze sady dostupných příznaků vybereme nejužitečnější příznaky pro náš model
  - extrakce příznaků (feature extraction)
    - ze stávajících příznaků vytvoříme nový užitečnější příznak (např. pomocí redukce dimenzionality)
  - tvorba nových příznaků
    - získáme nová trénovací data s novými příznaky

Přeučení modelu na trénovacích datech

- přeučení (overfitting) je jedna z hlavních věcí, kterým se budeme učit předcházet
- znamená, že model funguje dobře na trénovacích datech, ale špatně (hůře) na nových datech (nedělá dobré predikce pro data, která neviděl)
- příklad: nevhodně zvolený polynom vysokého stupně



Zdroj: (Géron 2019)

Přeučení modelu vs. kapacita modelu

- komplexní modely (např. hluboké neuronové sítě s mnoha parametry) jsou schopné detekovat velmi detailní vzory v datech
  - pokud je však dataset velmi malý nebo obsahuje šum, potom hrozí, že model začne nacházet vzory v samotném šumu, nebo že se naučí cílové hodnoty “zpaměti” pro testovací data
    - pro nová data je to však bezcenné
- příklad: model životní spokojenosti podle HDP země
  - pokud použijeme také nerelevantní příznaky jako je název země, může se model naučit, že země mající v názvu určité písmeno mají vysokou/nízkou hodnotu životní spokojenosti
    - zcela zjevně je toto výsledkem náhody a pro budoucí predikce nepoužitelné
    - model však nemá jak poznat, jestli toto vzniklo důsledkem náhody nebo je to skutečný vzor

Způsoby předcházení přeučení modelu

- k přeučení dochází, pokud je model příliš komplexní vzhledem k množství trénovacích dat a šumu v nich obsaženém
- možná řešení:
  - zjednodušit model tím, že se sníží množství jeho parametrů
  - získat více trénovacích dat
  - snížit šum v trénovacích dat
- nástroj pro zjednodušení modelu = **regularizace**
  - cílem je najít rovnováhu mezi perfektním naučením se z trénovacích dat a udržením jednoduchosti modelu, aby dobře zobecňoval na nových datech
  - příklad: lineární regrese (y = ax + b) má dva stupně volnosti (a = sklon přímky, b), které model může použít
    - regularizace může zajistit, že se použije jen jeden z těchto parametrů, nebo že hodnoty těchto parametrů budou malé
    - pokud máme v modelu hodně příznaků, může regularizace pomoci odfiltrovat málo relevantní příznaky

Volba hyperparametrů

- např. míra regularizace se modelu zadává formou **hyperparametru**
- hyperparametr je parametr učícího se algoritmu, nikoliv modelu jako takového
  - hyperparametry se tedy nastavují “zvenku” před začátkem trénování a typicky zůstávají během trénování konstantní
- volba a ladění hodnot hyperparametrů je důležitou součástí strojového učení, protože hyperparametry zásadním způsobem ovlivňují výkon modelu

Podučení modelu

- podučení (underfitting) je logicky opakem přeučení
  - nastává, pokud je model příliš jednoduchý pro to, aby se naučil vzory v trénovacích datech
  - např. lineární model aplikovaný na nelineární problém

- způsoby řešení
  - volba výkonnějšího modelu s více parametry
  - volba (příp. vytvoření) lepších příznaků
  - redukce omezení modelu (např. snížení míry regularizace)

Testování a validace

- po trénování chceme získat představu, jak dobře bude model fungovat na nových datech
  - výsledek, který získáme na trénovacích datech, nemusí nic vypovídat vzhledem k možnému přeučení
- jediný způsob jak zjistit výkon modelu na nových datech, je ho na těchto datech vyzkoušet
  - můžeme spustit model do produkce a sledovat jeho výkon, nicméně toto je zdlouhavé a špatným modelem na produkci si můžeme způsobit případné škody
  - mnohem lepší způsob je rozdělit data, co máme k dispozici, do dvou množin:
    - trénovací dataset
    - testovací dataset
  - model pak trénujeme na trénovacím datasetu a testujeme ho na testovacím datasetu
  - chybovost na testovacím datasetu nám dává dobrý odhad, jak bude model fungovat na nových datech, které nikdy neviděl
  - rozdělení typicky 80 % trénovací a 20 % testovací set
- pokud je chybovost na trénovacích datech nízká, ale na nových (testovacích) datech vysoká, pak to značí přeučení modelu

Validační set vs. testovací set

- příklad: rozhodujeme se mezi lineárním a polynomických modelem a hodnotami jejich hyperparametrů
- jak ale získat nejlepší hodnoty hyperparametrů?
  - můžeme natrénovat desítky nebo stovky modelů s různými hodnotami hyperparametrů
  - pokud je však budeme ověřovat na testovacím datasetu, tak reálně hrozí, že vybereme model, který je přeučený specificky vůči testovacímu datasetu a na nových datech nebude fungovat dobře
- řešením je použít ještě validační dataset
  - část trénovacích dat dáme bokem a použijeme je na průběžné ověřování různých modelů a jejich hyperparametrů (místo testovacího datasetu)
  - vybere nejlepší model a natrénujeme ho znovu na všech trénovacích datech vč. validačních, tím získáme finální model
    - pro ověření finálního modelu použijeme testovací dataset, abychom získali představu, jak bude fungovat v reálném provozu
- problém nastává u malých datasetů, kde nám rozdělením dat na trénovací, validační a testovací vznikne jen velmi malá validační množina
  - řešení pomocí tzn. K-násobné křížové validace (K-fold cross-validation)
  - testovací data se rozdělí na K skupin, přičemž model se trénuje na K-1 skupinách a ověřuje na zbývající
    - každý model se tedy trénuje a ověřuje K-krát, výsledek ověření získáme zprůměrováním jednotlivých hodnot
    - výhodou je mnohem přesnější výsledek ověření na malých datasetech, nevýhodou nutnost trénovat mnoho modelů

Shrnutí

Shrnutí

- strojové učení umožňuje vytvářet systémy, které jsou schopné řešit různé problémy tím, že se řešení naučí z dat namísto toho, abychom museli vytvořit explicitní pravidla ručně
- existují různé typy systémů strojového učení (supervised/unsupervised, batch/online, instance/model based, ...)
- v projektu strojového učení získáme trénovací data a předáme je učícímu algoritmu
    - pokud je algoritmus model-based, učením se nastavují hodnoty jeho vnitřních parametrů tak, aby dokázal vystihnout vztahy v trénovacích (a doufáme že i nových) datech
    - pokud je algoritmus instance-based, naučí se trénovací data “nazpamět” a generalizaci na nových datech pak provádí porovnáním vůči naučeným datům pomocí určité míry podobnosti
- systém nebude poskytovat dobré výsledky, pokud máme málo dat, nebo jsou data zašuměná, málo reprezentativní nebo plná nerelevantních příznaků (“garbage in, garbage out”)
- natrénovaný model by neměl být ani příliš jednoduchý (pak hrozí podučení), ani příliš komplexní (pak hrozí přeučení)
- pro ověření výsledků modelu používáme rozdělení datasetu na trénovací, validační a testovací set

Zdroje

- Coelho, L. P.; Richert, W. (2013) Building machine learning systems with Python. Birmingham: Packt Publishing. ISBN 978-1-78216-140-0.

- Géron, A. (2019) Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow, 2nd Edition. O'Reilly Media, Inc. ISBN 9781492032649.

- Chollet, F. (2019) Deep Learning v jazyku Python. Knihovny Keras, TensorFlow. Grada Publishing, a.s. ISBN 978-80-247-3100-1.

- Segaran, T. (2007) Programming collective intelligence: building smart web 2.0 applications. Beijing: O'Reilly Media. ISBN 0-596-52932-5.