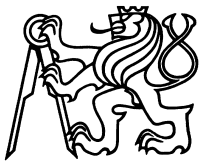
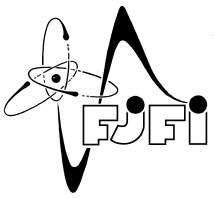


České vysoké učení technické v Praze
Fakulta jaderná a fyzikálně inženýrská



Nové přístupy k metaučení v regresním modelování

Diplomová práce

Autor: **Aleš Neoral**

Vedoucí práce: **RNDr. Jan Kalina, Ph.D.**

Akademický rok: 2019/2020

České vysoké učení technické v Praze, Fakulta jaderná a fyzikálně inženýrská

Katedra: matematiky

Akademický rok: 2018/2019

ZADÁNÍ DIPLOMOVÉ PRÁCE

Student:	Bc. Aleš Neoral
Studijní program:	Aplikace přírodních věd
Obor:	Aplikované matematicko-stochastické metody
Název práce (česky):	Nové přístupy k metaučení v regresním modelování
Název práce (anglicky):	New approach to metalearning in regression modeling

Pokyny pro vypracování:

1. Příprava datových souborů vhodných pro konkrétní studii metaučení.
2. Rešerše v literatuře na téma nestandardních přístupů k metaučení.
3. Implementace metaučení pro regresi. Regresní modelování predikční chyby pro jednotlivé datové soubory (oproti obvyklým klasifikačním postupům), hledání vhodné regresní metody.
4. Výpočty pro standardní i robustní predikční chyby.
5. Srovnání různých přístupů k metaučení. srovnání s různými aternativními postupy.
6. Interpretace výsledků získaných v numerické studii.

Doporučená literatura:

1. P. Brazdil, C. Giraud-Carrier, C. Soares, E. Vilalta. Metalearning: Applications to data mining. Springer, Berlin, 2009.
2. K. Smith-Miles. Cross-disciplinary perspectives on meta-learning for algorithm selection. ACM Comput. Surv. 41, 2009, Article 6.
3. K. Smith-Miles, D. Baatar, B. Wreford, R. Lewis. Towards objective measures of algorithm performance across instance spaces. Comput. Oper. Res. 45, 2014, 12-24.
4. O. Mersmann, M. Preuss, H. Trautmann, B. Bischl, C. Weihs. Analyzing the BBOB results by means of benchmarking concepts. Evol. Comput. 23, 2015.

Jméno a pracoviště vedoucí diplomové práce:

RNDr. Jan Kalina, Ph.D.
Ústav informatiky AV ČR, Pod Vodárenskou věží 2, 182 07 Praha 8

Jméno a pracoviště konzultanta:

Datum zadání diplomové práce: 31.10.2018

Datum odevzdání diplomové práce: 6.5.2019

Doba platnosti zadání je dva roky od data zadání.

V Praze dne 27. října 2018

.....
garant oboru

.....
vedoucí katedry

ČESKÉ VYSOKÉ UČENÍ TECHNICKÉ V PRAZE

.....
děkan

Poděkování:
Chtěl bych zde poděkovat především svému školiteli RNDr. Janovi Kalinovi, Ph.D. za pečlivost, ochotu, vstřícnost a odborné i lidské zázemí při vedení mé diplomové práce.

Čestné prohlášení:
Prohlašuji, že jsem tuto práci vypracoval samostatně a uvedl jsem všechnu použitou literaturu.

V Praze dne 29. června 2020 Aleš Neoral

Název práce:

Nové přístupy k metaučení v regresním modelování

Autor: Aleš Neoral

Obor: Matematické inženýrství

Zaměření: Aplikované matematicko–stochastické metody

Druh práce: Diplomová práce

Vedoucí práce: RNDr. Jan Kalina, Ph.D., Ústav informatiky AV ČR

Abstrakt: Metaučení se stává důležitou metodologií pro extrakci znalostí z dostupných trénovacích dat na nová (nezávislá) data. Zatím ale na tohle téma nebylo napsáno mnoho teorie a chybí také i praktické výpočty. Tato diplomová práce rozšiřuje praktický výzkum v této oblasti a zaměřuje se na citlivost metaučení na kontaminovaných datech. Metaučení jsme realizovali za použití čtyř různých regresních metod (včetně vysoce robustních) na 643 datových souborech s přidáváním různé míry kontaminace. Metaučení vykazuje vysoce nadprůměrné výsledky pro robustní míry predikce a při použití standardní střední kvadratické chyby metaučení selhává. Metaučení je citlivé na uměle přidanou kontaminaci, zejména na tu globální.

Klíčová slova: metaučení, robustnost, kontaminace dat, lineární regrese

Title:

Sensitivity of metalearning in regression modeling

Author: Aleš Neoral

Abstract: Metalearning is becoming an increasingly important methodology for extracting knowledge from a data base of available training dataset to a new (independent) dataset. There has not been written much theory about metalearning. Few practical experiments on metalearning research have been done. This diploma thesis expands empirical knowledge about metalearning. We performed metalearning using four regression methods (including highly robust ones) have been used on 643 data files, adding a various degree of contamination. Metalearning produces very good results for robust prediction measures and fails when used for standard mean squared error. Metalearning is sensitive to artificial contamination, especially to global contamination.

Key words: metalearning, robustness, data contamination, linear regression

Obsah

Seznam obrázků

Seznam tabulek

Úvod

Motivace

Dnešní svět se velmi rychle mění díky technologickému pokroku. Na burze už mezi sebou neobchodují lidé, ale algoritmy vysokorychlostního obchodování. Reklamu už nedoporučují lidem lidé, ale algoritmy kolaborativního filtrování. Na politických rozhodnutích se nepodílí jen politici, ale i algoritmy zpracování sentimentu a přirozeného jazyka. Snad už není průmyslové odvětví, které by nebylo zasaženo silnou automatizací. V této práci se na automatizaci podíváme z pohledu metaučení.

O metaučení se začínají zajímat lidé působící v oblasti umělé inteligence, protože v něm vidí další potenciální pokrok. Metaučení se totiž používá v systémech AutoML pro automatické strojové učení. Tyto systémy používají metaučení a nejrůznější algoritmy pro předzpracování dat, výběr příznaků, optimalizaci hyperparametrů, kombinaci metod a následný výběr modelu. Dále se přístupy metaučení uplatňují při kombinaci jednotlivých modelů, automatickém návrhu hlubokých neuronových sítí, učení z malého množství dat a v neposlední řadě při učení všestranných robotů.

Koncept metaučení má delší historii než počítačové vědy a poprvé se objevuje v oboru vzdělávací psychologie. Kde je metaučení popsáno jako schopnost být si vědom procesu a kontroly vlastního učení [1]. Výroky typu "Už mám zkušenost v ...", "Tohle se podobá na ..." "Tohle je speciální případ ..." implikují, že člověk, který prohlašuje tyto věty, se učí daný úkol rychleji, než jeho nezkušený protějšek. Tento fenomén byl proto silně motivován přenést do počítačových věd [2]. Ve standardních problémech strojového učení algoritmus typicky nebere v potaz žádnou předchozí zkušenost, v problémech metaučení ano.

Cíl práce

Hlavním cílem této práce je navrhnout a implementovat vlastní systém pro doporučení metody strojového učení pro nová data. Dílčí cíle můžeme zahrnout dvou rovin.

První, teoretická rovina, se zaměřuje na uvedení víceznačného pojmu metaučení. Záměrem je udělat takové teoretické shrnutí metaučení, že čtenář bude mít obecný přehled o pojmu metaučení a bude se pak lépe orientovat, pokud na tento pojem narazí v literatuře. Dalším úkolem je zaměřit se na jednu vybranou oblast metaučení, a to méně teoreticky prozkoumanou automatickou selekci algoritmu. Pro tuto oblast udělat detailní teoretickou studii zahrnující různé postupy a myšlenky.

Druhá, experimentální rovina, se zaměřuje na návrh vlastního automatického systému pro selekci algoritmu a diskutování jednotlivých úskalí. Po navržení vlastního systému pro automatickou selekci algoritmu je cílem provést několik experimentů na připravených datových souborech s různými nestandardními postupy. Těmito postupy jsou zkoumání citlivosti vytvořeného systému na základě uměle přidané kontaminace a použití různých nestandardních robustních měr predikce. Na závěr zhodnotíme a kvantifikujeme, zda a jak dobře námi navržený systém selekce algoritmu funguje na základě vybrané klasifikační metriky.

Struktura práce

V první kapitole definujeme základní pojmy strojového učení, na které se budeme odkazovat ve zbytku diplomové práce. Metaučení totiž zkoumáme právě v kontextu strojového učení.

Druhá kapitola je věnovaná teoretickému přehledu metaučení. Čtenář zjistí, že se pod pojmem metaučení skrývá velké množství různých myšlenek. Uvedeme několik praktických aplikací metaučení a jejich konkrétních příkladů. V této kapitole uvedeme, proč je metaučení důležité při vývoji všestranné umělé inteligence.

Třetí kapitola je zaměřená na jednu konkrétní podoblast metaučení, a to automatickou selekci algoritmu. Tuto oblast, kdy algoritmus strojového učení nevybírá člověk, ale počítač, jsme se rozhodli prozkoumat detailněji. Tato kapitola slouží jako teoretické shrnutí automatické selekce algoritmu.

V další kapitole popíšeme, jak jsme postupovali při řešení problému návrhu a implementace automatické selekce algoritmu. Uvedeme, jaké jsme použili algoritmy, datové soubory, predikční míry a další komponenty strojového učení. Čtenář se dozví, jaké jsme k tomuto problému používali softwarové nástroje a jakým způsobem jsme systém automatické selekce algoritmu implementovali. Také upřesníme několik oblastí, které nám v kontextu selekce algoritmu přišly zajímavé prozkoumat.

Poslední kapitola je věnována několika počítačovým experimentům pro automatickou selekci algoritmu. V každém z těchto experimentů jsme se rozhodli měnit některé komponenty systému selekce algoritmu a sledovat, jak se bude měnit jeho úspěšnost. Například jsme se rozhodli zkoumat systém pro různé míry predikce, přidávat kontaminaci a zkoumat jeho robustnost, vybrat jen několik důležitých příznaků a zkoumat jejich významnost. Nakonec jsme porovnali náš zkonstruovaný systém s náhodnou maticí.

Kapitola 1

Strojové učení

Strojové učení [3] je podoblastí umělé inteligence, zabývající se algoritmy a technikami, které umožňují počítačovému systému '*učit se*'. Učením v daném kontextu rozumíme takovou změnu vnitřního stavu systému, která zefektivní schopnost přizpůsobení se změnám okolního prostředí.

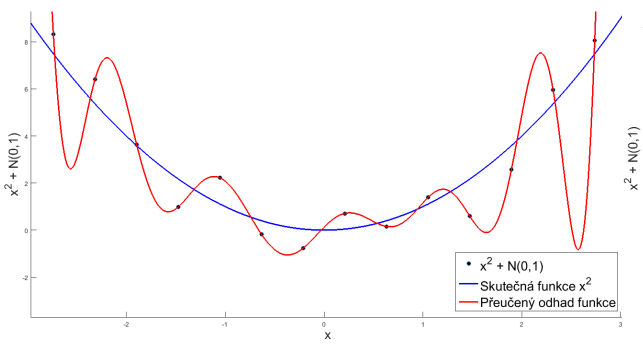
Aplikace strojového učení zaznamenaly v dvacátém prvním století obrovský úspěch. To zejména díky razantnímu rozvoji výpočetní techniky a masivní možnosti paralelizace. Snad žádná oblast lidské činnosti využívající data se nevyhla modernizaci systémů díky strojovému učení. Obory strojového učení, které zasáhl velký rozmach jsou například počítačové vidění [4], rozpoznávání mluvené řeči [5], zpracování přirozeného jazyka [6], strojový překlad [7], [8], doporučovací systémy [9], sentimentální analýza [10], predikce časových řad [11] a další. Tyto obory se uplatňují v aplikacích jako identifikace osob, rozpoznávání tváří, rozpoznávání hudby, přiřazení emocí danému textu, doporučení reklamy nebo predikci finančního trhu.

Algoritmy strojového učení lze podle způsobu učení rozdělit do kategorií: učení s učitelem, učení bez učitele, kombinace učení s učitelem a bez učitele a zpětnovazební učení. V teoretické i praktické části diplomové práce narážíme na metodu učení s učitelem a v teoretické části se často objevuje pojem zpětnovazebního učení. Je třeba uvést principy strojového učení před uvedením metaučení, protože na metaučení se můžeme dívat jako na proces automatizace strojového učení. Oblastem učení s učitelem a zpětnovazebnímu učení věnujeme další části této kapitoly. Zadefinujeme často používané koncepty nezbytné pro tyto typy učení. Tyto koncepty slouží k pochopení dalších částí diplomové práce, protože se s nimi bude čtenář dále v této práci setkávat.

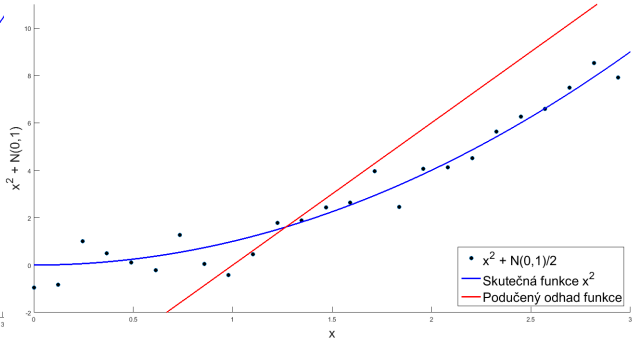
1.1 Učení s učitelem

Učení s učitelem je metoda strojového učení, pro kterou je cílem nalézt zobrazení zobrazující ze vstupních do výstupních proměnných. Datový soubor obsahuje n pozorování, které si můžeme označit jako $S = \{\mathbf{X_i}, Y_i\}_{i=1}^{n} \subset \mathbb{R}^p \times \mathbb{R}$, kde $\mathbf{X_i}$ je i-tý vstup a Y_i je i-tý výstup. Učící algoritmus hledá zobrazení $f : X \longrightarrow Y$, kde X je vstupní prostor a Y je výstupní prostor. Funkce f patří do prostoru všech možných funkcí, který se značí H a říká se mu prostor hypotéz. Často se vstupu do zobrazení f říká nezávislé proměnné a výstupu odezva.

Učení s učitelem se dělí na dvě kategorie, regresi a klasifikaci. Výstup zobrazení může být buď spojitá hodnota (regrese), nebo může předpovídat označení třídy výstupní proměnné (klasifikace). Často je při predikčních úlohách odezva ovlivněna nežádoucí kontaminací. Jak ale zařídit, aby zobrazení tuto kontaminaci ignorovalo a zachytilo obecný trend v datech? Na tuto otázku odpoví další podsekce.



Obrázek 1.1: Přeučení.



Obrázek 1.2: Podučení.

1.1.1 Generalizace

Jedním z velmi častých problémů, který může nastat v regresi, je přeučení (viz obrázek 1.1). Přeučení nastává, když až moc dobře předpovídáme naměřená data. Přeučená regresní křivka je velmi ovlivněna náhodným šumem. Vizuálně můžeme poznat, že došlo k přeučení, pokud se výsledná regresní křivka velmi vlní.

Opačným méně častým extrémem, který může nastat, je podučení (viz obrázek 1.2). Naším cílem je generalizovat pro nová nezávislá data a náhodnou kontaminaci ignorovat. Fenomén, kdy nemůžeme úplně přesně modelovat trend v trénovacích datech a zároveň generalizovat na nová testovací data se vžil pod anglickým názvem *bias-variance tradeoff* [12]. Praktické řešení pro dobrou generalizaci přinášejí metody validace modelu, ze kterých je nejčastěji používaná křížová validace.

1.1.2 Křížová validace

Křížová validace (CV) [13] je ve strojovém učení jedním nejčastěji používaných nástrojů k validaci modelu. Metoda křížové validace je založena na myšlence rozdělit data na trénovací a testovací množinu. Model se natrénuje na trénovací množině a na testovací, nezávislé množině se ověří jeho generalizující schopnost pomocí ztráty. Existuje několik variant křížové validace. Tyto varianty si v následujícím textu popíšeme. Nejdříve je ale potřeba zavést pár základních pojmů.

Mějme dány data $S = \{\mathbf{X_i}, Y_i\}_{i=1}^{n} \subset \mathbb{R}^p \times \mathbb{R}$ a f je funkce definovaná pro všechna $\mathbf{X_i}$. Pak hodnotu

$$E(f, S) = \sum_{i=1}^{n} (f(\mathbf{X_i}) - Y_i)^2$$

budeme nazývat chybou křížové validace na datovém souboru S.

Nejjednodušší varianta křížové validace je základní verze křížová validace, kdy je datová množina rozdělena náhodným algoritmem na dvě množiny. První množinu nazýváme trénovací S_{train}, která typicky obsahuje dvě třetiny dat, a druhou testovací S_{test}, která obsahuje zbytek. Z trénovacích dat je vypočten model, který je daný funkcí $f^{S_{train}}$. Chyba základní verze křížové validace $E_{holdout}$ je dána vzorcem:

$$E_{holdout} = E(f^{S_{train}}, S_{test}).$$

Další variantou křížové validace je k-násobná křížová validace (k-násobná-CV). Ta na rozdíl od předchozí základní verze rozdělí datovou množinu S na k podmnožin $S_1, S_2, \ldots, S_k$ takových, že platí $\cup_i S_i = S$ a $\forall i \neq j : S_i \cap S_j = \emptyset$. Proces k-násobné křížové validace se skládá z k iterací. V každé iteraci se $k-1$ podmnožin vyčlení jako trénovací množina pro trénování modelu a jedna podmnožina

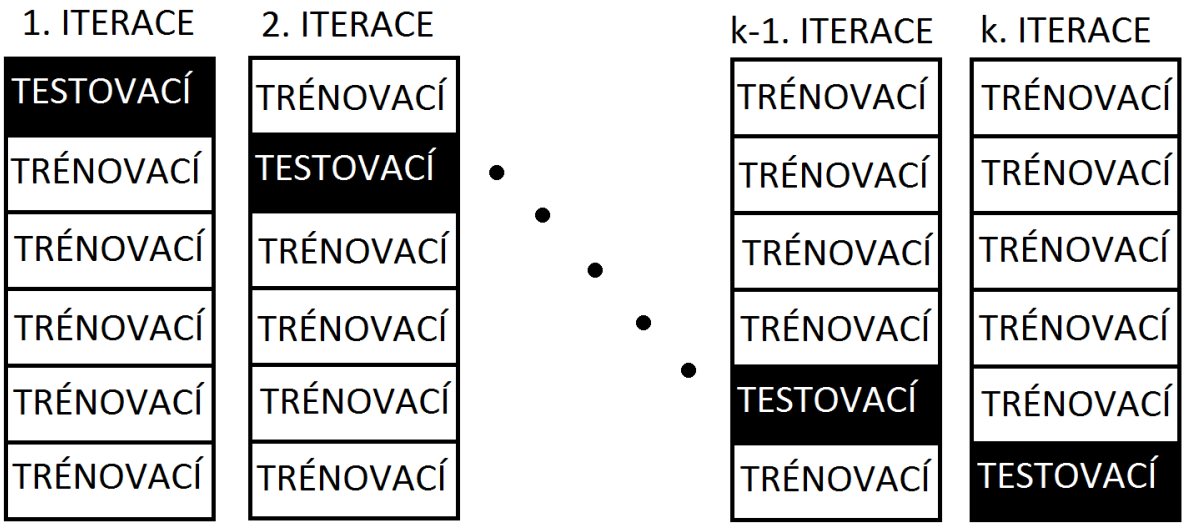
Vstup: Datový vzorek $S = \{\mathbf{X_i}, Y_i\}_{i=1}^{n} \subset \mathbb{R}^p \times \mathbb{R}$.
Výstup: Chyba $E_{holdout}$.

1. Rozděl data náhodně na dvě podmnožiny S_{train} a S_{test} takové, že platí: $S = S_{train} \cup S_{test}$ a $S_{train} \cap S_{test} = \emptyset$.
2. Vypočteme regresní křivku $f^{S_{train}}$ z dat S_{train}.
3. $E_{holdout} \longleftarrow E(f^{S_{train}}, S_{test})$.

Tabulka 1.1: Algoritmus základní verze křížové validace.

se vyčlení jako testovací množina pro ověření generalizace modelu, kde každá z k podmnožin je použita právě jednou jako testovací množina (viz obrázek 1.3). Nakonec jsou jednotlivé chyby zprůměrovány do celkového odhadu chyby E_{kfold}:

$$E_{kfold} = \frac{1}{k} \sum_{i=1}^{k} E(f^{\cup_{i \neq j} S_j}, S_i).$$



Obrázek 1.3: Schéma k-násobné křížové validace.

Speciální případ k-násobné křížové validace je tzv. *leave-one-out křížová validace* (LOOCV), při které se číslo k rovná velikosti datové množiny. V každé iteraci je jedno pozorování z datové množiny vyhrazeno jako testovací množina a zbylé pozorování jako trénovací množina.

V praxi se nejčastěji volí k-násobná-CV, protože základní varianta křížové validace generuje velmi hrubý odhad chyby, neboť je náchylná na rozdělení dat na trénovací a testovací podmnožiny. Druhá extrémní varianta LOOCV sice dává velmi přesný odhad chyby, ale je velmi výpočetně náročná. Velikost parametru k se volí podle toho, na čem nám více záleží (standardně $k = 10$).

Vstup: Datový vzorek $S = \{\mathbf{X_i}, Y_i\}_{i=1}^{n} \subset \mathbb{R}^p \times \mathbb{R}$. Dále $k \in \{2, 3, \ldots, n\}$.
Výstup: Chyba E_{kfold}.

1. Rozděl data náhodně na k podmnožin $S_1, S_2, \ldots, S_k$ takových, že $\cup_{i=1}^{k} S_i = S$ a $\forall i \neq j$: $S_i \cap S_j = \emptyset$.
2. $E_{kfold} \longleftarrow 0, \quad i \longleftarrow 1.$
3. TM $\longleftarrow \cup_{j \neq i} S_j$, kde TM označuje trénovací množinu.
4. Vypočteme regresní křivku f^{TM} z dat TM.
5. $E_{kfold} \longleftarrow E_{kfold} + E(f^{TS}, \text{TM})$.
6. $i \longleftarrow i + 1$, jestli je $i \leqslant k$, tak jdi na krok 3.
7. $E_{kfold} = \frac{1}{k} E_{kfold}$.

Tabulka 1.2: Algoritmus k-násobné křížové validace.

Nezodpovězenou otázkou zatím zůstává, kde se vezme funkce f, zobrazující ze vstupního do výstupního prostoru. Na tuto otázku odpoví následující podsekce.

1.1.3 Metody strojového učení

Metody strojového učení, někdy také označovány jako algoritmy, či modely strojového učení, zobrazují vstupní prostor X na výstupní prostor Y pomocí funkce f. Jak ale najít tuto funkci? Odpověď není stejná u každého algoritmu strojového učení a může se lišit jak pro regresní, tak klasifikační metody. Často se používají gradientní optimalizační metody, které iterují postupné řešení, dokud se nedostanou do lokálního minima (logistická regrese, neuronové sítě). Někdy se používá implicitní řešení (lineární regrese). Jindy stačí uchovat trénovací data v paměti (k-NN). Postup získání funkce f pro daný algoritmus nazveme trénování. Nejčastěji se totiž k získání funkce f používají trénovací data, protože na testovacích datech chceme predikční schopnost funkce f kvantifikovat. Pokud jsme již našli funkci f pro příslušný algoritmus, říkáme, že tento algoritmus je naučený, natrénovaný nebo nafitovaný.

Existuje celá řada metod strojového učení. Každá z metod má svoje předpoklady na vstupní a výstupní prostor. Tím pádem má každá metoda své slabiny i silné stránky a podle teoretické práce [14] neexistuje učící algoritmus, který by dominoval všechny ostatní algoritmy pro všechny problémy učení s učitelem. Proto se v praxi využívá znalosti problému k lepší volbě učícího algoritmu. V této diplomové práci jsme buď použili některé z těchto metod, nebo se o nich často zmiňujeme. Proto je vhodné si některé metody strojového učení, které jsou pevně zakotveny v rámci teorie statistického učení [15], v základu představit.

k-NN

Zkratka k-NN se do českého jazyka překládá jako metoda k-nejbližších sousedů. Tato metoda se převážně používá pro klasifikační problémy, a proto si popíšeme jen její klasifikační verzi, která není těžká

na pochopení. Model k-NN je už naučen tím, že trénovací množinu dat máme uschovanou v paměti. Klasifikaci nového pozorování ovlivňuje k jeho nejbližších sousedů z trénovací množiny. Pro měření vzdálenosti nového pozorování od jeho k nejbližších sousedů se nejčastěji používá Eukleidovská vzdálenost. U těchto sousedů je známa kategorie příslušnosti. Nejbližší sousedé nového pozorování provedou hlasování, kdy každý soused hlasuje pro svoji kategorii. Ve vážené verzi k-NN [16] se hlasování mohou vážit. Hodnota váhy je proporcionální k inverzní hodnotě vzdálenosti. Klasifikace nového pozorování připadne kategorii, která získala v hlasování největší zastoupení.

Speciálním případem metody k-NN je tzv. většinový klasifikátor, kdy se počet sousedů rovná počtu dat ($k = n$). Jak je podle názvu zřejmé, většinový klasifikátor predikuje většinovou kategorii zastoupenou v datech. Většinový klasifikátor je vhodné použít jako srovnání s ostatními klasifikačními metodami zejména při tzv. nevyváženém datasetu.[1] Metodu k-NN jsme se rozhodli použít v experimentální části selekce algoritmu, protože je to nejčastěji používaná metoda pro tuto oblast.

SVM

Anglickou zkratku SVM (*support vector machine*) překládáme do češtiny jako metoda podpůrných vektorů. Pod pojmem SVM jsou chápány ve skutečnosti dvě metody. První regresní, méně známá metoda, je zastíněna svým klasifikačním protějškem. Regresní variantou se v této práci zabývat nebudeme a při použití zkratky SVM budeme nadále chápat klasifikační metodu.

Tuto metodu navrhl Vapnik v [17]. Základní verze SVM klasifikuje data na dvě kategorie. Po natrénování se z SVM stává binární klasifikátor, který vytváří mezi dvěma kategoriemi co nejširší mezeru. Pokud nejsou data lineárně separovatelná, tak je SVM schopen provést i nelineární klasifikaci. U nelineární klasifikace se data nejprve transformují do prostoru příznaků o vyšší dimenzi, kde se použije tzv. jádrový trik. SVM separuje v tomto výše-dimenzionálním prostoru transformovaná data lineární nadrovinou tak, aby znovu byla co největší mezera mezi pozorováními z různých kategorií. Nakonec se pozorování s nadrovinou transformují zpět do původního prostoru. Zpětně transformovaná nadrovina už není lineární.

Pro klasifikaci dat do více kategorií se provede několik binárních klasifikací a ty se pak ve výsledku zkombinují. Při k kategoriích se celkově provede $k \cdot (k-1)/2$ binárních klasifikací, tedy všechny dvojice kategorií. Pozorování, u kterého chceme predikovat patřičnou kategorii, získá po této proceduře celkem $k \cdot (k-1)/2$ hlasů (z každé binární klasifikace jeden). Tyto hlasy se sečtou a pozorování následně získává klasifikaci s největším počtem hlasů.

Logistická regrese

Logistickou regresi lze použít ve skutečnosti i na klasifikační úlohu. Logistická regrese představuje nejrozšířenější případ zobecněného lineárního modelu [18]. Je to nejčastěji používaná metoda v situaci, kdy se modeluje závislost binární odezvy na jedné, nebo více nezávislých proměnných, které jsou buď spojité, nebo diskrétní.

Pomocí $\mathbf{Y} = (Y_1, \ldots, Y_n)^T$ označme hodnoty binární odezvy, jejíž hodnoty 1 (resp. 0) modelujeme pomocí alternativního (nula-jedničkového) rozdělení jako

$$Y_i \sim \mathrm{Alt}\,(\pi_i)\,, \quad i = 1, \ldots, n. \tag{1.1}$$

Hodnota odezvy se často označuje jako zdar, nebo nezdar. Označme přitom pomocí π_i pravděpodobnost zdaru pro i-té pozorování, což je zároveň i střední hodnota odezvy, tzn.

$$EY_i = P\,(Y_i = 1) = \pi_i, \quad i = 1, \ldots, n.$$

[1] V nevyváženém datasetu jsou jednotlivé třídy pro klasifikaci zastoupeny nerovnoměrně.

Takzvaný logit, čili hodnota logitové funkce, se definuje jako logaritmus poměru šancí

$$\text{logit}\,(\pi_i) = \log \frac{\pi_i}{1-\pi_i}, \quad i = 1, \ldots, n$$

a modeluje se jako lineární odezva nezávislých proměnných v modelu, který je plně popsán vzorcem (1.1) spolu s předpisem

$$\log \frac{\pi_i}{1-\pi_i} = \beta_0 + \beta_1 X_{i1} + \beta_2 X_{i2} + \cdots + \beta_p X_{ip} \quad i = 1, 2, \ldots, n.$$

kde X_{ij} je i-té pozorování j-té nezávislé proměnné. Odhady regresních parametrů β se typicky hledají metodou maximální věrohodnosti.

Tímto způsobem je možné odlišit pouze dvě kategorie. Pro odlišení více kategorií je možné použít logistickou regresi několikrát a tyto výsledky zkombinovat tak, jak to bylo popsáno u metody SVM.

Neuronové sítě

Velmi významným trendem je tzv. hluboké učení [19]. Hluboké učení je disciplína v rámci strojového učení, která se zabývá neuronovými sítěmi s velkým počtem vrstev. Inspirací neuronových sítí je snaha napodobit biologický mozek. Hloubkou modelu se nazývá počet vrstev, které jsou složeny z umělých neuronů. Vrstvy jsou za sebou zapojeny tak, že výstup jedné z nich je vstupem následující vrstvy. Hloubka se v praxi pohybuje v řádech desítek a více vrstev. Pro odhad parametrů sítě (trénování) se obvykle používá algoritmus zpětného šíření chyby [20]. Modely hlubokých neuronových sítí dosahují daleko lepších výsledků než klasické algoritmy strojového učení. To za cenu snížení interpretovatelnosti, což při některých aplikacích nehraje tak důležitou roli.

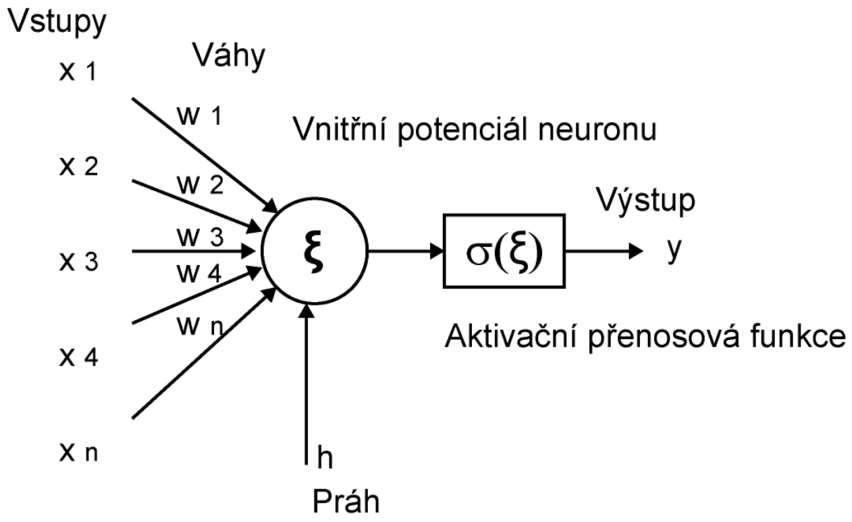
Neurony jsou vzájemně propojeny a navzájem si předávají signály a transformují je pomocí určitých přenosových funkcí. Neuron má libovolný počet vstupů, ale pouze jeden výstup (viz obrázek 1.4), kde výstupem neuronu je hodnota y daná tvarem

$$y = \sigma\left(\sum_{i=1}^{n} w_i x_i + h\right), kde$$

- x_i jsou vstupy neuronu
- w_i jsou synaptické váhy
- h je práh
- σ je přenosová neboli aktivační funkce

Čím je vyšší hodnota velikosti váhy w_i vstupující do neuronu, tím je daný vstup důležitější. Práh h označuje hranici, která rozděluje, kdy je neuron v aktivním, nebo pasivním stavu. Podle typu neuronu a typu neuronové sítě se použije vhodná přenosová funkce. Nejčastěji používanými přenosovými funkcemi jsou skoková, sigmoidální, přenosová funkce hyperbolické tangenty nebo radiální báze. Model neuronové sítě, který obsahuje jen jeden neuron (tzv. perceptron [21]) a používá sigmoidální přenosovou funkci, je ekvivalentní logistické regresi. Podle způsobů propojení neuronů a jejich přenosových funkcí existuje více různých architektur neuronových sítí.

První kategorií jsou vícevrstvé dopředné sítě, kdy výstup jedné vrstvy je připojen na vstup následující vrstvy a signál se šíří pouze ze vstupu sítě na její výstup. Speciálním případem těchto sítí je již zmiňovaný perceptron. Dopředné sítě se používají převážně pro problémy učení s učitelem, kdy poslední vrstvu tvoří jen jeden neuron (regrese), nebo poslední vrstvu tvoří několik neuronů (klasifikace).



Obrázek 1.4: Model umělého neuronu.

Speciální kategorií dopředných sítí jsou konvoluční neuronové sítě (CNN), které se terminologicky řadí do separátní kategorie. CNN pomocí konvolučních vrstvy[2] spojují jednotlivé neurony v sousedních vrstvách. Další, často používanou vrstvou CNN, je tzv. pooling vrstva, která nejčastěji aplikuje redukci dimenze. Na konci architektury CNN se typicky nalézá plně propojená vrstva, která slouží například ke klasifikaci obrázku do příslušné kategorie. CNN jsou například v oblasti počítačového vidění díky své vysoké úspěšnosti tzv. *state-of-the-art*.[3]

Poslední, hlavní kategorií neuronových sítí jsou rekurentní neuronové sítě (RNN). Předchozí modely architektur neuronových sítí vedly signál vždy jedním směrem a tvořily tedy acyklický graf. Tato architektura je vhodná při řešení problému, kde předchozí vstupy nějak ovlivňují vstupy následujícím. Architektury RNN tvoří cyklický graf a používají se tehdy, pokud mají být jejich vstupem posloupnosti různých délek, kde se jednotlivé položky ovlivňují.[4] Tím, že do architektury přidáme spoj vedoucí zpět, vznikne v RNN tzv. vnitřní stav. Za velkým množstvím zmedializovaných aplikací strojového učení stojí síť zvaná *Long Short Term Memory* (LSTM). LSTM, navržená v [22], je speciální druh RNN sítí, která je schopna se efektivně učit dlouhodobé závislosti. Toho je docíleno uchováváním vnitřního stavu buňky LSTM, který ovlivňují tři brány (zapomínající, vstupní a výstupní).

Hluboké neuronové sítě se používají především díky své schopnosti aproximovat jakoukoli funkci s dostatečnou přesností [23] a schopnosti pracovat s vysoce-dimenzionálními daty tak, že se adekvátně zvětší architektura neuronové sítě. Velmi často se pod pojmem umělá inteligence[5] myslí použití hlubokých neuronových sítí. Někteří autoři rozlišují mezi tzv. hlubokými a řídkými modely. Hluboké modely jsou neuronové sítě s takovým dostatečným počtem vrstev, že dokáží zachytit komplexní závislosti v datech. Řídké modely jsou všechny ostatní modely strojového učení.

[2] Konvoluční vrstva aplikuje techniku, kdy se použije menší čtvercová oblast (např. o velikosti 3 pixely), která se plynule posouvá po obrázku a slouží jako vstup to jednoho z neuronů další vrstvy.

[3] Anglický pojem state-of-the-art označuje technologie, které jsou v současnosti nejúspěšnější v dané oblasti.

[4] To může být v aplikacích jako předpovídání časových řad, strojový překlad, nebo generování textu.

[5] Umělá inteligence je tzv. buzzword pojem. Buzzword neboli umbrella-term je slovní spojení, nejčastěji nový technologický pojem, jehož význam není přesně definován, protože tento pojem často zahrnuje velmi mnoho oblastí. Užívá se často mezi lidmi, kteří znají v jakém kontextu je tento pojem použit. Jedním z buzzword pojmů je i metaučení.

Ačkoli máme kvůli přehlednosti zařazeny modely neuronových sítí do kategorie učení s učitelem, neuronové sítě se často používají i pro ostatní typy učení. Disciplínou učení bez učitele se v této práci zabývat nebudeme, proto jí nebudeme věnovat zvláštní sekci. Tato disciplína se zabývá metodikou, jak nejlépe seskupit množiny neoznačených[6] dat do skupin. To se využívá například v klastrovací analýze [24] a v odhadech hustoty pravděpodobnosti [25].

Parametry vs hyperparametry

Další často používané algoritmy strojového učení zahrnují metody lineární regrese, o kterých se zmíníme v kapitole 4. Často používanými metodami jsou rozhodovací stromy [26], náhodné lesy [27], lineární diskriminační analýza (LDA) [28] nebo Bayesův klasifikátor [15]. Skoro všechny modely strojového učení obsahují vnitřní parametry a hyperparametry. Jaký je ale mezi nimi rozdíl? A je mezi nimi přesně daná hranice?

V literatuře autoři často zaměňují tyto dva pojmy, a proto pro neznalého čtenáře může být orientace trochu obtížná. Strojové učení je disciplína, ve které je cílem sestrojit algoritmus, který nebude deterministický, ale bude se v průběhu učit zákonitosti na základě vstupních dat. Tento algoritmus se učí zákonitosti tak, že pozměňuje svoje vnitřní parametry. Algoritmus ale k učení používá jiný (optimalizační) algoritmus.[7] Při tomto nastavení má optimalizační algoritmus také nějaké vnitřní parametry.[8]

V podstatě se dá rozlišit mezi parametry a hyperparametry pouze z kontextu, v jakém o nich mluvíme. Parametry rozumíme takové proměnné, které systém optimalizuje sám. Hyperparametry rozumíme konstantní veličiny, které jsou do systému nastaveny pevně a už je dále v průběhu trénování neoptimalizujeme. Často se ale hovoří o optimalizaci hyperparametrů [29], což implikuje, že hyperparametry, na které jsme se dívali v systému jako na pevně dané, se najednou stávají proměnnými parametry, jejichž optimální hodnotu hledáme. Poté se mění i kontext, v jakém se díváme na celý systém a jako hyperparametry v tomto kontextu považujeme nově zanesené neměnné konstanty systému.[9] Pokud někdo mluví o optimalizaci hyperparametrů, tak má na mysli vylepšit stávající systém tím, že sestrojí novou meta-úroveň, která hledá původní pevně dané konstanty systému automaticky.[10] Tato myšlenka nechala vzniknout obor metaučení, které nad stávajícím systémem strojového učení vytvoří meta-systém. Podrobněji se na metaučení podíváme v další kapitole.

1.2 Zpětnovazební učení

V další kapitole se zmíníme o některých principech metaučení využívajících zpětnovazebního učení. Rozhodli jsme se tedy tento obor krátce uvést. Zpětnovazební učení, které je velmi podrobně popsáno v knize [31], je oblast inspirovaná neurovědou.[11] Konkrétní myšlenka, která využívá neurovědy, je napodobit odměňovací systém savců, ve kterém velmi významnou roli hraje neurotransmiter dopamin. Dopamin

[6] Neoznačená neboli neolabelovaná data jsou data, u kterých máme k dispozici vysvětlující proměnné, ale nemámme k dispozici odezvu.

[7] Optimalizační metoda, která nejčastěji iterativně pozměňuje vnitřní parametry algoritmu, který optimalizuje. Tento algoritmus můžeme chápat také jako meta-metodu, která je postavena nad základním algoritmem a automaticky vybírá optimální parametry základního algoritmu. Tím tento optimalizační algoritmus usnadňuje uživateli práci tak, že uživatel nemusí prohledávat celou množinu vnitřních parametrů ručně. Může to být metoda zpětného šíření chyby, metoda největšího spádu ap.

[8] Těmito parametry mohou být např. krok učení, počet iterací, typ predikční chyby, samotná architektura modelu, a také i samotný algoritmus.

[9] Nové hyperparametry metody optimalizace vyššího řádu, která optimalizuje původní hyperparametry, nyní parametry nižšího řádu.

[10] Takto by se dala meta-úroveň k dosavadnímu systému konstruovat donekonečna, protože nový meta-systém by měl další svoje pevně dané hyperparametry. Tomu se v literatuře říká nekonečný regres, o kterém přemýšlel už Aristoteles [30].

[11] Neurověda se zabývá problematikou, jak nervový systém ovlivňuje tělesné funkce a chování živočicha.

vyvolává příjemný pocit za takové chování, které vede k přežití a rozmnožení jedince. Jedinec se tímto procesem naučí opakovat prospěšné chování. Počítačové vědce proto napadlo tuto myšlenku přenést i do prostředí algoritmů, kde jedince nazýváme agentem.

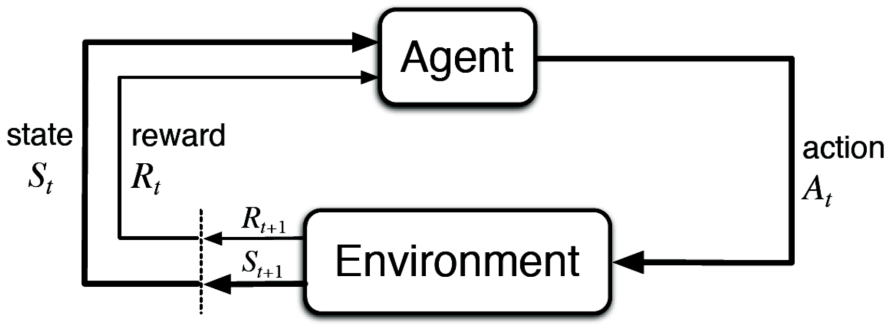
Cílem agenta ve zpětnovazebném učení je naučit se takovou strategii (chování), které maximalizuje jeho celkovou odměnu. Tuto odměnu získává z prostředí, pokud danou strategii používá. Typicky se na začátku agent nechá náhodně prozkoumávat pevně definované prostředí a vykonávat nějaké akce. Za svoji strategii v prostředí je náležitě odměněn, nebo potrestán a do další iterace je jeho strategie pozměněna s vizí dosáhnout lepší odměny. Prostředí můžeme formálně popsat jako markovský rozhodovací proces (MDP). Tento popis se zavedl z důvodu, že hodně algoritmů zpětnovazebného učení používá techniky dynamického programování [32]. MDP je zadaný čtveřicí (S, A, P, R), kde

- S je konečná množina stavů prostředí a stavů agenta
- A konečná množina akcí agenta
- $P_a\left(s, s^{'}\right)$ je přechodová funkce[12] udávající pravděpodobnost, že pokud agent vykoná akci a ve stavu s, tak přejde do stavu $s^{'}$
- $R_a\left(s, s^{'}\right)$ je funkce popisující odměnu, kterou agent získá, pokud ze stavu s provede akci a a dostane se do stavu $s^{'}$.

Chování agenta je popsáno pomocí strategie $\pi : S \times A \rightarrow (0, 1)$, kde $\pi(s, a)$ je pravděpodobnost provedení akce a ve stavu s. Cílem je najít takovou strategii π, která maximalizuje celkovou agentovu odměnu danou výrazem

$$\sum_{t=0}^{\infty} \gamma^t R_{a_t}\left(s_t, s_{t+1}\right), kde$$

$a_t = \pi(s_t)$ je akce provedená agentem v kroku t a $\gamma < 1$ je tzv. diskontní faktor zajišťující, že suma je konečná. Znázornění zpětnovazebného učení je na obrázku 1.5.



Obrázek 1.5: Zpětnovazební učení [31].

Trénování agenta probíhá tak, že se nejdříve pro každý stav zadefinuje tzv. hodnotová funkce, která udává, nakolik je výhodné se nacházet v daném stavu, nebo v něm provést danou akci. Tuto hodnotovou funkci na začátku neznáme a postupnými experimenty agenta v prostředí ji optimalizujeme tak, aby pro

[12] Přechodová funkce musí splňovat tzv. markovskou podmínku, tj. že závisí pouze na aktuálním stavu s a akci a a nikoli na stavech předchozích.

každý stav zvolil agent akci o nejvyšší hodnotě. To se dělá pomocí metod dynamického programování, temporální diference, nebo Monte Carlo metody [31]. Jedním z problémů, který nastává ve zpětnovazebném učení je otázka, jaký poměr má agent volit mezi průzkumem prostředí a jeho využitím [33]. Čistý průzkum zlepšuje model, ale model se nikdy nevyužije. Čisté využití zase vede k sub-optimální strategii. Jednou z dobrých variant je zpočátku více prozkoumávat prostředí s vírou v nalezení lepších cest a později s větším porozuměním prostředí stačí menší průzkum.

Zpětnovazební učení ještě není masivně používáno v průmyslu, ale je zatím převážně ve stavu výzkumu např. v robotice. Je to ale oblast, která ve vědecké komunitě často vyvolává nadšení, protože systémy používající zpětnovazebního učení se například dokázaly naučit hrát množství počítačových her na expertní úrovni a porazit ty nejlepší hráče. Zajímavou aplikací zpětnovazebního učení je optimalizace energetické náročnosti datových center společnosti Google o 40 procent.

Kapitola 2

Metaučení

Cílem této kapitoly je seznámit čtenáře s velmi širokým pojmem metaučení, často nazývaného učení se učit. Čtenář zjistí, proč je metaučení důležité v oblasti strojového učení a vytvoří si širší představu, kde se s metaučením může setkat. Dále uvedeme několik významných aplikací a velmi rychle se rozvíjejících oblastí, které mnozí autoři zařazují do oblasti metaučení.

2.1 Motivace

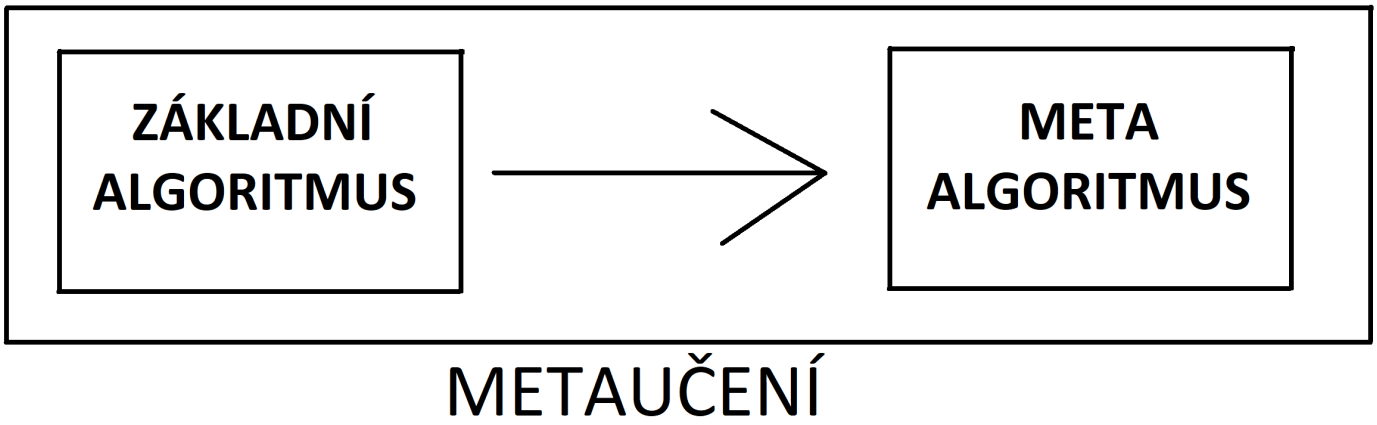
V předchozí kapitole byl čtenář seznámen s oborem strojového učení. Cílem strojového učení je sestavit takový systém, který nebude explicitně předprogramován, ale bude se učit průběžně danou úlohu řešit co nejlépe. To bylo dosaženo tak, že jsme původnímu deterministickému algoritmu přiřadili nějaké vnitřní parametry, které učícímu algoritmu dovolujeme v průběhu učení měnit tak, aby tyto parametry zachytily co nejlépe problém, který chceme řešit. Vložením těchto vnitřních parametrů jsme z původního algoritmu vyrobili algoritmus komplexnější, ale zároveň mnohem obecnější. Náš nový algoritmus je nyní schopen řešit mnohem širší třídu úloh než jeho deterministický protějšek. Nad deterministickým algoritmem jsme vytvořili tzv. meta-úroveň, meta-model neboli meta-algoritmus. Tato myšlenka se v širším kontextu nazývá strojovým učením.

Vytvořením tohoto systému jsme ale byli nuceni zavést jisté předpoklady a omezení, které omezují algoritmus strojového učení pro řešení jen jisté třídy úloh. Co když budeme chtít sestrojit algoritmus obecnější? Mohl by náš algoritmus řešit větší třídu úloh? Odpověď na tuto otázku je ano. Metodou, jak toho docílit, je sestrojit další algoritmus, který se na náš původní algoritmus, nebo na některé jeho komponenty dívá jako na své vstupní parametry, které může měnit tak, aby dosahoval svého cíle. Sestrojili jsme novou meta-úroveň nad algoritmem strojového učení, která tento algoritmus optimalizuje (viz obrázek 2.1). Právě tohle sestrojení nové meta-úrovně se nazývá metaučení.

2.2 Definice

Co když ale budeme chtít nad touto novou úrovní sestrojit další meta-úroveň, nebo dokonce libovolný počet meta-úrovní? Co poté můžeme chápat jako metaučení? Jako metaučení můžeme chápat každý přechod na novou vyšší úroveň, nebo všechny přechody ze základní úrovně, které vedly na novou úroveň. Proto pojem metaučení je těžké přesně definovat a někteří autoři tvrdí, že přesná definice ani není možná.[1] Není prý ani možné přesně rozlišit, co se myslí přechodem na další úroveň, protože můžeme

[1] http://www.scholarpedia.org/article/Metalearning, 2.6.2020



Obrázek 2.1: Schéma obecného metaučení.

zkombinovat algoritmus základní úrovně s algoritmem meta-úrovně a označit ho jako nový algoritmus základní úrovně.

Ovšem v komunitě strojového učení se s pojmem metaučení můžeme setkat poměrně často. Lidé, kteří používají tento pojem, ho intuitivně chápou ve svém konkrétním kontextu. Tento kontext zahrnuje znalost toho, co je základní algoritmus a jakým způsobem se vytvoří meta-úroveň k jeho optimalizaci. Nejčastěji se pojem metaučení vyskytuje v souvislostech, kdy mainstream[2] používá nějaké dobře fungující algoritmy strojového učení a další skupina vědců se tyto algoritmy snaží vylepšit pomocí vytvoření meta-úrovně. V praxi je často jednodušší rozeznat, co je metaučení. V teorii je ale určení pevné hranice velmi těžké.

Pro lepší intuici uveďme následující příklad [2]. Představme si člověka snažícího se vybrat ten správný parametr šířky jádra pro metodu odhadu pomocí jader [34]. V tomto příkladě člověk hraje roli meta-modelu a jádrová metoda hraje roli základního modelu. Člověk ví na základě předchozích zkušeností, jakým hodnotám hyperparametrů se má vyhnout, aby nedošlo například k přeučení. Cílem metaučení je tuto lidskou zkušenost přenést do meta-modelu a vytvořit tak automatický systém pro nalezení optimální šířky jádra. Tento automatický systém musí obsahovat objektivní funkci, která měří úspěšnost základního algoritmu a také pravidla, jak tento algoritmus měnit.

2.3 Terminologie

Hlavní motivací aplikace metaučení je vylepšení stávajících metod automatizace procesů. To představuje určitou metodologii ve strojovém učení, která se vžila pod názvem automatické strojové učení. To se stává stále více populární v počítačových vědách a dolování dat [35]. Pokud se na metaučení budeme dívat z pohledu Bayesovské statistiky, tak metaučení extrahuje apriorní informaci z tzv. meta-trénovací množiny úkolů neboli metadat[3], což mu pak usnadňuje učení nového úkolu. Učení nového úkolu využívá tuhle apriorní informaci, aby byla nalezena množina aposteriorních parametrů modelu. Pojmy metaučení, učení se učit, automatické strojové učení a učení se z metadat jsou ekvivalentní. Rozdíl je pouze v tom,

[2]Mainstream neboli hlavní proud je soubor myšlenek a postojů, které reprezentují či jsou konformní s názory, hodnotami, vkusem apod. většiny společnosti.

[3]Metadata neboli meta-znalosti jsou definována jako informace obsažené v datech. Ty můžeme použít pro metaučení jako apriorní informace z dat. U datového souboru to mohou být například základní statistiky jako je průměr z dat, počet dat, počet vysvětlujících proměnných, šikmost, špičatost, atd. U algoritmu to mohou být jeho hyperparametry nebo algoritmus samotný.

v jakém kontextu se tyto pojmy používají. To znamená, že se liší podle toho, jaká metadata používají k učení meta-modelu.

Hlavním cílem metaučení je použití takových metadat, aby se metaučení mohlo stát automatické a flexibilní. Použitím správných metadat lze vylepšit stávající algoritmy učení, nebo se samotný algoritmus učení naučit. Proto se pro metaučení používá alternativní pojem učení se učit. Algoritmy meta-úrovně používají předchozí zkušenosti (meta-znalostí) ke změně algoritmů základní úrovně, nebo jeho určitých aspektů tak, aby modifikovaný algoritmus základní úrovně byl lepší, než jeho nemodifikovaný protějšek při nevyužití meta-znalostí. Využití meta-znalostí může být buď z předchozí epizody učení na stejných datech (viz subsekce o kombinování modelů 2.5.2), nebo se může využít meta-znalostí společných pro různé problémy (viz subsekce o transferu znalostí 2.5.4).

Jednou z inspirací zavedení myšlenek metaučení do počítačových věd jsou metakognitivní vědy. Jejich počátky sahají do starověkého Řecka v dobách slavného filozofa Aristotela. Metakognitivní vědy popisují fenomén, kdy lidé nejen myslí, ale také myslí o samotném procesu myšlení a procesu učení. Metaučení v tomto kontextu znamená naučit se něco o procesu učení. Což je ekvivalentní definici metaučení v počítačových vědách. Člověk aplikující metaučení je schopný svoji strategii učení přizpůsobit tak, aby maximalizoval svůj užitek podle požadavků konkrétního úkolu.

V momentě, kdy přemýšlíme, jaký postup pro naučení matematiky bude nejefektivnější, probíhá metaučení. Zda bude lepší jít na přednášku, jít na cvičení, doma počítat příklady, nebo si číst skripta. Volíme samozřejmě na základě nahromaděných předchozích zkušeností tu metodu, o které si myslíme že nám nejvíce pomůže. To může být metoda, která byla v minulosti neefektivnější, kombinace potenciálně dobrých metod, nebo metoda kterou ještě neumíme, ale očekáváme od ní, že nám zvýší produktivitu učení. Na základě meta-znalostí pak optimalizujeme naše rozhodnutí, jak se co nejefektivněji učit na zkoušku z matematiky v omezeném čase [36]. Naše rozhodování se častokrát stává více-kriteriální, ve kterém hrají roli i další faktory, jako je například zábavnost metody, riziko neúspěchu metody ap. Někteří autoři tvrdí, že i samotná vědecká metodologie se dá považovat za formu metaučení [37].

2.4 Aplikace

Termín metaučení je tzv. buzzword, což znamená, že tento termín pokrývá širokou škálu konceptů a aplikací. V této sekci se pokusíme shrnout ty nejdůležitější aplikace metaučení a v další sekci se na tyto aplikace podíváme podrobněji.

Schopnost metaučení učit se na dvou úrovních[4] se považuje za velice zásadní přístup pro zlepšení umělé inteligence. Dále má metaučení významné postavení v systémech pro automatickou selekci modelu, při transferu znalostí a kombinaci modelů. Za metaučení se také považuje hledání optimálních architektur neuronových sítí, optimalizace hyperparametrů a v neposlední řadě i obor rychlého učení z malého množství dat. Metaučení se také používá při učení robotů konkrétního úkolu v neznámém prostředí, při učení algoritmů schopných se adaptovat na několik problémů podle potřeby a mnoho dalších.

Sestrojení aplikace s meta-úrovní má hned několik benefitů. Víceúrovňové aplikace častokrát vykazují daleko lepší predikční výsledky než jejich komponenty. Některé systémy využívající metaučení jsou schopny se velmi rychle adaptovat na změny v prostředí a přizpůsobit se na více úkolů.[5] Tento problém může být řešen pomocí transferu znalostí, který je podoborem více-úkolového učení [38]. Pro

[4] V první základní úrovni algoritmus akumulace meta-znalosti z každého jednotlivého úkolu. Tento proces je řízen druhým meta-algoritmem, který se učí z meta-znalostí extrahovaných základním algoritmem. Neboli učit se v rámci každého předloženého úkolu, zatímco shromažďovat znalosti o podobnostech a rozdílech mezi jednotlivými úkoly.

[5] Úkoly rozumíme různé druhy problémů vhodných pro učící algoritmus strojového učení. Například to mohou být následující problémy. Binární klasifikace obrázku do kategorií pes a kočka, lokace chodce z obrázku a zodpovězení otázky z obrázku. Nebo navigace robota v bludišti, držení rovnováhy robota při chůzi a přeskakování objektů robotem.

tyto problémy se na více úkolech může trénovat jedna sdílená neuronová síť. Zároveň požadujeme, aby tyto aplikace vyžadovaly co nejméně dat, tak k tomu můžeme využít obor zabývající se učením z velmi málého množství dat [39], pomocí kterého vhodně inicializujeme váhy neuronové sítě z množiny předchozích úkolů.

Velmi širokým oborem spadajícím pod techniky metaučení je obor optimalizace hyperparametrů, který se překrývá s mnoha aplikacemi uvedenými výše. Optimalizace hyperparametrů se zejména využívá v systémech pro automatické strojové učení (viz sekce 2.5.3). Hyperparametry jsou neměnné konstanty učícího algoritmu, na kterých závisí proces učení (viz kapitola 1). Proces metaučení postaví nad základním algoritmem meta-algoritmus[6], který optimalizuje základní algoritmus tak, že se snaží nalézt jeho optimální hodnoty hyperparametrů pro daný úkol. Jak ale prozkoumat co nejefektivněji prostor hyperparametrů? K tomu se používá několik standardně osvědčených metod. Dvě state-of-the-art metody pro prohledávání prostoru hyperparametrů, které se velmi často používají v systémech pro automatické strojové učení jsou tzv. Bayesovská optimalizace[7] [40] a Hyperband[8] [42].

Další aplikace zahrnují například hledání a konstrukce lepších algoritmů na základě meta-znalostí. V následující sekci se pokusíme uvést často používané praktické aplikace metaučení více dopodrobna. Tak by měl čtenář získat ucelenější představu a přehled toho, co může všechno nalézt v literatuře pod pojmem metaučení.

2.5 Příklady

Jakýkoli systém, který se řadí pod pojem metaučení, by měl odpovídat na alespoň jednu z následujících otázek. Co se má základní model učit? Který model se má základní model učit? Jak se má základní model učit?

Odpověď na otázku, co se má má základní model učit, přináší například přístup transferu znalostí [43], více-úkolového učení [38], nebo učení z málo dat [44]. Cílem je naučit se, jaké hodnoty parametrů základních modelů zachytí co nejlépe společné rysy jednotlivých úkolů tak, aby učení nového úkolu bylo co nejméně náročné a co nejrychlejší.

Odpověď na otázku, jaký model se má základní model učit, přinášejí přístupy automatické selekce algoritmu [45] a optimalizace hyperparametrů [46]. Cílem je naučit se pravidlo, které rozhodne, jaké základní modely dosáhnou nejlepší úspěšnosti na daném úkolu. Meta-informace zachycují korelace mezi jednotlivými úkoly a také úspěšnosti základních modelů na jednotlivých úkolech.

Odpověď na otázku, jak se má základní model učit, přinášejí optimalizačně založené přístupy metaučení [47], [48], kde cílem není učení pro predikci, ale učení o samotném procesu učení. Meta-informace zachytí podobnosti mezi dobrými základními modely.

[6] Nejčastěji se jedná o kombinaci algoritmu, který prozkoumává prostor hyperparametrů (standardně se používají metody mřížkového hledání, náhodného hledání, evoluční algoritmy a gradientní algoritmy) a algoritmu, který zhodnotí, jak je základní algoritmus úspěšný pro danou množinu hyperparametrů (např. křížová validace).

[7] V bayesovské optimalizaci je třeba vybrat tzv. akviziční funkci, do které vneseme informaci, jak očekáváme, že se hyperparametry budou chovat. Tedy při daném nastavení hyperparametrů, jakou hodnotu ztrátové funkce očekáváme. Akviziční funkce hraje roli apriorní informace a postupnými experimenty (měřením) aktualizujeme tuto apriorní informaci na aposteriorní. Tímto postupem je realizován tzv. poměr mezi využitím prostoru a jeho prozkoumáním. Cílem je nalézt rozumný poměr mezi prozkoumáním velké části prostoru hyperparametrů a zároveň tento prostor prozkoumávat podrobněji kolem již nalezených slibných míst.

[8] Hyperband kombinuje myšlenku Bayesovské optimalizace a myšlenku efektivně využívat výpočetní prostředky. Efektivní využívání výpočetních prostředků spočívá v tom, že se v prostoru hyperparametrů trénuje jistá množina modelů. Tyto modely jsou v průběhu trénování evaluovány a po určitých časových intervalech je horší polovina modelů zahozena a výpočetní prostředky jsou soustředěny do lepší poloviny modelů [41].

Některé aplikace metaučení se snaží odpovídat na více než jednu z těchto otázek, a proto je nelze zařadit jen do jedné z právě popsaných kategorií. V následujících podsekcích informujeme čtenáře, v jakých konkrétních aplikacích může čtenář narazit na pojem metaučení.

2.5.1 Automatická selekce algoritmu

Automatickou selekci algoritmu, která je v literatuře méně zastoupená, uvádíme jako první, protože právě automatický výběr algoritmu jsme se rozhodli prozkoumat více dopodrobna jak teoreticky v kapitole 3, tak i experimentálně v kapitolách 4, 5. Proto se k tomuto tématu v této kapitole nebudeme dále rozepisovat.

2.5.2 Kombinování modelů

Kombinováním modelů neboli průměrováním modelů rozumíme meta-algoritmy, které jsou v literatuře často nazývané pod anglickým pojmem *ensambles*. Použití těchto modelů je motivováno zejména dosažením lepších výsledků, než dosahují jejich individuální komponenty [49], [50]. Riziko výběru špatného algoritmu je sníženo, přesnost roste, ale interpretovatelnost modelu klesá [51]. Toho využívají například datoví vědci, kteří svůj um predikce z dat demonstrují v soutěžích, jako je například Kaggle[9], kde interpredovatelnost není nejdůležitějším faktorem. Je velmi časté, že vítězné modely jsou právě kombinací nějakých základních metod pro predikční analýzu dat. Jedním z úspěšných příkladů používajících algoritmy kombinování modelů je vítězný algoritmus v soutěži pro doporučování filmů pro společnost Netflix [52].

Kombinovány mohou být základní metody strojového učení (viz subsekce 1.1.3) např. lineární regrese, rozhodovací stromy, logistická regrese, SVM, k-NN atd. Uvedeme několik velmi často používaných metod kombinace modelů. Po prozkoumání české literatury jsme se jejich názvy rozhodli z angličtiny nepřekládat. Velmi často používanou technikou v metodách kombinace modelů je převzorkování, kdy vybíráme s opakováním z datového souboru podmnožinu dat neboli vzorek. Vzorek se pak často používá k trénování modelu.

Bagging

Technika baggingu byla vynalezena v práci [53]. Datový soubor se několikrát převzorkuje. Na každém vzorku se natrénuje základní model a tyto modely se na konci zprůměrují (v případě regrese), nebo hlasují o vítězné třídě (v případě klasifikace). Tato technika je nejvíce efektivní, pokud je základní model velice citlivý na variabilitu v datech.

Boosting

Technika boostingu byla navržena v [54]. Zatímco bagging využívá citlivosti základního modelu na variabilitu v datech, boosting využívá slabou schopnost predikce základního modelu.[10] Jednoduché modely jsou zesíleny opakovaným sekvenčním trénováním na stejných datech. V každé iteraci sekvenčního trénování se mění váhy pro různá pozorování. Na začátku mají všechna pozorování stejnou pravděpodobnost pro výběr do nového vzorku. Klasifikační algoritmus je aplikován na daný vzorek. Pozorování, která jsou špatně klasifikována, získávají větší váhu. Tím je do dalšího převzorkování zvýšena pravděpodobnost jejich výběru. Znovu je použit klasifikační algoritmus a váhy u pozorování jsou aktualizovány a procedura pokračuje v cyklu. Tímto procesem se získá několik modelů, které pak jsou zkombinovány

[9] https://www.kaggle.com/

[10] Jednoduchý model (*weak learner*) je algoritmus, který není o moc lepší než náhodný klasifikátor.

například váženým hlasovacím mechanismem. Velmi populárním algoritmem využívající boosting je Adaboost [55].

Stacking

Metoda stackingu [56] využívá rozdíly mezi základními modely. Několik základních modelů je trénováno na stejném datovém souboru. Jejich predikce slouží jako nezávislé proměnné pro jiný meta-model, kde odezva zůstává stejná jako u základních modelů. Nové pozorování pak necháme predikovat základními modely. Jejich predikce slouží jako vstup do meta-modelu, jehož predikce nás zajímá. U klasického přístupu se používá jedno číslo, udávající predikci základních modelů. Pokud jsou základní modely pravděpodobnostní klasifikační algortitmy, tak jako vstup do meta-modelu se používá vektor pravděpodobností udávajících příslušnosti do jednotlivých kategorií [57].

Cascade generalization

Tato technika využívá také predikční rozdíly základních modelů [58]. Algoritmus funguje sekvenčně, kde predikční vektor prvního základního modelu se přidá do matice nezávislých proměnných pro další základní model. Tento proces je opakován, dokud algoritmus nedojde k poslednímu základnímu modelu.

Cascading

Tato méně používaná metoda je navržena v [59], [60] a je velmi podobná boostingu. Rozdíl je ten, že se na různé vzorky nepoužívá stejný model, ale různé základní modely. U základních modelů typicky zvyšujeme komplexitu modelu se vzrůstajícím pořadím v sekvenci.

Delegating

Metoda delegování [61] funguje také jako sekvenční zřetězení základních modelů. Rozdíl je ten, že pokud si není daný klasifikátor dostatečně jistý klasifikací daného pozorování, přenechá ho dalšímu klasifikátorovi v řadě. Proces jistoty je určen nastavení prahu. Tento proces končí tehdy, když už nejsou žádná volná pozorování k dispozici, nebo jsme došli k poslednímu klasifikátorovi. Výhodou tohoto přístupu je snížená výpočetní náročnost, protože klasifikátory v sekvenci pracují se sníženým počtem dat. Dalším pozitivem je menší ztráta interpretovatelnosti, neboť se zde nepoužívá žádné průměrování, ale každé pozorování je klasifikováno právě jedním klasifikátorem.

Shrnutí

Uvedli jsme jen několik nejčastěji používaných metod pro kombinaci modelů. Pokud by čtenář chtěl tyto a další algoritmy z kategorie kombinace modelů prozkoumat více, může prostudovat literaturu [51], [62], [63]. Metody kombinace modelů ze řadí do oblasti metaučení, protože tyto metody skládají jednotlivé modely základní úrovně pomocí například klasifikačního meta-modelu.

Zatímco jiné formy metaučení se často zaměřují na získání znalostí ze samotného procesu učení, v těchto modelech jde zejména o zvýšení úspěšnosti. Nevýhodou těchto metod bývá, že v důsledku kombinace základních metod klesá interpretovatelnost výsledného modelu. Někteří autoři zastávají přístup Okamovy břitvy [64], že je lepší mít jednoduchý horší model, než lepší komplexní. U komplexního modelu může nastat přeučení.

2.5.3 Automatické strojové učení

Automatické strojové učení (AutoML) je snaha zjednodušit a automatizovat co nejvíce procesů v práci datového analytika, nebo kohokoli aplikující učení s učitelem. Častokrát vede tato snaha k tomu, že systémy AutoML jsou do takové míry automatizované, že uživatel zpracovávající data nemusí vůbec rozumět tomu, co se v pozadí děje. Stačí mu vědět, co do systému vložit a co z něho může očekávat jako výstup. Tento přístup se někdy označuje jako model černé skříňky.[11] Nové systémy provedou základní manipulaci s daty místo uživatele, což ušetří uživateli velké množství času. Tyto základní manipulace se řadí za sebe do tzv. workflow pipeline, která častokrát obsahuje čištění a předzpracování dat, extrakci příznaků, selekci modelu, optimalizaci hyperparametrů a predikci. Workflow pipeline znamená sekvenci všech procesů ve strojovém učení od začátku (sběr dat) po konec (evaluace modelu). Uveďme si některé AutoML systémy používané v dnešní době.

TPOT

Vývoj v oblasti AutoML systémů je stále velice aktuální téma. Návrháři se snaží přijít technikami, které co nejvíce zefektivní nalezení optimálního modelu. Za zmínku stojí například systém TPOT [65], který začíná s jednoduchou pipeline, kterou technikami genetického programování [66] postupně upravuje a rozšiřuje tak, aby byla vhodná pro konkrétní data. Techniky genetického programování používají pro nalezení optimální pipeline evoluční operace jako mutaci, křížení a selekci.[12] Nové pipeline mohou být také generovány technikou self-play.[13]

Auto-WEKA

Auto-WEKA [68] je velice jednoduchá pro použití komukoli, kdo neumí použít programovacího jazyka, neboť poskytuje grafické uživatelské rozhraní (GUI). Je to první AutoML systém, který začal kombinovat zároveň selekci nejvhodnějšího algoritmu a nastavení optimálních hodnot hyperparametrů k tomuto algoritmu. V literatuře je tento pojem znám jako tzv. CASH (*Combined Algorithms Selection and Hyperparameter Optimization*) problém [69]. Pro optimalizaci hyperparametrů používá již zmíněnou velmi populární Bayesovskou optimalizaci, která se ukazuje být velice efektivní [70].

Auto-SKLearn

V současnosti je nejznámějším AutoML systémem Auto-SKLearn [71]. Auto-SKLearn se používá jen pro selekci klasických modelů strojového učení (ne neuronových sítí). V této oblasti je Auto-SKLearn v současnosti state-of-the-art. Auto-SKLearn používá ve své pipeline tři základní pilíře. Na začátku se pomocí metod automatické selekce algoritmu (viz kapitola 3) vybere množina potenciálně dobrých modelů. Dalším pilířem je Bayesovská optimalizace, která najde optimální možné nastavení hyperparametrů k daným algoritmům. Posledním pilířem je kombinace těchto modelů pomocí některé z výše uvedených metod kombinace modelů. Důležitou vlastností systému Auto-SKLearn je, že uživatel může nastavit výpočetní čas, jaký chce věnovat hledání kombinace modelů pro svůj problém.

[11] Černou skříňkou se v kybernetice označuje zařízení, u kterého je zřejmé, jak se projevuje navenek, ale nevíme nebo nás nezajímá, co všechno musí probíhat uvnitř tohoto zařízení.

[12] Tyto evoluční operace se v počítačových vědách používají pro kombinaci genetické informace obsažené ve dvou vybraných rodičích z jisté populace jedinců tak, aby vznikl potomek, který bude mít kombinované vlastnosti obou dvou rodičů.

[13] Technika self-play učí algoritmus hraním proti sobě. Algoritmus typicky nevyžaduje lidský dohled a učí se pomocí zpětnovazebného učení. Tato technika se stala velmi silnou zbraní při konstrukci algoritmů, které porážejí expertní hráče v počítačových hrách [67].

H2O a AutoKeras

Hluboké učení a neuronové sítě hrají v dnešním světě velice zásadní roli. Systémy H2O a AutoKeras [72] byly sestrojeny právě pro automatický výběr nejvhodnější neuronové sítě. Tyto systémy požívají techniky automatického návrhu neuronových sítí (viz sekce 2.5.5) a metody kombinace modelů neuronových sítí.

Cloudové systémy

Výše uvedené AutoML systémy patří do kategorie softwaru fungujícího lokálně na počítači. Společnosti jako Google[14], DataRobot[15] a další začaly poskytovat AutoML systémy v tzv. cloudu. To znamená, že veškeré výpočty neprobíhají na lokálním počítači, ale na obrovských tzv. clusterech těchto společností. Slovíčko cloud také v dnešní době patří do kategorie buzzword. To je dáno i tím, že některé IT společnosti chtějí implementovat cloudové technologie do svých systémů. Častokrát manažeři těchto společností přesně neví, co znamená pojem cloudových technologií, ale chtějí je, protože je používají všechny ostatní společnosti. Cílová skupina cloudových AutoML služeb jsou například softwaroví inženýři, kteří nemají znalost strojového učení a potřebují z dat rychle vydolovat nějaké znalosti.

Čtenář si může klást otázku, zda standardní práce datového analytika už dále nebude potřeba kvůli těmto AutoML systémům. Vědecká komunita se ale shoduje na tom, že schopnosti datového analytika nebudou v nejbližší době počítačem úplně nahrazeny. Jednak je potřeba člověk, který rozumí tomu, co se v pozadí doopravdy děje a může popřípadě detekovat problém a nastavovat hyperparametry AutoML systému. Dále AutoML systémy mají limitace ve fázi extrakce příznaků. Extrakci příznaků je proto ve většině případů vhodné svěřit člověku. V oblasti učení bez učitele a zpětnovazebním učení jsou AutoML systémy ještě ve fázi vývoje a čekají na další pokrok.

2.5.4 Transfer znalostí

Transfer znalostí [43], [73] je mechanismus, který si klade za cíl zlepšit se v učení nového úkolu s využitím znalostí získaných v předchozích úkolech. Protože přístup transferu znalostí nezačíná učením každého úkolu zvlášť, ale využívá určitou informaci z předchozích úkolů, tak jej vědecká komunita zařazuje také do oblasti metaučení [74], [75].

Motivace

Cílem transferu znalostí je, aby se učící mechanismus dokázal naučit společné rysy podobných problémů. Nejčastěji se k transferu znalostí používají neuronové sítě, kde společné rysy jsou zakódovány do jejich vah. Jedním přístupem je sekvenční transfer, kde je model trénován jednotlivými úkoly sekvenčně. Další možností je trénovat modely neuronových sítí paralelně s tím, že si neuronové sítě mezi sebou sdílí svoje části [45].

Podmnožinou transferu znalostí je tzv. více-úkolové učení [38], kde typicky poslední vrstva neuronové sítě slouží jako přepínač, mezi jednotlivými úkoly. Sdružená informace o všech úkolech je obsažena ve vahách neuronové sítě.

Aplikace

Dříve se transfer znalostí používal zejména při řešení menších, velmi podobných úkolů. V současnosti je možně použít například velké předtrénované hluboké neuronové sítě [76] natrénované například na klasifikaci koček a přeučit je několika obrázky psů na rozpoznávání psů. Pod pojmem úkoly rozumíme

[14] https://cloud.google.com/automl
[15] https://www.datarobot.com/

třídu podobných učících problémů. Tato třída může obsahovat například binární klasifikaci obrázků do kategorií pes a kočka, lokace chodce z obrázku a zodpovězení otázky z obrázku.

Jedním z největších úspěchů tohoto odvětví bylo natrénování hluboké neuronové sítě [77] na velém datasetu ImageNet [78], obsahující více než 14 milionů obrázků a 20 tisíc kategorií. Tuto inicializovanou neuronovou síť si každý může stáhnout z internetu a doladit si ji na konkrétní problém nejlépe z oblasti rozpoznávání obrazu.

Další aplikace nachází transfer znalostí v robotice a zpětnovazebním učení [79], [80]. Chování robota v prostředí je zakódováno v neuronové síti tak, aby agent plnil jistý úkol a maximalizoval tím svůj dlouhodobý očekávaný užitek. Pokud mu úkol změníme, jeho neuronová síť je už inicializovaná z předchozího úkolu, a proto netrvá tak dlouho, než se robot adaptuje na úkol nový [81]. Tímto způsobem mohou být roboti trénováni na více úkolů a mohou se tak stávat více všestrannými [82]. Poslední vrstva neuronové sítě může obsahovat například tolik neuronů, kolik má robot úkolů.

Další využití transferu znalostí je v oblasti návrhu Bayesovských sítí [83], které se dle typu úkolu mohou lehce pozměnit. Transfer znalostí může být použit i v klastrování [84], kde se jednotlivé úkoly shlukují do klastrů na základě vybraných charakteristik, což je velmi podobný přístup, jakým se budeme zabývat v další kapitole a zbytku diplomové práce. Výhody použití transferu znalostí zahrnují spoustu ušetřeného času díky dobře inicializovanému modelu. Nevýhody zahrnují neefektivnost transferu znalostí, pokud se nezvolí podobné úkoly [85] a také náchylnost k přeučení.

2.5.5 Automatický návrh architektur neuronových sítí

Hluboké modely neuronových sítí naprosto dominují v oblastech počítačového vidění, rozpoznávání mluvené řeči, strojového překladu, zpracování přirozeného jazyka, zpětnovazebního učení a mnoho dalších. Hluboké sítě typicky obsahují desítky milionů parametrů (vah) a skládají se z různých vrstev, které na sebe navazují a mají určitou funkčnost. Většina slavných hlubokých sítí byla navržena člověkem. Člověk dokázal využít svoji apriorní znalost problému, aby navrhnul dobře fungující hlubokou neuronovou síť [86].

Jako velice úspěšné architektury neuronových sítí můžeme uvést například sítě AlexNet [87], ResNet [88], DenseNet [89], Xception [90], PolyNet [91], SENet [92] a GoogleNet [93]. Tyto člověkem navržené modely ve své době překonávaly veškeré ostatní modely strojového učení, zejména v oblasti rozpoznávání obrazu na standardních datových souborech jako menší CIFAR-10 [94] a větší ImageNet [5]. Častý postup ručního návrhu sítě zahrnuje postupné přidávání jednotlivých bloků za sebe a sledování změny ztrátové funkce neuronové sítě. Dále prvotní testování architektury na menším datovém souboru. Pokud se prvotní testování ukáže jako úspěšné, může být otestován i větší dataset.

Trend navrhování neuronových sítí se ovšem změnil. Dnes hluboké neuronové sítě nenavrhuje člověk, ale počítač. Navrhování různých hlubokých architektur neuronových sítí je pro člověka často obtížné a počítačem vytvořené architektury velmi často překonávají architektury navržené expertem. Návrh architektury hluboké neuronové sítě počítačem je velice výpočetně a finančně náročná záležitost. Výpočty hledání architektur probíhají na obrovských výpočetních clusterech obsahujících tisíce grafických karet. Na grafických kartách je možné experimenty urychlit tak, že výpočty jednotlivých modelů probíhají paralelně nezávisle na sobě na jednotlivých jádrech grafické karty [95]. Při experimentu [96] trvajícím 28 dní bylo použito 800 grafických karet NVIDIA P100, jejíž současná hodnota se pohybuje okolo 150 tisíc Kč.

Výpočetní náročnost těchto experimentů je dána zejména tím, že prostor neuronových sítí je diskrétní a obrovský. Prostor může zahrnovat hyperparametry architektury (počet vrstev, počet neuronů ve vrst-

vách, aktivační funkci) a hyperparametry optimalizačního algoritmu (krok učení, dropout[16] ap.). Jedna evaluace se v tomto prostoru rovná natrénování konkrétní neuronové sítě na daný problém. Což samo o sobě není výpočetně levná záležitost. Těchto evaluací je třeba udělat velký počet, a tím prohledat určitou část diskrétního prostoru architektur neuronových sítí. Proto vyvstává otázka, jak nejlépe vnést lidskou expertní apriorní informaci do počítače tak, aby se co nejvíce zredukoval prostor architektur. Na tuto otázku se staží odpovídat články [98]–[100].

Velké společnosti pracující s daty si mohou vyčlenit pro nalezení optimální architektury neuronové sítě omezený rozpočet a kontaktovat například společnost Google, která jim za tyto prostředky automaticky najde optimální architekturu. Pro návrh architektur neuronových sítí se používají zejména postupy neuroevoluce a zpětnovazebního učení.

Neuroevoluce

Neuroevoluce [101] používá k optimalizaci architektury neuronové sítě evoluční operace, jako jsou mutace, křížení a selekce. Architektura neuronové sítě se vhodně zakóduje tak, aby na ni šlo použít tyto evoluční operace, které hrají roli meta-algoritmu [102]. Neuroevoluce začíná s populací architektur neuronových sítí, kterou iteruje a evolučními algoritmy a mění populaci tak, že z populace mají větší šanci předat svoji genetickou informaci jedinci s vyšší hodnotou účelové funkce [100].[17]

Kromě grafických karet probíhají některé výpočty i na Googlem speciálně navržených čipech TPU (*Tensor processing unit*), které jsou navržené pro tenzorové operace, vyskytující se při optimalizačních algoritmech pro aktualizaci parametrů neuronových sítí. Tento čip je kompatibilní s velmi populární knihovnou TensorFlow [103], která je určená pro programování neuronových sítí. Architektury navržené neuroevolucí nenachází uplatnění jen v rozpoznávání obrazu, ale i třeba při vývoji umělé inteligence hrající počítačové hry [104], při obchodování na burze nebo při řízení robotů.

Zpětnovazebné učení

V tomto přístupu návrhu architektury neuronové sítě hraje klíčovou roli RNN, nejčastěji LSTM, která je v této technice meta-algoritmem. LSTM je v roli agenta, který vykonává akce v podobě změny aktuální architektury neuronové sítě (základní algoritmus) a dostává za tyto akce odměny. Odměna je dána hodnotou ztráty neuronové sítě na konkrétním problému, kterou tzv. kontrolor LSTM optimalizuje [105]. LSTM optimalizuje svoje váhy pomocí gradientní metody.

Díky chytrým optimalizačním technikám se podařilo výpočetní náročnost hledání architektury neuronové sítě pomocí zpětnovazebného učení snížit v řádech tisíců [106]. Další studie, které výrazně zredukovaly výpočetní čas technikami, jako jsou prohledání diskrétního prostoru aktuální neuronové sítě a znovupoužití jejich vah, jsou [107], [108].

Další postupy

Velmi úspěšnou a populární se stala metoda DARTS (*Differentiable Architecture Search*) [109], která na architekturu nahlíží jako kompozici jednotlivých uzlů, mezi kterými jsou operace. Operacemi můžeme rozumět například aktivační funkce. Mezi vybranými dvěma uzly neuvažujeme jen jednu operaci, ale normalizované exponenciální rozdělení několika operací. Neuronová síť je optimalizována pomocí metody největšího spádu, která přepíná mezi optimalizací vah a návrhem architektury. Postupnými iteracemi

[16] Dropout [97] je regularizační technika vhodná pro snížení přeučení používaná při trénování neuronových sítí. Typicky se pro jednu epizodu ignorují (nejsou aktualizovány) některé spoje neuronové sítě.

[17] V tomto případě je účelová funkce ekvivalentní ztrátové funkci neuronové sítě na daném problému. Účelová funkce je až na znaménko ekvivalentní pojmům: cenová funkce, objektivní funkce, odměnová funkce, užitková funkce nebo fitness funkce a její použití většinou závisí na oboru.

se tohle rozdělení ustálí a na konci je vybrána ta operace, jejíž hodnota v normalizovaném exponenciálním rozdělení je největší.

Dalším možným postupem je nechat síť růst inkrementálně pomocí tzv. hypersítě [110], která v roli metaučitele postupně predikuje váhy neuronové sítě do každé vrstvy [111]. Typicky menší hypersíť zakóduje váhy a spojení větší neuronové sítě, kterou optimalizuje.

2.5.6 Učení z malého množství dat

V poslední době vyšlo na toto téma velké množství vědeckých článků. Zejména na optimalizační přístupy neuronových sítí [112]–[114]. Začátek tohoto směru se připisuje článku [115] z roku 2015, kde autor sestrojil klasifikační model, který dokázal správně klasifikovat kategorie datového souboru Omniglot [116], který obsahuje 1623 písmen z 50ti různých abeced. V tomto datovém souboru se nevyskytuje žádné písmeno dvakrát. Klasifikátor dokázal na základě podobnosti písmen v abecedě klasifikovat písmeno do správné abecedy.

Aplikace

Obor učení se z velmi malého množství dat se velmi rychle rozvíjí díky rozsáhlým aplikacím hlubokého učení a faktu, že při některých aplikacích nemáme spoustu dat, které neuronové sítě často potřebují. Může se jednat o aplikace jako klasifikace medicínských dat, doporučení filmu z řídkých dat v doporučovacích systémech, imitace pohybů člověka robotem a rychlou adaptaci robota na nový úkol, generování dialogu určitého člověka na základě jen několika vzorků lidské řeči nebo překlad do vzácných jazyků. Dále jsou tyto metody potřeba, pokud pracujeme s daty, jejichž distribuce má těžké konce. Může jít o jevy, které nastávají s malou pravděpodobností. Například v některých dopravních situacích při řízení vozidla, v interakcích s jinou osobou nebo rozpoznávání řídce používaných slov.

Jak to u velké části nápadů v umělé inteligenci bývá, i tento nápad čerpal inspiraci z přírody. Děti při rozpoznávání nových objektů nepotřebují objekt vidět v řádech tisíců jako neuronové sítě, ale stačí jim ho vidět párkrát, aby si ho zapamatovaly [117]. Tohle slabé místo, kde je velký rozdíl v rychlosti mezi lidským a strojovým učením začali výzkumníci rychle atakovat [118].

Základní myšlenkou je, že metaučící systém je trénován na velkém množství úkolů a pak je otestován, jak je dobře schopen plnit úkol nový. Příkladem úkolu může být klasifikace obrázku do 5 kategorií s tím, že každá kategorie má zastoupení jen jedním obrázkem. Nebo učení robota se efektivně pohybovat v novém bludišti jen po jednom průchodu [119]. Tento přístup je odlišný od standardního postupu, kde klasifikační model, či robota trénujeme jen na jednom úkolu. Meta-algoritmus trénuje základní model na úkolech, které patří do meta-trénovací množiny.[18] V celém procesu jsou zároveň optimalizovány dva různé modely. Základní model, který se učí nový úkol a meta-model, který optimalizuje základní model.

Tento obor má velký překryv s transferem znalostí. Při učení algoritmu na novém úkolu požadujeme, aby nám k tomu stačilo co nejméně dat. Při vývoji efektivnějších modelů je velice nepraktické učit se nový úkol úplně od začátku, a proto je tento přístup metaučení klíčový k výzkumu všestranných agentů, kteří se kontinuálně dokáží učit více věcí [115]. Metody pro řešení úloh s málo daty mohou být rozděleny do kategorií modelově, metricky a optimalizačně založených přístupů.

Modelově založené přístupy

Tento přístup používá jako základní modely RNN sítě (např. LSTM), které sekvenčně zpracovávají data pro konkrétní úkol. Meta-model optimalizuje základní model přes množinu všech úkolů pomocí

[18] Meta-trénovací množina je trénovací množina pro učící algoritmus meta-úrovně. V tomto případě jsou jednotlivé úkoly chápany jako individuální pozorování.

gradientní metody největšího zpádu. Základním modelem je také RNN síť. Modelově založené přístupy pro učení z málého množství dat pro základní modely využívající učení s učitelem můžeme nalézt v [120]. Články pro základní modely využívající zpětnovazebního učení může čtenář nalézt v [121], [122]. V těchto případech se musí agent ve formě LSTM, který maximalizuje svůj očekávaný užitek, naučit svoji učící strategii od začátku.

Metricky založené přístupy

Cílem metricky založených přístupů je naučit se měřit vzdálenost mezi objekty v metrickém prostoru tak, aby v něm učení (měření vzdálenosti) bylo co nejefektivnější. Tento postup se zatím výhradně používá pro obor učení z velmi malého množství dat. K těmto účelům můžeme trénovat např. Siamskou neuronovou síť [123], která měří vzdálenost mezi dvěma objekty, které jsou kódovány do poslední vrstvy této sítě. Cílem je dosáhnout menší vzdálenosti mezi objekty ze stejné kategorie a větší vzdálenosti mezi objekty z různých kategorií. K měření podobnosti objektů můžeme použít např. eukleidovskou [124], nebo Kosinovou [125] vzdálenost. Síť je nejdříve natrénována na několika různých datasetech (např. ImageNet a CIFAR-10). Do poslední vrstvy si zakóduje rozdíly mezi těmito datasety (úkoly). Naším cílem je nechat predikovat model z jednoho obrázku. Síť porovná obraz, který se pokoušíme klasifikovat, s datasety v metrickém prostoru. Meta-modelem je v tomto přístupu neuronová síť a základním modelem je metoda, která umí měřit vzdálenost v metrickém prostoru (např. k-NN). Tyto postupy se velmi často používají pro klasifikační úlohy, protože vykazují uspokojivé výsledky [126]. Chybí ale studie těchto metod pro regresní úlohy a zpětnovazební učení.

Optimalizačně založené přístupy

Další velice zajímavou myšlenkou je učení se optimalizačního algoritmu [2]. Optimalizační algoritmy jako metoda největšího spádu, nebo její modifikace, které využívají hybnosti, nebo proměnlivé délky kroku, jsou často navrhovány na míru pro konkrétní problém. Proto vyvstává otázka, zda jsme schopni se tyto optimalizační algoritmy naučit automaticky bez přidané apriorní znalosti problému. Je možné sestrojit meta-model, který bude schopen nastavit optimalizační algoritmus základního modelu tak, že pro nový úkol bude základní model požívat efektivnější optimalizační algoritmus? Podle článků citovaných níže to možné je, a dokonce automaticky navržené optimalizační algoritmy jsou schopny překonat jejich ručně navržené protějšky. Meta-model vybírá optimalizační algoritmus pro základní model na základě zkušeností (meta-znalostí) z předchozích úkolů. Kromě potenciálního zrychlení optimalizačních algoritmů je další motivací tohoto přístupu také usnadnění lidské práce při výběru a nastavení optimalizačního algoritmu pro metodu strojového učení.

První pokusy naučit se optimalizační algoritmus postupovaly intuitivně tak, že v prostoru hyperparametrů optimalizačního algoritmu (například kroku učení a parametru hybnosti) hledaly minimální ztrátu pomocí gradientní metody. Tento postup ale dost dobře nefungoval kvůli tzv. skládajícím se chybám. Proto se muselo využít jiného přístupu, a to konkrétně zpětnovazebného učení. Přístup zpětnovazebního učení, kdy se meta-model učí aktualizovat parametry základního modelu, byl navržen ve studii [48]. Tato studie se zaměřuje na učení optimalizačního algoritmu pro specifický úkol. Článek [47] si klade poněkud ambicióznější cíl, a to učení optimalizačního algoritmu pro jakýkoli úkol. Tyto studie ukazují, že automaticky nalezené optimalizační algoritmy jsou schopny překonat ručně navržené a nastavené algoritmy pro optimalizaci modelů.

Za meta-model se nejčastěji volí RNN, protože je potřeba, aby si meta-model pamatoval předchozí aktualizaci parametrů základního modelu. Za základní model se často volí neuronová síť, a proto později vyšly články popisující učící proces, který vede k efektivnější optimalizaci neuronových sítí [112], [113]. Hochreiter [2] se na základní model dívá jako na černou skříňku, jejíž vstupem je sekvence trénovacích

dat a výstupem je sekvence predikcí a modeluje ho pomocí RNN. Ve fázi metaučení se pomocí pevně daného meta-modelu trénuje základní model. V meta-modelu je zakódován základní model v podobě historie optimalizací základního modelu díky paměti RNN, a tedy optimalizační algoritmus.

Meta-model je často optimalizován pomocí zpětnovazebního učení, kdy výše jeho odměny závisí na úspěšnosti základního modelu. Úspěšnost může být vyjádřena například multikriteriální charakteristikou, která může zahrnovat kombinaci ztrátové funkce a času konvergence. Na základní model (optimalizační algoritmus), se v tomto kontextu díváme jako na strategii meta-modelu, který přijímá odměny podle toho, jak úspěšný je jeho základní model. Takto lze hledáním optimální strategie meta-modelu nalézt optimální základní model (optimalizační algoritmus). Poté, co je trénování agenta (meta-modelu) u konce, může být jeho naučená strategie (optimalizační algoritmus) použita k optimalizaci nové objektivní funkce.

V článku [105] je tento velice obecný přístup prezentován pro učení z velmi malého množství dat. Také je zde navržený postup inicializace parametrů sítě LSTM, který je při učení z málého množství dat nepostradatelný. Díky dobré inicializaci parametrů základního modelu jsme schopni jen pomocí malého množství kroků gradientní optimalizační procedury dosáhnout velmi dobré generalizace na novém úkolu. Elegantní postup pro inicializaci parametrů pro jakýkoli model byl zobecněn v článku [44]. Tento postup může být použit pro klasifikační i regresní úlohy a také pro problémy zpětnovazebního učení. Myšlenky ve výše uvedených článcích výrazně vylepšují efektivitu zpětnovazebního učení ve smyslu využití dat díky velmi dobré inicializaci parametrů základních modelů.

První pokusy sestrojit adaptabilní optimalizační algoritmus můžeme nalézt v práci Bengia [127], který se snaží naučit tzv. Hebbovo učící pravidlo[19] pro parametry neuronové sítě. O standardně používaném zpětném šíření chyby měl totiž pochybnosti, zda vůbec probíhá v lidském mozku. Další velmi obecný pokus navrhnout algoritmus měnící adaptivně vlastní váhy můžeme nalézt v práci [129]. Tento návrh se nezdá být příliš použitelný v praxi, kvůli jeho přílišné obecnosti.

2.5.7 Genetické programování

Tento obor umělé inteligence a informatiky spadá do oblasti sebe-modifikujících se programů, a proto jej spousta autorů také nazývá metaučením. Jde o techniku, kdy je počítačový program schopen pracovat s ostatními programy jako se vstupními daty, nebo se sebou samotným [130]. To znamená, že program může číst, analyzovat, generovat nebo transformovat sebe, i jiné programy. K modifikaci svého vlastního zdrojového kódu se často používá genetických algoritmů. Cílem programu je plnit určitý úkol, který je měřený nějakou účelovou funkcí. Na jednotlivé části kódu je pohlíženo jako na komponenty vhodné pro genetické operace. Komponenty programu často bývají uspořádány jako listy do stromové struktury. Uzly v této stromové struktuře hrají roli operací mezi komponenty programu.

První pokusy sestrojení programu, který je schopný sám sebe rekurzivně vylepšovat se připisují britskému matematikovi Alanu Turingovi [131]. Další experimentální pokusy, kdy program pomocí genetických algoritmů měnil svoje vlastní genetické algoritmy pro sebe-modifikaci, byly provedeny v diplomové práci [132]. V této práci autor tvrdí, že skutečný meta-algoritmus je počítačový program, který se dokáže sám zlepšovat a také dokáže rekurzivně a neomezeně vylepšovat svoji metodu vlastního zlepšování. Tyto snahy byly inspirovány matematickou teorií popisující Gödelův stroj, který byl navržen v studiích [133], [134] stejného autora (Schmidhuber), který se na metaučení snaží dívat z velmi teoreticky obecného pohledu. Obecný pohled ale naráží v praxi na spoustu omezení.

Genetické programování přirozeně favorizuje programovací jazyky, které zahrnují stromové datové struktury (např. programovací jazyk Lisp). Genetické programování se používá v aplikacích zahrnujících

[19] Hebbovo pravidlo vychází z knihy [128]. Konstatuje, že pokud jsou dva neurony současně aktivní, měli by mít vyšší stupeň vzájemné interakce než neurony, jejichž aktivita korelaci nevykazuje. V praxi to znamená, že synapse mezi neurony je posílena při souhlasné pozitivní aktivaci neuronů na předložený vstup a oslabena při nesouhlasné aktivaci sousedních neuronů na předložený vstup.

automatické programovací nástroje [135], které jsou schopny vytvářet další části programu. Cíl tohoto generativního přístupu je vylepšit programátorovi produktivitu práce. Nástroje na sebe-modifikaci programu se používají jen zřídkakdy, protože ve většině softwarových aplikací funkčnost programu závisí na každé řádce kódu a komponenty programu na sebe navazují tak, že jakákoliv změna kódu s vysokou pravděpodobností vyvolá chybu.

2.5.8 Umělá inteligence

Aplikace metaučení mají obrovské využití v dnešní technologicky založené době. V posledních letech na téma metaučení vychází čím dál větší množství vědeckých článků. Hlavně v kontextu automatických systémů strojového učení, návrhů architektur neuronových sítí a učení z velmi malého množství dat. Nejcitovanější články o metaučení pochází zejména od předních světových výzkumných laboratoří, které zkoumají umělou inteligenci. Těmito laboratořemi jsou například Londýnská DeepMind[20], kalifornská OpenAI[21], Lugánská IDSIA[22] i česká GoodAI[23] založená Markem Rosou. Někteří nadšenci vidí v metaučení jednu z komponent takzvané obecné umělé inteligence [47], jejíž definice není zcela jasná. Jejím cílem je co nejvíce napodobit a možná i předčit lidský mozek. Jiná vědecká skupina je k vývoji obecné umělé inteligence více skeptická a argumentuje tím, že lidský mozek je ještě daleko složitější než ty nejsofistikovanější algoritmy hlubokého učení. Je ale zřejmé, že přístup metaučení vede k efektivnějším a všestrannějším metodám.

Jednou z otevřených otázek zůstává, jak sestrojit program, který se bude schopen graduálně učit [136] jeden úkol za druhým a nezapomínat předchozí naučené dovednosti, které využije při řešení nových problémů. Tohle téma lze také nalézt pod pojmem celoživotní učení [137] a byly navrženy teoretické strategie, jak ho dosáhnout [138]. Jednou z myšlenek, jak toho docílit, je vytvořit tzv. kolektivní paměť [139], ze které by jednotliví agenti mohli čerpat informace. I když touto pamětí může být internet, tak praktická aplikace celoživotního učení je velice těžká.

Umělá inteligence se dostala do povědomí široké veřejnosti díky několika velmi známým projektům. Byly to například systémy, které dokázaly porazit člověka v poměrně náročných deskových hrách jako TD-Gammon [140], Go [141] nebo počítačové hře Starcraft [142]. Tyto systémy používají hluboké neuronové sítě a několik sofistikovaných heuristických technik. Jednou z nejdůležitějších technik je trénování hlubokých neuronových sítí tak, že hrají proti sobě a soupeří. Úspěšnější neuronové sítě pak mohou být vybírány pomocí evolučních algoritmů. Dalším zmedializovaným úspěchem umělé inteligence byl systém Watson [143] od společnosti IBM, který byl připojen na internet, odkud získával informace a pomocí zpracování přirozeného jazyka [144] dokázal hrát vědomostní soutěž zvanou Jeopardy a porazit člověka.

Užití umělé inteligence a automatizace zasáhlo do většiny oblastí lidské činnosti jako zdravotnictví, doprava, komunikace, vzdělání a další. Častokrát tento zásah měl pozitivní důsledky. Existují ale i situace, kdy je těchto technologií zneužíváno. Příklady mohou být třeba využití strojového učení k masivnímu ukládání veškerých osobních dat lidí používajících mainstremové operační systémy, aplikace a prohlížeče internetu. Zvýšené bezpečí je vykopeno ztrátou soukromí. Dalším negativní důsledkem technologií může být úpadek mezilidského kontaktu vykoupený rychlou elektronickou komunikací a zábavou na sociálních sítích a jiných internetových portálech. Nadnárodní korporace používají technik strojového učení k tomu, aby jejich sociální sítě byly co nejvíce návykové a lidé na nich trávili co nejvíce času. To se jim velmi úspěšně daří, protože jen během pár let se průměrný čas mladého američana strávený na

[20] https://deepmind.com/
[21] https://openai.com/
[22] http://www.idsia.ch/
[23] https://www.goodai.com/

sociálních sítích dostal z nuly na 3 hodiny denně, což podle našeho názoru je velmi výrazná společenská změna, která významě ovlivní myšlení mladé generace. Další logickou aplikací umělé inteligence je inovace vojenské techniky. Ta je nezbytná, aby mezi sebou státy držely krok. Někteří přední političtí představitelé si pomalu začínají uvědomovat potenciál a zároveň i hrozbu umělé inteligence a začínají do jejího výzkumu investovat stále více peněz, a také ji aplikovat například ve vojenském průmyslu [145]. Bitevní pole v posledních desetiletích prodělalo jednu z největších změn v dějinách lidstva díky novým technologiím, jako jsou například na dálku řízené autonomní drony. Dovolujeme si v kontextu této sekce a metaučení požít následující citát.

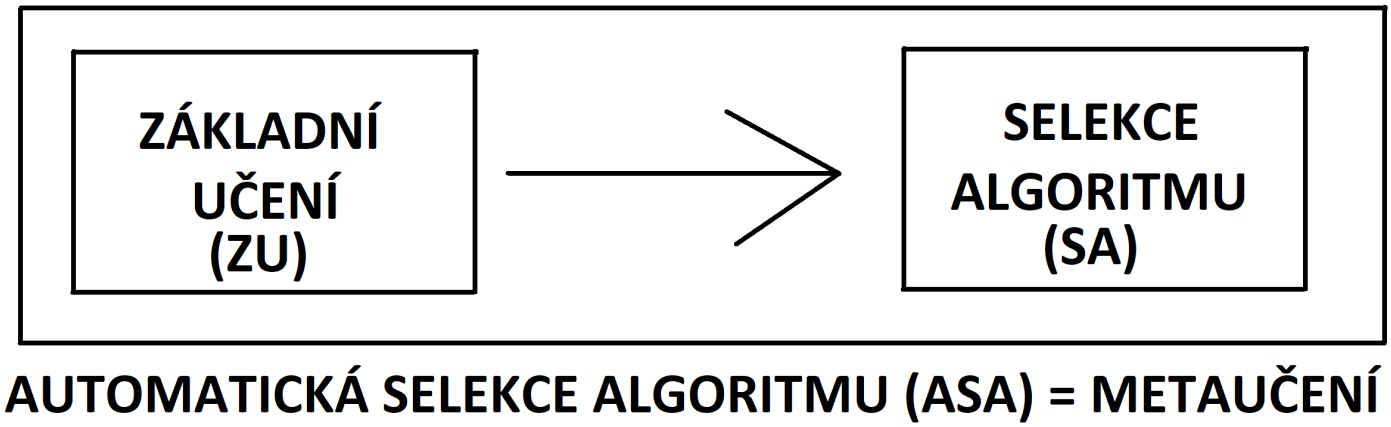
Jedna z nejdůležitějších schopností, kterou by měl člověk disponovat do rychle se měnícího světa technologií je schopnost učit se učit. Yuval Noah Harari

V této kapitole bylo naší snahou roztřídit pojem metaučení do jednotlivých kategorií tak, jak bylo v našich silách. Mnozí autoři tvrdí, že úplně přesná terminologie a kategorizace metaučení není možná, protože jednotlivé výše uvedené přístupy metaučení se často prolínají. Dále jsme se pokusili ukázat široké spektrum aplikací metaučení. I když terminologie metaučení je často obtížná, základní myšlenka zůstává pořád stejná. Metaučení je systém, který obsahuje dva podsystémy, kde nadsystém optimalizuje podsystém. To je z praxe většinou intuitivně jasné. V další kapitole se podrobněji podíváme na jednu konkrétní oblast metaučení, a to automatickou selekci algoritmu.

Kapitola 3

Automatická selekce algoritmu

Automatická selekce algoritmu (ASA) je obor, který byl prvně studován v práci [146]. ASA je obecný postup pro nalezení nejvhodnějšího algoritmu pro konkrétní problém. Algoritmy se myslí jakékoli metody, schopné plnit daný problém a v obecném pojetí to může znamenat i deterministické algoritmy nad rámec strojového učení. My se ale omezíme jen na problémy strojového učení a pod zkratkou ASA budeme rozumět úlohu automatické selekce algoritmu strojového učení. Ve strojovém učení se tomuto pojmu často říká metaučení, protože princip spočívá v učení na dvou úrovních. Pokud ve zbytku práce použijeme slovo metaučení, budeme myslet obor ASA. Dále se ve zbytku práce omezujeme pouze na učení s učitelem. To znamená, že se pomocí meta-metod učení s učitelem snažíme nalézt nejlepší základní metody z kategorie učení s učitelem. V této kapitole čtenář zjistí, proč je ASA důležitá, jak funguje a kde se používá.



Obrázek 3.1: Schéma metaučení v selekci algoritmu.

3.1 Motivace

Ve strojovém učení můžeme nalézt mnoho metod pro řešení dané úlohy. Hůře se ale naleznou nějaké návody, které by daly vodítko, jakou metodu pro danou úlohu zvolit. Lidé se v praxi většinou omezují na pár nejpoužívanějších metod. Doufají, že potřebné předpoklady pro použití daných metod budou splněny, což nemusí vždy platit. Můžeme si ale klást otázku, zda by se výběr vhodné metody nedal

zautomatizovat. Tedy, zda by se dala sestrojit nová meta-metoda, která bude automaticky doporučovat nejvhodnější metodu základní úrovně. Cílem metaučení je doporučit co nejlepší metodu základního učení pro jakýkoli učící problém.[1]

Pomocí metodologie ASA je možné doporučit nejen jediný nejlepší algoritmus, ale i množinu slibných algoritmů, či předpovědět úspěšnost algoritmu pro konkrétní problém. ASA není tak populárním oborem jako ostatní výše uvedené oblasti spadající pod metaučení. Proto na tohle téma nebylo zatím napsáno moc teorie. Velmi pěkné shrnutí této oblasti můžeme nalézt v [45, 147].

Obecný princip ASA spočívá v tom, že na základě nějakých charakteristik jistých instancí, hledá ASA optimální algoritmus pro danou instanci, kde optimalita se měří pevně danou evaluační metrikou. ASA může doporučovat algoritmy pro řešení logických úloh, softwarového návrhu, numerické a kombinaristické optimalizace [147], diferenciální rovnic, třídění [148], strojového učení ap.[2] V obecném případě pod pojmem instance můžeme chápat jednotlivé úkoly. Pro tento problém je cílem nalézt vhodné algoritmy na základě charakteristik úkolů. V našem případě jsou instance ekvivalentní datovým souborům.[3] Datové soubory budeme nazývat meta-pozorování.

3.2 Princip

Základní princip ASA spočívá v tom, že se pomocí učení s učitelem (metaučení s učitelem) snažíme naučit funkci zobrazující z charakteristik dat (meta-příznaky) do úspěšnosti algoritmů (meta-odezva). Učení s učitelem tedy probíhá na dvou úrovních, proto se pro ASA používá pojem metaučení. Cílem první úrovně (základní učení) je poskytnout meta-příznaky a meta-odezvu pro druhou úroveň (selekci algoritmu), ve které se meta-algoritmus učí z těchto meta-dat (viz obrázek 3.1).

Základní učení obsahuje konečnou množinu základních algoritmů, množinu datových souborů a jisté základní evaluační metriky. Meta-příznaky se získají charakterizací datových souborů (viz subsekce 3.5.1) a meta-odezva se získá jako výstup učení základních algoritmů na datových souborech (viz subsekce 3.4.3).

V meta-úrovni se učí meta-algoritmus z meta-trénovací množiny, což jsou meta-pozorování (datové soubory), kde vstupní nezávislé proměnné jsou meta-příznaky a odezvou je meta-odezva. Úspěšnost selekce algoritmu neboli schopnost predikce algoritmu pro nový datový soubor, se měří pomocí meta-evaluační metriky. Pro generalizaci učení, jak na základní, tak na meta-úrovni, se používá technika CV. V literatuře se pro celý proces ASA používá pojem metaučení. Často ale autoři používají pojem metaučení také pro její meta-úroveň, ve které se predikuje nejvhodnější algoritmus pro nová data. Tuto meta-úroveň ASA nazveme v této práci selekce algoritmu (SA) (viz obrázek 3.1).

3.3 Aplikace

ASA nalézá uplatnění všude tam, kde je zapotřebí vybrat konkrétní algoritmus pro nový problém. To se může hodit zejména tehdy, pokud nemáme apriorní znalost problému a nevíme, ze které třídy algoritmů máme čerpat. Nebo můžeme mít apriorní znalost problému, ale třída algoritmů je příliš velká na to, abychom je všechny testovali ručně.

[1] Problémem, úkolem, či úlohou budeme v této kapitole rozumět problém učení s učitelem. To znamená predikovat poslední závislou proměnnou na základě předchozích nezávislých proměnných pomocí metody strojového učení.

[2] Automatická selekce algoritmu je nejvíce prozkoumána v kontextu strojového učení, protože v ostatních aplikacích (návrh softwaru, diferenciálních rovnice atd.) je velmi těžké charakterizovat daný problém.

[3] Pod pojmem datové soubory rozumíme soubory vhodné pro učení s učitelem. Tedy soubory obsahující několik nezávislých proměnných a jednu odezvu. Odezva může být buď spojitá (regrese), nebo diskrétní (klasifikace)

Asi nejvíce rozšířenou aplikací ASA je její použití v AutoML systémech (viz 2.5.3). ASA používá většina state-of-the-art AutoML systémů na začátku pipeline, kdy se systém snaží rozhodnout, jakou množinu algoritmů má dále prozkoumávat a popřípadě kombinovat.

Dalšími aplikacemi ASA jsou doporučení algoritmů kolaborativního filtrování pro doporučovací systémy [149], konstrukce lepších algoritmů dolování dat [150] na základě meta-informací pro marketingová rozhodnutí. Může se jednat o algoritmy klasifikačních pravidel, asociativních pravidel nebo rozhodovacích stromů [151].

Separátní oblastí, která byla podrobněji zkoumána v kontextu ASA jsou časové řady [152]. Časové řady jsou klastrovány do shluků v prostoru jejich příznaků [153], [154]. Což je analogie ke klastrování datových souborů na základě charakteristik do shluků, čímž se zabýváme v experimentální části této diplomové práce. Pro časové řady mohou být použity příznaky jako délka časové řady, standardní odchylka, autokorelace nebo trendy v časové řadě [153], [154]. Bylo provedeno několik studií [155]–[157] zabývajících se charakterizací datových souborů a hledání nejvhodnějších příznaků. Ovšem další výzkum, který zhodnotí klady a zápory jednotlivých postupů je v této oblasti ještě zapotřebí, protože právě tohle je klíčová oblast pro úspěšnou ASA.

Historie

Od nového tisíciletí bylo vyvinuto několik automatických systémů pro selekci algoritmu. Tyto systémy jsou předchůdci výše zmíněných AutoML systémů a říkalo se jim virtuální asistenti. Dřívější systémy automatické selekce algoritmu nepoužívaly techniky optimalizace hyperparametrů, genetické algoritmy, či kombinaci modelů. Obsahovaly ale celou pipeline tak, že vstupem těchto systémů byl datový soubor a po zpracování v systému, který používal techniku ASA, bylo výstupem doporučení nejvhodnějšího algoritmu pro daná data. Nejčastěji tohle doporučení bylo ve formě seřazené množiny algoritmů.

Za zmínku stojí systém s názvem MiningMart [158], který se výhradně specializoval na fázi předzpracování dat. Další vyvinutý systém pro automatické doporučení algoritmu byl webový systém DMA (*Data Mining Advisor*) [159], který používal 10 základních modelů a umožňoval uživateli nastavit proměnnou, která vyjadřovala kompromis mezi rychlostí algoritmu a jeho přesností. Poslední z mnoha starších systémů vyvinutých pro ASA uvádíme systém s názvem IDA (*Intelligent Discovery Assistant*). Tento systém byl nejpokročilejší mezi výše zmíněnými systémy a mnoha dalšími staršími systémy pro ASA a zahrnoval předzpracování dat, výběr algoritmu a následné post-zpracování dat.

Konstrukce systému

Cílem této sekce je teoreticky nastínit problematiku konstrukce systému pro automatickou selekci algoritmu. Vstupem systému ASA je datový soubor s evaluační metrikou a výstupem systému je predikce algoritmu, který minimalizuje evaluační metriku na datovém souboru. Systém predikuje potenciálně slibný algoritmus pro nová data, aniž by se daný algoritmus trénoval. Obecně může systém ASA predikovat i množinu slibných algoritmů. Dokonce pokud je vstupem systému datový soubor, algoritmus a evaluační metrika, může systém predikovat ztrátu algoritmu na datovém souboru při evaluační metrice.

K tomu, aby bylo možné sestrojit systém ASA, je potřeba, aby na sebe jednotlivé části systému spojitě navazovaly. Části systému, neboli komponenty, jsou specifikovány například v [147]. Tyto komponenty jsou zahrnuty jak v základní úrovni (základní učení), tak v meta-úrovní (selekce algoritmu). Schéma celého systému je zobrazeno na obrázku 4.1. Komponenty mají ovšem svoje specifika správného nastavení a použití. Jsou to neměnné konstanty, které návrhář systému ASA předem stanoví tak, aby systém pro koncového uživatele fungoval dobře. Kvůli jejich pevnému nastavení můžeme také mluvit o hyperparametrech systému ASA. Tyto hyperparametry (vnitřní konstanty systému) odpovídají na ná-

sledující otázky. Z jaké množiny algoritmů ASA predikuje vhodný algoritmus? Pomocí jakého algoritmu predikuje ASA vhodný algoritmus? Pomocí jakých charakteristik systém predikuje vhodný algoritmus? Jak je predikce dosaženo? Jak je dosaženo generalizace systému ASA?

Je zřejmé, že ze systému se stává komplexnější černá skříňka, která má spoustu vnitřních hyperparametrů[4], které musí nastavit návrhář a uživateli, po kterém se požaduje předhodit systému datový soubor, tím ušetří velké množství práce při hledání nejvhodnějšího algoritmu.

Pojďme se v dalších sekcí podívat do černé skříňky systému ASA. Čtenář zjistí, jakým všem úskalím a problémům musí návrhář systému ASA čelit.

3.4 Základní učení

Cílem první fáze systému ASA, zvané základní učení (ZU), je vyprodukovat meta-příznaky a meta-odezvu pro druhou fázi ASA, zvanou selekce algoritmu (SA viz sekce 3.5). Meta-příznaky se získají vhodnou charakterizací datových souborů (viz subsekce 3.5.1). Charakterizace datového souboru je problém, který v oblasti ASA vyvolal největší zájem, protože úspěšnost celého systému závisí zejména na ní.

Meta-odezva je dána maticí ztrát[5] základních algoritmů pro množinu datových souborů (meta-pozorování). Jednotlivé ztráty v matici ztrát jsou získány pomocí nějaké varianty k-násobné-CV. Matici ztrát můžeme nechat v její původní podobě, transformovat ji na matici pořadí[6], nebo ji ztransformovat na vektor vítězů[7]. Meta-odezva tedy může obsahovat ztrátu všech základních algoritmů, pořadí základních algoritmů podle úspěšnosti, nebo nejlepší základní algoritmus pro každé meta-pozorování (datový soubor).

Základní učení se skládá ze tří základních komponent. Jsou to množina učících problémů (datových souborů), množina základních algoritmů a základní evaluační metrika.

3.4.1 Základní algoritmy

Cílem základních algoritmů je vyprodukovat meta-odezvu pro meta-algoritmus v meta-úrovni. Základní algoritmy jsou v křížové validaci trénovány na datových souborech. Ztráty jednotlivých algoritmů jsou vstupem do meta-odezvy. Ty algoritmy, které jsou úspěšnější v minimalizaci ztrátové funkce (evaluační metriky), favorizujeme do výsledné meta-odezvy. Konkrétní popis, jakým způsobem je meta-odezva vyprodukována lze nalézt v subsekci 3.5.2.

Hlavním cílem systému pro ASA je doporučit algoritmus základní úrovně pro učící problémy základní úrovně. Proto je na místě pečlivá volba množiny základních algoritmů. Pokud budeme v základní úrovní řešit regresní problém[8], tak je logické, že se množina základních algoritmů bude skládat jen z regresních algoritmů. Pokud budeme řešit klasifikační problémy, množina základních algoritmů se bude skládat jen z klasifikačních algoritmů.

[4] Znovu se zde uplatňuje princip nekonečného regresu, kdy jsme ve snaze odstranit hyperparametr typu a nastavení algoritmu pro konkrétní problém vnesli do systému několik dalších hyperparametrů. Těmito hyperparametry mohou být například množina a nastavení základních algoritmů, charakterizace datových souborů, meta-algoritmus, meta-evaluační metrika atd.

[5] Maticí ztrát budeme rozumět kvantifikovanou generalizovanou predikci všech základních algoritmů aplikovanou na všechna pozorování (datové soubory). Rozměry matice ztrát jsou dány počtem základních algoritmů (sloupce) a počtem datových souborů (řádky). Jedno políčko této matice znamená ztrátu příslušného algoritmu na příslušném datovém souboru. Ztrátou, úspěšností neboli výkonností algoritmu budeme chápat kvantifikovanou míru predikce algoritmu na datovém souboru. Ztráta je jedno číslo vyjadřující predikční schopnost algoritmu. Více v subsekci 3.4.3.

[6] Příslušnou ztrátu nahradíme pořadím algoritmu, které vyjadřuje, kolikátý nejlepší je daný algoritmus na daném datovém souboru.

[7] Vybereme nejlepší algoritmus pro každý datový soubor.

[8] Regresním problémem chápeme učení s učitelem na datovém souboru, u kterého je cílem predikovat poslední spojitou proměnnou. U klasifikačního problému je cílem predikovat poslední diskrétní neboli kategorickou proměnnou.

Je třeba uvést, že je rozdíl mezi základními algoritmy ze stejné třídy, ale jinými hyperparametry. Například algoritmus k-NN pro $k = 5$ považujeme za jiný algoritmus než pro $k = 10$. Touto úvahou se dostáváme k tomu, že množina možných algoritmů je nekonečná a nekonečná je i množina možných hyperparametrů pro jednotlivé algoritmy.[9] Proto je vhodné do základní množiny algoritmů volit různé algoritmy, které mají různou doménovou expertízu.[10] Různé vlastnosti algoritmů základního učení jsou prostudovány v práci [162]. V praxi je často postačující volba různé množiny běžně používaných algoritmů se standardním nastavením hyperparametrů.[11]

3.4.2 Datové soubory

Volba datových souborů hraje klíčovou roli v ASA, neboť z nich získáme meta-příznaky díky jejich správné charakterizaci a meta-odezvu díky základním algoritmům. Tyhle meta-data jsou nezbytná pro meta-úroveň ASA. Výsledky ASA pak mohou být velmi závislé na výběru datových souborů. Systém by měl vždy používat reálná data, protože každá náhodně generovaná data jsou určitým způsobem příliš specifická a mají jisté asymptotické vychýlení. Nicméně jsou navrženy postupy, jak si datové soubory nagenerovat uměle [163], [164].

Teoretické práce výzkumu vhodnosti a škálovatelnosti datových souborů pro jisté základní algoritmy můžeme nalézt v práci [162]. Je doporučeno použít větší počet datových souborů pro důkladnější studii metaučení, zejména pokud volíme více příznaků. Při menším počtu datových souborů může dojít k přeučení modelů kvůli prokletí dimensionality. Sběr velkého množství datových souboru ale v praxi není vždy jednoduchý, neboť ruční stahování mnoha souborů z internetových stránek je vhodné pouze pro dlouhé zimní večery.

Datové soubory mohou být získány z některých datových úložišť. Jedním z nejznámějších datových úložišť pro strojové učení je UCI [165] v současnosti obsahující 497 datových souborů. Další možností je stáhnout datové soubory ze stránky Kaggle.[12] Pokud uživatel neovládá techniky webscrapingu[13] [166], tak je odkázán na ruční pomalé stahování z těchto datových úložišť. Stovky datových souborů lze nalézt také na stránce pro sdílení počítačových kódů s názvem GitHub.[14] Pokud návrhář systému ASA hledá datové soubory ze specifické oblasti (finance, astronomie, medicína), doporučujeme použít internetový vyhledávač. Dnes velmi populární sdílenou meta-databází je OpenML[15], která obsahuje přes 21 tisíc datových souborů pro učení s učitelem. Proto bychom doporučili investovat určité množství času do učení se s touto meta-databází a datové soubory stáhnout z ní.

Jednou z potenciálních možností pro sběr datových souborů je využít tzv. extrémní dolování znalostí [167]. Velká databáze je segmentována na menší části, které můžeme po vhodných úpravách považovat za nové datové soubory. Další možností je využít datových toků[16] k segmentaci datového toku v různých časových úsecích, a tím vytvořit nové datové soubory.

[9] Někteří autoři se rozhodli provést studie zkoumající obrovskou množinu hyperparametrů příslušných k algoritmům základního učení [160], [161].

[10] Různá doménová expertíza znamená, že některé algoritmy jsou lepší na některé typy problémů než jiné algoritmy. Na jiné typy problémů jsou zase horší.

[11] Standardní nastavení hyperparametrů je takové nastavení hyperparametrů, které se v praxi ukázalo jako velmi uspokojivé pro běžnou třídu problémů. Algoritmy strojového učení implementované v softwarových balíčcích už typicky defaultně obsahují tohle standardní nastavení.

[12] https://www.kaggle.com/datasets

[13] Technika webscrapingu je programovací technika, pomocí které lze stáhnout díky automatickému programu přesně definované informace z webové stránky do počítače. V našem případě by šlo o datové soubory.

[14] https://github.com/

[15] https://www.openml.org/home

[16] Datový tok je kontinuální transfer dat generovaný digitálními technologiemi nebo senzory.

Pokud návrhář ASA nenalezne, nebo z nějakého důvodu nechce použít reálné datové soubory pro ASA, může se v literatuře inspirovat například tím, jak ztransformovat již existující datový soubor a vytvořit tím novou dostatečně velkou množinu datových souborů [163]. Tyto transformace zahrnují přidání nových nezávislých proměnných se specifickými hodnotami šikmosti nebo špičatosti [164], nebo přidání kontaminace ke jakýmkoli proměnným [168], [169]. Upozorňujeme, že vytváření umělých datových souborů by se měl návrhář metaučení za každou cenu vyhnout.

3.4.3 Predikční míra

K tomu, abychom byli schopni kvantitativně určit úspěšnost základního algoritmu na datovém souboru, potřebuje nějakou evaluační metriku, podle které budeme úspěšnost měřit. Pro terminologické rozlišení budeme evaluační metriku ve fázi ZU nazývat základní predikční míra a ve fázi SA meta-predikční míra. Predikční míra je úzce spojena se ztrátovou funkcí. Ztrátovou funkci chápeme jako obecný předpis, který udává, jakým způsobem se měří chyba. Predikční míra je kvantifikovaná hodnota úspěšnosti natrénovaného algoritmu na konkrétních datech, která je měřená ztrátovou funkcí. V průběhu trénování algoritmu se ztrátovou funkci snažíme iterativně minimalizovat a na konci trénování ohodnotíme kvalitu učícího algoritmu pomocí predikční míry.

Pomocí predikční míry jsme schopni kvantifikovat generalizující schopnost algoritmu učení. Toho se docílí sečtením individuálních chyb na testovacích datech. Pomocí k-násobné-CV se dosáhne toho, že testovací data jsou celý datový soubor a můžeme tedy kvantifikovat míru, jak dobře algoritmus predikuje data v celém datovém souboru. V praxi se používá několik druhů predikčních měr. Podle typu predikční míry jsou penalizovány chyby v predikci specifickým způsobem.

Ztrátové funkce můžeme dělit na dvě kategorie (pro regresi a pro klasifikaci). Při regresních úlohách ztrátová funkce typicky penalizuje predikci v závislosti na velikosti chyby od skutečné hodnoty. Podle typu úlohy je možné volit několik druhů ztrátových funkcí (kvadratická, absolutní, nebo Huberova ztrátová funkce) a k nim příslušné reziduální míry predikce (střední kvadratická chyba, absolutní chyba, Huberova chyba). Popis vybraných měr predikce pro experimentální část nalezne čtenář v další kapitole.

Pro klasifikační úlohy lze jako ztrátovou funkci použít křížovou entropii, Kullback-Leiblerovu divergenci a diskrétní $0-1$ ztrátovou funkci. K ohodnocení výkonnosti klasifikátorů se velmi často používají predikční míry jako přesnost, preciznost, úplnost, specificita, F-míra a plocha oblasti pod operační křivkou klasifikátoru zkracována AUC [170]. Pro důkladnější studium těchto měr predikce klasifikátorů doporučujeme prostudovat literaturu [171].

Hodnota predikční míry typicky hodnotí algoritmy učení s učitelem jen podle jejich predikční schopnosti a nezahrnují do ní například čas výpočtu. V takovém případě vybírá systém ASA jako nejúspěšnější algoritmus ten, jehož predikční schopnost může být jen o málo lepší než ostatní, ale čas jeho trénování může být řádově delší. Proto práce [172]–[174] kombinují obě kritéria do jednoho. Ve výsledném kritériu nastaví koncový uživatel podle potřeby parametr relativní důležitosti mezi úspěšností algoritmu a časem trénování algoritmu. Pro klasifikační algoritmy může například tento parametr udávat přesnost klasifikátoru, kterou je uživatel ochoten obětovat za desetinásobné urychlení ve fázi trénování klasifikátoru [175]. V případě zahrnutí vícekriteriální míry do systému ASA by mělo být pro koncového uživatele co nejjasnější, jakým způsobem se rozhoduje mezi kompromisem u více kritérií. Pokud výstupem systému ASA je seřazená množina algoritmů, tak správnou volbou parametru pro kompromis mezi výkonností a přesností se dosáhne toho, že rychlejší algoritmy budou na předních a pomalejší na koncových pozicích. Rychlejší modely pak mohou být provedeny dříve, pomalejší později [176].

3.5 Selekce algoritmu

Fází selekce algoritmu (SA) se v ASA rozumí meta-úroveň, ve které se meta-algoritmy učí z meta-příznaků predikovat meta-odezvu. Generalizací je možné dosáhnout toho, že pro nová meta-pozorování (datový soubor) je meta-algoritmus schopen predikce nejvhodnějšího algoritmu pro tento datový soubor. Pojďme se podívat na záležitosti, které návrháře systému ASA mohou potkat při konstrukci jednotlivých komponent ve fázi SA.

3.5.1 Meta-příznaky

V metaučení hraje volba meta-příznaků klíčovou roli, která si zasloužila největší pozornosti. Pokud totiž nejsou meta-příznaky voleny správně, meta-algoritmy nemohou zachytit korelace v datech a systém ASA nemusí fungovat. Meta-příznaky budeme chápat jako určité charakteristiky datových souborů. Pro úspěšnou SA by měly meta-příznaky co nejlépe diskriminovat neboli rozlišovat mezi jednotlivými soubory [177]. Soubory (meta-pozorování) si lze představit jako jednotlivá pozorování v prostoru meta-příznaků, a pokud meta-příznaky dobře diskriminují meta-pozorování, tak meta-pozorování tvoří v prostoru meta-příznaků několik separátních klastrů. Rozlišení klastrů se uskuteční pomocí klasifikačního meta-algoritmu.

Volba meta-příznaků velmi závisí na množině základních algoritmů [169]. Výpočetní náročnost meta-příznaků by neměla být příliš velká. Podle článku [178] by neměla přesahovat $O(n \log n)$. Počet meta-příznaků by také neměl být příliš velký, protože by poté mohl nastat problém prokletí dimensionality, kdy jsou meta-pozorování velmi řídce zastoupena ve zbytečně velkém prostoru meta-příznaků. Tento jev může nastat obzvlášť, pokud systém ASA pracuje s malým množstvím datových souborů a v prostoru meta-příznaků je tedy málo metapozorování. Tento problém může být vyřešen zredukování prostoru meta-příznaků a selekcí meta-příznaků s největší variabilitou [179]. Díky vhodné selekci meta-příznaků ve fázi SA se zachovají jen ty nejvíce diskriminující meta-příznaky (nejvíce korelované s odezvou). K selekci příznaků ve fázi SA se nejvíce hodí tzv. wrapper metody [180].[17]

3.5.1.1 Dělení meta-příznaků

Existují určitá obecná doporučení pro dobrou charakterizaci datového souboru. Tyto doporučení zahrnují velikost datového souboru, poměr signálu a šumu, nebo porušení statistických předpokladů. Zatím bylo vynalezeno několik desítek způsobů, jak aplikovat tyto doporučení a nalézt vhodné meta-příznaky. Meta-příznaky jsou rozděleny do třech základních skupin.

Jednoduché, statistické a informačně teoretické meta-příznaky

Tento typ meta-příznaku obsahuje základní informace o vlastnostech datových souborů [181]. Pomocí těchto meta-příznaků se snažíme diskriminovat algoritmy, jejichž úspěšnost je ovlivněna například různou velikostí datových souborů, typů distribucí, kontaminací v datech a chybějícím hodnotám.

Meta-příznaky mohou být jednoduché vlastnosti jako počet pozorování, počet nezávislých proměnných, poměr počtu pozorování a nezávislých proměnných, počet kategorických nebo numerických pro-

[17] Wrapper metody iterativně aplikují model strojového učení na množinu příznaků. Cílem je najít takovou množinu příznaků, při kterých vykazuje model nejmenší chybu. Wrapper metody mohou používat jeden z následujících postupů. Prvním postupem je dopředné hledání, kdy se začíná s jedním příznakem a další příznaky se postupně přidávají, dokud chyba nezačne růst. Dalším postupem je zpětné hledání, kdy se začíná s celou množinou příznaků a postupně se vyřazuje ten příznak, který je nejméně významný. Speciálním případem zpětného hledání jsou filter metody, kdy se provede jen první iterace a vybere se pevný počet nejvýznamnějších příznaků. Posledním postupem je rekurzivní hledání, kdy se prohledá každá permutace příznaků. U rekurzivního hledání roste výpočetní náročnost exponenciálně s lineárním přírůstkem příznaků.

měnných, poměr mezi počtem kategorických a numerických proměnných a počet tříd v odezvě při klasifikační úloze. Podle [45] je vhodnější používat poměry než počty jednoduchých meta-příznaků. Například poměr mezi počtem kategorických a numerických proměnných vyjadřuje lépe informaci o tom, zda je soubor spíše regresního, nebo klasifikačního typu, než jejich absolutní počty. Dalším příkladem je poměr mezi počtem pozorování a počtem proměnných, který lépe vyjádří efekt prokletí dimensionality než absolutní počty pozorování a proměnných.

Dalšími meta-příznaky mohou být statistické veličiny jako šikmost nebo špičatost jedné vybrané numerické proměnné, průměr šikmostí nebo špičatostí ze všech numerických proměnných, korelace dvou vybraných numerických proměnných (může být i s odezvou), průměrná korelace mezi všemi numerickými proměnnými a numerickou odezvou a poměr mezi směrodatnou odchylkou a průměrem odezvy pro regresi [182].

Další meta-příznaky mohou pocházet z oblasti teorie informace. Mohou to být například entropie jakýchkoli vybraných kategorických proměnných (i odezvy). Jako meta-příznak je možné použít vzájemnou informaci mezi jakýmikoli dvěma kategorickými proměnnými nebo průměrnou vzájemnou informaci mezi všemi nezávislými kategorickými proměnnými a kategorickou odezvou pro klasifikaci [183]. Je vhodnější použít souhrnné meta-příznaky využívající průměrování jednotlivých meta-příznaků pro charakterizaci datového souboru jako celku než meta-příznaky vyjadřující závislosti mezi dvěma vybranými proměnnými.

Landmarkers

Jednou z kreativních možností výběru příznaku je použití tzv. landmarkerů [184], což jsou predikce pro datové soubory velmi jednoduchých a rychlých algoritmů. Vektor složený z těchto predikcí uvažujeme jako nový meta-příznak.

Myšlenka landmarkerů je založena na předpokladu, že různé algoritmy učení mají pro různé třídy problémů různé odbornosti[18] ve kterých jsou lepší než ostatní algoritmy. A předpokládáme oblasti odbornosti časově náročných algoritmů se shodují s oblastmi odbornosti rychlých algoritmů. Tedy pokud rychlý jednoduchý algoritmus A porazí na dané úloze rychlý jednoduchý algoritmus B, tak očekáváme, že bude existovat komplexní pomalejší algoritmus X, který na stejné úloze porazí komplexní pomalejší algoritmus Y [178].

Rychlými algoritmy mohou být například rozhodovací pařezy.[19] Další studie, která se zabývá vhodnou volbou rychlých algoritmů je [162].

Modelově-založené meta-příznaky

Další z kreativních možností je charakterizovat datový soubor pomocí komplexnosti modelu, který je na těchto datech vytvořen. Proto mluvíme o modelově založených meta-příznacích. Modelem nejčastěji bývá rozhodovací strom. Algoritmus indukce[20] rozhodovacích stromů můžeme nalézt v [26].

Jako meta-příznak poslouží některý z hyperparametrů modelu (rozhodovacího stromu). Což mohou být například počet uzlů, tvar, nevyváženost, nebo hloubka stromu [185]. Tyto meta-příznaky jsou spočítány nepřímo z modelu. Studie zabývající se modelově založenými meta-příznaky jsou [186], [187].

[18] Metody strojového učení rozdělují třídy učících problémů (datových souborů) do skupin podle úspěšnosti metody. Metody mají často nějaké předpoklady. Pokud tyto předpoklady jsou splněny, metody jsou úspěšné. Pokud předpoklady nejsou splněny, metody jsou neúspěšné. Pomocí charakteristik (v tomto případě landmarkerů) datového souboru se snažíme tyto předpoklady zachytit.

[19] Rozhodovací pařez je rozhodovací strom, který má jen jeden větvící uzel.

[20] Algoritmus indukce využívá entropie ke konstrukci stromu a přidávání jednotlivých uzlů tak, aby maximálně odlišila jednotlivé kategorie.

3.5.1.2 Meta-databáze

Pokud návrhář systému ASA nenalezne dostatečně velkou množinu datových souborů, může si základní komponenty pro systém ASA stáhnout z velmi populární veřejně sdílené meta-databáze OpenML [188].[21] Mnoho studií zkoumajících ASA využívá právě tuto meta-databázi. Autor tohoto projektu (Vanshoren) se totiž sám do značné míry zabývá oborem ASA.

Pokud bychom měli dalšímu výzkumníkovi doporučit zdroj datových souborů a meta-dat (meta-pozorování, meta-příznaky, meta-odezva) pro studii ASA, tak mu doporučíme právě OpenML. Z této meta-databáze je možné čerpat pomocí datově zaměřených programovací jazyků (Python, R i Java a C++) a stáhnout a vyfiltrovat si z ní potřebná data.

OpenML funguje na principu sdíleného strojového učení přes internet. To znamená, že kdokoli může nahrát do databáze svoje experimenty učení s učitelem. U každého pokusu je zaznamenáno, jaký úkol se plnil[22], jaká metoda strojového učení byla na daný úkol použita[23], jak dlouho trvalo trénování algoritmu, typ predikční chyby a její velikost.

Z databáze je možné si filtrovat a zkoumat výsledky úspěšností jednotlivých metod pro konkrétní datový soubor. Dále je možné zkoumat, která metoda je nejúspěšnější v průměru pro všechny datové soubory, srovnávat časovou efektivnost metod, studovat úspěšnost algoritmů přes různé metriky, porovnávat člověkem navržené algoritmy a AutoML systémy a mnohé další. Filtrováním, které je možné přes úkoly, datové soubory, algoritmy, metriky a uživatele, je možné získat zajímavá metadata.

Uživatel si kromě nahrávání výsledků může data i modely stáhnout na vlastní počítač. To také znamená, že si může stáhnout binární soubory už naučených nejlepších modelů pro konkrétní úkol.

Tato meta-databáze obsahuje přes 21 tisíc datových souborů a 16 tisíc algoritmů, které jsou hodnoceny více než 30ti základními evaluačními metrikami. Do této meta-databáze už lidé celkem nahráli přes 10 milionů experimentů. Každý datový soubor je charakterizován pomocí 120 meta-příznaků. Správným vyfiltrováním a stažením dat z této meta-databáze je možné provést studii ASA s velkým množstvím dat.

3.5.2 Meta-odezva

Meta-odezva, získaná z fáze ZU, je proměnná, kterou je cílem predikovat ve fázi SA. Meta-odezva se může lišit podle toho, co požadujeme predikovat. Můžeme chtít predikovat například jeden nejlepší algoritmus, množinu algoritmů, seřazenou množinu algoritmů nebo ztrátu algoritmu základního učení. První tři typy jsou diskrétní proměnné, a proto se k tomu účelu používají klasifikační meta-algoritmy. Predikce ztráty je spojitá proměnná, ke které se používají regresní meta-algoritmy. V literatuře jsou návody zejména pro klasifikační přístupy. Základní typy meta-odezvy se dále pokusíme nastínit. Ve fázi ZU máme danou množinu n algoritmů $\{a_1, a_2, \ldots, a_n\}$ a m datových souborů $\{d_1, d_2, \ldots, d_m\}$.

Doporučení nejlepšího základního algoritmu

Doporučení systému ASA je v tomto případě jediný algoritmus a_j pro datový soubor d_i [178], [189]. Algoritmus a_j je predikován jako nejlepší z množiny základních algoritmů. Výstupem ze základního učení je vektor vítězných metod pro meta-pozorování (datové soubory). Tedy je vybrán algoritmus s nejmenší ztrátou pro konkrétní datový soubor. Vektor vítězných metod je tedy kategorická proměnná. Výhodou je rychlé použití jednoduchých klasifikačních meta-algoritmů. Nevýhodou je, pokud systém ASA předpoví základní algoritmus špatně, tak není k dispozici jiný alternativní algoritmus.

[21] https://www.openml.org/home

[22] Na kterém datovém souboru byla spuštěna metoda strojového učení a které jeho proměnné figurovaly jako nezávislé a která jako závislá proměnná.

[23] Metoda obsahuje i konfiguraci příslušných hyperparametrů.

Doporučení podmnožiny základních algoritmů

Z množiny základních algoritmů se uživateli doporučí jen jejich podmnožina. Nejprve se určí hodnota ztráty nejúspěšnějšího základního algoritmu. Do vybrané podmnožiny základních algoritmů se vyberou takové algoritmy, které nejsou horší více než nějaká pevně stanovená mez přičtená ke ztrátě nejúspěšnějšího základního algoritmu [190]. Mohou se také použít statistické testy, které porovnávají úspěšnost mezi základními algoritmy [189], [191]. Algoritmy, které nejsou signifikantně horší než nejlepší algoritmus jsou vybrány do podmnožiny. Výhodou této doporučovací metody je, že uživatel má na výběr mezi několika algoritmy. Nevýhodou je, že neví pořadí, v jakém je má zkusit otestovat. Tento problém atakuje další metoda.

Doporučení seřazeného pořadí základních algoritmů

Hodně publikací se zaměřuje na výběr nejlepšího algoritmu základního učení. Méně pozornosti je věnováno článkům zabývajících se doporučením pořadí algoritmů [152], [175]. Doporučení seřazené množiny slibných algoritmů je mnohem obecnější metoda než doporučení jen jednoho nejlepšího algoritmu [192]. Algoritmy základního učení seřadíme od pravděpodobně nejlepšího po pravděpodobně nejhorší. K tomu, abychom mohli získat pořadí algoritmů, se musí ve fázi ZU určit pořadí úspěšnosti základních algoritmů pro každý datový soubor. Pořadí daného algoritmu na novém datovém souboru bude dáno jako průměr pořadí tohoto algoritmu na k nejbližších datových souborech v prostoru meta-příznaků. Vzdálenost můžeme měřit například metodou k-NN.

Označme $R_{i,j}$ pořadí základního algoritmu $a_j\,(j = 1, \ldots, n)$ na datovém souboru i, kde n je počet základních algoritmů. Průměrné pořadí algoritmu a_j je dáno vztahem:

$$\bar{R}_j = \frac{\sum_{i=1}^{k} R_{i,j}}{k},$$

kde finální pořadí základních algoritmů pro nový datový soubor je dáno seřazením těchto průměrných pořadí pro každý základní algoritmus. Tato metoda se velmi často používá na začátku pipeline AutoML systémů.

Jak ale kvantitativně můžeme ohodnotit nalezené pořadí algoritmů? Například mějme dáno cílové pořadí základních algoritmů na zkoumaném datovém souboru je $(1, 2, 3, \ldots, n-1, n)$. Intuitivně je mnohem lepší predikovat pořadí $(2, 1, 3, \ldots, n-1, n)$ než $(n, n-1, \ldots, 2, 1)$ [193]. K ohodnocení podobnosti pořadí se používá tzv. Spearmanův pořadový korelační koeficient [194], [195]. Pomocí něj měříme vzdálenost mezi predikovaným a cílovým pořadím:

$$r_s = 1 - \frac{6 \sum_{i=1}^{n} \left(\hat{R}_i - R_i\right)^2}{n^3 - n},$$

kde n je počet algoritmů, $\hat{R}_i$ cílová hodnota na i-té pozici v cílovém pořadí algoritmů a R_i je predikovaná hodnota. Hodnota r_s rovna jedné znamená naprostou shodu a hodnota -1 naprostou neshodu. Pokud bychom predikované pořadí nahradili náhodnou permutací pořadí, hodnota Spearmanova pořadového korelačního koeficientu by se měla pohybovat okolo nuly. Pro konstrukci robustnějšího doporučovacího systému doporučujeme využít pořadí ke kombinaci seřazených algoritmů. Seřazené algoritmy zkombinovat se snižujícími se vahami.

Predikce ztráty základního algoritmu

Tato metoda předpoví pomocí regresního meta-algoritmu číselnou hodnotu ztráty základního algoritmu [156], [183]. Meta-odezvou je vektor úspěšností neboli ztrát algoritmu (spojitá proměnná) přes všechny

datové soubory. Protože jsou datové soubory různé a odezvy datových souborů jsou v jiných jednotkách, musí se použít přeškálovací techniky. Přeškálovací techniky založené na vzdálenosti k úspěšnosti nejlepšího algoritmu, vzdálenosti k úspěšnosti standardního algoritmu a normalizace úspěšnosti[24] jsou navrženy v [196].

Pro spojitou meta-odezvu je možné používat pouze jeden algoritmus základního učení, protože základní učení dalších základních algoritmů probíhá nezávisle na sobě. Na toto téma nebylo napsáno moc teorie nejspíše proto, že místo toho, aby systém zbytečně predikoval chybu jediného základního algoritmu, se tento algoritmus rovnou může použít. Rozdíl by byl v tom, pokud by výpočetní náročnost základního algoritmu byla mnohem větší než výpočetní náročnost meta-algoritmu. V praxi zřejmě tyto situace moc často nenastávají. Klasifikační přístupy přece jen ušetří trénování celé množiny základních algoritmů a vyberou z nich ten nejlepší, což je značné ušetření času.

3.5.3 Meta-algoritmy

Volba meta-algoritmu závisí a je omezena na typu meta-odezvy, kterou má meta-algoritmus predikovat. Pokud je meta-odezvou kategorická proměnná, volíme klasifikační meta-algoritmus. Pokud je meta-odezvou spojitá proměnná, volíme regresní meta-algoritmus. Obecná doporučení navrhují vybrat jeden ze standardních algoritmů strojového učení (viz subsekce 1.1.3).

V literatuře jsou nejčastěji zastoupeny klasifikační přístupy, kdy extrémním případem je použití stejných algoritmů na základní i na meta-úrovni (LDA, neuronové sítě, rozhodovací stromy). Pro meta-odezvu, obsahující pořadí algoritmů, je vhodné použít metodu k-NN, která měří vzdálenost v metrickém prostoru. Použití regresních meta-algoritmů je méně populární než jejich klasifikačních protějšků. I proto je množina regresních meta-algoritmů omezenější. V meta-úrovni může být pro regresi použita lineární regrese, či rozhodovací stromy.

Cílem celého systému ASA je dosáhnout generalizace. To znamená, že požadujeme, aby systém ASA byl schopný predikovat vhodný základní algoritmus pro nový datový soubor. Generalizace systému ASA se dosáhne pomocí metody CV v meta-úrovni. Pomocí k-násobné-CV se vypočte nějaká meta-predikční míra meta-algoritmu (viz subsekce 3.4.3), která kvantifikuje schopnost doporučit vhodný algoritmus pro nová data. Tato predikční míra udává kvalitu celého systému ASA.

Shrnutí

Metaučení v podobě automatické selekce algoritmu může ulehčit lidem spoustu času a práce tím, že jim poskytne dynamický systém v podobě asistenta, který jim doporučí nejvhodnější metodu strojového učení pro jejich data.

Konstrukce systému pro automatické doporučení algoritmu zahrnuje několik složitých problémů, kterým musí jeho návrhář čelit. Jednou z prvních otázek, které si návrhář musí klást je, co bude systém doporučovat. Odpověď na tuto otázku pak určuje množinu komponentů tohoto systému.

Klíčovou roli hraje sběr a charakterizace datových souborů, výběr a nastavení základních algoritmů, výběr vhodného meta-algoritmu a jeho meta-predikční chyby.

Na automatickou selekci algoritmu se nemůžeme dívat jako na zázračnou metodu strojového učení, která vyřeší jakýkoli problém. Za prvé, vybrané příznaky musejí být na tolik diskriminující, aby dokázaly rozlišit mezi jednotlivými datovými soubory. Za druhé, pokud je nový datový soubor velmi odlišný od souborů v trénovací množině systému, predikce algoritmu nebude fungovat. Komplexní systém pro automatickou selekci algoritmu se skládá z několika komponent, které na sebe spojitě navazují. Návrh

[24] Od úspěšnosti základního algoritmu se odečte průměr úspěšností všech základních algoritmů na stejném souboru a tohle číslo se vydělí standardní odchylkou.

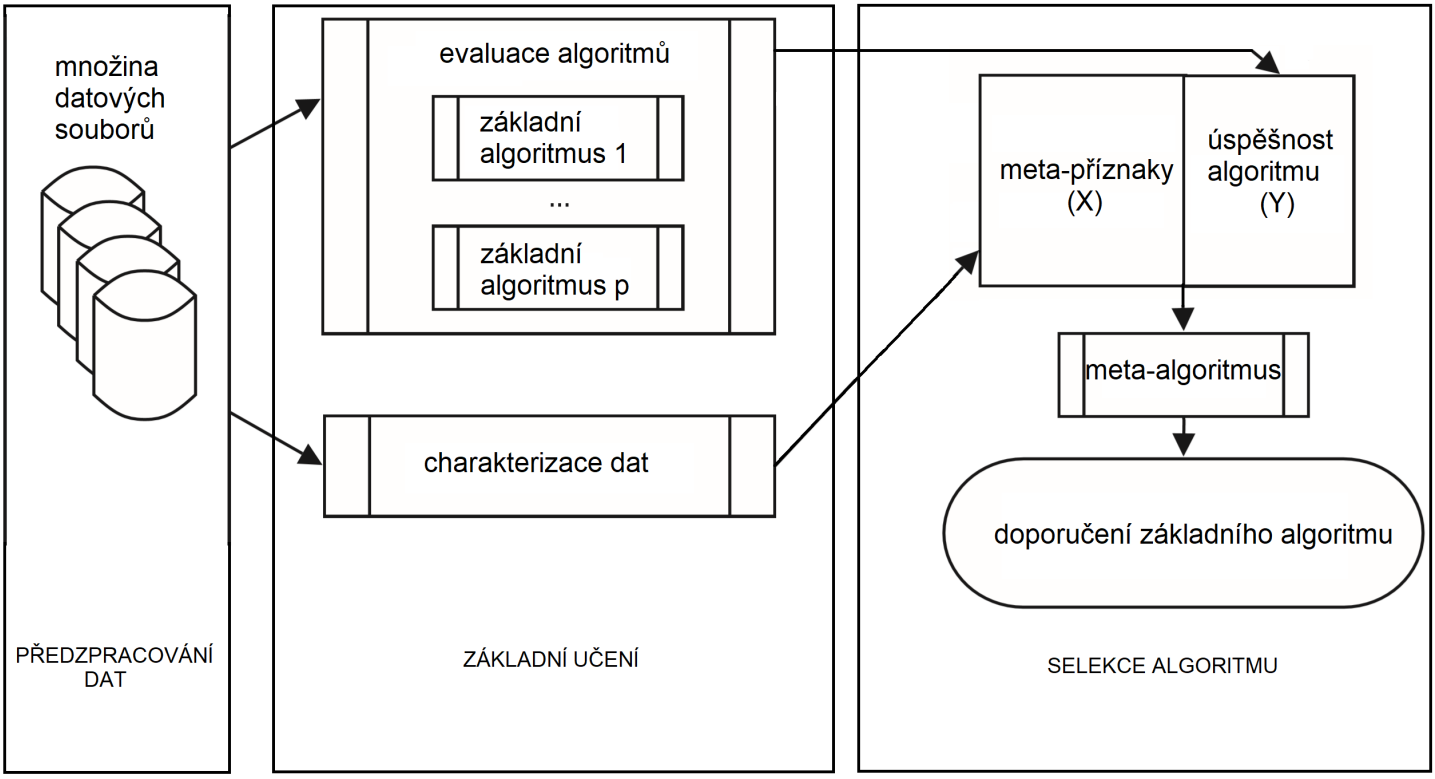
systému proto nesmí být brán na lehkou váhu, protože jedna špatně navržená komponenta systému znamená jeho nefunkčnost.

V další kapitole ukážeme, jak jsme výše uvedené komponenty systému ASA volili my a v poslední kapitole může čtenář nalézt výsledky našeho navrženého systému.

Kapitola 4

Implementace

Cílem této kapitoly je zkonstruovat vlastní systém pro automatickou selekci algoritmu (metaučení). V této kapitole budou popsány základní části systému, které jsme volili pro jeho návrh. Čtenář zjistí, jak jsme nastavili jednotlivé komponenty systému. V první řade tato kapitola uvede čtenáře do celého procesu od sběru datových souboru, jejich předzpracování, výběru základních algoritmů, charakterizaci datových souborů, výběr a nastavení meta-algoritmů a volbu predikční míry pro základní úroveň i meta-úroveň. Dále se pokusíme nastínit pracovní postup, jakým bylo konstrukce systému dosaženo. Na konci kapitoly uvedeme softwarové nástroje, pomocí kterých jsme konstruovali systém a stručný popis souborů obsahujících počítačové kódy.



Obrázek 4.1: Systém automatické selekce algoritmu.

4.1 Datové soubory

Jedním z našich cílů bylo provést studii metaučení na větším množství datových souborů. Z časových důvodů jsme nechtěli provádět stahování ani předzpracování dat ručně. Proto jsme museli ke stahování použít některé webscrapingové nástroje pro automatické stahování. Podařilo se nám stáhnout celkem přes 2000 datových souborů. Dalším automatickým předzpracováním a selekcí datových souborů jsme tento počet zredukovali na 643. Datové soubory pochází z různých oblastí lidské činnosti a většina z nich je stažena zejména z webových stránek uvedených v poznámce pod čarou.[1][2][3] Datové soubory jsou zejména formátů CSV a TXT a jsou určené pro učení s učitelem.

Datové soubory byly po fázi automatického předzpracování charakterizovány pomocí několika vybraných příznaků (viz sekce 4.4), které byly napočítány ke každému datovému souboru a předány do meta-úrovně meta-algoritmu jako vysvětlující proměnné. Číselné hodnoty příznaků pro náhodně vybrané datové soubory můžeme vidět v tabulce 4.2. Jako odezva pro meta-algoritmus posloužil vektor vítězů základních algoritmů, který byl získán po trénování všech základních algoritmů na všech datových souborech. Vektor vítězných algoritmů pro různé experimenty vybraných datových souborů můžeme vidět v tabulce 5.1.

Předzpracování dat

Část předzpracování dat patří při aplikacích strojového učení k těm nejvíce časově náročným. To je dáno zejména tím, že je velmi těžké do automatických systémů integrovat inteligenci, která rozhodne, jak správně data zpracovat. Inteligence (algoritmus), která by rozhodovala, které proměnné jsou důležité, které jsou zbytečné, které sloupce textového souboru nejsou vůbec proměnné, která pozorování s vynechanými hodnotami je vhodnější vyřadit a která je lepší doplnit například průměrem příslušné proměnné. Dále je důležitá schopnost identifikovat kategorické a numerické proměnné, identifikovat odlehlá pozorování a rozhodnout, zda je ponechat, či vyřadit.

Protože jsme na začátku neměli k dispozici čistá data a datových souborů bylo přes 2000, tak jsme byli nuceni si zkonstruovat vlastní automatický algoritmus pro předzpracování dat. A i kvůli konstrukci tohoto algoritmu byla, i v našem případě, část předzpracování dat velmi časově náročná. Na náš algoritmus pro předzpracování dat se můžeme dívat jako na černou skříňku, která má za úkol zchroustat všechny datové soubory v jejich surové podobě uložené v jedné složce. Výstupem předzpracování jsou upravené datové soubory v čisté podobě požadovaného formátu vhodného pro algoritmy základního učení. Pojďme se podívat dovnitř našeho sestrojeného algoritmu pro předzpracování dat. Algoritmus v cyklu prochází složku se staženými datovými soubory pro učení s učitelem a aplikuje na ně následující operace.

1. Nejprve byly vynechány řádky se znečištěnými nebo prázdnými hodnotami.

2. Dále byly z výpočetních důvodů zredukovány velikosti souborů. Délka velkých souborů byla zkrácena na 100 až 1000 řádků. Nová hodnota počtu řádků byla volena náhodně, aby se u velkých souborů nezredukoval příznak počtu pozorování na jednu hodnotu.

3. Byly detekovány kategorické a spojité proměnné. Proměnné ve formě textu byly označeny jako kategorické a byly překódovány pomocí umělých proměnných na několik[4] proměnných, které obsa-

[1] https://vincentarelbundock.github.io/Rdatasets/datasets.html, 5.8.2018
[2] https://github.com/curran/data, 5.8.2018
[3] https://data.fivethirtyeight.com/, 5.8.2018
[4] Z výpočetních důvodů jsme tohle číslo omezili jen na tři kategorie. Pokud kategorická proměnná obsahovala více než 3 různé kategorie, odstranili jsme ji.

hovaly na všech pozicích nuly, kromě pozic příslušné kategorie. Těchto nových umělých proměnných bylo o jednu méně než kategorií, protože jsme na konci předzpracování přidávali i sloupec jedniček (intercept) pro metody lineární regrese.

4. Kategorické proměnné v numerické formě jsme detekovali tak, že pokud daná numerická proměnná obsahovala více než pevně dané procento různých hodnot celkového počtu pozorování (volili jsme 10 %)[5] detekovali jsme ji jako kategorickou. Jinak jsme ji detekovali jako numerickou. Znovu jsme k nalezeným kategorickým proměnným v numerické podobě přidali umělé proměnné.

5. Protože jsme se rozhodli ve fázi ZU používat jen regresní metody (viz sekce 4.2), bylo nutné vhodně přeskládat kategorické a numerické proměnné. Kategorické proměnné byly přesunuty na první sloupce a numerické proměnné na další sloupce nového datového souboru. Pokud měl datový soubor jen kategorické proměnné, byl vyřazen. Tím jsme získali soubor, který měl v posledním sloupci numerickou proměnnou, a tedy soubor byl vhodný pro regresní algoritmy základní úrovně.

6. Z výpočetních důvodů jsme redukovali šířku souboru. Pokud měl výsledný soubor více než 15 sloupců, levé sloupce byly vyřazeny.

7. Dále jsme empiricky vypozorovali, že pokud má datový soubor proměnné ve velmi odlišných měřítkách (např. 10^{-3} a 10^{6}), metoda lineární regrese nebyla schopna vytvořit inverzní matici. Matice byla špatně podmíněná a její podmíněnost jsme vylepšili pomocí normalizace proměnných do rozmezí $(0, 1)$.

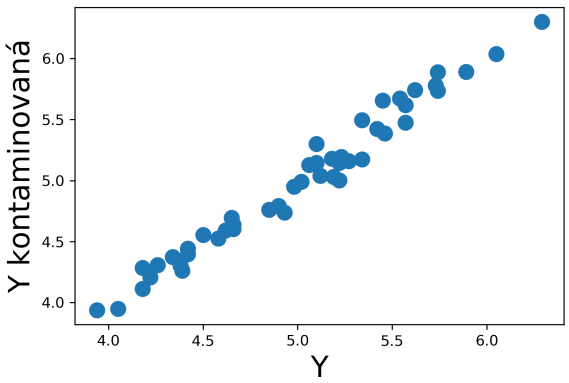
8. Dále způsobovalo základním algoritmům lineární regrese problémy, pokud byla matice skoro singulární, a tedy mezi některými sloupci této matice byla silná multikolinearita. Multikolineární sloupce nezávislých proměnných byly postupně vyřazovány metodou zpětného hledání. Tato metoda postupně vyřazovala sloupce s nejvyšším faktorem variance inflace [197], dokud maximální hodnota tohoto faktoru u proměnných nebyla vyšší než předem stanovená mez. Obecné doporučení nastavení meze je mezi 5 a 10, tak jsme ji nastavili na hodnotu 7. Pokud jsme touto metodou vyřadili počáteční sloupec jedniček, na konci jsme ho znova přidali.

9. Pokud po této proceduře byly vyřazeny všechny sloupce až na zmíněný sloupec jedniček, nebo zůstal soubor menší než čtyři pozorování, tak byl soubor vyhozen.

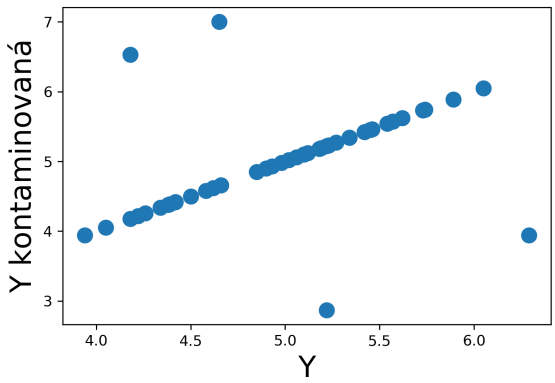
10. Nakonec byly otestovány všechny metody základního učení na tento nově vzniklý soubor. Pokud tohle testování proběhlo bez problému a program nevyhodil výjimku, byl tento nový soubor uložen do cílové složky. Pokud program vyhodil výjimku, soubor byl zahozen.

Celá tato procedura vedla ke zredukování počtu až na 643 souborů, což je podle našeho názoru pro naši studii metaučení dostačující. Dále je zřejmé, že tento hrubý postup předzpracování dat mohl vést k jejich poškození. Nenapadla nás ale lepší varianta, jak získat velký počet datových souborů stejného formátu vhodného pro fázi ZU. Jsme si vědomi, že jsme mohli u několika datových souborů změnit jejich příznaky předzpracováním natolik, že už nejsou vhodné pro metaučení. Podle našeho názoru je to pořád lepší postup, než si data generovat uměle. Rozhodli jsme se naši studii provést pro velký počet datových souborů, což pro ruční předzpracování není z časových důvodů ideální. Sázíme na velký počet datových souborů a myšlenky, že se v tomto velkém počtu najde hodně datových souborů, jejichž příznaky jsme do velké míry nezměnili a budou se tedy dobře dát kategorizovat v prostorů příznaků. V našem systému

[5] Při detekci a následném přidání umělých proměnných pro kategorické proměnné jsme se mohli dívat jen na to, jestli má daná proměnná 3 kategorie a ne dané procento. Chtěli jsme, aby systém detekoval všechny kategorické proměnné, a pak podle výpočetní náročnosti jsme se rozhodli, kolik jich uchováme.



Obrázek 4.2: Lokální kontaminace.



Obrázek 4.3: Globální kontaminace.

doufáme, že po hrubém předzpracování dat budeme schopni diskriminovat mezi jednotlivými datové soubory na základě jejich charakterizací.

Nejsme si vědomi nějaké studie, která zahrnuje přítomnost šumu nebo odlehlých pozorování v datech a jejich vliv na proces metaučení. Také jsme nenašli ani žádné snahy zrobustifikovat proces metaučení. Proto jsme se vydali tímto směrem a rozhodli jsme se oba tyto problémy prozkoumat. Proto jsme k předzpracovaným datovým souborům přidali umělou kontaminaci.

Kontaminace

Protože jsme nikde nenašli články zkoumající citlivost metaučení, rozhodli jsme se ji otestovat v této diplomové práci. K tomu, abychom toho docílili, jsme museli k původním datovým souborům přidat umělou kontaminaci. K datovým souborům jsme přidali dva typy kontaminace (lokální viz obrázek 4.2 a globální viz obrázek 4.3). Abychom byli schopni interpretovat citlivost metaučení, museli jsme celou studii provést celkem třikrát. Poprvé s původními daty, podruhé s přidanou lokálním kontaminací[6] a potřetí s přidanou globální kontaminací.[7]

- **Lokální kontaminace:** Každé pozorování v datovém souboru je kontaminováno nějakým malým šumem.
- **Globální kontaminace:** Malé procento pozorování je kontaminováno obrovským šumem, zatímco ostatní pozorování jsou neporušena.

4.2 Základní algoritmy

Cílem základních algoritmů je poskytnout meta-odezvu pro meta-algoritmus na meta-úrovni. Rozhodli jsme se omezit jen na meta-odezvu typu doporučení nejlepšího základního algoritmu (viz subsekce 3.5.2).

[6] Všechny odezvy byly kontaminovány aditivním Gaussovským šumem s rozptylem dvaceti procent interkvartilového rozpětí příslušné odezvy.

[7] K horní deseti procentům pozorování je s padesáti procentní pravděpodobností přičtena, nebo odečtena aditivní složka o velikosti dvacetinásobku variačního rozpětí příslušné odezvy. Variační rozpětí R je vzdálenost mezi nejmenším a největším pozorováním. $R = x_{min} - x_{max}$.

Ve fázi ZU se natrénuje k algoritmů na n datových souborech. Kde $k, n \in \mathbb{N}$. Pro každou dvojici (i, j), kde $(i \in 1, \ldots, k)$ a $(j \in 1, \ldots, n)$, je v CV[8] vypočtena ztráta algoritmu i na datovém souboru j. Tuto ztrátu označme $L_{i,j}$. Naším cílem je z těchto ztrát vybrat nejlepší algoritmus minimalizující ztrátu pro každý datový soubor j. Tedy j-tá složka vektoru meta-odezvy je dána vztahem:

$$Y_{j_{meta}} = \underset{i}{\operatorname{argmin}}\, L_{i,j}.$$

$Y_{j_{meta}}$ tedy obsahuje kategorickou proměnnou (vítězný algoritmus) pro datový soubor j (viz tabulka 5.1). Spojením dostaneme celkovou odezvu pro meta-algoritmus:

$$Y_{meta} = (Y_{1_{meta}}, Y_{2_{meta}}, \ldots, Y_{n_{meta}}).$$

V metaučení jsme použili čtyři algoritmy základního učení ($k = 4$) a 643 datových souborů ($n = 643$). Těmito algoritmy jsou metody lineární regrese. První z nich je nejčastěji používaná metoda nejmenších čtverců. Další tři metody spadají do oblasti robustní statistiky. Jsou to nejmenší vážené čtverce, nejmenší usekané čtverce a Huberův M-odhad.

Nejmenší čtverce

Metoda nejmenších čtverců (OLS) je zřejmě nejpoužívanější regresní metodou. Tuto metodu používáme v základní úrovni systému pro automatickou selekci algoritmu, proto si ji zde v základu nastíníme. Odvodíme odhad vektoru parametrů $\boldsymbol{\beta}$ pro standardní lineární model. Mějme p nezávislých proměnných. Zapíšeme model po složkách:

$$Y_i = \beta_1 X_{i1} + \beta_2 X_{i2} + \cdots + \beta_p X_{ip} + \varepsilon_i, \quad i = 1, 2, \ldots, n, \tag{4.1}$$

kde X_{ij} je i-té pozorování j-té vysvětlující proměnné a Y_i je i-té pozorování odezvy. V tomto vzorci předpokládáme n pozorování jedné odezvy Y_i a p vysvětlujících proměnných $X_{i1}, X_{i2}, \ldots, X_{ip}$. Hlavním cílem je odhadnout parametry β_j, $(j = 1, 2, \ldots, p)$. ε_i je i-tá náhodná chyba pocházející z normálního rozdělení.

Pomocí metody nejmenších čtverců odhadneme parametry β_j. Chceme minimalizovat součet čtverců reziduí od regresní křivky. Z rovnice (4.1) vyjádříme i-té reziduum:

$$u_i = Y_i - \hat{\beta}_1 X_{i1} - \hat{\beta}_2 X_{i2} - \cdots - \hat{\beta}_p X_{ip} = Y_i - \sum_{j=1}^{p} \hat{\beta}_j X_{ij}, \quad i = 1, 2, \ldots, n.$$

Zavedeme účelovou funkci

$$S = \sum_{i=1}^{n} {u_i}^2 = \sum_{i=1}^{n} \left(Y_i - \sum_{j=1}^{p} \hat{\beta}_j X_{ij} \right)^2,$$

kterou budeme minimalizovat. Funkce S je konvexní, tak pro získání minima zderivujeme S podle parametrů:

$$\frac{\partial S}{\partial \hat{\beta}_j} = 2 \sum_{i=1}^{n} u_i \frac{\partial u_i}{\partial \hat{\beta}_j}, \quad j = 1, 2, \ldots, p.$$

Dostáváme gradient, který když položíme nulovému vektoru, tak dostaneme stacionární bod. Snadno nahlédneme, že jde o minimum. Odhady parametrů $\hat{\beta}_1, \hat{\beta}_2, \ldots \hat{\beta}_p$ minimalizují rovnici

[8] Pro generalizaci algoritmů v základní i meta-úrovni používáme 10-násobnou-CV.

$$2\sum_{i=1}^{n}\left(Y_i - \sum_{k=1}^{p}\hat{\beta}_k X_{ik}\right)\left(-X_{ij}\right) = 0, \quad j = 1, 2, \ldots, p.$$

Po úpravě dostaneme p normálních rovnic

$$\sum_{i=1}^{n}\sum_{k=1}^{p} X_{ij}X_{ik}\hat{\beta}_k = \sum_{i=1}^{n} X_{ij}Y_i, \quad j = 1, 2, \ldots, p.$$

Tyto rovnice lze zapsat maticově jako

$$\left(\mathbf{X}^T\mathbf{X}\right)\hat{\boldsymbol{\beta}} = \mathbf{X}^T\mathbf{Y}.$$

Pokud existuje inverzní matice k matici $\mathbf{X}^T\mathbf{X}$, tak celou předchozí rovnici vynásobíme touto inverzní maticí zleva a dostaneme odhady parametrů $\hat{\boldsymbol{\beta}}$

$$\hat{\boldsymbol{\beta}} = \left(\mathbf{X}^T\mathbf{X}\right)^{-1}\mathbf{X}^T\mathbf{Y}, \tag{4.2}$$

kde

$$\mathbf{X} = \begin{bmatrix} X_{11} & X_{12} & X_{13} & \ldots & X_{1p} \\ X_{21} & X_{22} & X_{23} & \ldots & X_{2p} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ X_{n1} & X_{n2} & X_{n3} & \ldots & X_{np} \end{bmatrix}, \quad \mathbf{Y} = \begin{bmatrix} Y_1 \\ Y_2 \\ \vdots \\ Y_n \end{bmatrix}, \quad \hat{\boldsymbol{\beta}} = \begin{bmatrix} \hat{\beta}_1 \\ \hat{\beta}_2 \\ \vdots \\ \hat{\beta}_p \end{bmatrix}.$$

Metoda OLS je velmi náchylná k odlehlým pozorováním, protože minimalizuje čtverce reziduí. Velké čtverce odlehlých pozorování ji proto velmi snadno vychýlí. Pro kontaminovaná data je lepší použít metody robustní statistiky.

Robustní metody

Při statistickém zpracování dat je často potřeba nějakého mechanismu, který umožní to, že několik odlehlých pozorování nenaruší statistické odhady a predikce. Tato odlehlá pozorování mohou být způsobena například chybou měření, lidskou chybou, měřením za nestandartních podmínek atd. Proto vzniklo odvětví robustní statistiky, které se snaží odlehlá pozorování ignorovat, aby vůbec, nebo co nejméně narušila statistické predikce. V klasické nerobustní statistice je velmi populární metoda nejmenších čtverců, která je ovšem náchylná na odlehlá pozorování. V literatuře je popsáno několik metod [198, 199], které slouží jako robustní alternativy k metodě nejmenších čtverců. Díky svým robustním vlastnostem mají tyto metody větší praktické použití například v aplikované ekonometrii [200], [201]. Robustní statistické postupy jsou takové, které si zachovávají určitou optimalitu v okolí nějakého základního rozdělení (například normálního). Robustní postupy lze chápat jako určitá vylepšení klasických postupů, která neselžou při malých odchylkách od základních předpokladů. Základní přehled robustních statistických metod můžeme nalézt v knize [202].

Jedním ze základních charakteristik robustní statistiky je tzv. bod selhání (*breakdown point*) [203]. Udává, jaké procento dat s libovolnými hodnotami může být k datovému souboru přidáno tak, že tato přidaná data libovolně moc nevychýlí statistický odhad. Bod selhání může být maximálně 50 procent. Pokud je totiž kontaminováno více než 50 procent pozorování, tak už není možné rozlišit mezi kontaminovanými a původními pozorováními. Čím více jsou odhady robustnější, tím více ztrácejí na eficienci[9] a naopak. Správnou volbou bodu selhání se volí kompromis mezi robustností a eficiencí.

[9] Čím více je odhad eficientní, tím méně potřebuje pozorování, aby dosáhl dobré úspěšnosti.

Ve statistické literatuře chybí systematické porovnání výsledků robustních metod na reálných datech. Možná z důvodu nedostatečného srovnání robustních metod se robustní metody zatím tolik neprosadily v praxi jako jejich nerobustní konkurující metody [204]. Možná je to dáno i tím, že robustní metody jsou výpočetně náročnější a některé mají iterativní charakter. My jsme se tyto robustní metody rozhodli prozkoumat v rámci metaučení. Také jsme se pomocí robustních metod, které ignorují kontaminaci, pokusili zkoumat citlivost metaučení. První z implementovaných robustních metod je odhad pomocí nejmenších vážených čtverců.

Nejmenší vážené čtverce

Tento odhad se často plete s více známým odhadem vážených nejmenších čtverců. Metodu nejmenších vážených čtverců budeme značit zkratkou LWS (*Least Weighted Squares*). LWS je popsán v [205] a s lineárně klesajícími vahami v [204]. V této práci jsou ale voleny váhy následující

$$w_i = \begin{cases} 1 & \text{pro } i = 1, \dots, \left\lceil \frac{6n}{10} \right\rceil, \\ \text{lineárně se snižující} & \text{pro } i = \left\lceil \frac{6n}{10} \right\rceil + 1, \dots, \left\lceil \frac{8n}{10} \right\rceil, \\ 0 & \text{pro } i = \left\lceil \frac{8n}{10} \right\rceil + 1, \dots, n, \end{cases}$$

kde n je počet pozorování. V experimentálních pokusech (viz kapitola 5) volíme pro LWS váhu rovnou jedné pro nejmenších šedesát procent reziduí, dalším dvaceti procentům reziduí přiřadíme lineárně se snižující váhy a posledním největším dvaceti procentům reziduí přiřadíme nulové váhy.

Odhad parametru $\tilde{\beta}^{(LWS,n)}$ je definován jako

$$\tilde{\beta}^{(LWS,n)} = \underset{\beta \in \mathbb{R}^{p+1}}{\operatorname{argmin}} \sum_{i=1}^{n} w_i u^2_{(i)}(\beta),$$

kde $u_i(\beta)$ je reziduum příslušející i-tému pozorování pro dané $\beta \in \mathbb{R}^{p+1}$ a $u^2_{(1)}(\beta) \leq \cdots \leq u^2_{(n)}(\beta)$ jsou uspořádané hodnoty kvadrátů reziduí. Největším reziduím v absolutní hodnotě se přiřadí malá váha, a tím pádem se eliminuje jejich velký vliv. Stejná úvaha platí i pro seřazení reziduí v následujícím LTS odhadu.

Algoritmus pro nalezení odhadu LWS je iterační [206]. Jedna iterace totiž může vést do nějakého lokálního minima. Pro nalezení globálního minima je třeba těchto iterací provést několik. Počet iterací c je dalším hyperparametrem metody LWS. My jsme podle empirických pozorování tuto hodnotu nastavili na $c = 20$. Jelikož je LWS málo známá metoda, není součástí žádné ze softwarových knihoven. Byli jsme proto nuceni naimplementovat celý algoritmus LWS (viz tabulka 4.1).

Nejmenší usekané čtverce

Tato robustní metoda byla studována v [207]. Je to pravděpodobně nejpopulárnější robustní metoda s vysokým bodem selhání [208]. Tuto metodu budeme v celé této práci značit zkratkou LTS (*Least Trimmed Squares*). Odhad parametru $\tilde{\beta}^{(LTS,n,h)}$ je definován jako

$$\tilde{\beta}^{(LTS,n,h)} = \underset{\beta \in \mathbb{R}^{p+1}}{\operatorname{argmin}} \sum_{i=1}^{h} u^2_{(i)}(\beta),$$

kde $u_i(\beta)$ je reziduum příslušející i-tému pozorování pro dané $\beta \in \mathbb{R}^{p+1}$ a $u^2_{(1)}(\beta) \leq \cdots \leq u^2_{(n)}(\beta)$ jsou uspořádané hodnoty kvadrátů reziduí. Dále n je počet pozorování a číslo h pokládáme rovno dolní celé části z nějakého procenta počtu celkových pozorování. V této práci budeme odhad LTS počítat z osmdesáti procent počtu pozorování. Tedy $h = \lfloor 0.8 \cdot n \rfloor$.

Vstup: Datový vzorek $S = \{\mathbf{X_i}, Y_i\}_{i=1}^n \subset \mathbb{R}^p \times \mathbb{R}$. Váhy $\{w_i\}_{i=1}^n \subset [0, 1]$.
Výstup: Odhad regresních koeficientů b.

1. Nastavíme hodnotu ztrátové funkce na plus nekonečno: $L \leftarrow +\infty$. Náhodně vybereme m pozorování, ze kterých spočteme odhad b regresních koeficientů β metodou nejmenších čtverců.
2. Pro každé pozorování spočteme reziduum a přiřadíme mu váhu, která závisí na absolutní hodnotě tohoto rezidua. Rezidua i váhy jsou seřazeny. Velkým reziduím je přiřazena malá váha a naopak.
3. Porovnáme hodnotu ztrátové funkce vypočtené z daných vah (označme H) se současnou hodnotou ztrátové funkce L. Jestli $H < L$, jdi na bod 4. Jinak jdi na bod 5.
4. $L := H$, najdeme nový odhad b pomocí známější metody vážených nejmenších čtverců. Jdi na bod 2.
5. Opakuj algoritmus z bodu 1 c-krát. Kde c je zvolený hyperparametr. Výstupem bude odhad b, pro který je ztrátová funkce nejnižší ze všech c opakování. Díky dostatečně velké hodnotě c budeme mít větší pravděpodobnost nalezení globálního minima.

Tabulka 4.1: Algoritmus LWS.

LTS je speciálním případem LWS. LWS s vahami rovnými jedné pro h pozorování a pro zbytek vah rovnými nule je ekvivalentní LTS odhadu.

Huberův M-odhad

Huberův odhad patří do kategorie robustní statistiky zvané M-odhady a je navržen v [209]. M-odhady jsou v praxi nejpoužívanějšími robustními metodami. Kromě Huberova M-odhadu je ještě velmi používaný Hampelův M-odhad [198], který v této práci používat nebudeme. M-odhadem rozumíme odhad $\hat{\beta}$ minimalizující funkci

$$Q(u_i, \rho) = \sum_i \rho\left(\frac{u_i}{s}\right),$$

kde ρ je nezáporná, symetrická, monotónní funkce, která je v nule rovna nule a jejímž cílem je snížit vliv odlehlých pozorování. Dále u_i je i-té reziduum a s je odhad stupnice.[10] M-odhad je taková lineární přímka, která minimalizuje sumu funkčních hodnot reziduí. Každé reziduum má tedy jen omezený vliv do sumy v závislosti na funkci ρ. Funkce ρ Huberova M-odhadu je do jistého bodu kvadratická a potom lineární. To znamená, že menší rezidua jsou penalizována čtvercem vzdálenosti a větší už jen jejich vzdáleností od odhadu. Odhad $\hat{\beta}$ je pak získán iterativní metodou vážených nejmenších čtverců. Na rozdíl od metody LTS nedisponují M-odhady vysokým bodem selhání. Je třeba rozlišovat mezi lokální a globální robustností (robustnost, necitlivost). Z výše popsaných metod lineární regrese je pouze Hampelův M-odhad a LWS robustní v lokálním slova smyslu a pouze LTS je robustní v globální slova smyslu [205], [209].

Shrnutí

Použité algoritmy v základním učení jsou:

- **OLS**
- **LWS** s vahami rovnými jedné pro šedesát procent nejmenších reziduí, pro dalších dvacet procent lineárně se snižující váhy a pro zbytek nulové váhy. Hodnotu počtu iterací jsme volili jako $c = 20$, přestože při empirických pokusech jsme zpozorovali, že $c = 10$ je dostačující.
- **LTS** s useknutím $h = \lfloor 0.8 \cdot n \rfloor$
- **Huberův M-odhad**

4.3 Míra predikce základního učení

Na rezidua spočtená pomocí 10-násobné-CV v první části ZU jsme se rozhodli aplikovat tři různé predikční míry. Jsou to:

- Střední kvadratická chyba **MSE** (*mean squared error*)

$$\text{MSE} = \sum_{i=1}^{n} \frac{u_i^2}{n}, \quad \text{kde } u_i^2 \text{ je kvadrát } i\text{-tého rezidua a } n \text{ je počet pozorování.}$$

[10] Odhad stupnice je vypočten jako součin tzv. MAD (*Median absolute deviation*) a konstanty K. MAD je robustní míra variability vyjadřující medián z absolutních hodnot rozdílů jednotlivých reziduí a mediánu všech reziduí. Míra MAD je dána vztahem: $\text{MAD} = \text{median}\,(|\, u_i - \text{median}\,(u_i)\,|)$. Konstanta K je hodnota inverzní distribuční funkce normálního rozdělení nastavená tak, aby bylo dosaženo 95 procentní eficientnosti.

- Střední usekaná chyba **TMSE** (*trimmed mean squared error*) s hyperparametrem α. V této práci jsme volili hodnota $\alpha = 0.8$.

$$\text{TMSE}(\alpha) = \sum_{i=1}^{k} \frac{u_{(i)}^2}{k} \quad \text{pro } k = \lfloor \alpha \cdot n \rfloor, \text{ kde } u_{(i)}^2 \text{ je kvadrát } i\text{-tého uspořádaného rezidua a } \alpha \in \left[\frac{1}{2}, 1\right)$$

- Střední vážená chyba **WMSE** (*weighted mean squared error*) s lineárně se snižujícími vahami w_i rovnými

$$w_i = 1 - \frac{(i-1)}{n}, \quad \text{kde } i = 1, \ldots, n.$$

$$\text{WMSE} = \frac{\sum_{i=1}^{n} u_{(i)}^2 w_i}{\sum_{i=1}^{n} w_i}, \quad \text{kde } u_{(i)}^2 \text{ je kvadrát } i\text{-tého uspořádaného rezidua a } w_i \text{ je jeho váha.}$$

Připomeňme, že i-té neseřazené reziduum je dáno vztahem $u_i = Y_i - \hat{Y}_i$, pro $i = 1, \ldots, n$. Y_i značí skutečnou hodnotu a $\hat{Y}_i$ predikovanou hodnotu i-tého pozorování.

MSE je dána jako součet všech kvadrátů reziduí dělená jejich počtem. TMSE se od MSE liší tím, že do součtu nezapočítáme nějaké předem určené procento těch největších čtverců. WMSE se od MSE liší tím, že každý čtverec v sumě je vynásoben předem danou vahou.

Jedním z cílů základního metaučení je vyprodukovat meta-odezvu pro meta-algoritmus. Pro výpočet meta-odezvy se v CV používá jedna z výše uvedených měr predikce. My jsme se rozhodli použít všechny nezávisle na sobě a zkoumat tak náš navržený systém metaučení pro každou z nich.[11]

4.4 Meta-příznaky

Hlavním úkolem meta-úrovně je vybrat takové meta-příznaky, které budou diskriminovat mezi jednotlivými algoritmy základní úrovně. Pokud budou meta-příznaky dobře diskriminovat jednotlivé základní algoritmy, tak na základě meta-příznaků nového datového souboru je meta-algoritmus schopen predikovat nejvhodnější algoritmus pro tento soubor. Volba příznaků by měla souviset s algoritmy použitými v základním učení, aby meta-příznaky byly schopné lépe diskriminovat na základně predikčních vlastností jednotlivých algoritmů. Proto mezi standartně používané příznaky používáme i robustní alternativy a doufáme, že robustní příznaky budou schopny odlišovat mezi robustními algoritmy základní úrovně. Zvolili jsme deset příznaků, kde většina z nich je navržena v [210]. Vybranými příznaky jsou:

1. Počet pozorování datového souboru (označme n).
2. Přirozený logaritmus[12] podmíněnosti matice nezávislých proměnných daný vzorcem

$$\ln\left(\kappa\left(X\right)\right),$$

   kde X je matice nezávislých proměnných a $\kappa\left(X\right)$ je číslo podmíněnosti matice X.
3. Podíl n/p, kde n je počet pozorování a p je počet nezávislých proměnných.

[11] Není možné srovnávat jednotlivé míry predikce mezi sebou. Srovnávat lze pouze různé hodnoty měr predikce stejného typu pro různé algoritmy aplikované na daný datový soubor.

[12] Podmíněnosti matic datových souborů se typicky pohybují v exponenciální škále. Proto je nutné transformovat tento příznak do lineární škály pomocí logaritmu.

4. Desítkový logaritmus[13] p-hodnoty Shapiro-Wilk testu.
5. Šikmost reziduí napočítaných z OLS odhadu.
6. Špičatost reziduí napočítaných z OLS odhadu.
7. Koeficient determinace R^2.
8. Procento odlehlých pozorování, které je odhadnuté pomocí LTS vypočtené podle vzorce

$$\frac{1}{n}\sum_{i=1}^{n} I[u_i/\hat{\sigma} > 2.5],$$

   kde $u_1, \ldots, u_n$ jsou rezidua získaná metodou LTS s hyperparametrem $h = 0.8$, σ^2 je rozptyl $e_1, \ldots, e_n$ a $\hat{\sigma}$ je odhad σ, který dostaneme metodou LTS s hyperparametrem $h = 0.8$. Tento vzorec je navržen v [208].
9. Robustní koeficient determinace R^2_{LTS}. R^2_{LTS} vytvoříme tak, že jej napočítáme z 80 procent pozorování s nejmenšími rezidui (v absolutní hodnotě). Abychom získali tyto rezidua, tak musíme provést odhad LTS.
10. Podíl počtu kategorických a všech proměnných.

[13] Podmíněnosti p-hodnot se typicky pohybují v exponenciální škále. Proto je nutné transformovat tento příznak do lineární škály pomocí logaritmu.

datový soubor	číslo meta-příznaku									
	1	2	3	4	5	6	7	8	9	10
bcdeter	30	2.29	0.03	-7.73	-0.57	9.63	0.03	0.03	0.77	0
bigcity	29	1.82	0.03	-6.74	-0.71	8.81	0.03	0.1	0.09	0
biomassTill	123	0.97	0.01	-18.54	-1.18	9.02	0	0.11	0.2	0.33
bmt	45	5.67	0.02	-8.62	0.5	8.42	0.05	0.07	0.79	0
Bollcen	17	2.52	0.12	-1.47	-1.28	4.37	0.32	0	0.85	0.25
cd4.nested	37	5.05	0.03	-8.31	-0.37	9	0.03	0.1	0.04	0
ChildSpeaks	237	9.89	0	-25.35	1.06	10.4	0	0.07	0.07	0
chorSub	38	4.64	0.05	-7.51	0.17	8.74	0.04	0.05	0.05	0
cloth	146	1.78	0.01	-19.68	0.08	10.34	0	0.05	0	0
Clothing	429	2.27	0.01	-32.41	-0.56	10.76	0.01	0.09	0.19	0.38
CloudSeeding2	630	2.9	0	-39.89	0.04	10.67	0	0.14	0	0
codling	27	3.04	0.04	-7.77	0.88	8.74	0	0.22	0.09	0
CYGOB1	107	5.33	0.01	-12.95	0.22	8.76	0.08	0.07	0.22	0
delivery	18	0.89	0.06	-3.71	0.51	6.87	0.14	0.06	0.2	0.33
downs.bc	418	4.87	0	-32.25	-0.2	10.12	0	0.05	0	0
drughiv	66	1.7	0.03	-10.84	-2.13	6.95	0.14	0.03	0.32	0
elastic2	100	0.88	0.01	-15.69	-0.83	9.62	0.01	0.04	0.05	0.33
elasticband	512	1.51	0	-35.77	-0.19	10.2	0	0.1	0.99	0
FARSmiss	212	2.18	0	-23.89	-0.38	9.56	0	0.04	0.09	0
Fertility	125	3.43	0.01	-17.53	-1.73	8.53	0.02	0.06	0.02	0
FinalFourIzzo	314	2.37	0	-28.09	-0.69	9.88	0.01	0.13	0.59	0
FinalFourShort	928	7.87	0	-53.08	0.12	10.18	0	0.05	0.22	0
grav	117	1.93	0.01	-16.65	0.39	9.5	0.01	0.14	0.01	0
Hstarts	180	2.33	0.01	-22.44	0.41	10.5	0	0.16	0.01	0
humanpower2	365	2.23	0	-30.09	-0.53	10.26	0.01	0.07	0.07	0
hurricNamed	365	4.57	0	-30.46	0.51	9.74	0	0.08	0.02	0
immi2	863	1.13	0	-43.47	0.31	10.17	0	0.05	0.07	0.5
immigration	363	5.15	0	-31.2	0.21	10.95	0	0.1	0	0
Leinhardt	51	5.9	0.1	-7.94	0.26	8.63	0.24	0.2	0.63	0
leuk	205	0.89	0	-24.16	0.22	10.23	0	0.15	0.08	0.33
MathEnrollment	90	2.13	0.01	-15.6	-1.29	9.46	0.01	0.03	0.07	0
Mathlevel	753	2.32	0	-42.05	-0.18	10.15	0	0.07	0.02	0
nihills	101	7.38	0.02	-16.47	0.95	8.86	0	0.08	0.09	0
PPP	578	4.49	0	-38.09	0.18	10.67	0	0.06	0.03	0
Prestige	48	2.77	0.02	-8.31	-1.17	8.18	0.06	0.02	0.2	0
Pricing	41	3.2	0.02	-9.92	1.15	7.68	0	0.32	0.05	0
pulpfiber	198	4.6	0.01	-22.01	0.05	10.21	0.01	0.08	0.02	0
satact	123	1.15	0.01	-17.9	0.18	9.39	0	0.08	0.07	0.33
SeaSlugs	30	6.95	0.03	-8.44	2.57	7.83	0.03	0.1	0.67	0
stVincent	40	1.62	0.08	-8.19	2.58	9.02	0.12	0.88	0.94	0.33

Tabulka 4.2: Meta-příznaky náhodně vybraných datových souborů. Sloupce slouží jako vysvětlující proměnné pro klasifikační meta-algoritmus.

4.5 Meta-algoritmy

Cílem meta-algoritmu je na základě množiny meta-pozorování (datových souborů), které jsou v prostoru jejich charakterizací (meta-příznaků) označkovány[14] meta-odezvou predikovat pro nové meta-pozorování jeho meta-odezvu.

Protože je meta-odezva kategorická proměnná, jsme nuceni použít klasifikační meta-algoritmus. V meta-úrovni jsme nezávisle na sobě použili následující klasifikační meta-algoritmy:

- většinový klasifikátor[15]
- k-NN ($k = 20, 50$)
- SVM[16]
- logistická regrese s L1 a L2 regularizací[17]
- LDA

Díky metodě CV můžeme i na meta-úrovni dosáhnout generalizace. Tím zjistíme, jak je kvalitní náš navržený systém metaučení pro doporučení algoritmu. Kvalitu systému měříme nějakou mírou predikce (meta-míra predikce). A protože používáme klasifikační meta-algoritmus, jsme omezeni na výběr z množiny možných klasifikačních měr predikce. V 10-násobné-CV se jednotlivá meta-pozorování zařadí do matice záměn[18], ze které můžeme počítat některé míry[19] vyjadřující kvalitu klasifikátoru, a tedy našeho zkonstruovaného systému pro automatické doporučení algoritmu.

[14] Pojmem označkovány chápeme přiřazení meta-pozorování do jedné kategorie. V tomto případě se jedná o kategorie algoritmů základní úrovně (OLS, LWS, LTS, Huber).

[15] Na většinový klasifikátor pohlížíme jako na tzv. baseline klasifikátor. Většinový klasifikátor totiž klasifikuje vždy do většinové kategorie nezávisle na příznacích. Porovnávám s většinovým klasifikátorem pak můžeme určit schopnost klasifikátoru klasifikovat na základě příznaků.

[16] Úspěšnost SVM velmi silně závisí na dvou vnitřních hyperparametrech (c a γ). Tyto hyperparametry souvisí s jádrem v SVM metodě a se stupněm regularizace. My tyto hyperparametry volíme standardně přednastavené. Naším cílem není vytvářet další meta-úroveň, ve které bychom zkoumali optimální hyperparametry SVM.

[17] Do výsledné ztráty logistické regrese se kromě chyby z CV na testovacích datech připočte i hodnota závisející na velikostech parametrů logistické regrese. Tato připočtená hodnota je úměrná sumě absolutních hodnot parametrů logistické regrese (L1), nebo sumě kvadrátů parametrů logistické regrese (L2).

[18] Matice záměn je kontingenční matice, obsahující ve sloupcích skutečnou hodnotu předpovídané kategorie a v řádcích předpověď klasifikátoru. Buňky matice obsahují četnosti toho, kolikrát došlo na zkoumané datové množině k dané kombinaci skutečné a předpověděné hodnoty. Případy na diagonále matice záměn jsou klasifikovány správně, mimo diagonálu jde o chyby.

[19] Základní klasifikační míra je přesnost klasifikátoru. Tu vypočteme jako počet všech pozorování na diagonále matice záměn ku počtu všech pozorování. Velmi populární je použití F1-míry, protože je rovna harmonickému průměru preciznosti a úplnosti. Tyto míry se vyjadřují pro binární klasifikátor. Pro výpočet preciznosti a úplnosti při klasifikaci do více kategorií musíme tyto predikční míry spočítat pro všechny kategorie. To se provede tak, že se úloha rozdělí na k binárních klasifikací, kde k je počet kategorií. V jedné binární klasifikaci se daná kategorie označí jako pozitivní a všechny ostatní jako negativní. F1-míru pro úlohu klasifikace do více tříd získáme například jako aritmetický průměr F1-měr pro všechny kategorie.

4.6 Pracovní postup

V této sekci se pokusíme popsat jednotlivé implementační kroky, které vedly ke konstrukci systému pro automatický výběr algoritmu.

1. krok - softwarové nástroje

Nejprve bylo metaučení zkoumáno na 24 datových souborech s neuspokojivými výsledky. Proto jsme se rozhodli realizovat větší studii pro mnohem více datových souborů. Pro efektivní práci s velkým počtem datových souborů bylo zapotřebí se nejprve seznámit s některými softwarovými nástroji umožňující efektivní práci. Bez níže uvedených softwarových nástrojů si nedokážeme dost dobře představit práci s tak velkým počtem souborů. Pro realizaci experimentů pro tuto diplomovou práci byly použity následující softwarové nástroje:

- **Linuxový příkazový řádek** [211] je velice vhodný nástroj pro hromadnou manipulaci souborů. Také díky němu bylo pomocí příkazu *wget* staženo velké množství CSV souborů. Příkazový řádek byl používán také při hromadném kopírování, mazání, přesouvání a přejmenování souborů. Dále sloužil ke stahování balíčků pro programovací jazyk Python, zálohování práce a další.

- **Textový editor Vim** [212] je velmi mocný nástroj pro editaci textu. Ze začátku posloužil jako skvělý nástroj pro rychlou a efektivní editaci souborů. Spolu s textovým editorem Emacs je považován za nejlepší textový editor, což je ale vykoupeno velmi složitým a neintuitivním ovládáním.

- **Prohlížeč souborů Vifm** se skvěle doplňuje s Vimem a jsou spolu jedním stiskem klávesy propojeni. Ovládání má velmi podobné Vimu a umožňuje instantní náhled do souborů. To se hodí zejména ve chvílích, kdy chce člověk rychle nahlížet do souborů, měnit je zmáčknutím jediné klávesy a hlavně hledat rychle chyby v souborech, a tím odladʼovat celý systém pro metaučení. Vifm byl zpočátku také velmi užitečný při čištění a předzpracování dat. Později jsme zkonstruovali algoritmus pro automatické předzpracování.

- **Git** je nejpoužívanější systém správy verzí. Posloužil jako zálohovací nástroj na privátní úložiště Gitlab, později GitHub. Dále poskytl možnost se kdykoli vrátit k jakékoli verzi práce. Pomocí něho byly staženy stovky datových souborů z veřejného úložiště Github.

- **Programovací jazyk Python** se stává stále více populárním nástrojem pro zpracování dat, a hlavně pro strojové učení. To díky velké škále knihoven, které zajišťují komfortnější programování než například jazyk C++. Komfortnější prostředí pro práci s daty je ale vykoupeno pomalejším během. V Pythonu byly napsány všechny skripty.

  Z tohoto programovacího jazyka bylo využito několik výborných knihoven. Tři nejvíce používané knihovny v této práci byly:

  - **Numpy** je knihovna optimalizovaná pro maticové výpočty. Ekvivalent MATLABU v Pythonu. Hlavně díky ní byla provedena většina výpočtů.
  - **Pandas** je knihovna optimalizovaná pro práci s datovými soubory. Ekvivalent R v Pythonu. Díky ní bylo zautomatizováno předzpracování dat.
  - **scikit-learn** je velmi populární knihovna pro většinu metod strojového učení. Díky ní jsme nemuseli programovat klasifikační meta-algoritmy od začátku, ale využili jsme již přeprogramované klasifikační metody.

- **Spyder** je jednoduché vývojové prostředí se zaměřením na programovací jazyk Python. Toto prostředí je podobné vývojovému prostředí MATLAB. Má zabudovanou tzv. IPython konzoli, která je skvělým nástrojem pro odlaďování programů.

2. krok - Předzpracování dat

Při aplikovaném strojovém učení patří tato část k těm nejvíce časově náročným. To platilo i v našem případě. Nejdříve bylo metaučení provedeno pro 24 malých datových souborů s neuspokojivými výsledky. Bylo proto potřeba stáhnout více datových souborů a ty nějakým vhodným způsobem zpracovat do formátu vhodného pro metaučení. Jak již bylo zmíněno, tak bylo staženo přes 2000 datových souborů. Takový počet souborů není vhodné z časových důvodů předzpracovávat ručně. Na soubory byl hromadně spuštěn skript, který je hromadně zpracoval za použití cyklu a upravil do požadovaného formátu vhodného pro selekci algoritmu.

Požadovaný formát souborů byl z velké části docílen díky knihovnám *Pandas* a *Numpy*. Nejprve byly vynechány řádky se znečištěnými nebo prázdnými hodnotami. Dále byly z výpočetních důvodů zredukovány velikosti souborů. Délka velkých souborů byla zkrácena na 100 až 1000 řádků. Nové číslo počtu řádků bylo voleno náhodně, aby se nezredukoval tento příznak pouze na jednu hodnotu. Dále byly detekovány kategorické a spojité proměnné. Protože se v základním učení používají regresní metody, bylo nutné vhodně přeskládat kategorické a spojité proměnné. Kategorické proměnné byly přesunuty na první sloupce a spojité proměnné až za ně. Pokud měla kategorická proměnná maximálně tři kategorie, byly k ní přidány příslušné umělé proměnné. Pokud měla kategorická proměnná více než tři kategorie, byla z výpočetních důvodů vyřazena. Pokud měl výsledný soubor více než 15 sloupců, levé sloupce byly vyřazeny. Na první pozici byl přidán sloupeček jedniček, tedy konstantní člen v lineární regresi.

Bylo zpozorováno, že pokud je matice dat obsažená ve výsledném datovém souboru špatně podmíněná, tak některé regresní metody základního učení s touto maticí nedokázaly pracovat. Proto byla tato podmíněnost vylepšena. Dále způsobovalo problémy, pokud byla matice skoro singulární, a tedy mezi některými sloupci této matice byla multikolinearita. Silně multikolineární sloupce, kromě sloupce jedniček, byly postupně vyřazovány.

Pokud po této proceduře byly vyřazeny všechny sloupce (až na zmíněný sloupec jedniček), nebo zůstal soubor menší než zadaná mez, tak byl soubor vyhozen.

Dále byly otestovány všechny metody základního učení na tento nově vzniklý soubor. Pokud tohle testování proběhlo bez problému, byl nový soubor uložen do cílové složky. Pokud nastal nějaký další problém, soubor byl zahozen. Celá tato procedura vedla k zredukování až na 643 souborů, což je pro naši studii metaučení dostačující.

3. krok - Základní učení (1. část)

Poté, co byly datové soubory předzpracovány, následovalo základní učení. Všechny zbylé datové soubory byly znova postupně procházeny v cyklu. Poslední spojitý sloupec v každém datovém souboru byl označen jako odezva a zbývající sloupce jako nezávislé vysvětlující proměnné. V 10-násobné-CV byla predikována tato odezva pro čtyři základní metody lineární regrese. Rozdíl každé ze čtyř predikovaných a jedné skutečné odezvy vytvořil čtyři vektory reziduí. Každý vektor příslušející jedné metodě lineární regrese. 10-násobná-CV byla provedena ještě dvakrát. S odezvou lokálně kontaminovanou a dále s odezvou globálně kontaminovanou.

Tímto byl vytvořen čtyřrozměrný tensor reziduí, kde první rozměr udával číslo datového souboru, druhý rozměr typ kontaminace, třetí rozměr metodu lineární regrese a čtvrtý rozměr číslo řádku v datovém souboru. Tento čtyřrozměrný tensor reziduí následně posloužil při vyhodnocení nejlepších metod lineární regrese vzhledem k míře predikce. Tenzor byl předán do druhé části základního učení.

4. krok - Základní učení (2. část)

V této části byly aplikovány tři míry predikce (MSE, WMSE, TMSE) na čtyřrozměrný tensor reziduí získaný z první části základního učení. Pro každý datový soubor a každou míru kontaminace byly spočteny tyto míry predikce tak, aby se podle hodnoty míry predikce daly srovnat metody lineární regrese pro daný datový soubor, danou míru kontaminace a danou míru predikce. Metody lineární regrese byly srovnány a do další části procesu byla poslána jen kategorická proměnná v podobě vítězné metody (meta-odezva).

Druhá část základního učení poskytla jako výstup do fáze selekce algoritmu třídimenzionální tensor kategorických proměnných označujících vítězné metody lineární regrese ze základního učení, kdy první dimenze označovala číslo datového souboru, druhá dimenze míru kontaminace a třetí dimenze míru predikce.

5. krok - Selekce algoritmu (1. část)

V této fázi bylo spočteno deset základních příznaků napočítaných z každého datového souboru. Tyto příznaky vytvořily dvourozměrnou matici, kdy první rozměr udával číslo datového souboru a druhý rozměr udával číslo příznaku. Tato matice meta-příznaků následně posloužila jako vstup do druhé části selekce algoritmu.

6. krok - Selekce algoritmu (2. část)

Tato fáze má k dispozici dva vstupy. Prvním vstupem je matice základních příznaků napočítaných z datových souborů (meta-příznaky), druhým vstupem je trojrozměrný tenzor vítězných metod ze základního učení (meta-odezva). Nyní si v tomto tenzoru zvolíme druhou a třetí dimenzi, tedy míru kontaminace a míru predikce. Při volbě těchto dvou veličin se nám tensor zredukuje na vektor délky počtu datových souborů. Z tenzoru máme tedy devět vektorů, každý pro jednu ze tří měr kontaminace a jednu ze tří měr predikce.

Ve fázi selekce algoritmu využije meta-algoritmus právě matici základních meta-příznaků a meta-odezvu. Nyní každý řádek hraje roli jednoho datového souboru. Problém se dá interpretovat jako klasifikační úloha v deseti rozměrném prostoru příznaků, kde jednotlivé datové soubory chápeme jako jednotlivá pozorování a kategoriemi jsou vítězné metody ze základního učení.

Tato klasifikace je provedena pomocí 10-násobné-CV. Ke klasifikaci jsou použity již dříve popsané klasifikační metody. Každá z těchto klasifikačních metod může mít nějaké hyperparametry. Tyto hyperparametry mohou být voleny automaticky pomocí optimalizace hyperparametrů. My jsme nechtěli budovat nad metaučením další meta-úroveň, tak jsme se spokojili s většinou hyperparametrů přednastavených standardně v použitých balíčcích. Některé jsme odlaďovali empiricky pomocí několika experimentů. Sofistikovanější prohledávání prostoru hyperparametrů by mohlo vést ke zlepšení klasifikační přesnosti, které jsme si ale bohužel nemohli dovolit vzhledem k omezeným výpočetním kapacitám. Klasifikaci jsme samozřejmě nezávisle na sobě provedli pro všech devět vektorů, které jsou specifikovány daným typem kontaminace a danou míru predikce.

Predikcí naučeného meta-algoritmu pro nový datový soubor je dosaženo doporučení základního algoritmu pro tento soubor. Podle potřeby uživatele je náš systém je schopen doporučovat základní algoritmus pro tři predikční míry a pro tři druhy kontaminace.

4.7 Skripty

Všechny skripty jsou důkladně okomentované v anglickém jazyce a názvy proměnných jsou voleny co nejintuitivnějším způsobem. Přesto je zde pro čtenáře krátký popis jednotlivých skriptů. Všechny skripty

se nachází ve složce *kody* a jsou přiloženy na CD. Zájemce může také získat skripty na stránce GitHub.[20] Skripty byly spouštěny pomocí jazyka Python verze 3.6.6.

Pokud chce uživatel spouštět tyto skripty, musí mít nainstalovaný tzv. Python interpreter, pomocí kterého může v příkazové řádce spouštět skripty. Asi nejjednodušším způsobem je stáhnout si distribuci Pythonu zvanou Anaconda[21], která v sobě obsahuje všechno potřebné. Obsahuje i vývojové prostředí Spyder a správce balíčků conda. Pro funkčnost skriptů je potřeba pomocí správce balíčků stáhnout všechny potřebné balíčky. Balíčky potřebné pro daný skript se přidávají vždy na začátcích souborů pomocí klíčového slova *import*.

Celý systém jsme se snažili navrhnout pomocí metod objektově orientovaného programování a využít dědění a rozdělení problému na jednotlivé moduly. Kvůli tomu jsou na sobě skripty závislé. Všechny skripty mají koncovku *.py*, celkově obsahují 2521 řádků kódu a zde je jejich krátký výčet:

- **hyperparameters.py** - V tomto skriptu jsou nastaveny hyperparametry celého systému, které ovlivňují jeho běh a výsledky. Objekty definované v ostatních souborech dědí od objektu definovaném v tomto souboru jejich nastavení hyperparametrů. Jedná se o hyperparametry předzpracování dat, hyperparametry základních algoritmů, hyperparametry predikčních měr, hyperparametry kontaminace, hyperparametry meta-algoritmů, hyperparametry experimentů ap.

- **processing.py** - V tomto skriptu je naimplementován algoritmus předzpracování dat. Po čištění a předzpracování jsou datové soubory ve vhodném formátu překopírovány do složky *data* a podsložky příslušného experimentu. Pro přehledné odladʼování systému jsou tisknuty průběžné výsledky do konzole.

- **basic_features.py** - Tento skript generuje deset základních meta-příznaků (výše uvedených), které jsou napočteny z datových souborů.

- **basic_learning.py** - V tomto skriptu je provedeno základní učení na již předzpracované datové soubory. Pro přehlednou interpretaci výsledků a odladʼování systému jsou tisknuty průběžné výsledky do konzole. Běh tohoto skriptu trvá nejdéle ze všech. To kvůli výpočetní náročnosti CV, spousty datových souborů a devíti experimentům. Jedním z kritických hyperparametrů ovlivňující rychlost tohoto skriptu je počet iterací metody LWS. Jako výstupem je čtyřrozměrný tensor reziduí, který je dále vstupem do dalšího skriptu.

- **estimator_comparisson.py** - Funkce v tomto skriptu berou jako vstup čtyřrozměrný tenzor reziduí. Jsou zde spočteny tři druhy míry predikce a jejich velikost je porovnána a vždy nalezena vítězná metoda základního učení. Trojrozměrný tenzor kategorických proměnných vyjadřující vítězné metody slouží jako vstup do dalšího skriptu.

- **algorithm_selection.py** - Prvním vstupem tohoto skriptu jsou základní příznaky napočítané z datových souborů. Druhým vstupem je trojrozměrný tensor vítězných metod ze základního učení. V tomto skriptu je provedena klasifikace pomocí 10-násobné-CV. Pro přehlednou interpretaci výsledků a odladʼování systému jsou tisknuty průběžné výsledky do konzole. Ve výsledcích jsou v tabulkách srovnány všechny klasifikační metody, všechny míry kontaminace a míry predikce. Dále jsou zde důležité obrázky ukládány do cílové složky.

- **robust_methods.py** - Zde jsou implementovány robustní metody základního učení i nerobustní nejmenší čtverce.

[20] https://github.com/AlesNeoral/Diploma-Thesis/tree/master/kody. Zájemci pro stažení, nebo analýzu kódů musí zaslat žádost pro přístup. Repositáře mohou být veřejné jen pro předplacené verze.

[21] https://anaconda.org/

- **prediction_measures.py** - V tomto skriptu jsou implementovány všechny výše popsané tři míry predikce.
- **contaminate.py** - Zde jsou implementovány funkce umožňující přidat do odezvy buď lokální, nebo globální kontaminaci.
- **data_loading.py** - V tomto skriptu se nachází funkce, která načte všechny datové soubory do maticové proměnné, se kterou Python může pracovat.
- **multicollinearity.py** - Zde jsou postupně vyřazeny všechny multikolineární sloupce nezávislých proměnných, pokud multikolinearita, měřená variančním faktorem inflace, překročí předem stanovenou mez multikolinearity.
- **general_functions.py** - V tomto skriptu jsou obsaženy funkce umožňující uložit, nebo nahrát proměnnou na disk. Proměnné se ukládají do složky *saved_results*. Proměnné mohou být rezidua algoritmů základního učení, vypočtené míry predikce, vypočtené příznaky, nebo vypočtené meta-predikční míry pro meta-algoritmy. Ve vhodné si uložit na disk výsledky časově náročnějších výpočtů. Pokud zkoumáme například úspěšnost meta-algoritmů, je vhodné mít meta-data předpočítaná, a tím si ušetřit nějaký čas.
- **latex_tables.py** - Zde jsou implementovány funkce, které převedou některé z výsledků do tabulek LaTeXového formátu.
- **plots.py** - Tento skript slouží jako zdroj některých obrázků objevujících se v diplomové práci.
- **metalearning.py** - Tento skript slouží pro spuštění jednotlivých experimentů, aby experimentátor nemusel jednotlivé dílčí metody volat individuálně.

Kapitola 5

Experimenty

V této kapitole se pokusíme zhodnotit a kvantifikovat úspěšnost našeho navrženého systému metaučení pro automatickou selekci algoritmu. Celkem provedeme několik experimentů, na kterých otestujeme schopnost systému predikovat vhodný algoritmus pro tři typy měr predikce. Ověříme, jak se změní predikční schopnosti systému v závislosti na uměle přidané kontaminaci. Vyzkoušíme několik klasifikačních meta-algoritmů a pokusíme se zhodnotit jejich výsledky. Pomocí selekce vhodných meta-příznaků a techniky převzorkování se pokusíme vylepšit úspěšnost klasifikačních meta-algoritmů. V posledním experimentu otestujeme funkčnost metaučení pomocí náhodné matice.

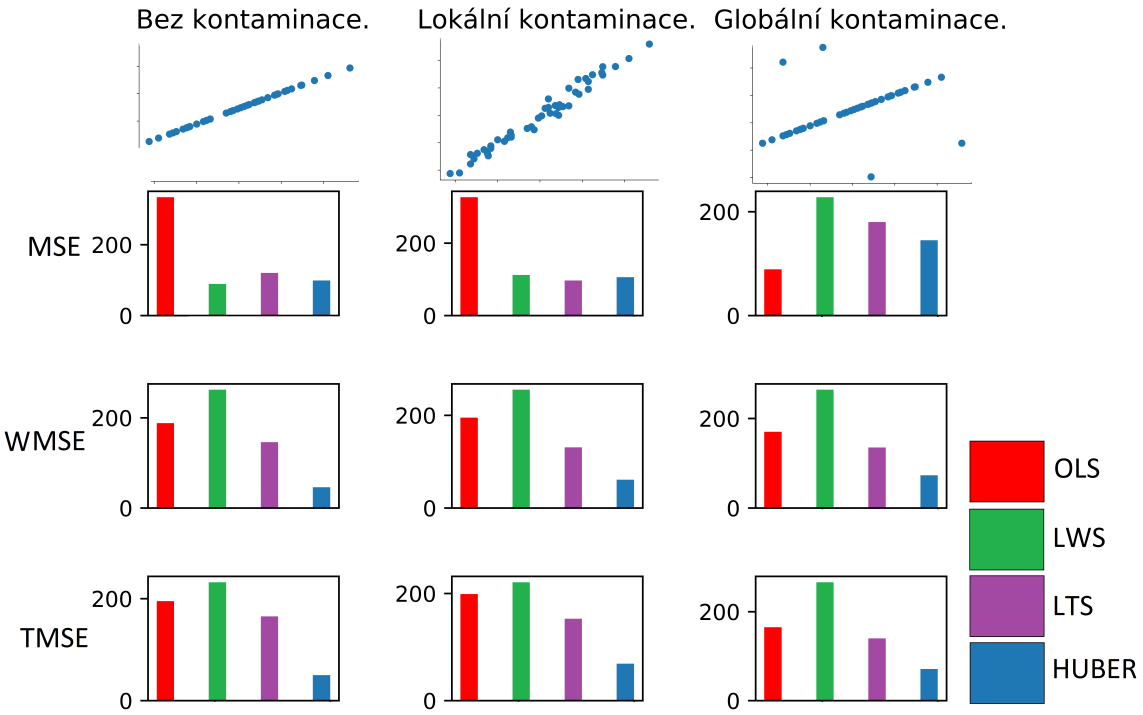
5.1 Základní učení

Ve fázi základního učení byla vyprodukována meta-odezva v podobě vektoru kategorické proměnné. Tento vektor vyjadřuje kategorie vítězných metod základního učení. Pro náhodně zvolené datové soubory můžeme část z tohoto vektoru vidět v tabulce 5.1. Tato tabulka obsahuje jednotlivé pozorování meta-odezvy pro různé experimentální realizace. Konkrétně pro experimenty s různými predikčními mírami a různými typy kontaminace. Z každého tohoto vektoru můžeme zkoumat početní zastoupení jednotlivých kategorií. Toto početní zastoupení můžeme vidět pro všechny kombinace v podobě histogramů na obrázku 5.1. V histogramech můžeme zhodnotit úspěšnosti základních algoritmů na základě jejich absolutní frekvence.

Frekvenční analýza základního učení

Pomocí frekvenční analýzy v základním učení můžeme vytvářet některé předpoklady pro následnou fázi selekce algoritmu. Tyto předpoklady se mohou hodit pro pozdější interpretaci výsledků systému metaučení. Z obrázku 5.1 si můžeme všimnout hned několika následujících skutečností.

- Metoda OLS naprosto dominuje pro experiment s původními daty bez kontaminace a při použití míry predikce MSE. To v praxi, kde se nejčastěji měří úspěšnost regresních metod pomocí MSE, znamená, že je nejvhodnější doporučit metodu OLS. Odhad OLS je ze své podstaty lineární odhad, který minimalizuje MSE. Proč tedy není četnost OLS 100 procent? Je to dáno tím, že se predikční míra počítá v 10-násobné-CV a OLS minimalizuje MSE na jiných datech, než na kterých je natrénováno.



Obrázek 5.1: Četnosti vítězných základních algoritmů v základním učení.

- Experimenty s lokální kontaminací se příliš neliší od experimentů bez kontaminace. Lokální kontaminace mírně zvýšila četnost méně úspěšného Huberova odhadu, a tím lehce snížila podíl většinové kategorie (viz tabulka 5.2).
- S globální mírou kontaminace výrazně klesá podíl zastoupení nerobustní OLS a roste podíl robustních metod.
- S použitím robustních měr predikce pro původní a lokálně kontaminovaná data klesá podíl OLS a roste podíl robustních metod.
- Četnost Huberova odhadu je až na první dva experimenty dominována četností LWS a LTS odhadu.
- Huberův odhad je až na první dva případy v řádku s MSE vždy dominován také četností LWS.
- Nejméně známý odhad LWS předčil veškerá očekávání a v počtu četností dokázal konkurovat ostatním metodám. V sedmi případech z devíti se stal dokonce nejčastější vítěznou metodou.
- Z tabulky 5.2 si můžeme všimnout, že podíl nejčastěji zastoupené kategorie je v meta-odezvě nižší buď při přidání globální kontaminace, nebo při použití robustních měr predikce.

	typ kontaminace		
	A	B	C
MSE	0.52	0.51	0.36
WMSE	0.41	0.4	0.41
TMSE	0.36	0.34	0.41

Tabulka 5.2: Podíl nejčastěji zastoupené kategorie vítězného základního algoritmu. (A) původní data, (B) lokální kontaminace, (C) globální kontaminace.

Podle experimentálních pokusů je pro následnou fázi selekce algoritmu velmi důležitý poměr kategorií v datech. Jak se později ukáže, selekce algoritmu funguje mnohem lépe pro rovnoměrnější rozložení kategorií. To například nastává pro experiment s mírou predikce TMSE a lokální kontaminaci, kdy je zastoupení většinové kategorie nejnižší ze všech provedených experimentů. Podíl většinové kategorie je 34 procent (viz tabulka 5.2). Naopak ze stejné tabulky a obrázku histogramů můžeme pozorovat dominantní převahu metody OLS pro experiment s predikční mírou MSE a původními daty bez kontaminace. Několika experimentálními pokusy, kde výsledky některých z nich jsou zobrazeny v této kapitole, jsme se přesvědčili, že rozložení poměru kategorií v meta-odezvě je klíčová charakteristika pro úspěšný meta-algoritmus. Čím více je toto rozložení nerovnoměrné a v meta-odezvě je převládající kategorie, tím více se z meta-algoritmu stává většinový klasifikátor. Tyhle poznatky budeme diskutovat v dalších částech této kapitoly.

5.2 Selekce algoritmu

Po základním učení zkoumáme úspěšnost metaučení pomocí klasifikační přesnosti (absolutní i relativní) na různých experimentech. Experimenty jsou identifikovány pomocí míry predikce a typu kontaminace. Pro každý experiment testujeme několik klasifikačních meta-algoritmů, aby jsme v případě potřeby mohli vybrat ten nejlepší. Tři realizované experimenty můžeme vidět například v tabulce 5.3, kde každý sloupec vyjadřuje jiný experiment (pro jiný typ kontaminace) a každý řádek vyjadřuje jiný klasifikační meta-algoritmus. Pro přehlednost budeme v tabulkách značit písmenem **A** data původní, písmenem **B** data lokálně kontaminovaná a písmenem **C** data globálně kontaminovaná. Tučným písmem jsou zvýrazněny nejlepší klasifikační výsledky pro každý druh kontaminace.

Tabulky napravo vyjadřují absolutní klasifikační přesnosti. Absolutní klasifikační přesnost znamená podíl správně klasifikovaných pozorování z celkového počtu pozorování. Tabulky nalevo vyjadřují relativní klasifikační přesnosti. Relativní klasifikační přesnost vyjadřuje podíl klasifikační přesnosti dané klasifikační metody ku klasifikační přesnosti většinového klasifikátoru. Relativní klasifikační přesnost větší než jedna znamená úspěšnější klasifikační metodu oproti většinovému klasifikačnímu pravidlu. Relativní klasifikační přesnost menší než jedna znamená horší klasifikační metodu než většinový klasifikátor.

Důvod přidání relativní klasifikační přesnosti je ten, že pokud jsou četnosti kategorií silně nevyvážené a je v nich přítomna převládající kategorie, klasifikátor může velmi snadno dosáhnout vysoké klasifikační přesnosti. Míra absolutní klasifikační přesnosti je u nevyvážených datových souborů velmi zavádějící, a proto se nedoporučuje měřit nevyvážené datové soubory pomocí klasifikační přesnosti. Pro první experimenty uvedeme tabulky absolutních klasifikačních přesností, ale dále se už budeme držet jen relativní klasifikační přesnosti, na které budeme vyhodnocovat výsledky klasifikace meta-algoritmů.

Základní experiment

Zanalyzujme si potupně výsledky metaučení pro míry MSE, WMSE a TMSE. Z tabulky experimentů pro MSE (viz tabulka 5.3) je zřejmé, že klasifikační meta-algoritmy nejsou lepší než většinový klasifikátor. I když meta-algoritmy překonaly v absolutním měřítku náhodný klasifikátor[1], tak mírně zaostaly za většinovým klasifikátorem. Pro základní experiment s MSE se metaučení nezdá být ovlivněno přidanou kontaminací.

Při experimentech s WMSE (viz tabulka 5.5) se o několik procent zlepšily klasifikační výsledky. Zejména u logistické regrese s oběma typy regularizace a LDA. Nepatrně za ostatními metodami zaostal SVM. Zajímavou skutečností je, že při přidání umělé kontaminace (zejména lokální) se zlepšují klasifikační výsledky.

Použití TMSE ještě více zlepšilo klasifikační výsledky než WMSE. Všechny klasifikační metody vykázaly na původních a lokálně kontaminovaných datech lepší výsledky, než většinové klasifikační pravidlo. To je důkaz toho, že meta-algoritmy se nerozhodují náhodně, ale meta-příznaky nesou určitou informaci o tom, který algoritmus základního učení si na daných datech povede nejlépe.

klasifikační metoda	typ kontaminace		
	A	B	C
většinový klasifikátor	**1.0**	1.0	**1.0**
k-NN (k = 50)	**1.0**	**1.01**	0.91
k-NN (k = 20)	0.96	**1.01**	0.87
SVM	**1.0**	1.0	0.97
L1 - logistická regrese	0.97	0.99	0.97
L2 - logistická regrese	0.97	0.99	0.94
LDA	0.96	0.96	0.97

Tabulka 5.3: Relativní klasifikační přesnost pro MSE. (A) původní data, (B) lokální kontaminace, (C) globální kontaminace.

klasifikační metoda	typ kontaminace		
	A	B	C
většinový klasifikátor	**0.52**	0.51	**0.36**
k-NN (k = 50)	**0.52**	**0.51**	0.32
k-NN (k = 20)	0.5	**0.51**	0.31
SVM	**0.52**	0.51	0.35
L1 - logistická regrese	0.51	0.5	0.35
L2 - logistická regrese	0.5	0.5	0.33
LDA	0.5	0.49	0.34

Tabulka 5.4: Absolutní klasifikační přesnost pro MSE. (A) původní data, (B) lokální kontaminace, (C) globální kontaminace.

[1] Náhodný klasifikátor se náhodně rozhoduje mezi jednotlivými kategoriemi. V tomto kontextu myslíme náhodný klasifikátor s uniformním rozdělením. V našem případě absolutní úspěšnost náhodného klasifikátoru je 25 procent, protože klasifikujeme do čtyřech kategorií.

klasifikační metoda	typ kontaminace		
	A	B	C
většinový klasifikátor	1.0	1.0	1.0
k-NN (k = 50)	0.94	1.15	0.94
k-NN (k = 20)	0.88	1.13	0.88
SVM	0.99	0.91	1.0
L1 - logistická regrese	**1.05**	**1.39**	1.0
L2 - logistická regrese	1.02	1.38	**1.02**
LDA	**1.05**	1.37	1.01

Tabulka 5.5: Relativní klasifikační přesnost pro WMSE. (A) původní data, (B) lokální kontaminace, (C) globální kontaminace.

klasifikační metoda	typ kontaminace		
	A	B	C
většinový klasifikátor	0.41	0.4	0.41
k-NN (k = 50)	0.38	0.45	0.39
k-NN (k = 20)	0.36	0.45	0.36
SVM	0.4	0.36	0.41
L1 - logistická regrese	**0.43**	**0.55**	0.41
L2 - logistická regrese	0.42	0.55	**0.42**
LDA	**0.43**	0.55	0.42

Tabulka 5.6: Absolutní klasifikační přesnost pro WMSE. (A) původní data, (B) lokální kontaminace, (C) globální kontaminace.

klasifikační metoda	typ kontaminace		
	A	B	C
většinový klasifikátor	1.0	1.0	1.0
k-NN (k = 50)	1.0	1.3	0.9
k-NN (k = 20)	1.03	1.2	0.92
SVM	1.02	1.09	1.0
L1 - logistická regrese	1.14	1.49	**1.04**
L2 - logistická regrese	1.09	**1.52**	**1.04**
LDA	**1.16**	1.49	0.98

Tabulka 5.7: Relativní klasifikační přesnost pro TMSE. (A) původní data, (B) lokální kontaminace, (C) globální kontaminace.

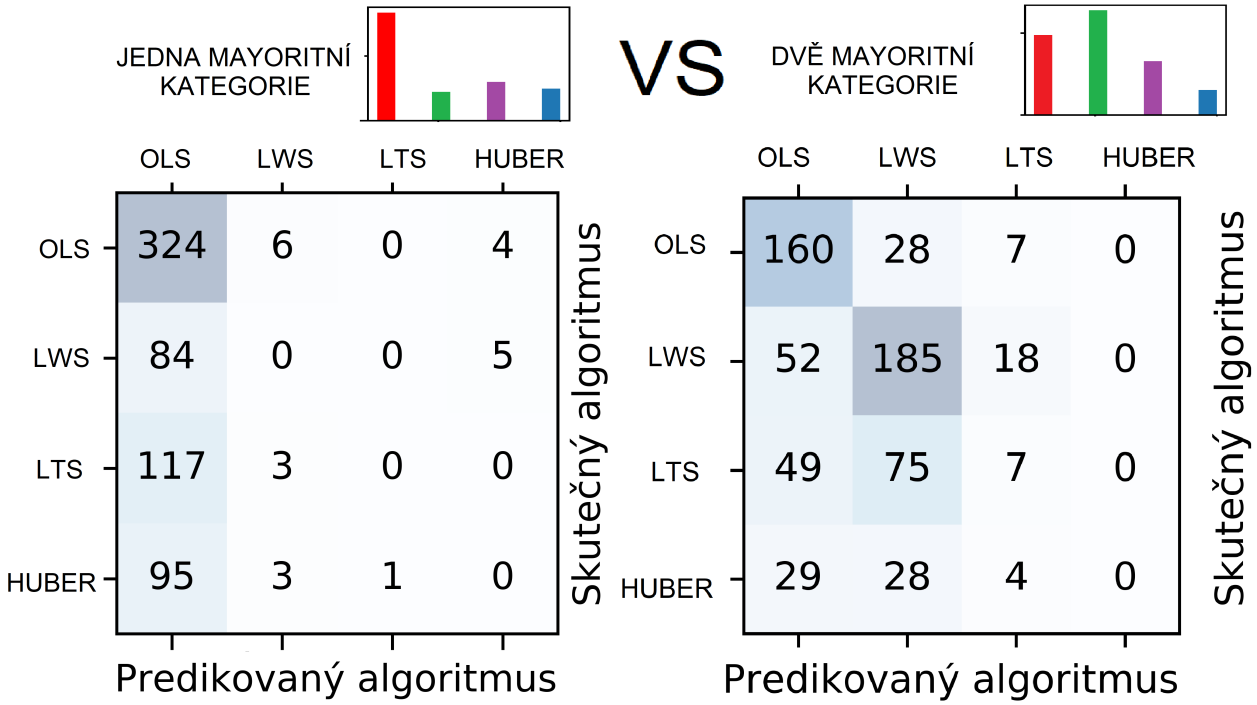
klasifikační metoda	typ kontaminace		
	A	B	C
většinový klasifikátor	0.36	0.34	0.41
k-NN (k = 50)	0.36	0.45	0.37
k-NN (k = 20)	0.37	0.41	0.38
SVM	0.37	0.37	0.41
L1 - logistická regrese	0.41	0.51	**0.43**
L2 - logistická regrese	0.4	**0.52**	**0.43**
LDA	**0.42**	0.51	0.41

Tabulka 5.8: Absolutní klasifikační přesnost pro TMSE. (A) původní data, (B) lokální kontaminace, (C) globální kontaminace.

Proč ale metaučení nedává lepší výsledky než většinový klasifikátor pro MSE? Tuto skutečnost jsme prozkoumávali pomocí několika počítačových experimentů. Po důkladné analýze jsme zjistili, že výsledky klasifikátoru jsou velmi silně ovlivněny četností klasifikačních kategorií. Pokud je v datech přítomna převládající kategorie, klasifikátor se snaží klasifikovat zejména do ní. Tento jev můžeme pozorovat na matici záměn na obrázku 5.2. Levá část obrázku ukazuje skutečnost, že při nerovnoměrném zastoupení kategorií klasifikuje klasifikátor logistické regrese zejména do většinové kategorie. Pro logistickou regresi a SVM je tento jev ještě zesílen strategií klasifikace.[2] Pokud je rozdělení kategorií rovnoměrnější, většinová kategorie nehraje tak velkou roli a klasifikátor se musí rozhodovat na základě příznaků.

Při experimentu zobrazeném v levé části obrázku 5.2 dosáhla logistická regrese 97 procentní úspěšnosti většinového klasifikátoru. Proto je v tomto případě zbytečné se pomocí logistické regrese rozhodovat o výběru základního algoritmu. Při experimentu zobrazeném v pravé části obrázku 5.2 dosáhla logistická regrese 152 procentní úspěšnosti většinového klasifikátoru. Znamená to, že v tomto případě

[2]SVM a logistická regrese používají pro klasifikaci strategii jeden versus všechny kategorie. Pro minoritní kategorii (např. Huberův odhad) se při použití této strategie tento poměr ještě zvýší, kdy klasifikujeme minoritní kategorii proti všem kategoriím.

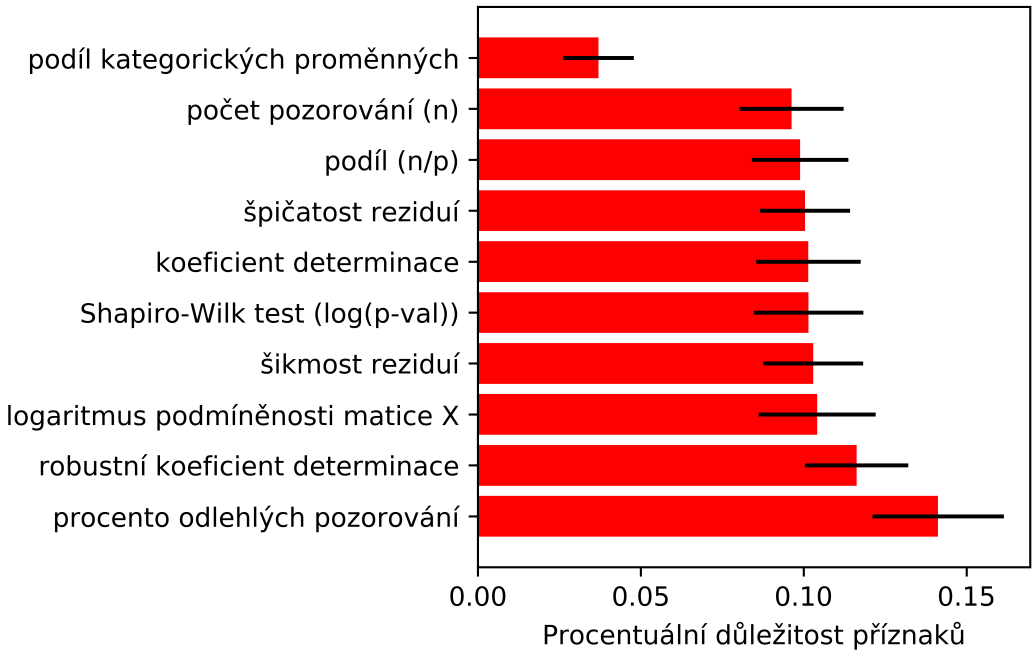


Obrázek 5.2: Rozdíl úspěšnosti klasifikace logistické regrese v závislosti na poměru zastoupení kategorií. Nevyvážené kategorie (vlevo), vyváženější (vpravo) velmi ovlivňují klasifikaci.

logistická regrese na základě příznaků datového souboru dokázala predikovat nejlepší základní algoritmus o 52 procent lépe než většinový klasifikátor, a tedy dokázala využít informaci v příznacích k predikci základního algoritmu. Proč ale v tomto případě klasifikuje logistická regrese zejména do dvou kategorií? Příčinou jsou dvě mayoritní kategorie. Logistická regrese se naučila rozlišovat mezi robustní metodou LWS (zeleně) a nerobustní OLS (červeně). Tento problém se budeme snažit atakovat pomocí tzv. převzorkování, kdy uměle vytvoříme nová pozorování, abychom vyrovnali poměr kategorií. Skutečnost, kdy klasifikátor je výrazně ovlivněn poměrem kategorií, jsme pozorovali nezávisle na sobě pro několik různých klasifikátorů a pro všechny provedené experimenty. Obrázek 5.2 je pouze ilustrací námi pozorovaného jevu.

Zajímavým fenoménem je, že přidáním kontaminace (zejména lokální) se relativní úspěšnost klasifikátoru zlepšuje. Tento paradox si vysvětlujeme následující úvahou. Přidaná kontaminace zredukuje podíl majoritní kategorie a zrovnoměrní rozložení kategorií (viz tabulka 5.2). Klasifikátor se potom nemůže spoléhat na hlas většiny, ale musí se rozhodovat podle příznaků. Meta-odezva není zjevně pro lokální kontaminaci do značné míry ovlivněna (viz druhý sloupec tabulek 5.5 a 5.7). Proto je dosaženo vyšších klasifikačních výsledků než u většinového klasifikátoru. Pro globální kontaminaci je zjevně meta-odezva ovlivněna mnohem více (viz třetí sloupec tabulek 5.5 a 5.7).

Zjistili jsme, že klasifikátor je ovlivněn dvěma jevy. První jev, poměr kategorií, se budeme snažit vylepšit v experimentech s převzorkováním. Druhým jevem je síla příznaků. K tomu abychom zvýšili úspěšnost klasifikátoru, musíme zvýšit sílu příznaků. To můžeme docílit buď výběrem nové množiny příznaků, nebo selekcí několika nejdůležitějších příznaků ze stávající množiny. Ve výše popsaných experimentech mohlo v deseti-rozměrném prostoru příznaků docházet k prokletí dimensionality a klasifikátory mohly trpět přeučením. Proto jsme se rozhodli tento problém řešit. Vhodnou množinu příznaků pro metaučení budeme vybírat v experimentech se selekcí příznaků.



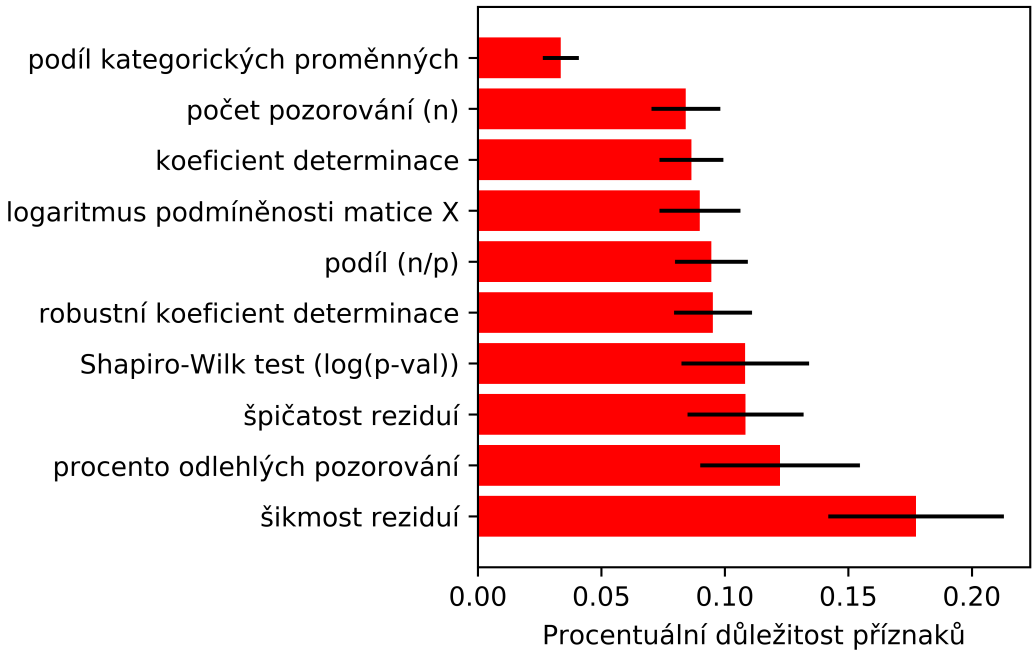
Obrázek 5.3: Důležitost meta-příznaků pro experiment MSE a lokální kontaminací.

Selekce příznaků

V těchto experimentech se pokusíme vybrat jen ty nejvhodnější příznaky a provést učení na meta-úrovni jen s nimi. Důležitost příznaků jsme počítali metodou zpětného hledání, ve které jsme jako evaluační algoritmus použili náhodný les. V každé iteraci zpětného hledání byl náhodný les natrénován na trénovacích datech (podmnožina meta-příznaků a meta-odezvy). Pro natrénovaný model byla spočtena chyba CV na testovacích datech. Množina příznaků, která vykazovala na testovacích datech nejmenší chybu, byla vybrána jako množina příznaků pro klasifikační meta-algoritmus. Množiny příznaků se při každém experimentu lehce lišily, i když základní pořadí a rysy zůstávaly napříč různými experimenty stejné. My jsme pro ilustraci vybrali srovnání důležitosti příznaků pro experiment MSE s lokální kontaminací (viz obrázek 5.3) a experiment s TMSE a lokální kontaminací (viz obrázek 5.4). U obout těchto experimentů bylo vybráno 6 nejdůležitějších meta-příznaků pro meta-algoritmus. Typicky se hodnota optimálního počtu příznaků pohybovala mezi 4 a 8.
V následujících bodech se pokusíme zamyslet nad tím, jakou roli mohou mít jednotlivé meta-příznaky volené v této práci, a zkusit teoretizovat nad jejich významem.

- **procento odlehlých pozorování**

  Jako nevýznamnější příznak se podle obrázku 5.3 jeví procento odlehlých pozorování. Tento příznak jsme v této práci volili úmyslně s nadějí, že bude rozlišovat mezi nerobustní OLS a robustními metodami. Pravděpodobně se naše hypotéza potvrdila, což může dokládat i obrázek 5.4. Vyšší procento by mohlo favorizovat robustnější metody. Robustní metody lépe proloží vysoce kontaminovaná data než nerobustní metody. Odhad robustních metod není tak citlivý na odlehlá pozorování, a proto mohou v CV dosáhnout nižší chyby a na datovém souboru vyhrát.



Obrázek 5.4: Důležitost meta-příznaků pro experiment TMSE a lokální kontaminací.

- **robustní koeficient determinace**

  Robustní koeficient determinace jsme znovu volili úmyslně, aby odlišoval mezi robustními a neurobustními metodami. Vyšší hodnoty tohoto koeficientu udávají vyšší úspěšnost robustní lineární regrese, a tedy její vhodnost pro daná data. Nižší hodnoty robustního koeficientu determinace by mohly favorizovat OLS.

- **šikmost reziduí**

  Šikmost byla napočítána z reziduí OLS odhadu. Kladná šikmost značí, že vpravo od průměru se vyskytují odlehlejší hodnoty než vlevo. U záporné šikmosti je tomu naopak. Hodnoty šikmosti by mohly být indikátorem vhodnosti OLS odhadu.

- **špičatost reziduí**

  Špičatost porovnává rozdělení s normálním. Normální rozdělení má špičatost rovnou nule. Kladná špičatost znamená špičatější rozdělení než normální. Záporná špičatost znamená rovnoměrnější rozdělení než normální. Pokud se špičatost hodně liší od nuly, může to být indikátor nesplněné normality reziduí, která je nutnou podmínkou lineární regrese. Jelikož byla špičatost napočítána z reziduí pocházejících z OLS odhadu, velké absolutní hodnoty špičatosti mohou diskriminovat OLS.

- **normalita reziduí**

  Nízká p-hodnota normality reziduí by mohla favorizovat metody lineární regrese, u kterých je normalita reziduí nutným předpokladem. P-hodnota normality se typicky pohybuje v exponenciální škále. Proto jsme ji logaritmem transformovali do lineární škály.

- **koeficient determinace**

  Vyšší hodnoty koeficientu determinace udávají vyšší úspěšnost lineární regrese, a tedy její vhodnost pro daná data. Nižší hodnoty koeficientu determinace by mohly favorizovat jiné metody. Koeficient determinace byl napočítán z reziduí pocházejících z OLS odhadu, proto by mohla jeho hodnota odlišovat mezi OLS a ostatními metodami.

- **podíl počtu pozorování a vysvětlujících proměnných**

  Podle literatury odlišuje tento příznak mezi metodami, které jsou náchylné na prokletí dimensionality a mezi metodami, které jsou na něj náchylné méně. Podíl počtu pozorování a vysvětlujících proměnných datového souboru totiž měří náchylnost algoritmu k přeučení.

- **podmíněnost matice vysvětlující proměnných**

  Tato hodnota může odlišovat OLS od ostatních metod, neboť se v odhadu OLS počítá inverzní matice vysvětlujících proměnných. Pro výpočet inverzní matice musí být původní matice regulární. Míru regularity měříme pomocí tohoto příznaku. Typicky je hodnota podmíněnosti matice pohybuje v exponenciální škále. Aplikací logaritmu transformujeme tuto hodnotu do lineární škály.

- **počet pozorování**

  Hodnota počtu pozorování by mohla rozlišovat mezi robustními a nerobustními metodami. Nerobustní metody potřebují pro dobrý odhad méně pozorování než robustní metody a jsou tzv. eficientnější.

- **podíl kategorických a všech proměnných**

  Tento příznak určuje vhodnost datového souboru pro klasifikaci. Mohl by rozlišovat mezi klasifikačními a regresními metodami základního učení. My ale požíváme v základním učení jen regresní metody, proto se jeho diskriminační vlastnost nemohla uplatnit. To se nicméně potvrzuje i na obrázcích 5.3 a 5.4. Navíc v našich experimentech nemá takovou variabilitu (viz poslední sloupec tabulky 5.1).

S vybranými příznaky jsme provedli znova několik experimentů a zkoumali jsme úspěšnost meta-algoritmů. Jako klasifikační metriku meta-algoritmů budeme uvažovat pouze relativní klasifikační přesnost, protože zkoumáme stále nevyvážená data a absolutní klasifikační přesnost je zavádějící.

Pro experimenty s predikční mírou MSE se relativní úspěšnosti meta-algoritmů nijak výrazně nezměnily (viz tabulka 5.9). To je dáno nejspíše stále převládající kategorií. Zatím ani selekce příznaků nepomohla. Rozložení kategorií je v tomto případě pro meta-algoritmus daleko důležitější než síla meta-příznaků. To ale neplatí pro experimenty v tabulce 5.10, ve kterých je rovnoměrnější rozložení kategorií. Oproti předchozímu ekvivalentnímu experimentu bez selekce příznaků se relativní klasifikační přesnost meta-algoritmů zvýšila o desítky procent. Díky tomu můžeme považovat selekci příznaků za úspěch.

Zajímavou skutečnost můžeme pozorovat v druhém sloupci tabulky 5.10 pro experiment s lokálně kontaminovanými daty. Oproti předchozímu experimentu bez selekce příznaků relativní úspěšnost klasifikačních meta-algoritmů poklesla. Je pravděpodobné, že v procesu selekce příznaků vyřadila metoda náhodného lesu některé důležité příznaky právě pro experiment s mírou WMSE a lokálně kontaminovaná data. Pro selekci příznaků používá metoda náhodného lesu svoji vlastní klasifikační metriku. Výsledky experimentů s globální kontaminací se podle očekávání výrazně nezměnily.

Pro experimenty s predikční mírou TMSE (viz tabulka 5.11) se výsledky zlepšily oproti WMSE. Pro původní data a data s uměle přidanou globální kontaminací se relativní klasifikační přesnosti zlepšily jen mírně, u lokální kontaminace se oproti předchozímu WMSE zlepšily výsledky o desítky procent. To

by potvrzovalo předchozí hypotézu, že v selekci příznaků se pro WMSE a lokální kontaminaci vyřadil nějaký důležitý příznak.

klasifikační metoda	typ kontaminace		
	A	B	C
většinový klasifikátor	**1.0**	**1.0**	**1.0**
k-NN (k = 50)	0.98	0.99	0.92
k-NN (k = 20)	0.95	0.95	0.87
SVM	**1.0**	**1.0**	0.97
L1 - logistická regrese	**1.0**	**1.0**	0.99
L2 - logistická regrese	**1.0**	**1.0**	0.98
LDA	0.99	**1.0**	0.94

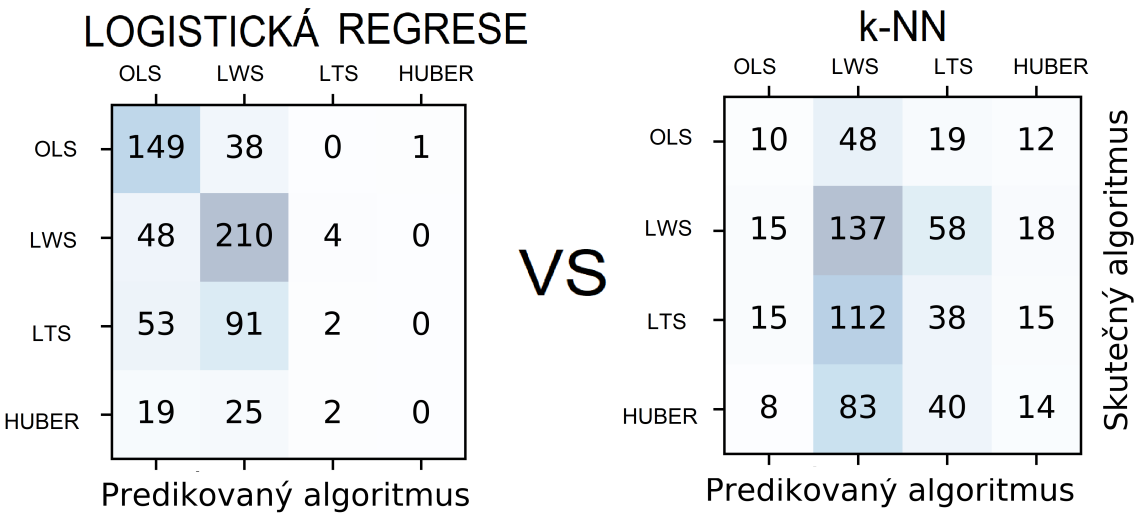
Tabulka 5.9: Relativní klasifikační přesnost pro MSE po selekci příznaků. (A) původní data, (B) lokální kontaminace, (C) globální kontaminace.

klasifikační metoda	typ kontaminace		
	A	B	C
většinový klasifikátor	1.0	1.0	1.0
k-NN (k = 50)	1.21	0.98	0.9
k-NN (k = 20)	1.23	0.92	0.84
SVM	1.27	0.95	1.0
L1 - logistická regrese	**1.38**	1.0	1.0
L2 - logistická regrese	1.37	1.01	**1.01**
LDA	1.29	**1.04**	0.97

Tabulka 5.10: Relativní klasifikační přesnost pro WMSE po selekci příznaků. (A) původní data, (B) lokální kontaminace, (C) globální kontaminace.

klasifikační metoda	typ kontaminace		
	A	B	C
většinový klasifikátor	1.0	1.0	1.0
k-NN (k = 50)	1.26	1.06	0.99
k-NN (k = 20)	1.27	1.0	0.94
SVM	1.35	1.0	1.0
L1 - logistická regrese	**1.42**	**1.19**	**1.03**
L2 - logistická regrese	1.41	1.13	1.01
LDA	1.3	1.18	0.97

Tabulka 5.11: Relativní klasifikační přesnost pro TMSE po selekci příznaků. (A) původní data, (B) lokální kontaminace, (C) globální kontaminace.



Obrázek 5.5: Matice záměn pro experimenty se selekcí příznaků.

Pojďme se nyní podívat na matice záměn vybraných klasifikátorů pro některé experimenty. Z tabulky 5.10 si vybíráme nejlepší klasifikátor (logistickou regresi s L1 regularizací) pro původní data. Srovnejme ho s relativně úspěšným klasifikátorem k-NN ($k = 20$) pro TMSE a původní data. Relativní klasifikační úspěšnost logistické regrese je o 38 procent vyšší a metody k-NN o 23 procent vyšší než většinový klasifikátor. Srovnání obou klasifikátorů můžeme vidět na obrázku 5.5. Klasifikátor k-NN se snaží predikovat do všech kategorií. To mu jde výrazně lépe, neboť ke klasifikaci využívá metrické vzdálenosti a nepoužívá strategii jedna vs všechny kategorie jako logistická regrese (nalevo). Stále je tedy zřejmé, že logistická regrese se drží dvou významných kategorií a není schopna predikovat do kategorií LTS a Huber. I když metoda selekce příznaků výrazně vylepšila predikční schopnosti meta-algoritmů, tak nezměnila jejich nerovnoměrnou klasifikaci do kategorií základních algoritmů. Tento problém se budeme snažit vyřešit pomocí převzorkování.

Selekce příznaků s převzorkováním

I když meta-algoritmy dokáží predikovat základní algoritmy mnohem lépe než většinový klasifikátor, často se rozhodují mezi dvěma početně výraznými kategoriemi. V těchto experimentech se budeme snažit vylepšit klasifikační schopnosti pomocí standardně používané techniky převzorkování. Technikou převzorkování se pokusíme zeslabit vliv převládající kategorie, a tím donutit klasifikátory se rozhodovat více na základě příznaků.

Technika převzorkování datového souboru se standardně používá v aplikacích, při kterých jsou k dispozici nevyvážená data. Může jít o bankovní podvody, emailové spamy a další vzácné jevy. K méně zastoupeným kategoriím se pro blízká pozorování nageneruje nové pozorování v náhodném bodě na jejich spojnici. Tímto generováním je možné dosáhnout rovnoměrnějšího zastoupení kategorií. My jsme se rozhodli převzorkovat data tak, aby v každé kategorii byl stejný počet pozorování (viz tabulka 5.12). V experimentech s převzorkováním používáme stále metodu selekce příznaků. Tím doufáme, že úspěšnost klasifikátorů vzroste jednak díky selekci příznaků, a také díky převzorkování. Atakujeme tedy oba dva problémy nastíněné v základních experimentech.

	typ kontaminace		
	A	B	C
MSE	0.25	0.25	0.25
WMSE	0.25	0.25	0.25
TMSE	0.25	0.25	0.25

Tabulka 5.12: Zastoupení většinové kategorie po převzorkování.

klasifikační metoda	typ kontaminace		
	A	B	C
většinový klasifikátor	1.0	1.0	1.0
k-NN (k = 50)	1.76	1.67	1.63
k-NN (k = 20)	**1.87**	**1.84**	**1.87**
SVM	1.19	1.31	1.17
L1 - logistická regrese	1.2	1.26	1.32
L2 - logistická regrese	1.18	1.26	1.26
LDA	1.17	1.26	1.33

Tabulka 5.13: Relativní klasifikační přesnost pro MSE po převzorkování. (A) původní data, (B) lokální kontaminace, (C) globální kontaminace.

klasifikační metoda	typ kontaminace		
	A	B	C
většinový klasifikátor	1.0	1.0	1.0
k-NN (k = 50)	2.16	1.7	1.79
k-NN (k = 20)	**2.38**	**2.09**	**2.03**
SVM	2.05	1.26	1.2
L1 - logistická regrese	1.77	1.65	1.58
L2 - logistická regrese	1.77	1.57	1.56
LDA	1.76	1.62	1.55

Tabulka 5.14: Relativní klasifikační přesnost pro WMSE po převzorkování. (A) původní data, (B) lokální kontaminace, (C) globální kontaminace.

klasifikační metoda	typ kontaminace		
	A	B	C
většinový klasifikátor	1.0	1.0	1.0
k-NN (k = 50)	2.08	1.77	2.08
k-NN (k = 20)	**2.35**	**1.96**	**2.29**
SVM	2.15	1.43	1.69
L1 - logistická regrese	1.88	1.65	1.6
L2 - logistická regrese	1.88	1.55	1.57
LDA	1.72	1.6	1.61

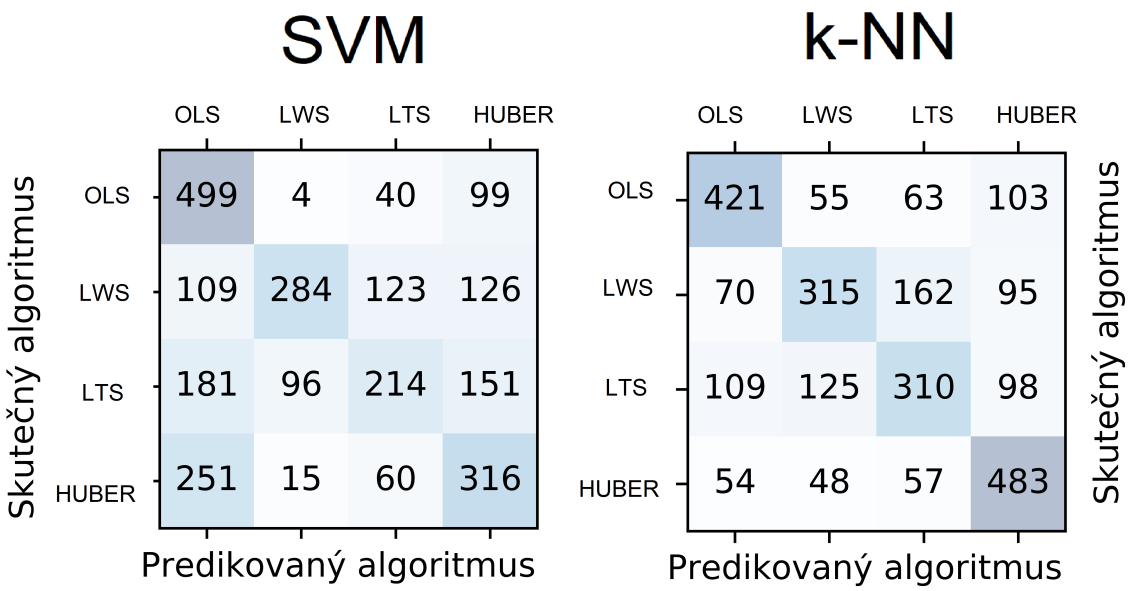
Tabulka 5.15: Relativní klasifikační přesnost pro TMSE po převzorkování. (A) původní data, (B) lokální kontaminace, (C) globální kontaminace.

V experimentech se selekcí příznaků a převzorkováním dosahujeme až dvojnásobných klasifikačních přesností než většinový klasifikátor. Meta-algoritmy jsou úspěšné dokonce i pro experimenty s MSE (viz tabulka 5.13). Tím se nám jen potvrzuje hypotéza, že úspěšnost meta-algoritmů velmi závisí na většinové kategorii. Meta-algoritmy vykazují ještě lepší výsledky pro experimenty s WMSE a TMSE (viz tabulky 5.14 a 5.15).

Překvapením je, že i pro globální kontaminaci se úspěšnost meta-algoritmů výrazně zvýšila. To může být způsobeno navzorkováním Huberova odhadu, který se podle frekvenční analýzy příliš neuplatnil pro původní a lokálně kontaminovaná data a jeho četnost vzrostla pro globální kontaminaci.

Ze všech tří tabulek je jasně vidět, že metoda *k*-NN dominuje pro všechny experimenty. To ale není žádným velkým překvapením. Při generování nových pozorování se převzorkováním vytváří nové vzorky mezi nejbližší sousedy. Tím pádem vznikne několik oddělených clusterů, ve kterých pak *k*-NN měřením vzdálenosti může ostatní meta-algoritmy dominovat. Podle literatury může někdy při aplikacích převzorkování docházet k přeučení. Proto tyto výsledky budeme brát s rezervou. Často se nepřevzorkovává na rovnoměrný poměr kategorií. My jsme ale převzorkovali rovnoměrně, protože v naší klasifikaci mají meta-příznaky slabou sílu. V praxi se používá i podvzorkování, kdy jsou některé pozorování vyhozena z většinové kategorie. Podvzorkováním může zase docházet k podučení, proto se často používá kombinace převzorkování a podvzorkování.

Podívejme se nyní na matici záměn pro experiment s převzorkováním a vybrané klasifikátory. Srovnání metod SVM a *k*-NN můžeme vidět na obrázku 5.6. Predikce klasifikátorů nyní vypadají mnohem lépe, než při předchozích experimentech. Obrázek zobrazuje predikce metod SVM a *k*-NN pro původní data měřená mírou WMSE. SVM dosahuje relativní klasifikační přesnosti 205 procent a *k*-NN 238 procent většinového klasifikátoru. Často se udává úspěšnost klasifikátoru také pomocí F1-míry. F1 míra je harmonickým průměrem přesnosti a úplnosti klasifikátoru. F1-míra pro SVM je 0.5 a pro *k*-NN 0.59 (viz tabulka 5.16).



Obrázek 5.6: Matice záměn pro experiment se selekcí příznaků a převzorkováním.

klasifikační metoda	typ kontaminace		
	A	B	C
většinový klasifikátor	0.0	0.0	0.0
k-NN (k = 50)	0.53	0.42	0.44
k-NN (k = 20)	**0.59**	**0.52**	**0.51**
SVM	0.5	0.28	0.26
L1 - logistická regrese	0.41	0.41	0.39
L2 - logistická regrese	0.41	0.39	0.39
LDA	0.43	0.4	0.39

Tabulka 5.16: F1 míra pro experimenty WMSE po převzorkování.

Ověření úspěšnosti pomocí náhodné matice

Jelikož se chceme přesvědčit o pozitivních výsledcích metaučení, napadlo nás zrealizovat experimenty s náhodnou maticí příznaků. Pokud jsou meta-algoritmy úspěšné, a jsou tedy schopny na základě skutečných meta-příznaků rozhodnout, která metoda základního učení bude pro nová data fungovat nejlépe, tak by měly na základě uměle nagenerovaných falešných příznaků selhat. Následujícími experimetny ověříme, zda předchozí úspěšné experimenty nejsou jenom náhodné fluktuace. Zjistíme, jestli opravdu meta-příznaky nesou nějakou informaci, která předurčuje nejlepší metodu pro datový soubor.

Hodnoty náhodné matice jsou generována z normálního rozdělení. Meta-odezvu jsme samozřejmě ponechali stejnou, jako v předchozích experimentech. Pokud má platit naše hypotéza, očekáváme, že relativní klasifikační přesnosti při použití náhodné matice meta-příznaků nepřekročí jedničku. Opravdu se tak prokázalo, což ostatně můžeme vidět v tabulkách 5.17, 5.18 a 5.19. Při náhodném experimentu nezvýrazňujeme tučně nejlepší úspěšnosti klasifikátorů, protože naším cílem není předpovídat základní

algoritmy z náhodné matice. Navíc odchylky od jedničky jsou nejspíše náhodné fluktuace kolem většinového klasifikátoru.

klasifikační metoda	typ kontaminace		
	A	B	C
většinový klasifikátor	1.0	1.0	1.0
k-NN (k = 50)	1.0	1.0	1.0
k-NN (k = 20)	0.97	0.97	0.94
SVM	1.0	1.0	0.94
L1 - logistická regrese	1.0	1.0	0.99
L2 - logistická regrese	1.0	1.0	0.96
LDA	1.0	1.0	0.92

Tabulka 5.17: Relativní klasifikační úspěšnost náhodné matice příznaků pro MSE.

klasifikační metoda	typ kontaminace		
	A	B	C
většinový klasifikátor	1.0	1.0	1.0
k-NN (k = 50)	0.89	0.96	0.98
k-NN (k = 20)	0.85	0.91	0.95
SVM	0.98	0.91	0.98
L1 - logistická regrese	0.98	0.95	1.02
L2 - logistická regrese	0.96	0.96	1.02
LDA	0.95	0.95	0.99

Tabulka 5.18: Relativní klasifikační úspěšnost náhodné matice příznaků pro WMSE.

klasifikační metoda	typ kontaminace		
	A	B	C
většinový klasifikátor	1.0	1.0	1.0
k-NN (k = 50)	0.85	0.9	0.94
k-NN (k = 20)	0.92	0.88	0.85
SVM	0.82	0.91	0.95
L1 - logistická regrese	0.84	0.98	0.97
L2 - logistická regrese	0.84	1.02	0.94
LDA	0.86	1.03	0.92

Tabulka 5.19: Relativní klasifikační úspěšnost náhodné matice příznaků pro TMSE.

Zajímavým fenoménem je výrazně nižší relativní klasifikační přesnost pro náhodné matice (zejména tabulka 5.18 a 5.19). Relativní klasifikační přesnost je nižší hlavně pro experimenty, kde s reálnou maticí meta-příznaků vykazovalo metaučení pozitivní výsledky. Tento jev nejspíše nastává znovu z větší vyvážeností kategorií. Poměr kategorií pro meta-algoritmus hraje nižší roli a meta-algoritmus se snaží rozhodovat na základě úplně zbytečných příznaků, což z něho dělá náhodný klasifikátor. Pokusy s náhodnou maticí jen utvrzují naše teorie a můžeme usoudit, že náš systém pro automatickou selekci algoritmu funguje!

5.3 Diskuse

Klíčovým parametrem úspěšného systému pro automatickou selekci algoritmu jsou zvolené příznaky. Pokud tyto příznaky nemají dostatečnou sílu, klasifikační meta-algoritmy nejsou schopny doporučit nejlepší základní algoritmus. Podle některých autorů je velmi těžké nalézt vhodné příznaky pro metaučení. Na základě několika experimentů se domníváme, že jsme pro metaučení vybrali příznaky, které dokáží diskriminovat mezi jednotlivými datovými soubory. Tyto příznaky pravděpodobně nejsou velmi silně korelovány s meta-odezvou, ale existuje jistá slabá korelace mezi námi zvolenými meta-příznaky a meta-odezvou. Byli jsme schopni naučit klasifikační meta-algoritmy na meta-úrovni závislosti mezi meta-příznaky a meta-odezvou. O tom jsme se utvrdili v experimentu s náhodnou maticí, který doporučujeme použít pro ověření úspěšnosti metaučení. Pro zlepšení diskriminační síly příznaků doporučujeme analyzovat důležitost příznaků pomocí metod selekce příznaků a vybrat jen ty nejvíce korelované s meta-odezvou. Dále doporučujeme analyzovat rozložení poměru kategorií v meta-odezvě a při vysoce nevyvážených poměrech použít techniky převzorkování, nebo podvzorkování datového souboru.

Výběr správných příznaků je velmi úzce provázán se zkoumanými základními metodami. V naší studii se nám úspěšně podařilo diskriminovat mezi vybranou množinou základních metod lineární regrese na základě vybraných příznaků. Pro rozlišení robustních metod jsme vybrali robustní příznaky. Robustní příznaky počtu odlehlých pozorování a robustní koeficient determinace se v metodě selekce příznaků ukázaly jako významné. Podle matic záměn mohou rozlišovat zejména mezi OLS a LWS. Dalšími příznaky, které se ukázaly jako významné, byly šikmost a špičatost reziduí z OLS odhadu. Tyto příznaky mohou vyjadřovat stupeň vhodnosti metody OLS. Systém pro automatickou selekci algoritmu jsme navrhli tak, aby bylo možné na základě jednotlivých příznaků rozlišovat mezi metodami základního učení. Příznaky rozlišují mezi jednotlivými robustními metodami a mezi robustní a nerobustní skupinou metod. Obecně je důležité, aby rozsah metod v základním učení byl co nejpestřejší a různé metody byly vhodné pro různé problémy. Každý z vybraných příznaků by měl favorizovat jednu z metod základního učení.

Jedním z důležitých vlastností systému pro automatickou selekci algoritmu je jeho citlivost na kontaminaci. V experimentální části jsme ukázali, že metaučení je na uměle přidanou kontaminaci v datech citlivé. Při přidání mírné lokální kontaminace mohou v některých případech relativní klasifikační výsledky paradoxně růst. To je dáno snížením poměru většinové kategorie v meta-odezvě. Tím se donutí klasifikační meta-algoritmus přihlížet více na příznaky, a ne tolik na většinovou kategorii. Poměr většinové kategorie se také výrazně sníží přidání globální kontaminace. Metaučení ale při globální kontaminaci nevykazuje lepší výsledky než většinový klasifikátor. Silná globální kontaminace nejspíše poškodila datové soubory natolik, že to výrazně ovlivnilo meta-odezvu. Proto doporučujeme dalším studiím se globálně kontaminovaným datům vyhnout.

Studii metaučení jsme provedli pro tři různé predikční míry (MSE, WMSE a TMSE). Pro standardní míru MSE nevykazuje metaučení lepší výsledky než většinový klasifikátor. To je dáno návrhem našeho systému, který nám způsobil nerovnoměrnost kategorií. MSE totiž favorizuje odhad OLS, který chybu MSE na trénovacích datech minimalizuje. Proto byla velká šance, že bude chybu MSE minimalizovat i na testovacích datech a zvítězí mezi základními metodami. Při použití MSE doporučujeme se odhadu OLS

vyhnout, nebo do základních metod přidat jinou konkurující metodu. Při použití predikčních měr WMSE a TMSE už nebylo tak jasné, jaká metoda by mohla vyhrát, a proto rozložení kategorií meta-odezvy bylo rovnoměrnější. Rovnoměrnější rozdělení způsobilo mnohem lepší predikci základního algoritmu pro nový datový soubor než většinový klasifikátor. Predikci doporučujeme měřit pomocí relativní klasifikační přesnosti, protože při použití absolutní klasifikační přesnosti se můžeme dopustit mylných výsledků.

Výsledky metaučení by se pravděpodobně daly vylepšit lepším předzpracováním datových souborů. Při automatickém předzpracování datových souborů může docházet k výrazné změně jejich příznaků. Na druhou stranu se zároveň změní i algoritmus, který je na tomto novém datovém souboru nejlepší. To může znamenat jen přesunutí meta-pozorování (datového souboru) z jednoho clusteru do druhého. Klasifikační výsledky by se mohly také zvýšit prozkoumáním větší množiny metod strojového učení na základní úrovni i meta-úrovni. Pro úspěšné metaučení doporučujeme zůstat u většího množství souborů. To je vhodné zejména při větším množství příznaků, kde hrozí prokletí dimensionality. Pro stažení většího množství datových souborů bychom doporučili použít meta-databázi OpenML.

	míra predikce a typ kontaminace								
	MSE			WMSE			TMSE		
datový soubor	A	B	C	A	B	C	A	B	C
bcdeter	1	1	1	3	3	3	2	3	3
bigcity	3	3	3	3	2	2	2	1	1
biomassTill	3	3	3	3	2	2	1	1	1
bmt	3	3	3	3	3	3	1	2	2
Bollen	2	2	2	1	3	3	3	3	3
cd4.nested	3	2	2	3	2	2	2	3	3
ChildSpeaks	1	1	1	1	1	1	1	3	3
chorSub	1	1	1	2	1	1	3	2	4
cloth	1	4	4	1	1	1	2	4	4
Clothing	1	2	2	1	4	4	2	1	1
CloudSeeding2	1	2	2	1	2	2	4	2	2
codling	1	2	2	1	2	3	3	2	2
CYGOB1	1	2	2	1	3	3	2	3	3
delivery	2	3	3	2	2	2	3	3	2
downs.bc	1	2	2	1	2	2	2	2	2
drughiv	4	3	3	2	2	2	1	1	1
elastic2	3	2	2	4	2	2	2	1	1
elasticband	1	1	1	1	1	1	3	1	1
FARSmiss	1	2	2	1	2	2	2	1	1
Fertility	1	2	2	1	2	2	1	1	1
FinalFourIzzo	3	2	2	2	2	2	4	2	2
FinalFourShort	3	3	4	3	3	3	3	3	3
grav	1	2	2	1	2	2	1	2	2
Hstarts	1	2	2	1	2	2	4	2	2
humanpower2	1	2	2	1	2	2	2	1	1
hurricNamed	1	2	2	1	2	2	4	2	2
immi2	1	1	1	1	1	1	3	2	2
immigration	1	2	2	1	2	2	3	2	2
Leinhardt	4	3	3	4	3	3	4	4	4
leuk	1	2	2	1	2	2	4	2	2
MathEnrollment	1	3	3	1	3	4	2	2	2
Mathlevel	1	2	2	1	2	2	2	1	1
nihills	3	3	3	1	2	2	2	2	2
PPP	4	2	2	4	2	2	3	2	2
Prestige	3	3	3	3	1	1	1	1	1
Pricing	1	1	1	1	1	1	2	3	3
pulpfiber	1	2	2	1	2	2	2	2	2
satact	1	2	2	1	2	3	2	2	2
SeaSlugs	3	1	1	2	2	2	1	2	2
stVincent	3	3	3	3	2	2	2	2	2

Tabulka 5.1: Vítězné metody z prvního experimentu pro náhodně vybrané datové soubory. (1) OLS, (2) LWS, (3) LTS, (4) HUBER. (A) původní data, (B) lokální kontaminace, (C) globální kontaminace. Sloupce této tabulky slouží jako odezva do následné fáze metaučení.

Závěr

Tato diplomová práce je vůbec první praktickou studií zkoumající citlivost metaučení na uměle přidanou kontaminaci. V experimentální části diplomové práce se nám podařilo zkonstruovat úspěšný systém pro automatickou selekci algoritmu (metaučení). Systém dokáže na základě jistých charakteristik datového souboru doporučit pro datový soubor nejvhodnější algoritmus strojového učení. Doporučení je až o 138 procent lepší než doporučení většinového klasifikátoru. Systém je schopen doporučit základní algoritmus i pro kontaminované datové soubory.

Zkonstruovaný systém pro automatickou selekci algoritmu používá v základní úrovni 643 datových souborů. Datové soubory jsou určené k učení s učitelem a pocházejí z několika internetových stránek pro strojové učení. Systém pro automatickou selekci algoritmu se na základě jistých charakteristik celé množiny datových souborů naučí klasifikační pravidlo pro doporučení algoritmu strojového učení pro nový datový soubor. Pro dobrou funkčnost systému hraje charakterizace datových souborů pomocí příznaků klíčovou roli. Jedny z nejvýznamnějších příznaků charakterizujících naše datové soubory jsou šikmost a špičatost reziduí a zejména robustní příznaky jako podíl odlehlých pozorování a robustní koeficient determinace. Tyto příznaky diskriminují mezi metodou nejmenších čtverců a robustními metodami nejmenších vážených čtverců, nejmenších usekaných čtverců a Huberovým odhadem. Veškerá očekávání předčila méně známá metoda nejmenších vážených čtverců. Další důležitá komponenta systému pro automatickou selekci algoritmu je predikční míra. V této diplomové práci je provedena studie metaučení pro tři různé predikční míry. Zkoumané predikční míry jsou standardní střední kvadratická chyba, robustní vážená střední kvadratická chyba a robustní usekaná střední kvadratická chyba. Zkonstruovaný systém pro automatickou selekci algoritmu dokáže mnohem lépe doporučit algoritmus při požití jedné z robustní predikčních měr. Robustní metody jsou typicky lepší při použití robustních měr predikce.

Dalšími důležitými komponenty systému pro automatickou selekci algoritmu je meta-algoritmus a typ meta-odezvy. Volba meta-algoritmu závisí na typu meta-odezvy. V této diplomové práci zkoumáme metaučení pouze pro doporučení nejlepšího algoritmu strojového učení, proto používáme kategorickou meta-odezvu. Pro tuto meta-odezvu jsme zvolili několik standardních klasifikačních algoritmů. Velmi dobře funguje penalizovaná logistická regrese a metoda nejbližších sousedů. Mírně za ostatními metodami zaostala metoda podpůrných vektorů, která je velmi silně závislá na nastavení vnitřních hyperparametrů. Učení klasifikačních algoritmů probíhá na meta-úrovni. Učení základních metod lineární regrese probíhá na základní úrovni. V obou dvou úrovních bylo dosaženo generalizace pomocí metody křížové validace. Experimentální studie byla velmi výpočetně náročná, protože křížovou validaci bylo nutné provést pro všechny datové soubory a všechny kombinace základních algoritmů, predikčních měr a typů uměle přidaných kontaminací. K původním datovým souborům byla přidána lokální a globální kontaminace. Na těchto kontaminacích byla studována citlivost metaučení.

Metaučení je citlivé na kontaminaci v datech. Úspěšnost metaučení může záviset na lokální kontaminaci. Při správné selekci příznaků je úspěšnost metaučení typicky nižší pro lokálně kontaminovaná data než pro data původní. Pokud má ale meta-odezva nevyvážené kategorie, tak je možné, že úspěšnost metaučení paradoxně vzroste vlivem lokální kontaminace. Úspěšnost metaučení při uměle přidané glo-

bální kontaminaci není typicky o moc lepší než většinový klasifikátor. Tato studie slouží jako výstraha dalším studiím, které by chtěly metaučení zkoumat pro globální kontaminaci. Pokud jsou datové soubory globálně kontaminované odlehlými pozorováními, není vhodné používat k evaluaci nejčastěji užívanou střední kvadratickou chybu, ale raději některou robustnější míru.

Další studie metaučení by se mohla podrobněji zaměřit na klíčovou oblast pro úspěšné metaučení, a to charakterizaci datových souborů pomocí vhodných meta-příznaků. Meta-příznaky by měly rozlišovat mezi jednotlivými základními metodami. Proto by pro budoucí studii bylo vhodné zvolit co nejrozličnější základní metody strojového učení. Podle silných stránek základních metod zvolit správné diskriminující příznaky. Kromě množiny základních algoritmů by další studie mohla prozkoumávat také jejich hyperparametry, nebo sestrojit další meta-úroveň a prozkoumávat hyperparametry meta-algoritmů. Prostor pro zlepšení je i ve fázi předzpracování dat. Datové soubory by se mohly ponechat ve své původní velikosti. Také by další studie mohla k metaučení použít ještě větší množství datových souborů, než bylo voleno v této práci.

Bibliografie

[1] J. B. Biggs, "The role of metalearning in study processes", *British Journal of Educational Psychology*, vol. 55, pp. 185–212, 1985.

[2] S. Hochreiter, A. S. Younger, and P. R. Conwell, "Learning to learn using gradient descent", in *Proceedings of the International Conference on Artificial Neural Networks*, Springer, 2001, pp. 87–94.

[3] C. M. Bishop, *Pattern recognition and machine learning*. Springer, 2006.

[4] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition", in *Proceedings of the IEEE Conference*, vol. 86, 1998, pp. 2278–2324.

[5] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks", in *Proceedings of the Advances in Neural Information Processing Systems*, 2012, pp. 1097–1105.

[6] Y. Wu, M. Schuster, Z. Chen, Q. V. Le, M. Norouzi, W. Macherey, and M. Krikun, "Google's Neural Machine Translation System: Bridging the gap between human and machine translation", *arXiv:1609.08144*, 2016.

[7] I. Sutskever, O. Vinyals, and Q. V. Le, "Sequence to sequence learning with neural networks", in *Proceedings of the Advances in Neural Information Processing Systems Conference*, 2014, pp. 3104–3112.

[8] J. K. Chorowski, D. Bahdanau, D. Serdyuk, K. Cho, and Y. Bengio, "Attention-based models for speech recognition", in *Proceedings of the Advances in Neural Information Processing Systems Conference*, 2015, pp. 577–585.

[9] J. Lu, D. Wu, M. Mao, W. Wang, and G. Zhang, "Recommender system application developments: A survey", *Decision Support Systems*, vol. 74, pp. 12–32, 2015.

[10] B. Pang and L. Lee, "Opinion mining and sentiment analysis", *Foundations and Trends in Information Retrieval*, vol. 2, pp. 1–135, 2008.

[11] C. Chatfield, *Time-series forecasting*. CRC press, 2000.

[12] S. Geman, E. Bienenstock, and R. Doursat, "Neural networks and the bias/variance dilemma", *Neural Computation*, vol. 4, pp. 1–58, 1992.

[13] R. R. Picard and R. D. Cook, "Cross-validation of regression models", *Journal of the American Statistical Association*, vol. 79, pp. 575–583, 1984.

[14] D. H. Wolpert and W. G. Macready, "No free lunch theorems for optimization", *IEEE Transactions on Evolutionary Computation*, vol. 1, pp. 67–82, 1997.

[15] J. Friedman, T. Hastie, and R. Tibshirani, *The elements of statistical learning*. Springer, 2001.

[16] S. A. Dudani, "The distance-weighted k-nearest-neighbor rule", *IEEE Transactions on Systems, Man, and Cybernetics*, pp. 325–327, 1976.

[17] V. Vapnik, *Statistical learning theory*. Wiley, 1998.

[18] P. McCullagh, *Generalized linear models*. Routledge, 2018.

[19] I. Goodfellow, Y. Bengio, and A. Courville, *Deep learning*. MIT Press, 2016.

[20] R. Hecht-Nielsen, "Theory of the backpropagation neural network", in *Neural Networks for Perception*, Elsevier, 1992, pp. 65–93.

[21] F. Rosenblatt, "The perceptron: A probabilistic model for information storage and organization in the brain", *Psychological review*, vol. 65, p. 386, 1958.

[22] S. Hochreiter and J. Schmidhuber, "Long short-term memory", *Neural Computation*, vol. 9, pp. 1735–1780, 1997.

[23] B. C. Csáji, "Approximation with artificial neural networks", *Faculty of Sciences, Eötvös Loránd University, Hungary*, vol. 24, p. 7, 2001.

[24] C. Romesburg, *Cluster analysis for researchers*. Lulu, 2004.

[25] D. W. Scott, *Multivariate density estimation: theory, practice, and visualization*. Wiley, 2015.

[26] J. R. Quinlan, "Induction of decision trees", *Machine Learning*, vol. 1, pp. 81–106, 1986.

[27] A. Liaw and M. Wiener, "Classification and regression by randomForest", *R news*, vol. 2, pp. 18–22, 2002.

[28] J. Cohen, P. Cohen, S. G. West, and L. S. Aiken, *Applied multiple regression/correlation analysis for the behavioral sciences*. Routledge, 2013.

[29] J. Bergstra and Y. Bengio, "Random search for hyper-parameter optimization", *Journal of Machine Learning Research*, vol. 13, pp. 281–305, 2012.

[30] P. H. Byrne, *Analysis and science in Aristotle*. SUNY Press, 1997.

[31] R. S. Sutton and A. G. Barto, *Introduction to reinforcement learning*. MIT Press, 1998.

[32] R. Bellman, "Dynamic programming", *Science*, vol. 153, pp. 34–37, 1966.

[33] S. B. Thrun, "Efficient exploration in reinforcement learning", 1992.

[34] Z. I. Botev, J. F. Grotowski, and D. P. Kroese, "Kernel density estimation via diffusion", *The Annals of Statistics*, vol. 38, pp. 2916–2957, 2010.

[35] R. Vilalta, C. Giraud-Carrier, P. Brazdil, and C. Soares, "Using meta-learning to support data mining", *IJCSA*, vol. 1, pp. 31–45, 2004.

[36] P. Slovic, B. Fischhoff, and S. Lichtenstein, "Behavioral decision theory", *Annual Review of Psychology*, vol. 28, pp. 1–39, 1977.

[37] K. B. Korb, "Introduction: Machine learning as philosophy of science", *Minds and Machines*, vol. 14, pp. 433–440, 2004.

[38] R. Caruana, "Multitask learning", *Machine Mearning*, vol. 28, pp. 41–75, 1997.

[39] C. Finn, K. Xu, and S. Levine, "Probabilistic model-agnostic meta-learning", in *Proceedings of the Advances in Neural Information Processing Systems Conference*, 2018, pp. 9516–9527.

[40] M. Feurer, J. T. Springenberg, and F. Hutter, "Initializing bayesian hyperparameter optimization via meta-learning", in *Proceedings of the 29th AAAI Conference on Artificial Intelligence*, 2015.

[41] L. Li, K. Jamieson, G. DeSalvo, A. Rostamizadeh, and A. Talwalkar, "Hyperband: A novel bandit-based approach to hyperparameter optimization", *The Journal of Machine Learning Research*, vol. 18, pp. 6765–6816, 2017.

[42] S. Falkner, A. Klein, and F. Hutter, "BOHB: Robust and efficient hyperparameter optimization at scale", in *Proceedings of the 35th International Conference on Machine Learning*, ICML, 2018.

[43] S. Thrun and L. Pratt, *Learning to learn*. Springer, 2012.

[44] C. Finn, P. Abbeel, and S. Levine, "Model-agnostic meta-learning for fast adaptation of deep networks", in *Proceedings of the 34th International Conference on Machine Learning*, JMLR, 2017, pp. 1126–1135.

[45] P. Brazdil, C. G. Carrier, C. Soares, and R. Vilalta, *Metalearning: Applications to data mining*. Springer, 2008.

[46] M. Feurer and F. Hutter, "Hyperparameter optimization", in *Automated Machine Learning*, Springer, 2019, pp. 3–33.

[47] K. Li and J. Malik, "Learning to optimize", in *Proceedings of the 5th International Conference on Learning Representations*, ICLR, 2017.

[48] M. Andrychowicz, M. Denil, S. Gomez, M. W. Hoffman, D. Pfau, T. Schaul, B. Shillingford, and N. de Freitas, "Learning to learn by gradient descent by gradient descent", in *Proceedings of the Advances in Neural Information Processing Systems Conference*, 2016, pp. 3981–3989.

[49] I. Guyon, A. Saffari, G. Dror, and G. Cawley, "Model selection: Beyond the bayesian/frequentist divide", *Journal of Machine Learning Research*, vol. 11, pp. 61–87, 2010.

[50] A. Lacoste, M. Marchand, F. Laviolette, and H. Larochelle, "Agnostic Bayesian learning of ensembles", in *Proceedings of the International Conference on Machine Learning*, 2014, pp. 611–619.

[51] T. G. Dietterich, "Ensemble methods in machine learning", in *Proceedings of the International Workshop on Multiple Classifier Systems*, Springer, 2000, pp. 1–15.

[52] J. Bennett and S. Lanning, "The Netflix prize", in *Proceedings of KDD Cup and Workshop*, Citeseer, 2007, p. 35.

[53] L. Breiman, "Bagging predictors", *Machine Learning*, vol. 24, pp. 123–140, 1996.

[54] R. E. Schapire, "The strength of weak learnability", *Machine Learning*, vol. 5, pp. 197–227, 1990.

[55] Y. Freund and R. E. Schapire, "A desicion-theoretic generalization of on-line learning and an application to boosting", in *Proceedings of the European Conference on Computational Learning Theory*, Springer, 1995, pp. 23–37.

[56] D. H. Wolpert, "Stacked generalization", *Neural Networks*, vol. 5, pp. 241–259, 1992.

[57] K. M. Ting and I. H. Witten, "Issues in stacked generalization", *Journal of Artificial Intelligence Research*, vol. 10, pp. 271–289, 1999.

[58] J. Gama and P. Brazdil, "Cascade generalization", *Machine Learning*, vol. 41, pp. 315–343, 2000.

[59] C. Kaynak and E. Alpaydin, "Multistage cascading of multiple classifiers: One man's noise is another man's data", in *Proceedings of the ICML Conference*, 2000, pp. 455–462.

[60] E. Alpaydin and C. Kaynak, "Cascading classifiers", *Kybernetika*, vol. 34, pp. 369–374, 1998.

[61] C. Ferri, P. Flach, and J. Hernández-Orallo, "Delegating classifiers", in *Proceedings of the 21st International Conference on Machine Learning*, 2004, p. 37.

[62] D. Opitz and R. Maclin, "Popular ensemble methods: An empirical study", *Journal of artificial intelligence research*, vol. 11, pp. 169–198, 1999.

[63] J. Kittler, M. Hatef, R. P. Duin, and J. Matas, "On combining classifiers", *IEEE transactions on pattern analysis and machine intelligence*, vol. 20, pp. 226–239, 1998.

[64] P. Domingos, "The role of Occam's razor in knowledge discovery", *Data Mining and Knowledge Discovery*, vol. 3, pp. 409–425, 1999.

[65] R. S. Olson, N. Bartley, R. J. Urbanowicz, and J. H. Moore, "Evaluation of a tree-based pipeline optimization tool for automating data science", in *Proceedings of the Genetic and Evolutionary Computation Conference*, 2016, pp. 485–492.

[66] N. L. Cramer, "A representation for the adaptive generation of simple sequential programs", in *Proceedings of the First International Conference on Genetic Algorithms*, 1985, pp. 183–187.

[67] D. Silver, T. Hubert, J. Schrittwieser, I. Antonoglou, M. Lai, A. Guez, M. Lanctot, L. Sifre, D. Kumaran, and T. Graepel, "Mastering chess and shogi by self-play with a general reinforcement learning algorithm", *arXiv:1712.01815*, 2017.

[68] L. Kotthoff, C. Thornton, H. H. Hoos, F. Hutter, and K. Leyton-Brown, "Auto-WEKA 2.0: Automatic model selection and hyperparameter optimization in WEKA", *The Journal of Machine Learning Research*, vol. 18, pp. 826–830, 2017.

[69] C. Thornton, F. Hutter, H. H. Hoos, and K. Leyton-Brown, "Auto-WEKA: Combined selection and hyperparameter optimization of classification algorithms", in *Proceedings of the 19th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2013, pp. 847–855.

[70] F. Hutter, H. H. Hoos, and K. Leyton-Brown, "Sequential model-based optimization for general algorithm configuration", in *Proceedings of the International Conference on Learning and Intelligent Optimization*, Springer, 2011, pp. 507–523.

[71] M. Feurer, A. Klein, K. Eggensperger, J. Springenberg, M. Blum, and F. Hutter, *Efficient and robust automated machine learning*. Curran Associates, 2015, pp. 2962–2970.

[72] H. Jin, Q. Song, and X. Hu, "Auto-keras: An efficient neural architecture search system", in *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2019, pp. 1946–1956.

[73] R. Vilalta and Y. Drissi, "A perspective view and survey of meta-learning", *Artificial Intelligence Review*, vol. 18, pp. 77–95, 2002.

[74] S. Thrun, "Lifelong learning algorithms", in *Learning to Learn*, Springer, 1998, pp. 181–209.

[75] L. A. Rendell, R. Sheshu, and D. K. Tcheng, "Layered concept-learning and dynamically variable bias management", in *Proceedings of the International Joint Conferences on Artificial Intelligence*, IJCAI, 1987, pp. 308–314.

[76] A. Sharif Razavian, H. Azizpour, J. Sullivan, and S. Carlsson, "CNN features off-the-shelf: An astounding baseline for recognition", in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*, 2014, pp. 806–813.

[77] J. Donahue, Y. Jia, O. Vinyals, J. Hoffman, N. Zhang, E. Tzeng, and T. Darrell, "Decaf: A deep convolutional activation feature for generic visual recognition", in *Proceedings of the International Conference on Machine Learning*, 2014, pp. 647–655.

[78] J. Deng, W. Dong, R. Socher, L. Li, K. Li, and L. Fei-Fei, "Imagenet: A large-scale hierarchical image database", in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, IEEE, 2009, pp. 248–255.

[79] D. Andre and S. J. Russell, "State abstraction for programmable reinforcement learning agents", in *Proceedings of the AAAI Conference*, 2002, pp. 119–125.

[80] T. G. Dietterich, D. Busquets, R. L. de Mantaras, and C. Sierra, "Action refinement in reinforcement learning by probability smoothing", in *Proceedings of the ICML Conference*, 2002, pp. 107–114.

[81] M. E. Taylor and P. Stone, "Transfer learning for reinforcement learning domains: A survey", *Journal of Machine Learning Research*, vol. 10, pp. 1633–1685, 2009.

[82] B. Hengst, "Discovering hierarchy in reinforcement learning with HEXQ", in *Proceedings of the ICML Conference*, vol. 19, 2002, pp. 243–250.

[83] A. Niculescu-Mizil and R. Caruana, "Predicting good probabilities with supervised learning", in *Proceedings of the 22nd International Conference on Machine Learning*, 2005, pp. 625–632.

[84] S. Thrun and J. O'Sullivan, "Clustering learning tasks and the selective cross-task transfer of knowledge", in *Learning to Learn*, Springer, 1998, pp. 235–257.

[85] J. Yosinski, J. Clune, Y. Bengio, and H. Lipson, "How transferable are features in deep neural networks?", in *Advances in Neural Information Processing Systems*, 2014, pp. 3320–3328.

[86] C. Szegedy, S. Ioffe, V. Vanhoucke, and A. Alemi, "Inception-v4, Inception-Resnet and the impact of residual connections on learning", in *Proceedings of the 31st AAAI Conference on Artificial Intelligence*, 2017.

[87] M. Z. Alom, T. M. Taha, C. Yakopcic, S. Westberg, P. Sidike, M. S. Nasrin, B. C. Van Esesn, A. A. S. Awwal, and V. K. Asari, "The history began from Alexnet: A comprehensive survey on deep learning approaches", *arXiv:1803.01164*, 2018.

[88] S. Targ, D. Almeida, and K. Lyman, "Resnet in resnet: Generalizing residual architectures", *arXiv:1603.08029*, 2016.

[89] F. Iandola, M. Moskewicz, S. Karayev, R. Girshick, T. Darrell, and K. Keutzer, "DenseNet: Implementing efficient convnet descriptor pyramids", *arXiv:1404.1869*, 2014.

[90] F. Chollet, "Xception: Deep learning with depthwise separable convolutions", in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2017, pp. 1251–1258.

[91] X. Zhang, Z. Li, C. Change Loy, and D. Lin, "Polynet: A pursuit of structural diversity in very deep networks", in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2017, pp. 718–726.

[92] J. Hu, L. Shen, and G. Sun, "Squeeze-and-excitation networks", in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018, pp. 7132–7141.

[93] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich, "Going deeper with convolutions", in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2015, pp. 1–9.

[94] A. Krizhevsky and G. Hinton, "Convolutional deep belief networks on CIFAR-10", *Unpublished manuscript*, vol. 40, pp. 1–9, 2010.

[95] J. D. Owens, M. Houston, D. Luebke, S. Green, J. E. Stone, and J. C. Phillips, "GPU computing", in *Proceedings of the IEEE Conference*, vol. 96, 2008, pp. 879–899.

[96] B. Zoph and Q. V. Le, "Neural architecture search with reinforcement learning", in *Proceedings of the 5th International Conference on Learning Representations*, ICLR, 2016.

[97] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Dropout: A simple way to prevent neural networks from overfitting", *The Journal of Machine Learning Research*, vol. 15, pp. 1929–1958, 2014.

[98] T. Elsken, J. H. Metzen, and F. Hutter, "Neural architecture search: A survey", *Journal of Machine Learning Research*, 2019.

[99] K. O. Stanley and R. Miikkulainen, "Evolving neural networks through augmenting topologies", *Evolutionary Computation*, vol. 10, pp. 99–127, 2002.

[100] E. Real, A. Aggarwal, Y. Huang, and Q. V. Le, "Regularized evolution for image classifier architecture search", in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 33, 2019, pp. 4780–4789.

[101] T. Elsken, J. H. Metzen, and F. Hutter, "Efficient multi-objective neural architecture search via lamarckian evolution", in *Proceedings of the 7th International Conference on Learning Representations*, ICLR, 2018.

[102] K. O. Stanley, "Neuroevolution: A different kind of deep learning", *arXiv:1712.06567*, 2017.

[103] M. Abadi, P. Barham, J. Chen, Z. Chen, A. Davis, J. Dean, M. Devin, S. Ghemawat, G. Irving, and M. Isard, "Tensorflow: A system for large-scale machine learning", in *Proceedings of the 12th USENIX Symposium on Operating Systems Design and Implementation OSDI*, 2016, pp. 265–283.

[104] B. Pell, "METAGAME: A new challenge for games and learning", 1992.

[105] S. Ravi and H. Larochelle, "Optimization as a model for few-shot learning", 2016.

[106] B. Zoph, V. Vasudevan, J. Shlens, and Q. V. Le, "Learning transferable architectures for scalable image recognition", in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018, pp. 8697–8710.

[107] H. Pham, M. Y. Guan, B. Zoph, Q. V. Le, and J. Dean, "Efficient neural architecture search via parameter sharing", in *Proceedings of the 35th International Conference on Machine Learning*, ICML, 2018.

[108] H. Cai, T. Chen, W. Zhang, Y. Yu, and J. Wang, "Efficient architecture search by network transformation", in *Proceedings of the 32nd AAAI Conference on Artificial Intelligence*, 2018.

[109] H. Liu, K. Simonyan, and Y. Yang, "DARTS: Differentiable architecture search", in *Proceedings of the 7th International Conference on Learning Representations*, ICLR, 2018.

[110] D. Ha, A. Dai, and Q. V. Le, "Hypernetworks", in *Proceedings of the 5th International Conference on Learning Representations*, ICLR, 2016.

[111] A. Brock, T. Lim, J. M. Ritchie, and N. Weston, "Smash: One-shot model architecture search through hypernetworks", in *Proceedings of the 6th International Conference on Learning Representations*, ICLR, 2017.

[112] K. Li and J. Malik, "Learning to optimize neural nets", *arXiv:1703.00441*, 2017.

[113] O. Wichrowska, N. Maheswaranathan, M. W. Hoffman, S. G. Colmenarejo, M. Denil, N. de Freitas, and J. Sohl-Dickstein, "Learned optimizers that scale and generalize", in *Proceedings of the 34th International Conference on Machine Learning*, JMLR, 2017, pp. 3751–3760.

[114] Y. Chen, M. W. Hoffman, S. G. Colmenarejo, M. Denil, T. P. Lillicrap, M. Botvinick, and N. de Freitas, "Learning to learn without gradient descent by gradient descent", in *Proceedings of the 34th International Conference on Machine Learning*, JMLR, 2017, pp. 748–756.

[115] B. M. Lake, R. Salakhutdinov, and J. B. Tenenbaum, "Human-level concept learning through probabilistic program induction", *Science*, vol. 350, pp. 1332–1338, 2015.

[116] B. Lake, R. Salakhutdinov, and J. Tenenbaum, "The Omniglot challenge: A 3-year progress report", *Current Opinion in Behavioral Sciences*, vol. 29, pp. 97–104, 2019.

[117] B. Lake, R. Salakhutdinov, J. Gross, and J. Tenenbaum, "One shot learning of simple visual concepts", in *Proceedings of the Annual Meeting of the Cognitive Science Society*, vol. 33, 2011.

[118] H. Edwards and A. Storkey, "Towards a neural statistician", *arXiv:1606.02185*, 2016.

[119] P. Mirowski, R. Pascanu, F. Viola, H. Soyer, A. J. Ballard, A. Banino, M. Denil, R. Goroshin, L. Sifre, and K. Kavukcuoglu, "Learning to navigate in complex environments", *arXiv:1611.03673*, 2016.

[120] A. Santoro, S. Bartunov, M. Botvinick, D. Wierstra, and T. Lillicrap, "Meta-learning with memory-augmented neural networks", in *Proceedings of the International Conference on Machine Learning*, 2016, pp. 1842–1850.

[121] Y. Duan, J. Schulman, X. Chen, P. L. Bartlett, I. Sutskever, and P. Abbeel, "Rl2: Fast reinforcement learning via slow reinforcement learning", *arXiv:1611.02779*, 2016.

[122] N. Mishra, M. Rohaninejad, X. Chen, and P. Abbeel, "A simple neural attentive meta-learner", in *Proceedings of the 6th International Conference on Learning Representations*, 2017.

[123] G. Koch, R. Zemel, and R. Salakhutdinov, "Siamese neural networks for one-shot image recognition", in *Proceedings of the ICML Conference*, Lille, vol. 2, 2015.

[124] J. Snell, K. Swersky, and R. Zemel, "Prototypical networks for few-shot learning", in *Proceedings of the Advances in Neural Information Processing Systems Conference*, 2017, pp. 4077–4087.

[125] O. Vinyals, C. Blundell, T. Lillicrap, and D. Wierstra, "Matching networks for one shot learning", in *Proceedings of the Advances in Neural Information Processing Systems Conference*, 2016, pp. 3630–3638.

[126] P. Shyam, S. Gupta, and A. Dukkipati, "Attentive recurrent comparators", in *Proceedings of the 34th International Conference on Machine Learning*, JMLR, 2017, pp. 3173–3181.

[127] Y. Bengio, S. Bengio, and J. Cloutier, "Learning a synaptic learning rule", in *Proceedings of the International Joint Conference on Neural Networks*, IJCNN, vol. 2, 1991, p. 969.

[128] D. O. Hebb, *The organization of behavior: A neuropsychological theory*. Psychology Press, 2005.

[129] J. Schmidhuber, "A 'self-referential'weight matrix", in *Proceedings of the International Conference on Artificial Neural Networks*, Springer, 1993, pp. 446–450.

[130] L. Spector and A. Robinson, "Genetic programming and autoconstructive evolution with the push programming language", *Genetic Programming and Evolvable Machines*, vol. 3, pp. 7–40, 2002.

[131] A. M. Turing, "Computing machinery and intelligence", in *Parsing the Turing Test*, Springer, 2009, pp. 23–65.

[132] J. Schmidhuber, "Evolutionary principles in self-referential learning, or on learning how to learn", Ph.D. dissertation, Technische Universität München, 1987.

[133] J. Schmidhuber, "Optimal ordered problem solver", *Machine Learning*, vol. 54, pp. 211–254, 2004.

[134] J. Schmidhuber, "Gödel machines: Fully self-referential optimal universal self-improvers", in *Artificial General Intelligence*, Springer, 2007, pp. 199–226.

[135] R. Poli, W. B. Langdon, N. F. McPhee, and J. R. Koza, *A field guide to genetic programming*. Lulu, 2008.

[136] P. Boersma and B. Hayes, "Empirical tests of the gradual learning algorithm", *Linguistic Inquiry*, vol. 32, pp. 45–86, 2001.

[137] S. Thrun and T. M. Mitchell, "Lifelong robot learning", *Robotics and Autonomous Systems*, vol. 15, pp. 25–46, 1995.

[138] J. Schmidhuber, J. Zhao, and M. Wiering, "Shifting inductive bias with success-story algorithm, adaptive Levin search, and incremental self-improvement", *Machine Learning*, vol. 28, pp. 105–130, 1997.

[139] V. Kvasnička and J. Pospíchal, "Artificial intelligence and collective memory", in *Emergent Trends in Robotics and Intelligent Systems*, Springer, 2015, pp. 283–291.

[140] G. Tesauro, "TD-Gammon, a self-teaching backgammon program, achieves master-level play", *Neural Computation*, vol. 6, pp. 215–219, 1994.

[141] J. X. Chen, "The evolution of computing: AlphaGo", *Computing in Science Engineering*, vol. 18, pp. 4–7, 2016.

[142] K. Arulkumaran, A. Cully, and J. Togelius, "Alphastar: An evolutionary computation perspective", in *Proceedings of the Genetic and Evolutionary Computation Conference Companion*, 2019, pp. 314–315.

[143] D. Ferrucci, A. Levas, S. Bagchi, D. Gondek, and E. Mueller, "Watson: Beyond jeopardy!", *Artificial Intelligence*, vol. 199, pp. 93–105, 2013.

[144] C. D. Manning and H. Schütze, *Foundations of statistical natural language processing*. MIT Press, 1999.

[145] S. Cave and S. S. ÓhÉigeartaigh, "An AI race for strategic advantage", in *Proceedings of the 2018 AAAI/ACM Conference on AI, Ethics, and Society*, 2018, pp. 36–40.

[146] J. R. Rice, "The algorithm selection problem", in *Advances in Computers*, vol. 15, Elsevier, 1976, pp. 65–118.

[147] K. Smith-Miles, D. Baatar, B. Wreford, and R. Lewis, "Towards objective measures of algorithm performance across instance space", *Computers & Operations Research*, vol. 45, pp. 12–24, 2014.

[148] K. Smith-Miles, "Cross-disciplinary perspectives on meta-learning for algorithm selection", *ACM Computing Surveys*, vol. 41, pp. 1–25, 2009.

[149] T. Cunha, C. Soares, and A. de Carvalho, "Metalearning and recommender systems: A literature review and empirical study on the algorithm selection problem for collaborative filtering", *Information Sciences*, vol. 423, pp. 128–144, 2018.

[150] S. C. Suh, *Practical applications of data mining*. Jones and Bartlett Publishers, 2011.

[151] R. Lior, *Data mining with decision trees: Theory and applications*. World Scientific, 2014.

[152] R. Prudêncio and T. Ludermir, "Meta-learning approaches to selecting time series models", *Neurocomputing*, vol. 61, pp. 121–137, 2004.

[153] X. Wang, K. Smith-Miles, and R. Hyndman, "Rule induction for forecasting method selection: Meta-learning the characteristics of univariate time series", *Neurocomputing*, vol. 72, pp. 2581–2594, 2009.

[154] C. Lemke and B. Gabrys, "Meta-learning for time series forecasting and forecast combination", *Neurocomputing*, vol. 73, pp. 2006–2016, 2010.

[155] C. Köpf and I. Iglezakis, "Combination of task description strategies and case base properties for meta-learning", in *Proceedings of the 2nd International Workshop on Integration and Collaboration Aspects of Data Mining, Decision Support and Meta-learning*, 2002, pp. 65–76.

[156] H. Bensusan and A. Kalousis, "Estimating the predictive accuracy of a classifier", in *Proceedings of the European Conference on Machine Learning*, Springer, 2001, pp. 25–36.

[157] L. Todorovski, H. Blockeel, and S. Dzeroski, "Ranking with predictive clustering trees", in *Proceedings of the European Conference on Machine Learning*, Springer, 2002, pp. 444–455.

[158] K. Morik and M. Scholz, "The miningmart approach to knowledge discovery in databases", in *Intelligent Technologies for Information Analysis*, Springer, 2004, pp. 47–65.

[159] M. Consortium, "Esprit project METAL", *was available at www.metal-kdd.org*, 2002.

[160] T. Gomes, R. Prudêncio, C. Soares, A. Rossi, and A. Carvalho, "Combining meta-learning and search techniques to select parameters for support vector machines", *Neurocomputing*, vol. 75, pp. 3–13, 2012.

[161] P. de Miranda, R. Prudêncio, A. de Carvalho, and C. Soares, "An experimental study of the combination of meta-learning with particle swarm algorithms for SVM parameter selection", in *Proceedings of the International Conference on Computational Science and Its Applications*, Springer, 2012, pp. 562–575.

[162] J. Vanschoren, "Understanding machine learning performance with experiment databases", Ph.D. dissertation, KU Leuven, 2010.

[163] C. Soares, "UCI++: Improved support for algorithm selection using datasetoids", in *Proceedings of the Pacific-Asia Conference on Knowledge Discovery and Data Mining*, Springer, 2009, pp. 499–506.

[164] J. Vanschoren and H. Blockeel, "Towards understanding learning behavior", in *Proceedings of the First Research Contact Day of the CIL Doctoral School*, 2006.

[165] K. Bache and M. Lichman, *UCI machine learning repository*, 2013.

[166] D. Glez-Peña, A. Lourenço, H. López-Fernández, M. Reboiro-Jato, and F. Fdez-Riverola, "Web scraping technologies in an API world", *Briefings in Bioinformatics*, vol. 15, pp. 788–797, 2014.

[167] F. S. Fogelman, "Data mining in the real world: What do we need and what do we have?", 2006.

[168] M. Hilario and A. Kalousis, "Quantifying the resilience of inductive classification algorithms", in *Proceedings of the European Conference on Principles of Data Mining and Knowledge Discovery*, Springer, 2000, pp. 106–115.

[169] D. W. Aha, "Generalizing from case studies: A case study", in *Machine Learning Proceedings*, Elsevier, 1992, pp. 1–10.

[170] J. Huang and C. X. Ling, "Using AUC and accuracy in evaluating learning algorithms", *IEEE Transactions on Knowledge and Data Engineering*, vol. 17, pp. 299–310, 2005.

[171] D. M. Powers, "Evaluation: From precision, recall and F-measure to ROC, informedness, markedness and correlation", 2011.

[172] C. Köpf, C. Taylor, and J. Keller, "Multi-criteria meta-learning in regression", in *Proceedings of the PKDD-01 Workshop on Integration of Data Mining, Decision Support and Meta-Learning*, 2001.

[173] N. Jankowski and K. Grąbczewski, "Universal meta-learning architecture and algorithms", in *Meta-Learning in Computational Intelligence*, Springer, 2011, pp. 1–76.

[174] G. Nakhaeizadeh and A. Schnabl, "Development of multi-criteria metrics for evaluation of data mining algorithms", in *Proceedings of the KDD Conference*, 1997, pp. 37–42.

[175] P. Brazdil, C. Soares, and J. P. Da Costa, "Ranking learning algorithms: Using IBL and meta-learning on accuracy and time results", *Machine Learning*, vol. 50, pp. 251–277, 2003.

[176] N. Jankowski, "Complexity measures for meta-learning and their optimality", in *Algorithmic Probability and Friends. Bayesian Prediction and Artificial Intelligence*, Springer, 2013, pp. 198–210.

[177] K. Alexandros and H. Melanie, "Model selection via meta-learning: A comparative study", *International Journal on Artificial Intelligence Tools*, vol. 10, pp. 525–554, 2001.

[178] B. Pfahringer, H. Bensusan, and C. Giraud-Carrier, "Meta-learning by landmarking various learning algorithms", in *Proceedings of the ICML*, 2000, pp. 743–750.

[179] I. Guyon and A. Elisseeff, "An introduction to variable and feature selection", *Journal of Machine Learning Research*, vol. 3, pp. 1157–1182, 2003.

[180] L. Todorovski, P. Brazdil, and C. Soares, "Report on the experiments with feature selection in meta-level learning", in *Proceedings of the PKDD-00 Workshop on Data Mining, Decision Support, Meta-learning and ILP*, Citeseer, 2000.

[181] P. Brazdil, J. Gama, and B. Henery, "Characterizing the applicability of classification algorithms using meta-level learning", in *Proceedings of the European Conference on Machine Learning*, Springer, 1994, pp. 83–102.

[182] G. Lindner and R. Studer, "AST: Support for algorithm selection with a CBR approach", in *Proceedings of the European Conference on Principles of Data Mining and Knowledge Discovery*, Springer, 1999, pp. 418–423.

[183] S. Y. Sohn, "Meta analysis of classification algorithms for pattern recognition", *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 21, pp. 1137–1144, 1999.

[184] H. Bensusan and C. Giraud-Carrier, "Discovering task neighbourhoods through landmark learning performances", in *Proceedings of the European Conference on Principles of Data Mining and Knowledge Discovery*, Springer, 2000, pp. 325–330.

[185] H. Bensusan, C. Giraud-Carrier a C. Kennedy, "A higher-order approach to meta-learning", GBR, tech. zpr., 2000.

[186] Y. Peng, P. A. Flach, C. Soares, and P. Brazdil, "Improved dataset characterisation for meta-learning", in *Proceedings of the International Conference on Discovery Science*, Springer, 2002, pp. 141–152.

[187] H. Bensusan, "God doesn't always shave with occam's razor—learning when and how to prune", in *Proceedings of the European Conference on Machine Learning*, Springer, 1998, pp. 119–124.

[188] J. Vanschoren, J. N. Van Rijn, B. Bischl, and L. Torgo, "OpenML: Networked science in machine learning", *SIGKDD Explorations Newsletter*, vol. 15, pp. 49–60, 2014.

[189] A. Kalousis, "Algorithm selection via meta-learning", Ph.D. dissertation, University of Geneva, 2002.

[190] L. Todorovski a S. Džeroski, "Experiments in meta-level learning with ILP", in *Proceedings of the European Conference on Principles of Data Mining and Knowledge Discovery*, Springer, 1999, s. 98–106.

[191] A. Kalousis and T. Theoharis, "Noemon: Design, implementation and performance results of an intelligent assistant for classifier selection", *Intelligent Data Analysis*, vol. 3, pp. 319–337, 1999.

[192] C. Soares and P. Brazdil, "Zoomed ranking: Selection of classification algorithms based on relevant performance information", in *Proceedings of the European Conference on Principles of Data Mining and Knowledge Discovery*, Springer, 2000, pp. 126–135.

[193] P. Brazdil and C. Soares, "A comparison of ranking methods for classification algorithm selection", in *Proceedings of the European Conference on Machine Learning*, Springer, 2000, pp. 63–75.

[194] H. R. Neave and P. L. Worthington, *Distribution-free tests*. Unwin Hyman, 1988.

[195] C. Spearman, "The proof and measurement of association between two things", *The American Journal of Psychology*, vol. 100, pp. 441–471, 1987.

[196] J. Gama and P. Brazdil, "Characterization of classification algorithms", in *Proceedings of the Portuguese Conference on Artificial Intelligence*, Springer, 1995, pp. 189–200.

[197] T. A. Craney and J. G. Surles, "Model-dependent variance inflation factor cutoff values", *Quality Engineering*, vol. 14, pp. 391–403, 2002.

[198] J. Jurečková, J. Picek, and P. K. Sen, *Methodology in robust and nonparametric statistics*. CRC Press, 2012.

[199] P. Rousseeuw, C. Croux, V. Todorov, A. Ruckstuhl, M. Salibian-Barrera, T. Verbeke, M. Koller, and M. Maechler, *Robustbase: Basic robust statistics. R package version 0.92-3*, 2015.

[200] A. Alfons, M. Templ, and P. Filzmoser, "Robust estimation of economic indicators from survey samples based on Pareto tail modelling", *Journal of the Royal Statistical Society*, vol. 62, pp. 271–286, 2013.

[201] J. Kalina, "On multivariate methods in robust econometrics", *Prague Economic Papers*, vol. 21, pp. 69–82, 2012.

[202] J. Jurečková, *Robustní statistické metody*. Karolinum, 2001.

[203] P. L. Davies and U. Gather, "Breakdown and groups", *The Annals of Statistics*, vol. 33, pp. 977–1035, 2005.

[204] J. Kalina, "Implicitly weighted methods in robust image analysis", *Journal of Mathematical Imaging and Vision*, vol. 44, pp. 449–462, 2012.

[205] J. Á. Víšek, "Regression with high breakdown point", in *Proceedings of ROBUST Conference*, 2000, pp. 324–356.

[206] J. Kalina, "Some robust estimation tools for multivariate models", in *Proceedings of the 9th International Days of Statistics and Economics*, University of Economics, Prague, 2015, pp. 713–722.

[207] J. Á. Víšek, "The least trimmed squares", *Kybernetika*, vol. 42, pp. 1–36, 2006.

[208] A. M. Leroy and P. J. Rousseeuw, "Robust regression and outlier detection", *Wiley*, 1987.

[209] F. R. Hampel, E. M. Ronchetti, P. J. Rousseeuw, and W. A. Stahel, "Robust statistics", *Wiley*, 1986.

[210] B. P. Jan Kalina, "Robust regression estimators: A comparison of prediction performance", Conference Mathematical Methods in Economics, University of Hradec Králové, 2017, pp. 307–312.

[211] R. Blum, *Linux command line and shell scripting bible*. Wiley, 2008.

[212] A. Robbins, E. Hannah, and L. Lamb, *Learning the vi and Vim Editors*. O'Reilly Media, 2008.