Mendelova univerzita v Brně
Provozně ekonomická fakulta

Částečně řízené učení algoritmů strojového učení (semi-supervised learning)

Diplomová práce

Vedoucí práce:
doc. Ing. Jan Žižka, CSc.

Bc. Jiří Pavlík

Brno 2016

Touto cestou bych rád poděkoval svému vedoucímu diplomové práce doc. Ing. Janu Žižkovi, CSc. za jeho cenné rady a přípomínky, které mi pomohly při tvorbě této práce. Poděkování patří i mé rodině, která mě podporovala po celou dobu studia a poskytla mi příležitost studovat.

Čestné prohlášení

Prohlašuji, že jsem tuto práci: **Částečně řízené učení algoritmů strojového učení**
vypracoval/a samostatně a veškeré použité prameny a informace jsou uvedeny v seznamu použité literatury. Souhlasím, aby moje práce byla zveřejněna v souladu s § 47b zákona č. 111/1998 Sb., o vysokých školách ve znění pozdějších předpisů, a v souladu s platnou *Směrnicí o zveřejňování vysokoškolských závěrečných prací.*

Jsem si vědom/a, že se na moji práci vztahuje zákon č. 121/2000 Sb., autorský zákon, a že Mendelova univerzita v Brně má právo na uzavření licenční smlouvy a užití této práce jako školního díla podle § 60 odst. 1 Autorského zákona.

Dále se zavazuji, že před sepsáním licenční smlouvy o využití díla jinou osobou (subjektem) si vyžádám písemné stanovisko univerzity o tom, že předmětná licenční smlouva není v rozporu s oprávněnými zájmy univerzity, a zavazuji se uhradit případný příspěvek na úhradu nákladů spojených se vznikem díla, a to až do jejich skutečné výše.

V Brně dne 19. května 2016 _____

Abstract

Pavlík, J. Semi-supervised learning. Diploma thesis. Brno: Mendel University, 2016. Brno: Mendel University, 2009.
This diploma thesis focuses on comparison of semi-supervised learning methods with methods of unsupervised and supervised machine learning. These methods are compared based on results of designed text mining experiments from relatively large collection of textual data. Data collection is formed by user textual reviews of electronic products from Amazon server which are written in natural English language. Based on experiment results recommendations and conclusions are proposed related to used methods of machine learning.

Keywords

Semi-supervised learning, classification, clustering, feature selection, product reviews, Amazon, text mining, stop words removal, Weka, Keel, Cluto, Rel-RASCO, RASCO, Self-training, Tri-training, Naive Bayes, k-means, SMO, k-NN, C4.5

Abstrakt

Pavlík, J. Částečně učené řízení algoritmů strojového učení. Diplomová práce. Brno: Mendelova univerzita v Brně, 2016.
Tato diplomová práce se zabývá srovnáním metod částečně řízeného učení s metodami řízeného a neřízeného strojového učení. Srovnání metod strojového učení je realizováno na základě výsledků navržených experimentů dolování znalostí z poměrně rozsáhlé kolekce textových dat. Datová kolekce je tvořená z uživatelských textových hodnocení produktů elektroniky ze serveru Amazon, která byla napsána v přirozeném anglickém jazyce. Na základě výsledků z realizovaných experimentů jsou vyvozeny doporučení a závěry týkající se použitých metodách strojového učení.

Klíčová slova

Částečně řízené učení, klasifikace, shlukování, výběr rysů, hodnocení produktů, Amazon, dolování znalostí z textových dat, odstranění stopslov, Weka, Keel, Cluto, Rel-RASCO, RASCO, Self-training, Tri-training, Naive Bayes, k-means, SMO, k-NN, C4.5

Obsah

Seznam obrázků

Seznam tabulek

1 Úvod a cíl práce

1.1 Úvod

Metody strojového učení nachází široké uplatnění v praxi při řešení reálných problémů, jako jsou například klasifikace dokumentů, rozpoznávání psaného textu nebo kategorizace obrazu. Další využití tyto metody nachází v oblasti podpory při rozhodování, kde se znalosti získané pomocí metod strojového učení následně využívají při rozhodování různých problémů. Dalším typickým použitím metod strojového učení je analýza textu psaného v přirozeném jazyce, která v současné době plné sociálních médií a internetových obchodů zažívá velký rozvoj. Řada internetových obchodů a prodejců umožnuje na svých webových stránkách nakupujícím zákazníkům poskytnout zpětnou vazbu na jednotlivé produkty. Typickou formou vyjádření zpětné vazby k produktu je obvykle krátké textové hodnocení psané v přirozeném jazyce. Tato hodnocení v přirozeném jazyce mohou být cenným zdrojem informací, který následně slouží jako pomoc při rozhodování v oblasti řízení vztahu se zákazníky. Na základě získaných znalostí z hodnocení lze využít řadu marketingových metod, jejichž účelem je především oslovení zákazníka a zvýšení jeho spokojenosti s nákupem.

Hodnocení psaná v přirozeném jazyce jsou obvykle při zpracování pomocí metod strojového učení velice náročná na přípravu a volbu použitých metod pro dolování znalostí. Uživatelská hodnocení obsahují celou řadu nedostatků typických pro přirozenou formu jazyka, jako jsou překlepy, gramatické chyby a speciální znaky, které činí tyto data obtížně zpracovatelnými pro metody strojového učení. Při zvolení vhodných metod předzpracování je však možné většinu těchto problémů odstranit a získat znalosti ukryté v datech pomocí metod strojového učení.

Tato práce se zabývá srovnáním metod strojového učení na základě provedení experimentů nad reálnými textovými daty. Zkoumaná data tvoří kolekce textových uživatelských hodnocení produktů elektroniky psaná v anglickém jazyce, která byla získána ze zahraničního serveru *Amazon*.

1.2 Cíl práce

Hlavním cílem diplomové práce je navržení a uskutečnění experimentů nad reálnými textovými daty za účelem srovnání výsledků dolování znalostí prostřednictvím algoritmů neřízeného, řízeného a částečně řízeného strojového učení.

Pro splnění hlavního cíle diplomové práce je nutné se nejprve seznámit se základní problematikou strojového učení, metodami předzpracování dat a následně s jednotlivými směry strojového učení, řízeným, neřízeným a částečně řízeným strojovým učením. Dalšími kroky jsou volba demonstračních a reálných dat, návrh a uskutečnění experimentů nad zvolenými daty. Posledním krokem je provedení vyhodnocení, interpretace a diskuze výsledků, na jejichž základě budou vyvozeny příslušné závěry a doporučení.

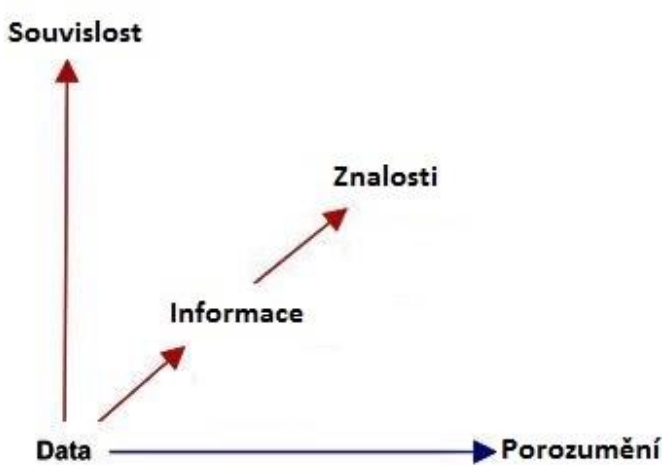
2 Problematika strojového učení

2.1 Úvod do problematiky

Před samotným popisem jednotlivých metod a algoritmů strojového učení je nezbytné si definovat několik základních pojmů, které jsou základními stavebními kameny v oblasti strojového učení. V následující teoretické části práce jsou popsány základní pojmy z oblasti zpracování dat, metody extrakce termů z textových dokumentů a způsoby jejich možné reprezentace. Následně jsou popsány jednotlivé metody strojového učení, řízené, neřízené a částečně řízené učení. V posledních kapitolách této části jsou popsány základní principy klasifikačních a shlukovacích algoritmů a způsoby jejich evaluace, které byly použity při realizaci experimentů v praktické části práce.

2.2 Zpracování dat

Strojové učení lze popsat jako soubor technik využívaných pro vyhledávání a popisování strukturálních vzorů ukrytých v datech. Proces zpracování dat je tedy jeho nedílnou součástí sloužící k získávání informací a znalostí o datech. Pojmy data, informace a znalost jsou často vzájemně zaměňovány i přes jejich poměrně jasnou a pevně danou hierarchii (viz Obr. 1) z hlediska porozumění zkoumané problematiky (Mohri a kol., 2012).



Obr. 1 Hierarchie dat, informace, znalosti z pohledu porozumění zkoumané problematiky
Zdroj: Upraveno podle Bellinger a kol., 2004.

Nejníže postavená v této hierarchii jsou **data**, která tvoří nějak nashromážděné, naměřené nebo jinak získané údaje nebo hodnoty, které se snažíme zpracovat za účelem získání informace (Mohri a kol., 2012). Obvykle se jedná o velké množství dat, které obsahuje z části i pro nás nepodstatná data tzv. šum. Šum může vznik-

nout například při přenosu díky rušení, nebo se může jednat o zaznamenanou chybu vzniklou během měření (Cejpek, 2005).

Filtrací šumu a následným výběrem vhodných dat pro řešený problém a jejich transformací získáme **informace**. Díky informacím získáme konkrétní význam dat, který může vést k lepšímu pochopení relačních vztahů mezi daty v kontextu s řešeným problémem. Zisk konkrétního významu dat ovšem ještě nezaručuje fakt, že se jedná o informaci, která je pro nás nějak užitečná nebo použitelná. Definici informace z matematického hlediska vytvořil jako první Shannon (1948), který řekl, že „informaci lze definovat jako míru množství neurčitosti nebo nejistoty o nějakém náhodném jevu, odstraněnou realizací tohoto jevu“. Dále Shannon (1948) prakticky dokázal že, pro měření množství informačního obsahu obsaženého v realizaci jevu lze využít entropie (Stone, 2015).

Náhodný proces s *n* možnými realizacemi lze označit jako **jev**. Oblíbeným příkladem jevu je hod mincí, který má právě dvě svoje možné realizace respektive výsledky. Samotnou realizací nějakého náhodného jevu je tedy myšleno právě jedno provedení jevu se získáním jednoho výsledku, například pokud při hodu mincí padne orel (Stone, 2015).

Entropie se používá v mnoha oborech a má celou řadu definicí z matematického hlediska ji lze definovat jako nějakou míru neurčitosti nebo nejistoty obsažené ve zkoumaném systému (Martin, England, 2010). Další z často používaných definic entropie je to, že entropie je dána střední hodnotou množství informací obsažených ve všech realizacích jevu. Shannon ve svém teorému objevil, že nejlepší funkcí pro vyjádření informace při zachování jejích vlastností je použití logaritmické funkce s libovolným základem větším než 1. Běžně je využíván logaritmus o základu dvou, jehož výsledkem je vyjádření množství informace v bitech (Stone, 2015). Výpočet množství informace, kterou získáme jednou realizací náhodného jevu je definován následujícím vztahem (Shannon, 1948):

$$S(p) = -\log_b p,\ p \in \langle 0,1 \rangle \tag{1}$$

kde *b* značí nějaký libovolný základ větší než 1, obvykle *b = 2*, *p* značí pravděpodobnost realizace náhodného jevu, *S(p)* vyjadřuje množství informace získané z realizace náhodného jevu.

Entropie *H(X)* pro všechny realizace náhodného jevu X je definována následujícím vztahem (Stone, 2015):

$$\mathrm{H(X)} = -\sum_{i=1}^{n} \mathrm{p(x_i)} \log_b p(\mathrm{x_i}),\ p \in \langle 0,1 \rangle \tag{2}$$

kde $p(x_i)$ je pravděpodobnost výsledku x_i při realizaci náhodného jevu *X*, značí nějaký libovolný základ větší než 1, obvykle *b = 2, n* je počet realizací náhodného jevu X.

Zobecněním informací získáme znalost, která nelze z hlediska informatiky formálně definovat zcela přesně, nicméně existuje mnoho různých definic, které se snaží o co nejlepší vysvětlení tohoto pojmu. Například Liebowitz (1999) ve své knize uvádí, že **znalost** je kombinace zkušeností, údajů, kontextových informací a expertního pohledu, která poskytuje rámec pro hodnocení a začleňování nových zkušeností a informací. Znalost lze také definovat jako míru porozumění či povědomí o řešení nějakého problému na základě faktů, informací, popisů nebo pravidel získaných ze zkušeností na základě zkoumání nebo pozorování nějakého systému nebo jevu (Cerra a kol., 2015).

2.3 Extrakce termů z textových dokumentů

Existuje celá řada problémů strojového učení pracujících s textovými dokumenty například klasifikace dokumentů, příspěvků nebo slovního hodnocení. Aby bylo vůbec možné zpracovat jednotlivé dokumenty pomocí algoritmů strojového učení, je nutné je nejprve transformovat z přirozeného jazyka do jiné reprezentace, která půjde dobře zpracovat pomocí metod strojového učení a zároveň zachová co nejvíc informací z původního dokumentu. Obvykle se textové dokumenty převádí do nějakého typu vektorové reprezentace (Hackeling, 2014). Před tvorbou samotné reprezentace dokumentu je nejprve nutné provést extrakci termů z textových dokumentů. Pod pojmem **term** si lze představit jednotlivé části dokumentu, které jako celek tvoří jeho sémantiku. V běžném kontextu strojového učení jsou termy chápany obvykle jako jednotlivá slova textového dokumentu. Nejjednodušší možnou reprezentací jednotlivých dokumentů je použití vektorové reprezentace tzv. **příznakovým vektorem** (angl. *features vector*), který je složený z jednotlivých termů. Vektorová reprezentace je obvykle vysoce dimenzionální, a proto je nutné využití extrakčních metod, které pomohou snížit dimenzionalitu pomocí odstranění nedůležitých termů (Aggarwal, 2012). Pro extrakci termů z textových dokumentů a jejich následnou reprezentaci ve strojovém učení existuje již celá řada zavedených formátů, které se běžně využívají (Hackeling, 2014).

2.3.1 Extrakční metody

Tokenizace

Tokenizace je jednou z nejdůležitějších částí procesu zpracování přirozeného textu, jejímž účelem je rozdělení textu na jednotlivá slova, na tzv. tokeny. Jednotlivé tokeny (angl. *tokens*) se následně používají pro tvorbu slovníku jako jeho instance čili jako termy. Proces tokenizace je podstatně ovlivněn jazykem zpracovávaného textu. V závislosti na jazyce analyzovaného textu existuje celá řada rozdílných přístupů k tokenizaci textu (Jurafsky, Martin, 2009). Obecně nejjednodušší metodou tokenizace je využití bílých znaků (mezera, tabulátor, atd.) a interpunkce pro rozdělení textu na jednotlivé tokeny. Tento způsob se osvědčil především u zpracování anglických textů. Nicméně tato metoda má celou řadu nedostatků spočívající především v nedokonalé tokenizaci hovorově stažených tvarů jako jsou například v anglickém jazyce tvary (i'am (já jsem), i'll (já budu), a další). V těchto

případech by rozdělením slov podle interpunkce často znamenalo možný vznik neexistujících slov, případně by mohlo dojít ke ztrátě jejich významu. Další z koncepčních nedostatků použití metody tokenizace na principu rozdělení slov bílým znakem spočívá v rozdělení víceslovných názvů nebo frázi na samostatné tokeny. Nejběžnějším případem výskytu tohoto problému je rozdělení víceslovného názvů měst (Kostelec nad Orlicí, New York) nebo společností či institucí (Mendelova univerzita, Honeywell Aerospace s.r.o.), na jednotlivé slova. Jako další typické problémy lze zmínit tokenizaci telefonního čísla nebo například emailu, které běžně obsahují speciální a interpunkční znaky. Univerzální přístup k procesu tokenizace prakticky neexistuje a použité metody je nutné uvážlivě volit na základě požadavků na analýzu textu podle jazyka a formátu analyzovaného textu (Manning a kol., 2008).

Stemming

Stemming je metoda extrakce termů na základě společného kořene slov (angl. *stem*), která se používá ke snížení dimenzionality příznakového vektoru. Princip stemmingu je založen na odstranění všech slov (termů), které mají stejný kořen a jejich nahrazení v příznakovém vektoru pouze pomocí tohoto kořene. Například všechna následující slova „training“, „trainer“, „trained“, „trainee“ a „trains“ mohou být nahrazena pomocí jednoho termu „train“. Algoritmus realizující stemming se nazývá **stemmer** (Ikonomakis a kol., 2005). Nejznámějším algoritmem provádějící stemming je Porterův stemmer (Porter, 1980), který publikoval a implementoval pro anglický jazyk. Porterův stemmer je jednoduchý algoritmus zaměřující se pouze na odstraňování přípon slov (angl. *suffix stripping*), který pracuje v několika fázích na základě pravidel vytvořených podle běžných anglických přípon (Jones, Willet, 1997). Nejčastějšími příklady odstraněných přípon jsou přípony „ing“ a „ed“ u sloves nebo přípony u tvarů množného čísla „s“ podle následujících pravidel (Porter, 1980):

S → „“	cars → car
(*v*) ING → „“	motoring → motor
(*v*) ED → „“	plastered → plaster

Existuje celá řada rozličných algoritmů realizujících stemming, které využívají odlišné metody stemmingu nebo například zpracovávají jiný jazyk než angličtinu. Přestože stemming je všeobecně v oblasti klasifikace textu považován za metodu, která zvyšuje přesnost rozhodování klasifikátorů, objevuje se celá řada pochybností při rozhodování o volbě vhodného algoritmu. Například Porterův algoritmus je často označován jako agresivní stemovací algoritmus (Ikonomakis a kol., 2005).

Lemmatizace

Proces nahrazování slov pomocí jejich transformací do normalizované formy (lemma) se nazývá **lemmatizace** (angl. *lemmatization*). Tento proces je velice důležitý při zpracování přirozených jazyků, které mají mnoho různých forem pro vyjádření stejného slova, jako je například vliv pohlaví na slovesnou formu nebo

změna slova na základě jeho skloňování (Sammut, Webb, 2010). Lemmatizace je hojně využívána ve všech oblastech zpracování přirozeného jazyka, například v lingvistice nebo ve strojovém učení při analýze sentimentu. Princip transformace slov u lemmatizace lze vidět na následujícím příkladu pro anglická slova „be“, „car“ a španělské slovo „querer“ (Jurafsky, Manning, 2015):

am, are, is → be
car, cars, car's, cars' → car
quiero (já chci), quieres (ty chceš) → querer (chtít)

Na předchozím příkladu je jasně vidět zásadní rozdíl mezi lemmatizací a stemmingem. Při použití stemmingu by nebylo možné transformovat slova „am“, „are“ a „is“ na slovo „be“. Lemmatizace je tedy podstatně sofistikovanější přístup, který však vyžaduje prvotní morfologickou analýzu a použití podstatně složitější algoritmů v porovnání se stemmingem. (Devi a kol., 2016). **Morfologická analýza** je nepostradatelnou součástí procesu lemmatizace a oblasti zkoumání přirozeného jazyka, která se zabývá rozlišováním a popisem gramatických kategorií. Výsledkem této analýzy je popis gramatiky jazyka ve formě sady značek pomocí níž lze vygenerovat základní tvar (lemma) slova (Ikeda a kol., 2009).

Stop slova

Ne všechna slova extrahované z dokumentů se používají pro jejich následnou reprezentaci. Existuje velký počet tzv. „bezvýznamových“ slov se běžně používají v souvislosti s jakýmkoliv kontextem, jako jsou například spojky nebo předložky. Tyto slova se nazývají **stop slova** (angl. *stopwords*). Tyto slova nepředstavují žádnou důležitou sémantickou informaci, ale díky četnosti jejich výskytu mohou podstatně ovlivnit formu reprezentace dokumentů a následně například i výsledky jejich klasifikace. Stop slova jsou obvykle definovány pomocí seznamu stop slov, který slouží jako vzor při jejich odstraňování během předzpracování dokumentů (Ikonomakis a kol., 2005). Existuje celá řada vytvořených seznamů stop slov pro různé jazyky, například pro angličtinu tyto seznamy běžně obsahují slova, jako jsou „is“, „the“, „about“ nebo „and“ (Rajaraman, Ullman, 2011). Odstraněním stop slov lze podstatně snížit dimenzionalitu reprezentace dokumentů v řádu desítek až stovek slov. Při použití seznamu stop slov je nutné brát v potaz i kontext, ve kterém probíhá analýza dokumentů, například obvyklým postupem při předzpracování je odstranění slov (termů) s délkou rovnou tří písmen nebo menší. Nicméně pokud například zpracováváme dokumenty týkající se chemických sloučenin nebo lékařské terminologie, případně z jiných oblastí hojně využívajících zkratek, může odebrání těchto termů způsobit ztrátu podstatných sémantických informací. V těchto případech je obvykle lepší vytvořit si vlastní seznam stop slov nebo případně modifikovat některý z běžně dostupných vytvořených seznamů (Ganesan, Subotin, 2014).

Odstranění termů s určitou četností
Další z možností redukce počtu termů ve slovníku za účelem snížení paměťové náročnosti při dolování znalostí, je odstranění termů s určitou četností (frekvencí) jejich výskytu v dokumentech. Obvykle je vhodné odebrat slova s vysokou a nízkou četností výskytu, která obecně mají malý nebo téměř žádný vliv na výsledky dolování znalostí. V případě slov s vysokou četností výskytu se obvykle jedná o slova, které mají v dané problematice obecný význam a malou informační hodnotu. V případě slov s nízkou četností výskytu se jedná o jedinečná slova, které se běžně nepoužívají nebo se také často může jednat například o neexistující nebo nesmyslná slova vzniklá díky překlepům. Odebráním slov s nízkou a vysokou četností výskytu lze výrazně zredukovat velikost slovníku bez negativního vlivu na výkonost metod strojového učení (Žižka, Dařena, 2011).

2.3.2 Reprezentace textových dokumentů

Bag-of-words
Bag-of-words v překladu „pytel nebo batoh slov" je jedna z nejznámějších a nejběžnějších reprezentací textu nebo dokumentů. V této reprezentaci je text převeden do posloupnosti slov vyskytujících se v textu, bez ohledu na pořadí jednotlivých slov ve větách a jejich gramatického smyslu. Jedná se o nejjednodušší způsob reprezentace textových dokumentů, který reprezentuje obvykle každé vyskytující se slovo v dokumentu jedním termem. Tato forma reprezentace dokumentů se běžně používá pro třídění nebo vyhledávání dokumentů, kde se využívá předpokladu, že dokumenty obsahující stejné slova budou pravděpodobně patřit do stejné kategorie (Hackeling, 2014). Tato forma reprezentace i přes svoji jednoduchost není zcela ideální z důvodu ztráty sémantických informací o dokumentech. Převedením dokumentů na tento způsob reprezentace dochází nejen ke ztrátě gramatické informace o smyslu vět, ale také již nelze sestavit text dokumentu do původního znění, což pro řešení některých problému strojového učení může činit značný problém (Biemann, Mehler, 2014). Další z nevýhod této reprezentace je to, že díky ztrátě sémantické informace vůbec neuvažuje speciální případy slov, jako jsou synonyma, homonyma a mnohoznačná slova (Ni, Xu, Cao a kol., 2016).

Vektorová reprezentace
Vektorová reprezentace (Vector Space Models, VSM) je další z často využívaných způsobů reprezentace textu v oblasti strojového učení. Tento způsob reprezentace byl vyvinut pro systém SMART (System for the Mechanical Analysis and Retrieval of Text) určený pro získávání informací v roce 1975 Gerardem Saltonem a jeho kolegy (Salton, Wong, Yang, 1975). Hlavní myšlenkou VSM je jednotná vektorová reprezentace textových dokumentů z kolekce vstupních dokumentů pomocí bodů ve společném prostoru (Turney, Pantel, 2010). Tento prostor je také často nazýván jako tzv. sémantický prostor a je definován na základě vytvořeného slovníku. **Slovník** (angl. *vocabulary*) je množina termů, která se vyskytuje v dokumentech a zároveň svým rozsahem definuje velikost vektoru, jímž jsou reprezentovány jednotlivé dokumenty. Jednotlivé body takto umístěné v prostoru respektive doku-

menty lze poté porovnávat na základě jejich vzájemné vzdálenosti v sémantickém prostoru. Body ležící v prostoru blízko u sebe jsou zcela jistě vzájemně sémanticky podobnější v porovnání s body vzdálenými daleko v prostoru. Tohoto faktu se využívá například při webovém vyhledávání, kdy se uživatelův dotaz převede na tuto reprezentaci a vloží se do sémantického prostoru dokumentů, následně jsou jako výsledek vyhledávání uživateli zobrazeny dokumenty seřazené sestupně od nejbližších dokumentů po nejvzdálenější dokument (Ni, Xu, Cao a kol., 2016). Dalším pojem často zmiňovaným v souvislosti s vektorovou reprezentací je **korpus** (angl. *corpus*). Pod tímto pojmem si lze představit nějakou definovanou kolekci dokumentů, nad kterou provádíme nějaké operace, jako například klasifikaci (Wołk, Marasek, 2014). Existuje několik způsobů udržování vektorové reprezentace celé kolekce dokumentů, obvykle se využívá některá z následujících maticových reprezentací (Turney, Pantel, 2010):

- Matice term-dokument (angl. *term-document matrix*, *T-D*)
- Matice term-term (*term-term matrix*, *T-T*)
- Matice dokument-dokument (*document-document matrix*, *D-D*)

U matice typu *T-D* matice jsou jednotlivé dokumenty reprezentovány pomocí řádků, kde každý řádek vždy představuje jeden dokument z korpusu. Sloupce této matice číselně reprezentují obvykle výskyt nebo váhu (důležitost) jednotlivých termů pro každý dokument (Turney, Pantel, 2010).

Příklad *T-D* matice lze prezentovat na následujícím jednoduchém příkladu (viz Tab. 1), tří krátkých dokumentů o sportu, kde čísla ve sloupcích jsou vyjádřeny pomocí reprezentace přítomnosti termu (viz dále):

dokument 1 Tenis je rychlý sport.
dokument 2 Nejlepší sport je fotbal.
dokument 3 Hokej je tvrdý sport.

Tab. 1 Příklad matice T-D s reprezentací váhy termů vyjádřené pomocí přítomnosti termu TP

	fotbal	hokej	nejlepší	rychlý	sport	tenis	tvrdý
dokument 1	0	0	0	1	1	1	0
dokument 2	1	0	1	0	1	0	0
dokument 3	0	1	0	0	1	0	1

U matice typu *T-T* jsou řádky i sloupce tvořeny symetricky pomocí jednotlivých termů. Tato reprezentace se obvykle využívá pro vyjádření sémantické podobnosti termů, proto je často označována pod pojmem matice slovního-kontextu (angl. *word-context matrix*). Pro vyjádření podobnosti mezi jednotlivými termy se využívá například lze využít kosinovou podobnost (viz podkapitola 2.6.1), ale existuje celá řada dalších metrik (Turney, Pantel, 2010).

U matice typu D-D jsou řádky i sloupce tvořeny znovu symetricky stejně jako je to u předchozího typu reprezentace. Jediným rozdílem je, že jednotlivé řádky

a sloupce vyjadřují dokumenty namísto termů. Cílem této reprezentace je vyjádřit podobnost mezi jednotlivými dokumenty (Turney, Pantel, 2010).

Obvyklou vlastností vektorové reprezentace rozsáhlé kolekce dokumentů je problém těchto reprezentací, který se vyznačuje nutností uchovávání výrazně převyšujícího počtu nulových hodnot vůči hodnotám nenulovým. Reprezentace dokumentů z pohledu uchování hodnot ve vektorech lze rozlišit na hustou a řídkou maticovou reprezentaci. **Hustá** (angl. *dense*) maticová reprezentace uchovává všechny složky vektorů bez rozdílu, jak nulové tak i nenulové. **Řídká** (angl. *sparse*) maticová reprezentace uchovává pouze nenulové složky vektoru, které reprezentuje pomocí indexu příznaku vektoru a jeho hodnotou (Yan a kol., 2013).

Reprezentace váhy termů

Jeden z nejjednodušších způsobů reprezentace váhy jednotlivých termů textového dokumentu je pomocí **přítomnosti termu** (*Term Presence, TP*). Jednotlivé prvky jsou reprezentovány binárně na základě jejich výskytu v dokumentu. V případě, že se daný term vyskytuje v dokumentu alespoň jednou tak je reprezentován hodnotou 1. Pokud se v dokumentu daný term nevyskytuje, je reprezentován hodnotou 0. Jinou variantou této reprezentace je její rozšíření reprezentace váhy termů v dokumentu pomocí **frekvence termu** (*Term Frequency, TF*), kde je každý term reprezentován pomocí celého přirozeného čísla udávajícího počet jeho výskytů v dokumentu. Problémem této reprezentace je možnost nadhodnocování některých termů na základě velké četnosti jejich výskytu v nepoměrně dlouhých dokumentech. Obvykle se tento problém řeší normalizací hodnoty výskytu termu na relativní hodnotu *TF*, která lze vyjádřit pro nějaký term *i* vyskytující se v dokumentu *j* pomocí následujícího vztahu (Rajaraman, Ullman, 2011):

$$TF_{ij} = \frac{n_{i\,j}}{\sum n_{i\,j}} \tag{3}$$

kde n_{ij} značí počet výskytů termu *i* v dokumentu *j* a suma ve jmenovateli značí celkový počet výskytů všech termů v dokumentu *j*.

Využití pouze samotné frekvence termů jako způsobu reprezentace dokumentů nemusí dosahovat dobrých výsledků při zpracování přirozeného textu. Jedním z nedostatků této reprezentace je například to, že nadměrný počet výskytů termu v jednom dokumentu může ovlivnit jeho důležitost bez ohledu na celkový kontext problému. Proto je často využívána metoda hodnocení (vážení) jednotlivých termů napříč celým korpusem, která se nazývá **Inverse Document Frequency** (*IDF*). Hlavní myšlenkou metody *IDF* je předpoklad, že term vyskytující s vysokou četností ve velkém počtu dokumentů z korpusu má menší význam než pojem, který se vyskytuje pouze v několika dokumentech. Tato myšlenka zaznamenala velký úspěch v praxi a přispěla k výraznému pokroku v oblasti dolování znalostí a strojového učení. Výpočet složky *IDF* lze definovat pomocí následujícího základního vztahu (Robertson, 2004):

$$IDF(t_i) = \log \frac{N}{n_i} \tag{4}$$

kde N značí celkový počet dokumentů v kolekci, t_i je term vyskytující se v n_i dokumentech.

Metoda *IDF* je obvykle slučována s reprezentací *TF* a dohromady tvoří velice robustní reprezentaci dokumentů pomocí váhového schématu termů. Tato reprezentace je obecně známá pod názvem **TF-IDF** nebo **TF*IDF** (angl. *Term Frequency-Inverse Document Frequency*) a díky své výjimečné robustnosti je velice oblíbenou a často využívanou reprezentací v oblasti strojového učení. Váhy pro určení důležitosti jednotlivých termů v této reprezentaci jsou určeny na základě součinu složek TF a IDF (Zhang a kol., 2011).

Sémantická vektorová reprezentace

Sémantická vektorová reprezentace je upravená verze klasické vektorové reprezentace (VSM), která se snaží o její vylepšení využitím sémantických informací z dokumentů při tvorbě jejich reprezentace. Existuje celá řada přístupů k řešení sémantického vylepšení vektorové reprezentace, například pomocí rozšíření slovníků o dvojice nebo n-tice slov na základě jejich výskytu v podobném kontextu. Další variantou zahrnutí sémantické informace je použití metody Latentní sémantické analýzy (angl. *Latent Semantic Analysis ,LSA*) nebo Latentní Dirichletovy alokace (*Latent Dirichlet Allocation, LDA*) vytvářejících asociace mezi termy objevujícími se ve stejném kontextu (Ni, Xu, Cao a kol., 2016). Žádná ze zmíněných variant sémantické vektorové reprezentace nezaručuje jistotu zlepšení výsledků strojového učení, protože špatná volba sémantické úrovně může zapříčinit zhoršení výsledků v porovnání s reprezentacemi, které neuvažují sémantickou stránku dokumentu. Naopak při správném výběru slovníkových n-tic nebo počtu témat u *LSA* a *LDA* lze dosáhnout velmi dobrých výsledků (Turney, Pantel, 2010).

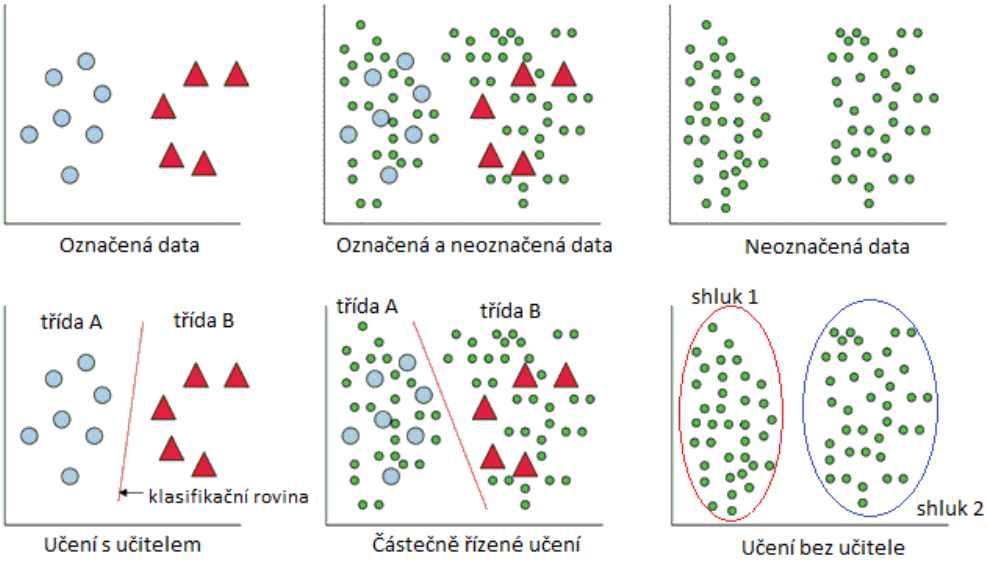
N-gramy

N-gramy (angl. *N-grams*) jsou další z hojně využívaných reprezentací dokumentů, kdy jsou jednotlivé dokumenty reprezentovány pomocí uspořádaných n-tic slov vyskytujících se v dokumentu. Jednotlivé termy u této reprezentace tvoří n-gramy, které jsou tedy vytvořeny na základě po sobě jdoucí n-tice slov (termů) vyskytující se v dokumentu. V kontextu terminologie n-gramů jsou často zmiňovány pojmy **bi-gram** a **tri-gram**, které jsou složeny z dvojic (bi-gram) případně trojic (tri-gram) slov. Hlavní myšlenkou vytváření reprezentace dokumentu pomocí n-gramů je nalezení uspořádaných n-tic slov, u kterých jejich spojením získáme více kontextových informací o dokumentu z pohledu řešeného problému než z jednotlivých slov samostatně (Mejova, Srinivasan, 2011). Příkladem mohou být anglická slova „not“ a „bad“ jdoucí po sobě například v nějakém dokumentu recenzujícím výrobek, může tvorba n-gramů zcela změnit výslednou klasifikaci. Pokud bychom klasifikovali dokument na základě jednotlivých slov, asi by téměř jistě byl

označen za negativní recenzi, ovšem v případě vytvoření n-gramu „not bad“ by byl tento dokument spíše zařazen do kategorie pozitivních recenzí. Tvorba n-gramů vyžaduje rozsáhlé zkušenosti a specifikaci řešeného problému, protože při špatné volbě můžeme podstatně zhoršit výsledky klasifikace díky přílišné konkrétnosti jednotlivých n-gramů nebo naopak díky jejich nejednoznačnosti (Agarwal, Mittal, 2016).

2.4 Strojové učení

Strojové učení je v praxi velice rozšířenou podoblastí umělé inteligence, která nachází významné uplatnění při řešení problémů napříč různými obory lidské činnosti. Existuje celá škála problémů, které je výhodné řešit za pomoci algoritmů strojového učení, například rozpoznávání lidské tváře nebo psaného textu v obraze, rozhodovací mechanismy pro spamové filtry, klasifikace dokumentů atp. Strojové učení lze definovat jako proces, na jehož základě počítače mění či adaptují prováděnou činnost (nezáleží na typu realizované činnosti, ať už se například jedná o řízení robota či hraní šachů), tak aby bylo dosaženo přesnějších respektive lepších výsledků (Marsland, 2009). Strojové učení je také často popisováno jako soubor technik využívaných pro vyhledávání a popisování strukturálních vzorů ukrytých v datech. Pod pojmem **strukturálního vzoru** v datech si lze představit nějaký objekt, funkci nebo proces popisující skupinu dat na základě jejich vlastností (Witten, Eibe, Hall, 2011). Strojové učení lze rozdělit podle typu učení do tří kategorií a to na učení s učitelem (angl. *supervised learning*), částečně řízené učení (*semi-supervised learning*) a učení bez učitele (*unsupervised learning*). Základní principy jednotlivých typů učení jsou demonstrovány na následujícím obrázku (viz Obr. 2) a budou podrobně popsány v následujících podkapitolách.



Obr. 2 Typy strojového učení podle typu učení
Zdroj: Upraveno podle Shah, Oehmen, Webb-Robertson, 2008.

2.4.1 Učení s učitelem

Metody učení s učitelem nebo také tzv. „řízeného učení“ jsou jednou z technik strojového učení, pomocí kterých se snažíme objevit závislosti mezi vstupními vlastnostmi (často zmiňované jako nezávislé proměnné) a cílovou vlastností (často zmiňované jako závislá proměnná). Nalezené vztahy mezi atributy jsou reprezentovány ve struktuře, která je obvykle v odborné literatuře nazývána jako model. Tento model nějakým způsobem popisuje a vysvětluje objevené závislosti, které byli skryty ve vstupních datech a může být použit pro predikci cílové proměnné pro další data na základě jejich vstupních vlastností (Maimon, Rokach, 2010).

Jiný autor (Mohri a kol., 2012) popisuje učení s učitelem jako techniku strojového učení, kdy se na základě trénovací množiny dat snažíme vytrénovat nějakou výstupní funkci. **Trénovací sada** (angl. *training set*) se typicky skládá z dvojic prvků, kde první z této dvojice je obvykle *příznakový vektor* popisující daný objekt a druhým prvkem je požadované cílové zařazení objektu. Díky informaci o cílovém zařazení jsou tyto data označována jako tzv. „označená data“ (angl. *labeled data*). Získání trénovací sady označených dat obvykle vyžaduje manuální označení jednotlivých položek od experta v daném oboru, což je časově i finančně náročný proces. Nicméně metody učení s učitelem dosahují podstatně lepších výsledků než metody učení bez učitele. Natrénovaná výstupní funkce může být použita pro predikci cílové hodnoty neznámých dat z *testovací množiny* na základě jejich příznakových vektorů. **Testovací množina** (angl. *testing set*) ve většině případů musí obsahovat pro klasifikátor zcela neznámá data, aby byla zaručena objektivita hodnocení natrénovaného klasifikátoru. Pokud by nebylo dodrženo toto pravidlo, může dojít k výraznému zkreslení výsledků klasifikace. Takto natrénovaná výstupní funkce může být použita pro řešení reálných problému v oblastech, jako jsou například marketing, bankovnictví nebo v oblasti zpracování textu. Další oblastí kde se tyto funkce v dnešní době hojně využívají je pro řešení problému detekce spamu v emailových klientech. Dvěma hlavními přístupy učení s učitelem jsou *klasifikace* a *regrese* (Mohri a kol., 2012).

Klasifikace

Klasifikace (angl. *classification*) je proces, při kterém je každá vstupní položka přiřazena do určité předem definované cílové skupiny. Známým příkladem klasifikačního problému je klasifikace dokumentů, kdy jednotlivé dokumenty zařazujeme do různých kategorií na základě jejich obsahu například sport, politika nebo náboženství (Maimon, Rokach, 2010). Obvykle se vstupní položky rozřazují do relativně malého počtu skupin, v případě složitých problémů však může počet kategorií razantně růst a tím se problém stává těžce řešitelným nebo i prakticky neřešitelným. Typickými příklady složitého klasifikačního problému jsou rozpoznávání řeči nebo rozpoznávání psaného písma (Mohri a kol., 2012). Pro účely klasifikace existuje celá řada metod, které budou popsány v následující kapitole této práce (viz kapitola 2.5) například algoritmus podpůrných vektorů, rozhodovací stromy nebo Naivní Bayesův klasifikátor.

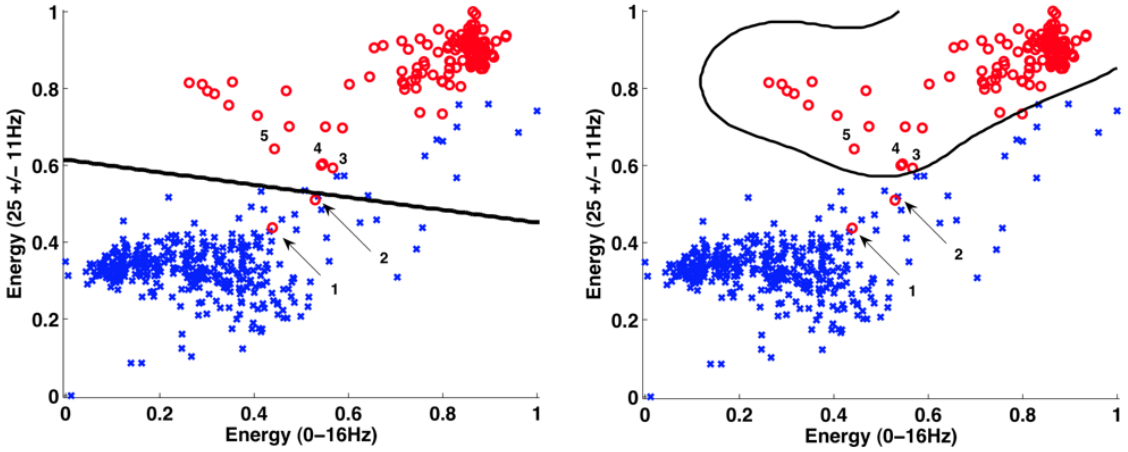
Regrese

Regrese (angl. *reggresion*) nebo také tzv. regresní modelování (angl. *reggresion modeling*) je metoda strojového učení, která se běžně používá na učení se vztahů mezi atributy vyjádřených v podobě reálných čísel (Aggarwal, 2012). Jiná definice říká, že regrese slouží pro předpověď nějaké reálné hodnoty každé vstupní položky v závislosti na dosud získaných hodnotách. Příkladem použití regrese je například její uplatnění v ekonomické oblasti pro předpověď růstu hodnoty cenných papírů nebo při předpovědi trendu ekonomických indexů. Výhodou regrese je snadné určování velikosti chyby na základě rozdílu mezi reálnou a predikovanou hodnotou v porovnání s klasifikací, kde typicky neexistuje žádný přesný ukazatel pro měření vzdálenosti mezi jednotlivými kategoriemi (Mohri a kol., 2012).

2.4.2 Částečně řízené učení

Jak již napovídá samotný název částečně řízené učení, myšlenkou za tímto způsobem strojového učení je zisk nějaké částečné informace od učitele (obvykle člověka) za účelem vylepšení výsledků. Klasickým příkladem je vylepšení distribuce velkého počtu neoznačených vstupních dat za pomoci mnohonásobně menší skupiny ručně označených dat k dosažení lepších výsledků klasifikace. Označená data v tomto případě slouží jednotlivým algoritmům jako tzv. násada (angl. *seed*), podle níž je natrénován počáteční klasifikátor, který se následně snaží označit zbylá neoznačená data zcela samostatně. Získání označených dat je mnohonásobně náročnější proces než získání dat neoznačených, která jsou obvykle volně dostupná například na internetu nebo zaznamenaná v přístrojích. Označování dat je ve většině případů časově i finančně náročný proces, který vyžaduje označení jednotlivých položek expertem orientujícím se v dané problematice. Použití kombinace označených a neoznačených dat v trénovací sadě je však pouze jedním ze způsobů částečného řízení. Existuje celá řada dalších metod částečně řízeného učení, pracující s různými typy částečné informace, jako jsou například omezující podmínky klasifikace nebo několik pravidel vygenerovaných rozhodujícím stromem (Abney, 2008).

Částečně řízené učení je stejně jako ostatní metody strojového učení závislé na vhodné parametrizaci a výběru poskytnuté informace. Při špatně zvolených předpokladech pro vstupní data dochází k podstatnému snížení kvality dosahovaných výsledků. Příkladem jsou grafově založené metody částečného řízené učení nebo transduktivní SVM (angl. *Support Vector Machines*), které dostanou jako počáteční předpoklad informaci o tom, že hranice mezi jednotlivými skupinami prochází skrz řídké oblasti. Pokud tyto metody následně necháme zpracovat data, která jsou vygenerovaná pomocí dvou silně překrývajících se Gaussových rozdělení, výsledný klasifikátor (viz Obr. 3) se bude rozhodovat velice špatně pro data, u kterých optimální rozhodovací hranice leží v zahuštěné oblasti (Søgaard, 2013).



Obr. 3 a) Ukázka výsledků při použití špatné parametrizace pro transduktivní SVM, b) Ukázka výsledků při použití optimální parametrizace pro transduktivní SVM
Zdroj: Verma, Lee, Shoeb, 2011.

Self-training

Self-training je nejznámější a zároveň pravděpodobně jedna z nejstarších metod částečně řízeného učení, často je také nazývána jako tzv. sebe učící metoda (angl. *self-teaching*). Myšlenka *Self-trainingu* se poprvé objevila někdy okolo roku 1960, nicméně první výraznou zmínkou je její použití v oblasti zpracování přirozeného jazyka až v roce 1995. Její hlavní myšlenkou je použití libovolného klasifikátoru, který vytrénujeme na základě malé množiny označených trénovacích dat a jeho následném použití při postupné klasifikaci množiny neoznačených dat a přidáváním nově označených (klasifikovaných) dat do trénovací množiny (Yarowsky, 1995). *Self-training* je pravděpodobně nejjednodušší možná metoda částečně řízeného učení. Princip nejjednoduššího algoritmu realizující *Self-training* lze popsat pomocí následujícím pseudokódu (Søgaard, 2013, str. 43):

1: L = $\{\langle y_i x_i \rangle\}_{i=1}^{N}, U = \{x_i\}_{i=1}^{M}$
2: c ← train(L)
3: **while** není splněno ukončující kritérium **do**
4: L ← L + select (label(U, c))
5: c ← train(L)
6: **end while**
7: **return** c

Obr. 4 Pseudokód algoritmu realizující *Self-training*
Zdroj: Søgaard, 2013

Tento algoritmus využívá klasifikátor *c* natrénovaný na základě označené datové množiny bodů $L = \{\langle y_i x_i \rangle\}_{i=1}^{N}$, a zároveň se snaží o následné využití dodatečných informací v podobě velké neoznačené datové množiny $U = \{x_i\}_{i=1}^{M}$, kde velikost

neoznačené datové množiny *M* je typicky mnohonásobně větší, než označené datové množiny *N*. Funkce *label* slouží pro klasifikaci všech prvků množiny neoznačených dat pomocí klasifikátoru *c*. Výsledkem této klasifikace je přiřazení jistoty (pravděpodobnosti) správného zařazení (angl. *confidence score*) prvku všem prvků z neoznačené množiny dat *U*. Funkce *train* slouží pro natrénování klasifikátoru za použití všech prvků z množiny označených dat *L.* Počáteční klasifikátor *c* je obvykle natrénován na poměrně malé množině označených dat, a proto můžeme zcela určitě říci, že při klasifikaci bodů z množiny neoznačených dat, která obsahuje mnohonásobně více položek, bude poměrně často chybovat v jejich označování. Pokud bychom tedy následně chtěli využít přidání nově označených dat k dalšímu vylepšení tréninku původního klasifikátoru *c,* i s celou řadou špatně označených dat, dosáhli bychom podstatného snížení celkové přesnosti klasifikace. Neoznačená data obsahují celou řadu pro klasifikátor neznámých dat, takže se nejedná o nic překvapivého. Tento problém se obvykle řeší pomocí funkce *select*, která se používá pro výběr pouze určité podmnožiny nově označených dat pro použití ke zpřesnění klasifikátoru *c*. Výběr nově označených položek je obvykle založen na principu výběru pouze těch položek, které dosáhnou určité hranice jistoty jejich klasifikace (angl. *confidence score of classification*), například výběr položek s jistotou klasifikace minimálně 90 % a vyšší. Nicméně i přes fakt, že díky funkci *select* můžeme podstatně rozšířit množinu označených trénovacích dat, ani její použití nezaručuje zlepšení přesnosti klasifikátoru. Funkce *select* může například vybírat z nově označených položek pouze jednu nebo několik kategorií, které dokáže označit s naprostou jistotou a tím rozšíří trénovací množinu nerovnoměrně pouze o položky těchto kategorií, čímž dojde k celkovému zkreslení výsledků klasifikace (Søgaard, 2013). Existuje celá řada dalších upravených verzí *Self-trainingu*, které se snaží o vylepšení této metody částečně řízeného učení. Například lze zmínit *Self-training* s možností průběžného promazávání nově přidaných položek s využitím validace nových položek pouze na základě původní trénovací sady (Abney, 2008).

Jednou z nejdůležitějších částí algoritmu realizující *Self-training* je nastavení ukončující podmínky (angl. *stopping criterion*, Obr. 4, řadek 3), typicky se využívá určitý počet *k* iterací algoritmu nebo ukončení při dosažení určité hranice přesnosti klasifikátoru. Vhodně zvolená ukončující podmínka společně s použitím technik pro zlepšení robustnosti *Self-trainingu* umožnují podstatné zlepšování výsledku až do bodu konvergence (Søgaard, 2013). Existuje celá řada technik, které se snaží o vylepšení robustnosti *Self-trainingu*, jako jsou například *Throttling*, *Pooling* nebo *Balancing*. *Throttling* je technika, kdy při každé iteraci algoritmu funkce *select* vybere pouze určitý počet *k* položek s nejvyšší jistotou klasifikace ze všech neoznačených dat. *Balancing* pracuje téměř naprosto stejně jako *Throttling* s tím rozdílem, že vybere *k* nejjistěji klasifikovaných položek pro každou klasifikační skupinu. Technika *Pooling* využívá principu výběru nově označených položek z náhodně zvolené podmnožiny neoznačených dat v kombinaci s využitím výběru na základě klasifikační jistoty (Abney, 2008).

Metoda *Self-training* je hojně využívaná v oblasti zpracování přirozeného jazyka (angl. *Natural Language Processing, NLP*), kde byla s úspěchem použita například při analýze sentimentu (Liu a kol., 2015) nebo při klasifikaci dokumentů (Fakeri-Tabrizi a kol., 2015). Je nutné zmínit i negativní výsledky při použití *Self-trainingu*, na které poukazuje řada autorů ve svých pracích například Abney (Abney, 2008) nebo Spreyer s Kuhnem (Spreyer, Kuhn, 2009). *Self-training* je podstatně závislý na nalezení správné konfigurace, kterou ovlivňuje celá řada vstupních faktorů. Dále je nutné zvýšit robustnost této metody pomocí vhodných technik, ošetřit problém přílišného přetrénování klasifikátoru a zajistit vhodné nastavení výběru nově označených položek (Søgaard, 2013).

Co-training

Co-training je další známou metodou částečně řízeného učení, která byla vytvořena za účelem zvýšení robustnosti *Self-trainingu* a poprvé byla představena v roce 1998 (Blum, Mitchell, 1998). Tato metoda reagovala na největší problém při *Self-trainingu*, kterým je tzv. jednosměrné učení, kdy jsou výsledky produkovány pouze na základě jedné perspektivy. *Co-training* nabízí řešení tohoto problému pomocí poskytnutí druhého pohledu na řešený problém. Prvotní myšlenkou *Co-trainingu* bylo rozdělení příznakového vektoru reprezentujících data na dvě části, pomocí kterých se následně vytvoří dva různé pohledy na data. Blum s Mitchelem ve své práci dokázali (Blum, Mitchell, 1998), že pokud dokážeme vhodně vytvořit dva zcela nezávislé pohledy na data lze pomocí *Co-trainingu* dosáhnout velice dobrých výsledků (Søgaard, 2013). Abney vylepšil myšlenku *Co-trainingu* tím, že ve své práci dokázal (Abney, 2002), že *Co-training* je možné úspěšně využít i v případě existence slabých závislostí mezi příznakovými vektory reprezentujících jednotlivé pohledy. Princip algoritmu realizující *Co-training* lze popsat pomocí následujícího pseudokódu (Søgaard, 2013, str. 46):

1: L = $\{\langle y_i x_i \rangle\}_{i=1}^{N}, U = \{x_i\}_{i=1}^{M}$
2: c ← train(L)
3: **while** není splněno ukončující kritérium **do**
4: c_1 ← train(view$_1$(L))
5: c_2 ← train(view$_2$(L))
6: L ← L + select (label(U, c_1)) + select (label(U, c_2))
7: **end while**
8: c ← train(L)
9: **return** c

Obr. 5 Pseudokód algoritmu realizující *Co-training*

Algoritmus realizující *Co-training* je téměř totožný s předcházejícím algoritmem realizujícím *Self-training* (viz Obr. 4), pouze s tím rozdílem, že výběr nově označených položek funkce *select* je realizován na základě porovnání dvou různých pohledů. Realizace rozdílných pohledů je obvykle vytvářena pomocí rozdělení příznakového vektoru jednotlivých instancí dat na dva na sobě nezávislé vektory.

V novějších pracích (Goldman, Zhou, 2000) nebo (Wang, Zhou, 2007) byla představena a v praxi úspěšně ověřena myšlenka *Co-trainingu*, který využívá k vytvoření rozdílných pohledů na řešený problém dva odlišné učící algoritmy (klasifikátory) natrénované na stejné množině dat. Tato verze *Co-trainingu* je obvykle v literatuře zmiňována jako tzv. „jedno pohledový" (angl. *single-view*) *Co-training* (Søgaard, 2013).

Co-training s výběrem náhodného podprostoru

Varianta *Co-trainingu* s výběrem náhodného podprostoru (Random Subspace method for Co-training, RASCO) vytváří různé pohledy na vstupní data pomocí skupin náhodně vybraných atributů z příznakového vektoru. Všechny vytvořené skupiny náhodně vybraných atributů jsou následně použity pro natrénování několika rozdílných klasifikátorů, které jako celek většinově rozhodují o klasifikaci vstupních dat. Zcela náhodný výběr podprostoru atributů pro trénování klasifikátoru, může mít v případě výběru nevhodných atributů negativní vliv na výslednou úspěšnost klasifikace. Protože v rámci omezeného rozsahu diplomové práce nelze popsat veškeré aspekty do detailu autor si dovoluje v případě zájmu čtenáře odkázat se na literaturu, ze které byly zároveň čerpány výše uvedené informace (Wang a kol., 2008).

Co-training s výběrem relevantního náhodného podprostoru

Co-training s výběrem relevantního podprostoru (Relevant Random Subspace method for Co-training, Rel-RASCO) je upravenou verzí předchozího zmíněného algoritmu RASCO. V případě Rel-RASCO jsou rozhodovací poprostory vytvářeny náhodně s využitím průměrné relevance atributů. Průměrná relevance atributů je vypočítana podle hodnoty informace, kterou atribut může poskytnout vzhledem k jednotlivým třídám. Pro zachování náhodného výběru jsou jednotlivé atributy do podprostoru vybírány na základě pravděpodobnosti relevance ohodnocení atributů. V rámci omezeného rozsahu práce není možné popsat do detailu všechny algoritmy, proto byly popsány pouze základní informace o tomto algoritmu. Další podrobnosti o algoritmu Rel-RASCO lze získat z následující doporučené literatury, ze které bylo čerpáno v této práci (Yaslan, Cataltepe, 2010).

Tri-training

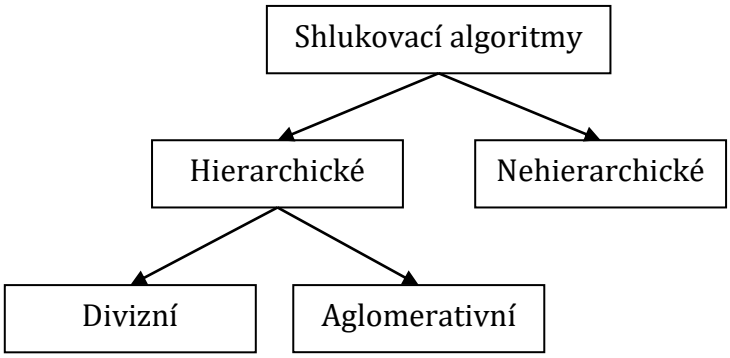
Posledním algoritmem použitým v rámci experimentů částečně řízeného učení byl *Tri-training*. Tento algoritmus z koncepčního pohledu pouze jednoduchým rozšířením jednopohledového *Co-trainingu*, které využívá pro klasifikaci dat do cílové třídy tří nezávislých klasifikátorů. Princip je tedy velice podobný stejně jako u *Co-trainingu*, všechny tři klasifikátory jsou nejprve natrénovány na počáteční označené násadě a poté rozhodují o zařazení neoznačených položek na základě většinového hlasování (angl. *majority voting*). Pro klasifikaci neoznačené položky je tedy nutné, aby se na jejím zařazení shodly alespoň dva ze tří klasifikátorů. V případě neshody všech tří klasifikátorů je obvykle neoznačená položka ponechána jako neoznačená nebo je nutné označit jeden z klasifikátorů jako rozhodující a neoznačená položka je zařazena do cílové třídy podle něho (Søgaard, 2013).

2.4.3 Neřízené učení

Pro neřízené učení se vstupní množina dat skládá z neoznačených dat, bez jakékoliv informace o jejich cílové hodnotě nebo příslušnosti. Neoznačená data je obvykle velice snadné získat, například přímo dolováním dat z internetu nebo jiných datových zdrojů a proto jsou metody neřízeného učení velice oblíbenou a zkoumanou oblastí strojového učení. Cílem při řešení problémů za použití metod neřízeného učení může být několik variant řešení. Prvním možným řešením je nalezení skupin dat se společnými vzory v jejich vlastnostech a na základě nalezených vzorů jsou data rozdělena do několika skupin. Tyto skupiny se nazývají shluky (angl. *clusters*) a metoda pro jejich vyhledávání se nazývá shlukování (angl. *clustering*). Dalším variantou řešení neřízeného učení je určování rozložení dat v rámci vstupního prostoru, tato metoda je známá jako odhad hustoty. Případně se může jednat o řešení ve formě projekce dat z mnoharozměrného prostoru do dvou nebo tří rozměrného prostoru za účelem vizualizace řešení (Marsland, 2015).

Shlukování

Shlukování je velice oblíbenou a zároveň v praxi nejpoužívanější metodou neřízeného strojového učení. Jak již bylo naznačeno v předchozí části, metody shlukování jsou použity tehdy, když nelze predikovat žádnou cílovou třídu položek, ale chceme vstupní data rozdělit do několik přirozených skupin (Maimon, Rokach, 2011). Shlukování lze označit jako speciální případ klasifikace na konečné množině dat, která se snažíme rozdělit do skupin na základě jejich podobnosti vyjádřené pomocí nějaké metriky měření vzdálenosti například Euklidovskou vzdáleností. Metody shlukování lze rozdělit do dvou hlavních kategorií podle přístupu ke tvorbě struktury dat na *hierarchické* a *nehierarchické*. Hierarchické shlukovací metody se dále dělí na *aglomerativní* a *divizní*, vztahy mezi jednotlivými metodami shlukování lze vyznačit pomocí jednoduchého schématu (viz Obr. 6). **Hierarchické** shlukovací metody (angl. *hierarchical*) jsou založeny na principu vytváření zanořených sekvencí (hierarchie) rozdělení dat podle jejich blízkosti v prostoru vyjádřené maticí blízkosti (angl. *proximity matrix*). Hierarchické metody jsou velice oblíbené především v oblasti strojového učení z pohledu biologie, sociologie nebo marketingu, kde se obvykle zkoumají vztahy na základě vytvořené taxonomie. Nevýhodou hierarchických metod je to, že se nehodí pro zpracování velkých kolekcí dat. **Nehierarchické** shlukovací metody (angl. *partional*) jsou založeny na principu vytvoření jednotlivých shluků dat na základě pouze jediného a předem určeného kritéria rozdělení, které se snaží objevit přirozené skupiny ukryté v datech. V dalších krocích se pak snaží upravit počáteční kritérium na základě výpočtu optimální vzdálenosti mezi jednotlivými podmnožinami (shluky). Nejznámějším algoritmem nehierarchického shlukování je algoritmus *k*-means, jehož princip je podrobně popsaný v následující části této práce (viz kapitola 2.6). Nehierarchické shlukování je typicky hojně využívané v aplikacích, kde je nutné vytvořit efektivní reprezentaci velkého množství dat (Jain, Dubes, 1998).



Obr. 6 Rozdělení shlukovacích metod
Zdroj: Upraveno podle Jain, Dubes, 1998

Prvním ze dvou možných přístupů k hierarchickému shlukování jsou **divizní** shlukovací metody, které považují vstupní množinou dat za jeden velký shluk a ten se pokouší dále rozložit do několika menších shluků. Rozklad probíhá na základě nějakého kritéria, podle kterého se původní shluk rozdělí do dvou menších shluků. Během rozkladu se vytváří hierarchická struktura a na konci této struktury se obvykle nacházejí jednoprvkové shluky. Druhým možným přístupem jsou **aglomerativní** shlukovací metody, které pracují se vstupními daty pomocí přístupu zdola nahoru, tedy zcela opačně než metody divizní. Vstupní data jsou u aglomerativního shlukování brány jako *n* separátních jednoprvkových shluků, které tvoří tzv. „nultý rozklad“. Jednotlivé shluky jsou pak postupně spojovány do dvojic podle nějaké kriteriální funkce, dokud nevznikne jeden velký konečný shluk. Nevýhodou všech metod hierarchického shlukování je jejich exponenciální časová složitost $O(n^2)$ při hledání optimálního rozkladu *n* vstupních objektů do dvou podmnožin, která při velkém počtu vstupních dat je prakticky nerealizovatelná v reálném čase (Zaki, Meira, 2014).

2.5 Klasifikace

Samotný proces klasifikace byl už popsán v předchozí kapitole, a proto se tato část věnuje především popisu jednotlivých přístupů ke klasifikaci a způsobu jejich hodnocení. V první podkapitole jsou popsány klasifikační algoritmy, které byly použity v praktické části práce pro klasifikaci v rámci učení s učitelem a zároveň byly využity pro metody částečně řízeného učení. Následující druhá kapitola je pak zaměřena na evaluační metody klasifikace, které se běžně využívají pro hodnocení klasifikačních algoritmů a zároveň byly použity v rámci praktické části.

2.5.1 Klasifikační algoritmy

Mezi nejznámější klasifikační metody, které se běžně využívají pro klasifikaci textových dokumentů, patří následující algoritmy (Baharudin, 2010):

- Bayesův klasifikátor (angl. *Bayes classifier*)
- rozhodovací stromy (*Decision Trees*)
- umělé neuronové sítě (*Artificial Neural Networks*)
- *k*-nejbližších sousedů (*k-nearest neighbor, k-NN*)
- algoritmy podpůrných vektorů (*Support Vector Machines, SVM*)
- asociační pravidla (*Association rules*)
- fuzzy korelace (*Fuzzy correlation*)
- výběr rysů (*Feature selection*)
- genetické algoritmy (*Genetic Algorithms*)

V rozsahu této diplomové práce není možné popsat všechny zmíněné algoritmy, a proto jsou v této podkapitole popsány pouze algoritmy, které byly použity při realizaci její praktické části.

Výběr rysů
Za výběrem rysů (angl. *feature selection*) se skrývá celá řada algoritmů a metod strojového učení, které se používají za účelem výběru podprostoru relevantních rysů pro natrénování klasifikátoru. Hlavním cílem použití těchto metod jsou především snížení dimenzionality vytvořeného datového modelu odstraněním nerelevantních atributů, urychlení procesu trénování a případné zamezení nebo redukce přeučení natrénovaného klasifikátoru. Metody výběrů rysů lze rozdělit podle principu do dvou hlavních skupin, na obálkové metody (angl. *wrappers*) a filtrovací metody (angl. *filters*). Obálkové metody využívají pro hodnocení rysů přesnost klasifikace prediktivní modely, které jsou natrénovány pro každý z hodnocených atributů. Vzhledem k přirozené vysoké dimenzionalitě textových dat je použití těchto metod pro jejich analýzu nevhodné. Protikladem obálkových metod jsou filtrovací metod, které pro hodnocení rysů provádějí nezávisle na prediktivních modelech. V případě filtrovacích metod jsou pro hodnocení rysů použity evaluační metriky zaměřené na určení atributů, na jejichž základě lze nejlépe rozpoznat cílové zařazení prvku (Baharudin, 2010). V rámci praktické části byla pro finální výběr rysů použita řadící metoda (angl. *ranking*) Pearsonova chí-kvadrátu na základě jejího řazení byly vybírány nejdůležitější atributy pro klasifikaci. Další zkoumané metody z rozsahových a rámcových důvodů nebudou v této práci popsány. Pearsonův chí-kvadrát je test nezávislosti mezi dvěma náhodnými veličinami X a Y. Test nezávislosti je vyjádřen pomocí rozdílu mezi očekávanou a naměřenou frekvencí dle následujícího vztahu (Kirk, 2014):

$$\chi^2 = \sum_{i=1}^{k} \frac{(O_i - E_i)^2}{E_i} \tag{5}$$

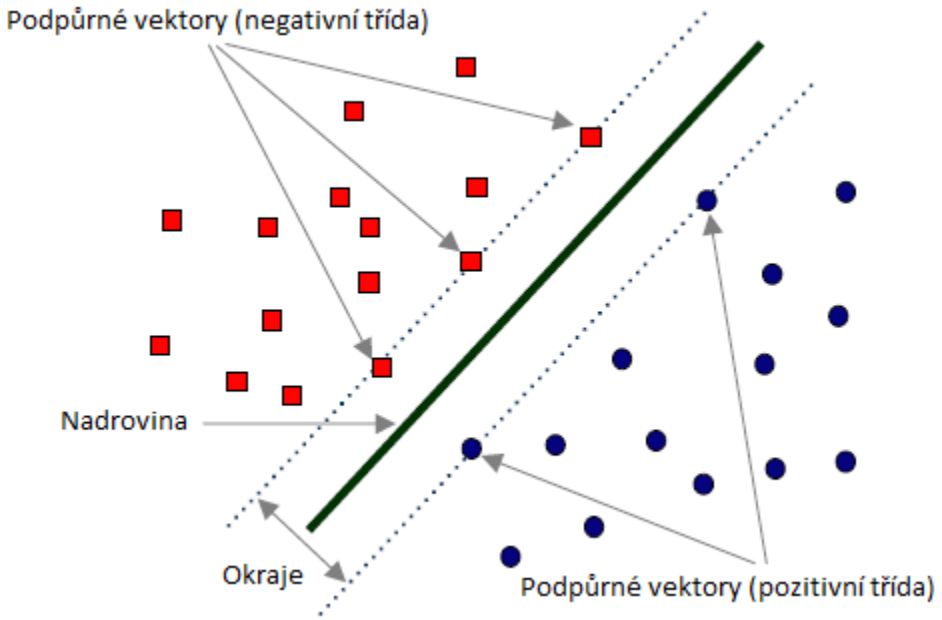
kde O_i je naměřená frekvence pro případy i, E_i je očekávaná frekvence v případě i a vypočítá se dle následujícího vztahu:

$$E_i = N(F(Y_u) - F(Y_l)) \tag{6}$$

kde N je celkový počet případů, F je distribuční funkce pro zkoumané rozložení, Y_u je horní hranice pro případ i, Y_l je spodní hranice pro případ i.

Algoritmy podpůrných vektorů

Algoritmy podpůrných vektorů (*Support Vector Machines, SVM*) jsou v dnešní době jednou z nejpopulárnějších metod moderního strojového učení. Poprvé byly představeny Vapnikem v roce 1992 a od té doby byla vytvořena celá řada vylepšených verzí těchto algoritmů, především díky tomu, že tyto algoritmy obvykle dosahují podstatně lepších klasifikačních výsledků, než ostatní klasifikační algoritmy (Marsland, 2015). SVM se využívá především pro řešení problémů tzv. binární klasifikace, kde je třeba stanovit lineární nebo nelineární hranici mezi dvěma třídami v prostoru. SVM lze například použít pro zhodnocení, zda poroste nebo bude klesat kurz měny na základě řady ekonomických faktorů. Dále se často používá na rozhodování problému, zda má banka poskytnout půjčku svému klientovi na základě jeho osobních finančních informací.



Obr. 7 Ukázka principu algoritmu podpůrných vektorů (SVM) na příkladu binární klasifikace, vykreslení nadroviny a podpůrných vektorů v prostoru.
Zdroj: Hyremath, Jyothi, 2013

Hlavním principem SVM je umístění nadroviny (angl. *hyperplane*) v prostoru ideálně tak, aby byla maximálně vzdálená od obou tříd označených bodů a zároveň, aby body každé z tříd byly pouze v jednom poloprostoru. Z obou nově vzniklých poloprostorů se vyberou body, které měli největší vliv na umístění nadroviny v prostoru a ty tvoří hranice při klasifikaci nových položek (viz Obr. 7). Tyto body tvořící hranice jednotlivých tříd, se nazývají jako tzv. podpůrné vektory (angl. *support vectors*)(Campbell,Ying, 2011).

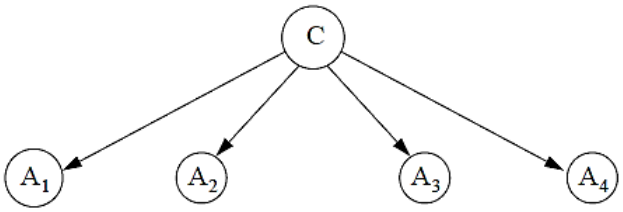
Rozdělující nadrovina je v prostoru definována následující rovnicí, kde b je posun nadroviny od počátku ve vstupním prostoru, $\vec{x}$ je vektor bodů ležících v nadrovině a současně kolmých na nadrovinu, $\vec{w}$ je vektor vah určujících orientaci nadroviny v prostoru (Campbell,Ying, 2011):

$$\vec{w} \cdot \vec{x} + b = 0 \tag{7}$$

Hlavní nevýhodou algoritmů SVM je především jejich poměrně velká složitost tréninkové a klasifikační fází, která způsobuje také jejich vysokou paměťovou a prostorovou náročnost během těchto výpočetních fází (Baharudin, 2010). Výpočetní časová složitost implementace algoritmů SVM je obvykle kvadratická O (n^3) v závislosti na použitém jádru (angl. *kernel*) lze dosáhnou i složitosti O (n^2) za jednu iteraci výpočtu (Zaki, Meira, 2014).

Naivní Bayes

Naivní Bayes (angl. *Naive bayes*) je jednoduchý klasifikační algoritmus založený na Bayesovském teorému o podmíněných pravděpodobnostech. Je to jeden z nejvíce efektivních a nejrychlejších učících algoritmů strojového učení, a proto se těší velké oblibě v této oblasti a je jedním z nejpoužívanějších algoritmů strojového učení vůbec. Překvapivá je především jeho efektivnost v praxi i přes to, že je založen na předpokladu, který obvykle nekoresponduje s reálnými problémy. *Naivní Bayesův* klasifikátor upravuje původní teorém za využití předpokladu, který říká, že všechny atributy z příznakového vektoru *A* příkladu jsou vzájemně zcela nezávislé vzhledem k hodnotě třídy *C*, do které daný objekt patří (viz Obr. 8). Tento předpoklad je často označován jako tzv. podmíněná nezávislost. Samotná naivita Bayesovského klasifikátoru tedy spočívá právě v tomto předpokladu podmíněné nezávislosti, který je při řešení reálných problému splněn jen ve zcela výjimečných případech (Zhang, 2004).



Obr. 8 Ukázka předpokladu podmíněné nezávislosti u Naivního Bayesovského klasifikátoru
Zdroj: Zhang, 2004, str. 1

Mezi výhody *Naivního Bayese* patří především fakt, že tomuto algoritmu stačí velice malá trénovací množina dat, na jejímž základě je schopen odhadnout parametry nutné pro správnou klasifikaci. Další z jeho výhod spočívá v jeho lineární výpočetní náročnosti O (*nd*), kde *n* značí počet dokumentů a *d* je počet dimenzí příznakového vektoru (Zaki, Meira, 2014).

Hlavní nevýhodou tohoto přístupu ke klasifikaci je relativně nízká přesnost klasifikace v porovnání s jinými klasifikačními algoritmy, jako jsou například SVM, které jsou obvykle podstatně přesnější (Baharudin, 2010).

Rozhodovací stromy

Rozhodovací stromy jsou klasifikační algoritmy vytvářející stromovou sktrukturu na základě trénovací množiny, kterou je následně možné použít pří klasifikaci neznámých případů. Vytvořená struktura je tvořená pomocí sady rozhodovacích pravidel, která je velice intuitivní a snadno interpretovatelná v přirozeném jazyce. Díky přehledné a snadno pochopitelné reprezentaci výsledných pravidel se tyto algoritmy teší velké popularitě v oblasti klasifikace a podpory při rozhodování. Rozhodovací strom se skládá z kořenového uzlu (*root node*), vnitřních uzlů (*internal nodes*) a listů (*leaf nodes*). Kořenový uzel tvoří počátek stromu a na rozdíl od vnitřních uzlů nemá žádné vstupní hrany. Uzly tvořící výslednou stromovou strukturu reprezentují atributy zkoumaných objektů, v případě této práce jsou to jednotlivé termy z uživatelských hodnocení. Vstupní hrany reprezentují skutečnost, zda se daný term v uživatelském hodnocení nachází nebo ne. Listy stromu představují cílovou třídu zařazení objektu. Následná klasifikace nově příchozích neoznačených hodnocení je velice jednoduchá, sestupně od kořenového uzlu jsou vyhodnocována jednotlivá pravidla na základě výskytu termů v neoznačeném hodnocení. Výsledná třída neoznačené položky je určena na základě třídy, kterou obsahoval list stromu, ve kterém skončilo sestupné vyhodnocování pravidel. Cílem rozhodovacích stromů je získat co nejmenší počet pravidel, které v ideálním případě budou schopny bezchybně klasifikovat všechny nově příchozí hodnocení v co nejmenším počtu kroků. V reálném případě se snažíme o vytvoření dostatečně obecného rozhodovacího stromu a při tvorbě stromu je nutné se vyvarovat tzv. stavu přeučení (ang. *overfitting*), při kterém listy stromu tvoří pouze jeden prvek. Za účelem udržení určité úrovně obecnosti stromu se obvykle používá operace prořezávání (angl. *pruning*), která odstraňuje příliš detailní větve a listy stromu (Baharudin, 2010). V praktické části práce je využívána implementace algoritmu rozhodovacího stromu C4.5 v Javě z nástroje Keel.

C4.5

C4.5 je klasifikační algoritmus založený na principu vytváření rozhodovacího stromu z trénovací množiny dat, který vytvořil Ross Quinlan v roce 1993. Jedná se o rozšířenou verzi Quinlanova předchozího algoritmu ID3. Rozhodovací strom je vytvářen stejně jako jeho předchůdce na principu výběru atributů (termů v případě dolování znalostí z textových dat) s využitím informační entropie. Atributry jsou vybírány na základě jejich schopnosti snížit celkovou entropii, tedy pod-

le informačního zisku. Cílem tohoto algoritmu je rozdělení vstupní hetoregenní množiny objektů na podmnožiny, které mají nižší hodnotu entropie než původní vstupní množina. V ideálním případě by toto rozdělení tvořily pouze dokonalé homogenní množiny (množina obsahující pouze prvky jedné cílové třídy), které mají nulovou entropii a přesně rozhodují o zařazení všech položek do správné cílové třídy. Velikost vytvořeného stromu je lineárně úměrná celkovému počtu vstupních příkladů. Výška stromu určující výpočetní časovou složitost je v závislosti na počtu příkladů polynomická (Marsland, 2015).

k-NN

Klasifikační metoda *k-nejbližších sousedů (*angl. *k-nearest neighbors)* patří k používaným metodám zejména díky její jednoduchosti. Princip této metody spočívá v umístění neoznačeného prvku (v kontextu práce uživatelského hodnocení) do vytvořeného *n*-rozměrného prostoru, kde jsou rozloženy prvky trénovací množiny a následného hledání n-nejbližších sousedů. Nejbližší sousedé neoznačeného prvku jsou určeni na základě výpočtu jejich vzdálenosti v *n*-rozměrném prostoru, která je vyjádřena pomocí nějaké metriky měření vzdálenosti ve společném prostoru, obvykle je použita Euklidovská vzdálenost respektive podobnost (viz podkapitola 2.6.1). Cílová třída zařazení neoznačeného prvku je určena podle cílové třídy jeho *k*-nejbližších sousedů. Pro výpočet cílové třídy neoznačeného prvku je nutné provést výpočet vzdálenosti mezi ním a všemy prvky trénovací množiny, proto je časová složitost *k-NN* lineární O(*n*). Prostorová složitost je v případě *k-NN* rovněž lineární O (*n*) v závislosti na celkovém počtu prvků *n* obsažených v trénovací množině (Mohri a kol., 2012).

2.5.2 Evaluační metriky klasifikace

Existuje celá řada principiálně rozdílných přístupů ke klasifikaci, ale všechny klasifikační algoritmy mají společný princip, který spočívá ve využití trénovací datové sady k natrénování klasifikátoru. Pro posouzení kvality natrénovaného klasifikátoru se využívají jeho výsledky klasifikace testovací množiny dat. Obvykle se pro účely hodnocení klasifikace využívají metody založené na vytvoření matice záměn pro jednotlivé třídy. Příklad matice záměn pro binární klasifikaci, tedy nejjednodušším případu klasifikace do dvou tříd, znázorňuje následující tabulka 0 (Zaki, Meira, 2014):

Tab. 2 Matice záměn pro binární klasifikaci

	Skutečná třída	
Predikovaná třída	Pozitivní (c_1)	Negativní (c_2)
Pozitivní (c_1)	True Positive (TP)	False Positive (FP)
Negativní (c_2)	False Negative (FN)	True Negative (TN)

Zdroj: Zaki, Meira, 2014

Při binární klasifikaci se obvykle jednotlivé třídy označují jako pozitivní a negativní. Správně klasifikované instance tříd se poté na základě jejich příslušnosti do pozitivní nebo negativní třídy nazývají jako skutečně pozitivní (angl. *true positive, TP*) nebo skutečně negativní (*true negative, TN*). Nesprávně klasifikované instance zařazené do špatné třídy se nazývají podle jejich skutečné příslušnosti jako falešně pozitivní (*false positive, FP*) a falešně negativní (*false negative, FN*). Na základě příkladu binární klasifikace vznikla celá řada metrik, které se běžně používají pro hodnocení kvality (úspěšnosti) klasifikátoru (Powers, 2011).

Přesnost
Přenost (angl. *accuracy*) je jednou z nejpoužívanějších evaluačních metrik klasifikace. Tato metrika je zároveň velice intuitivní, je vyjádřena jako poměr všech správně klasifikovaných objektů do jejich patřičných tříd. Přesnost klasifikátoru c lze vypočítat pomocí následujícího vztahu (Powers, 2011):

$$\mathrm{P}\,(c) = \frac{TP + TN}{TP + FP + TN + FN} \tag{8}$$

Určitost
Určitost (angl. *precision*) je jednou z běžně používaných metrik v oblasti strojového učení, v literatuře zabývající se dolováním znalostí je také často zmiňována pod jiným označením jako jistota (*confidence*). Tato metrika značí množství označených dokumentů nebo jiných dat během klasifikace, které byly označeny za pozitivní a jsou i skutečně pozitivní. Určitost *U* pro klasifikátor *c* se dá vyjádřit pomocí následujícího vzorce (Zaki, Meira, 2014):

$$\mathrm{U}\,(c) = \frac{TP}{TP + FP} \tag{9}$$

Senzitivita
Senzitivita (angl. *Sensitivity, recall*) je hodnotící metrika klasifikátoru určující míru skutečně pozitivních případů, které byly správně klasifikovány jako pozitivní. Senzitivita *S* klasifikátoru *c* se dá vyjádřit pomocí následujícího vztahu (Powers, 2011):

$$\mathrm{S}\,(c) = \frac{TP}{TP + FN} \tag{10}$$

2.6 Shlukování

V této kapitole jsou popsány nejznámější metody měření podobnosti dokumentů v prostoru a shlukovací algoritmy, které se v dnešní době běžně používají na kategorizaci dokumentů. Protože tato práce se nezabývá primárně shlukováním je zde popsán pouze nehierarchický shlukovací algoritmus K-means, který je používán v praktické části při realizaci experimentů. Zbylé shlukovací algoritmy jsou v této kapitole zmíněny pouze okrajově s odkazem na patřičnou literaturu.

2.6.1 Metody měření podobnosti objektů ve společném prostoru

Metody měření vzájemné podobnosti (angl. *similarity*) nebo vzdálenosti (*distance*) objektů ve společném prostoru jsou obecně základem shlukování. Tvorba jednotlivých shluků při kategorizaci dokumentů je založena právě na jejich vzájemné vzdálenosti ve společném prostoru. Před samotným měřením vzájemné vzdálenosti (podobnosti) mezi jednotlivými dokumenty je nutné převést všechny dokumenty z textové podoby na jejich vektorovou reprezentaci, aby je bylo možné reprezentovat ve společném prostoru. Důvodem vektorové transformace dokumentů je také to, že většina nejznámějších metod měření podobnosti je založena právě na principu práce s vektory (Zaki, Meira, 2014).

Kosinova podobnost

Kosinova podobnost (angl. *cosine similarity*) je v současnosti nejpoužívanější metodou měření podobnosti mezi vektorovými reprezentacemi při zpracování přirozeného textu (Žižka a kol. 2012; Sidorov a kol. 2014). Její princip je založený na výpočtu velikosti úhlu svíraného mezi porovnávanými vektory (viz Obr. 9), jenž reprezentují datové objekty například dokumenty. Funkce pro výpočet kosinu *cos* mezi dvěma vektory *a* a *b* lze vyjádřit pomocí následujícího vzorce (Zaki, Meira, 2014):

$$\cos\theta = \frac{\vec{a}^{T}\vec{b}}{\left\|\vec{a}\right\| \cdot \left\|\vec{b}\right\|} = \left(\frac{\vec{a}}{\left\|\vec{a}\right\|}\right)^{T}\left(\frac{\vec{b}}{\left\|\vec{b}\right\|}\right) \tag{11}$$

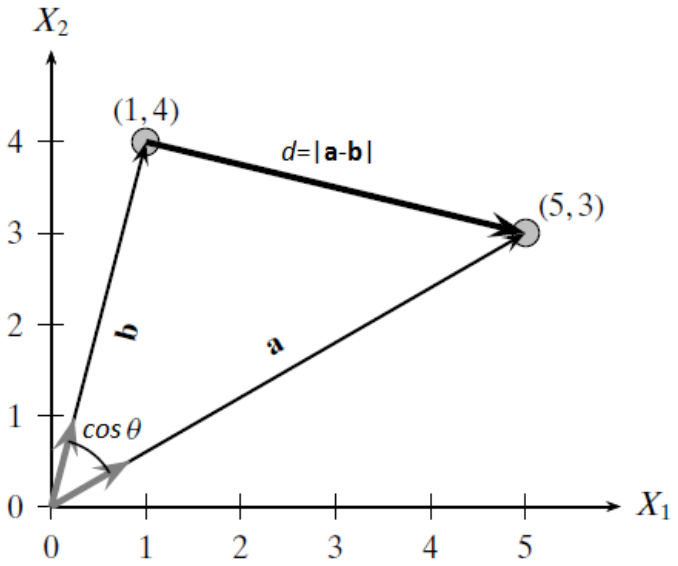
Euklidovská podobnost

Euklidovská podobnost (angl. *Euclidean distance*) je další ze známých metrik pro měření vektorové podobnosti. Tato metrika je založená na měření vzdálenosti mezi dvěma body na základě směru a velikosti vektorů v n-dimenzionálním prostoru. Euklidovská vzdálenost mezi dvěma libovolnými body ve společném prostoru lze vyjádřit pomocí následujícího vztahu (Zaki, Meira, 2014):

$$d(\vec{a},\vec{b}) = \left\|\vec{a}-\vec{b}\right\| = \sqrt{(\vec{a}-\vec{b})^{T}(\vec{a}-\vec{b})} = \sqrt{\sum_{i}^{m}(a_i - b_i)^2} \tag{12}$$

kde *a* a *b* jsou vektory reprezentující objekty, a_i a b_i jsou souřadnice těchto objektů v *i*-té dimenzi společného prostoru.

Na následujícím obrázku je znázorněna ukázka měření Kosinové *cos* θ a Euklidovské podobnosti *d* mezi dvěma vektory **a**(5,3) a **b**(1,4) v dvourozměrném prostoru.



Obr. 9 Ukázka měření kosinové a euklidovské vzdálenosti v dvourozměrném prostoru
Zdroj: Upraveno podle: Zaki, Miera, 2014, str. 8.

2.6.2 Algoritmy shlukování

Hierarchické shlukování

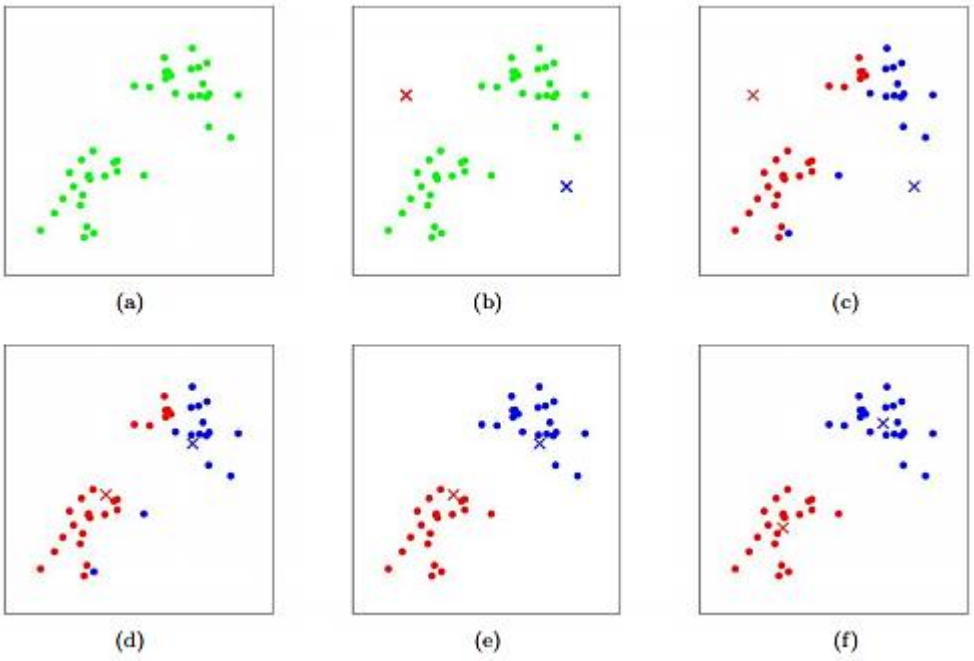
Algoritmy hierarchického shlukování nebyly při realizaci experimentů v praktické části použity z důvodu zaměření této práce především na částečně řízené učení. Existuje však celá řada známých algoritmů hierarchického shlukování, které se využívají v závislosti na formátu zpracovávaných dat a potřebné parametrizaci (například cílového počtu shluků). Mezi nejznámější aglomerativní algoritmy patří například algoritmus *Nejbližšího souseda* (angl. *Single Link*), algoritmus Skupinového průměru (*average link, UPGMA*) nebo Wardův algoritmus (*Ward's method*). Nejznámější zástupci algoritmů realizujících divizní shlukování jsou *opakovaná bisekce* nebo *PDDP* (angl. *Principal Direction Divisive Partitioning*). Podrobnější informace o algoritmech hierarchického shlukování popisují ve svých pracích například následující autoři (Jain, Dubes, 1988; Zaki, Meira, 2014, Kirk, 2014).

Nehierarchické shlukování

***k*-means**

Algoritmus *k*-means je velice známým a nejběžněji používaným algoritmem ze skupiny nehierarchických shlukovacích metod. Jedná se o velice jednoduchý a časově efektivní shlukovací algoritmus, který rozdělí vstupní data na předem definovaný počet *K* shluků. Algoritmus *k*-means pracuje na následujícím principu, který je zobrazen na následujícím obrázku (viz Obr. 10), nejprve náhodně vybere ze vstupních dat *K* položek a udělá z nich středy (angl. *centroids*) shluků (viz Obr. 10b). Následně zařadí všechny zbylé datové body do nejbližšího shluku v jejich okolí (viz Obr. 10c). Tím získáme shlukování založené na metodě náhod-

ných středů, na jejíž myšlence je k-means založeno. Úprava u *k*-means vůči metodě náhodných středů spočívá v pohybu středů, kdy na základě průměru hodnot všech datových prvků vybereme v každém z *K* nově vzniklých shluku nový střed. Na základě nově získaných středů provedeme vytvoření nových shluku přerozdělením jednotlivých datových bodů, které díky pohybu středů patří do jiného shluku (viz Obr. 10d-f). Proces shlukování u algoritmu *k*-means končí v případě, kdy všechny středy zůstanou neměnné (Kirk, 2014). Výhody základního algoritmu *k*-means spočívají především v jeho logaritmické časové složitosti O (log n) a lineární prostorové složitosti závislé na počtu shluků a celkovém počtu shlukovaných objektů (Wu, 2012).



Obr. 10 Průběh tvorby dvou shluků pomocí algoritmu *k*-means (křížky značí středy shluků). Zdroj: Piech, 2013.

2.6.3 Evaluační metriky shlukování

Existuje velké množství zcela rozdílných shlukovacích metod, které přistupují k tvorbě shluků na základě rozdílných datových charakteristik. Vzhledem k rozmanitosti přístupu shlukovacích algoritmů a jejich možné parametrizaci bylo nutné vyvinout objektivní přístupy pro objektivní posouzení výsledků shlukování. Proces posouzení výsledků shlukovacích metod zahrnuje tři hlavní směry validace, **evaluaci shlukování** (angl. *clustering evaluation*), která posuzuje správnost nebo kvalitu shlukování. Dalším směrem je **stabilita shlukování** (*clustering stability*), která zkoumá citlivost výsledků vzhledem ke změnám parametrů shlukování například počtu cílových shluků. Posledním směrem je zkoumání **tendence shlukování** (*clustering tendency*). V případě, že známe nebo dokážeme určit správnost

zařazení dat do shluků (tříd) lze použít následující hodnotící metriky shlukovacích algoritmů (Zaki, Meira, 2014):

- Metriky založené na shodě (angl. *Matching based measures*) - čistota (*purity*), F-measure, Maximální shoda (*Maximum matching*), přesnost (accuracy)
- Metriky založené na entropii (*Entropy-based measures*) - Entropie
- Metriky založené na párování (*Pairwise measures*) – Jaccardův koeficient
- Metriky založené na korelaci (*Correlation measures*)– C-index, Dunnův index

V rámci experimentů neřízeného učení je shlukování posuzováno pouze na základě přesnosti, jejíž princip byl popsán v předchozí kapitole. Nicméně výsledky shlukování z nástroje CLUTO obsahují i metriky entropii a čistotu shluků, a proto je v následující části krátce zmíněn jejich princip. Ostatní existující evaluační metriky nebyly v praktické části použity a proto jejich principy i z důvodu omezených rozsahových možností nejsou dále rozvedeny.

Entropie

Entropie je jedna z nejpoužívanějších metrik měření kvality shlukování, která shlukování posuzuje na základě rozložení tříd prvků v rámci shluků. Entropie pro jeden shluk S_r lze vyjádřit pomocí následujícího vzorce:

$$E(S_r) = \frac{1}{\log q} \sum_{i=1}^{q} \frac{n_r^i}{n_r} \log \frac{n_r^i}{n_r} \qquad (13)$$

kde q značí počet tříd v datové množině, n_r^i je počet prvků *i-té* třídy, které byly zařazeny do shluku r. Výsledná vážená entropie shlukování E pak lze vyjádřit jako součet entropií jednotlivých shluků vážených jejich velikostí:

$$E = \sum_{r=1}^{k} \frac{n_r}{n} E(S_r) \qquad (14)$$

kde k značí celkový počet shluků, n je celkový počet prvků v datové kolekci a n_r je počet prvků z datové kolekce v *r-tém* shluku.

V případě ideálního průběhu shlukování by hodnota celková entropie byla nulová, což by znamenalo, že každý shluk obsahuje pouze prvky z jedné třídy. V rámci shlukování se tedy snažíme získat shluky s co nejmenší hodnotou výsledné entropie (Zhao, Karypis, 2001).

Čistota

Čistota je nejjednodušší a nejintuitivnější metrikou měření kvality shlukování. Tato metrika zkoumá kvalitu shlukování na základě počtu položek majoritní třídy přiřazených ve vytvořeném shluku vůči prvkům ostatních tříd. Čistota shluku S_r lze vyjádřit pomocí následujícího vzorce:

$$P(S_r) = \frac{1}{n_r} \max_i (n_r^i) \tag{15}$$

kde n_r značí celkový počet prvků z datové kolekce ve shluku *r* a n_r^i je počet prvků *i-té* třídy, které byly zařazeny do shluku *r.* Celková čistota výsledků shlukování *P* je dána jako vážená suma čistot jednotlivých shluků:

$$P = \sum_{r=1}^{k} \frac{n_r}{n} P(S_r) \tag{16}$$

kde *k* značí celkový počet shluků, *n* je celkový počet prvků v datové kolekci a n_r je počet prvků z datové kolekce v *r-tém* shluku.
Obecně se snažíme dosáhnout co nejvyšší hodnoty čistoty, která značí kvalitu výsledného shlukování (Zhao, Karypis, 2001).

3 Zhodnocení současného stavu

V současné době existuje rozsáhlá řada nástrojů vytvořených za účelem analýzy dat a dolování znalostí. Tyto nástroje jsou hojně využívány ve všech oblastech lidské činnosti, ať už se jedná například o akademický výzkum, finanční analýzu, marketing nebo analýzu sentimentu uživatelů. Nástroje pro dolování znalostí napomáhají analyzovat nashromážděná data za účelem podpory při rozhodování a plánování. Výběr vhodných nástrojů pro účely analýzy dat a dolování znalostí je obvykle velice závislý na typu zpracovávaných dat a požadovaných informacích, které chceme získat. V této kapitole jsou popsány známé a obecně hojně používané a rozšířené nástroje, které byly zvažovány pro použití v praktické části práce. Nejprve jsou popsány komerční nástroje a následně jejich možné open source varianty, které jsou volně dostupné.

3.1 Komerční nástroje

V současné době je vyvíjena celá řada nástrojů, které se používá v oboru analýzy dat a dolování znalostí. Silnou stránkou komerčních nástrojů je obvykle kvalitní a robustní uživatelské rozhraní s rozsáhlou dokumentací. Další z obvyklých výhod je poskytovaná podpora od zkušených vývojářů těchto společností. Komerční nástroje jsou obvykle cíleny na potřeby koncových zákazníku a často jsou speciálně zaměřeny pouze na některé algoritmy nebo oblasti dolování znalostí. Mezi klasické nevýhody těchto nástrojů obvykle patří jejich špatná rozšiřitelnost, co se týče přidávání vlastních algoritmů a funkcí, vysoká cena a nutnost pracovat s celou sadou nástrojů od vybrané společnosti.

IBM SPSS Modeler (původně Clementine)
IBM SPSS Modeler (dále jen „SPSS“)je nástroj od společnosti IBM určený pro analýzu dat a dolování znalostí z dat, který jednou ze špiček v této oblasti na současném trhu. Nalezl široké využití především v komerční sféře, kde slouží pro strategické rozhodování jedinců či celých společností. Dále je pak hojně využíván na akademické půdě pro výuku analýzy dat, dolování znalostí a vědecký výzkum. Největší výhodou tohoto nástroje je grafické uživatelské rozhraní zpracované na vysoké úrovni. SPSS obsahuje celou řadu implementovaných algoritmů strojového učení, jejichž využití nevyžaduje žádné programátorské schopnosti. Dále zde můžeme využít rozsáhlé možnosti pro analýzu a předzpracování dat (IBM, 2016).
Nevýhodou SPSS je závislost na grafickém rozhraní, které limituje jeho rozšiřitelnost o nové algoritmy a neumožnuje propojení s žádným programovacím jazykem. Uživatel je nucen používat implementovanou sadu a na nově objevené metody a algoritmy dolování znalostí musí počkat, dokud nejsou implementovány do vyšších verzí programu. Jako nevýhodu SPSS také můžeme zmínit poměrně vysokou cenu tohoto programu a neúplnou kompatibilitu mezi mnoha existujícími verzemi SPSS.

RapidMiner Studio
RapidMiner Studio je multiplatformní systém zaměřující se na analýzu dat, dolování znalostí, prediktivní analytiku a business intelligence. Zpočátku se jednalo o open source nástroj, který díky spolupráci s velkými společnostmi jako jsou například Intel, Pepsi nebo eBay přešel do komerční sféry a začal konkurovat produktům od společností SAS, IBM a Oracle. RapidMiner Studio je vyvíjený v programovacím jazyce Java, ale dobře do něj lze integrovat funkce a algoritmy, které poskytují open source nástroje jako jsou například Weka nebo R. Výhodou tohoto nástroje je velmi dobře odladěné grafické uživatelské rozhraní, které od uživatele nevyžaduje žádné programátorské schopnosti ani obsáhlou znalost jednotlivých algoritmů. Další výhodou je podpora zpracování dat z libovolného zdroje, ať už z databáze nebo přímo ze souborů uložených na disku. Algoritmy pro dolování znalostí dokáží pracovat nejen s tabulkovou a číselnou formou dat, ale dokáží hledat znalosti i v textu, zvuku, obraze nebo videu. RapidMiner Studio nabízí i neplacenou počáteční verzi toho programu obsahující pouze základní funkce a řadu omezení, které jí činí prakticky nepoužitelnou pro složitější problémy (RapidMiner, 2016).

3.2 Open source nástroje

V současné době existuje celá řada volně dostupných nástrojů pro analýzu dat a dolování znalostí, které jsou alternativou komerčních nástrojů v této oblasti. Open source nástroje obvykle byly vytvořeny za účely výzkumu na akademické půdě a následně se rozšířily mezi odbornou veřejnost. Silnou stránkou volně dostupných open source nástrojů je to, že se obvykle teší velké oblibě a mohou se pochlubit velkou uživatelskou a programátorskou základnou. Další z častých výhod těchto nástrojů lze uvést jejich snadnou rozšířitelnost a možnost implementace vlastních funkcí. Mezi klasické nevýhody těchto nástrojů lze zmínit obvykle složité a nepříliš robustní uživatelské rozhraní. Negativem těchto nástrojů je také obvykle neexistující uživatelská podpora a velice střídmá nebo i dokonce žádná dokumentace některých funkcí.

Weka
Weka (Wakaito Enviroment for Knowledge Analysis) je softwarový nástroj disponující širokou kolekci algoritmů řízeného i neřízeného strojového učení, který je vyvíjen na univerzitě Wakaito na Novém Zélandu. Tento nástroj pro dolování znalostí byl původně vytvořen pro akademické účely, dnes však nachází i široké uplatnění při prediktivní analýze v oboru business inteligence. Weka je napsána v programovacím jazyce Java a díky tomu se jedná o multiplatformní nástroj. Tento nástroj se těší velké uživatelské a programátorské základně, protože se jedná o software s otevřeným zdrojovým kódem, legálně dostupným s dodržením licenčních podmínek GPL (General Public License). Některé z dalších otevřených softwarů pro dolování znalostí dat dokonce využívají algoritmy přímo z nástroje WEKA (Weka, 2016). Implementace pomocí jazyka Java přináší celou řadu nedostat-

ků, například při zpracování velkého množství dat z důvodů prostorové náročnosti. Nevýhodou je také to, že nástroj Weka nepodporuje multitasking, tedy provádění více běžících úloh dolování znalostí z dat současně. Další slabší stránkou tohoto softwaru jsou možnosti importování a exportování dat, které podporují omezenou sadu souborových typů a často dochází k chybám při načítání některých formátů (Ondrejka, 2013).

KNIME

KNIME (Konstanz Information Miner) je další ze současných open source nástrojů využívaných pro dolování znalostí pomocí algoritmů strojového učení. Původně byl vyvíjen jako proprietární software pro farmaceutické společnosti, jako podpůrný nástroj při analýze dat a rozhodování. Brzy však byla uvolněna open source verze tohoto programu a ta se stala velice oblíbeným nástrojem pro analýzu dat a dolování znalostí. KNIME je implementováno v programovacím jazyce Java, ale obsahuje celou řadu rozšíření díky modulům napsaných v oblíbených skriptovacích jazycích Perl nebo Python. Výhodou KNIME je také poměrně snadná možnost načítání dat z jiných open source nástrojů jako jsou např. Weka a R. Další výhodou je také poskytované grafické uživatelské rozhraní, které je jednoduché, intuitivní a nevyžaduje programátorské schopnosti. KNINE je tedy velice komplexní a robustní nástroj, který nachází uplatnění v celé řadě oblastí vědy a výzkumu (Knime, 2016). KNIME velice dobře pracuje z hlediska paměťové náročnosti. Nicméně jako nevýhodu je nutné zmínit, že oproti jiným open source nástrojům zaostává z hlediska grafické vizualizace dat a výsledků (Ondrejka, 2013).

KEEL

KEEL (Knowledge Extraction based on Evolutionary Lerning) je open source nástroj na dolování znalostí dat vytvořený v programovacím jazyce Java. Byl vyvinut skupinou španělských vědců pro účely výzkumu a vzdělávání v rámci Španělského národního programu pro vzdělávání. Jedná se o legálně dostupný software s otevřeným zdrojovým kódem při dodržení podmínek licence GPLv3 (General Public License verze 3). KEEL nabízí jednoduché uživatelské rozhraní umožňující návrh experimentů pro rozličné sady dat za použití některého z rozsáhlé řady poskytovaných algoritmů. Tento nástroj poskytuje celou řadu algoritmů řízeného a částečně řízeného strojového učení. Dále obsahuje implementaci algoritmů pro zpracování a diskretizaci dat, ošetření chybějících hodnot, statistické a vyhodnocovací metody (García, Luengo, Herrera, 2015). Velkou výhodou tohoto softwaru je rozsáhla sada implementovaných algoritmů strojového učení a jeho snadná rozšiřitelnost pomocí nových modulů napsaných v programovacím jazyce Java. Další výhodou oproti jiným nástrojům určeným k dolování znalostí je implementace řady algoritmů částečné řízeného strojového učení jako jsou například *Self-training* nebo *Co-training*. Nevýhodou jsou velice omezené možnosti uživatelského rozhraní, které slouží pouze pro sestavení experimentů, bez možnosti grafického zobrazení výsledků. Jako další nevýhodu je nutné zmínit problematické

importování nových datových sad do programu, které špatně identifikuje nečíselné nominální hodnoty a má pouze omezené možnosti editace.

CLUTO

CLUTO je softwarový balíček zaměřený výhradně na shlukovací analýzu datových sad, pomocí různých typů implementovaných metod shlukování. Jedná se o multiplatformní nástroj implementovaný v programovacích jazycích C a C++ dostupný pro systémy Windows, Linux a MacOS. V současnosti se stále řadí mezi jedny z nejpoužívanějších a velice populárních nástrojů používaných na shlukovací analýzu i přesto, že CLUTO byl vytvořen profesorem Georgem Karypisem již v roce 2006. CLUTO obsahuje také celou řadu implementovaných algoritmů na hodnocení podobnosti či měření vzdálenosti mezi daty a vizualizačních metod. Celkem existují tři verze tohoto nástroje CLUTO, gCLUTO a wCLUTO. CLUTO je původní nástroj, který pracuje jako konzolová aplikace a na kterém jsou založeny další nadstavby. Nástroj gCLUTO (Graphical Clustering Toolkit) je grafická nadstavba původní konzolové aplikace, která poskytuje jednoduché grafické uživatelské rozhraní nad původní konzolovou aplikací. Další nadstavba nástroje CLUTO je wCLUTO (Web-based Clustering of Microarray Data) sloužící pro shlukovací analýzu přímo na webu, kam si uživatel nahraje data, která chce analyzovat (Karypis, 2006).

R

R je programovací jazyk a open source prostředí vyvinuté pro statistickou analýzu dat, dolování znalostí a grafickou vizualizaci dat. Jedná se o GNU (GNU is Not Unix) projekt vyvinutý Johnem Chambersem a jeho kolegy, který je založený na podobném komerčním prostředí S vyvinutém v Bellových laboratořích. Prostředí a jazyk R jsou multiplatformní a jsou přeložitelné a spustitelné na většině současných distribucí systémů Linux, UNIX, Windows a MacOS. R nabízí rozsáhlou sadu statistických a dolovacích metod jako jsou lineární a nelineární modelování, klasické statistické testy, shlukování a klasifikace. Výhodou je snadná tzv. balíčková rozšiřitelnost tohoto prostředí pomocí jazyka R. Dále pro náročné výpočetní úlohy je možné využít podpory programů implementovaných v C, C++ nebo Fortranu. Další z výhod R je vlastní dokumentační nástroj podobný LaTexu a jednoduchá možnost publikace kvalitních grafů včetně složitých matematických výrazů a znaků. R obsahuje i řadu balíčků, které implementují jednoduché grafické uživatelské rozhraní zaměřených na dolování znalostí, jako jsou například Deducer, RWe nebo Rattle GUI (The R Foundation for Statistical Computing, 2016). Nevýhodou tohoto prostředí může být zmatečnost a menší robustnost, díky rozsáhlé řadě verzí prostředí a rozšiřujících balíčků. Další z nevýhod je také neefektivnost a velká časová náročnost zpracování velkých objemů dat (Lantz, 2013).

4 Použitá datová kolekce

4.1 Obecné informace

Zvolená datová kolekce dat, na které byly realizovány praktické experimenty pro porovnání metod částečně řízeného strojového učení, pochází z rozsáhlé textové množiny uživatelských hodnocení produktů na serveru *Amazon.com*. Analyzovaná data jsou zaměřena na uživatelská hodnocení u jednotlivých produktů, které jsou na serveru Amazon nabízeny zákazníkům ke koupi. Jedná se o kolekci nashromážděných textových recenzních příspěvků zákazníků u produktů v různých kategoriích, jako jsou například elektronika, hudba, video nebo knihy. Použitá datová kolekce byla stažena z následujících webových stránek http://www.cs.jhu.edu/~mdredze/datasets/sentiment/, kde je volně dostupná ke stažení díky pánům Blitzerovi a kol., kteří ji použili ve své práci týkající se analýzy sentimentu (Blitzer a kol., 2007), a následně volně poskytli na internet pro výzkumné účely. Tato data byla nashromážděna pomocí skriptů přímo z amerického serveru *Amazon.com* a roztříděna podle kategorií produktů. Celková velikost dat činí po rozbalení 4,71 GB a obsahuje celkem 27 kategorií, které obsahují zcela rozdílný počet hodnocení. Vzhledem k experimentální formě této diplomové práce bylo cílem zpracování co největšího množství dat s ohledem na výpočetní výkon použité výpočetní techniky a reálnou dobu zpracování experimentů. Nejprve byla prozkoumána velikost všech 27 kategorií a následně byly vyřazeny všechny kategorie, které obsahovaly méně než deset tisíc uživatelských hodnocení. Po vyřazení rozsahově méně obsáhlých kategorií zůstalo pouze sedm kategorií (viz Tab. 3). Největší počet recenzí obsahuje kategorie s hodnocením knih, kde se celkový počet uživatelských hodnocení produktů pohybuje okolo půl milionu recenzí. Mezi další početnější kategorie patří kategorie hudba a DVD, video a elektronika.

Tab. 3 Celkový počet zakaznických hodnocení pro jednotlivé kategorie datové kolekce

Kategorie	**Celkový počet hodnocení**
knihy	500 000
hudba	174 180
DVD	124 438
video	36 180
elektronika	23 009
kuchyňské potřeby	19 856
hračky	13 147

Vzhledem k omezenému rozsahu práce, časovým možnostem a použité výpočetní technice, byla použita datová sada hodnocení elektroniky, která obsahuje celkem 23 009 zákaznických hodnocení. Jedná se o poměrně rozsáhlou kolekci dat, která

se jevila jako zcela dostačující pro realizaci plánovaných experimentů i s ohledem na použitou výpočetní techniku a reálnou dobu zpracování.

Charakteristika dat

Jak již bylo uvedeno na začátku této kapitoly, datová množina má textový charakter v podobě uživatelských hodnocení elektronických produktů na serveru *Amazon.com*. Vzhledem k tomu, že se jedná o zahraniční server, všechna uživatelská hodnocení byla napsána v anglickém jazyce. Jednotlivá uživatelská hodnocení z datové kolekce byla napsána v přirozeném jazyce a obsahují tedy celou řadu nedostatků typických pro text v přirozené formě. Textová data v přirozeném jazyce mohou typicky obsahovat celou řadu překlepů, gramatických chyb, špatného slovosledu či zcela nesmyslných slov. Všechna hodnocení byla stažena pomocí skriptů přímo z internetu, kde byla uložena ve formě struktury xml, která obsahovala následující údaje pro jednotlivá hodnocení:

- unikátní identifikační číslo hodnocení
- standardní identifikační číslo produktu
- jméno produktu
- jméno kategorie produktu
- počet označení daného hodnocení za nápomocné od jiných uživatelů
- číselné hodnocení produktu v intervalu od 1 (nejhorší) do 5 (nejlepší)
- titulek hodnocení
- text hodnocení produktu
- datum hodnocení
- země původu hodnotícího
- uživatelské jméno hodnotícího uživatele

Za účelem získání samotného textového hodnocení z této struktury bylo nutné aplikovat několik kroků předzpracování, které budou podrobněji popsány v kapitole popisující proces přípravy dat (viz kapitola 5.3). Jako nejpodstatnější údaje z xml struktury byly určeny tři položky, titulek hodnocení, samotný text hodnocení produktu a číselné hodnocení daného produktu. Titulek hodnocení produktu byl označen jako podstatný díky tomu, že často obsahoval anglická slova (například *outstanding, horrible, perfect*), která by měla pomáhat při rozhodování o zařazení hodnocení do patřičné třídy. Dalším podstatným údajem z xml struktury byl samozřejmě obsah textového hodnocení produktu. Číselné hodnocení produktu bylo použito za účelem rozřazení jednotlivých hodnocení do tříd. Celkem byly vytvořeny dvě třídy uživatelských hodnocení, negativní a pozitivní. Negativní třída vznikla sloučením všech uživatelských hodnocení s číselným hodnocením 1 a 2. Uživatelská hodnocení s číselným hodnocením 4 a 5 pak po sloučení vytvořily pozitivní třídu. Po aplikaci metod předzpracování a sloučení hodnocení byla získána samotná uživatelská hodnocení, která jsou vzájemně disjunktní. Tedy mohou vždy

patřit právě do jedné ze dvou tříd a to buď do pozitivní třídy, nebo do třídy negativní.

Příklady uživatelských hodnocení

V rámci popisu použité datové množiny a její charakteristiky je vhodné ukázat několik příkladů, jak vypadala zpracovávaná uživatelská hodnocení. Následující prezentované ukázkové příklady jsou data po extrahování všech podstatných informací z xml struktury a to bez dalších úprav. Tudíž si lze v příkladech všimnout běžných nedostatků vyskytujících se v psaném přirozeném jazyce a speciálních znaků. Číslo na prvním řádku značí číselné hodnocení produktu, hned za ním se nachází titulek hodnocení a poté na dalších řádcích následuje již samotné znění textového hodnocení produktu.

1.0 Waste Of Money
I bought these discs at CompUSA because I needed a few before I got the ones I ordered online. Well, I just wated a good 25 bucks because out of 5 discs, I got one that burned good. Not much bang for the buck. I have burned many discs but never Memorex, usually Verbatim or Ridata or MAM. Very disgruntled with these discs making coasters. Maybe my burners don't like them but whatever the case, I won't buy them ever again. I thought Memorex made good products

2.0 Coming Apart at the Seams
In theory an excellent product -- room for lots of disks, relatively compact and attractive HOWEVER I purchased three of these (2 for me 1 for a friend) I can't say how his is doing but both of mine have faulty zippers, making it impossible to close the binders and therefore losing much of their attractiveness and compactness. (...). All in all, a disappointment

4.0 Works well
Bought this card along with a Rebel XT as a gift for my girlfriend. It's been performing without a hitch so far, though on burst mode, there is transfer lag: in that, the pictures take a lot faster than can be transferred to the card, and so after the burst, the pictures are still being written to memory

5.0 Good memory card
I bought this card to go with my Canon 30D and have not been let down. I can get about 500 JPEGs and 300 RAW. SanDisk makes highly constant products. I can set the 30D to take high speed continuos shooting and take 5 RAW pic's and not have the card slow down my camera. Very happy with it

Statistické údaje

Před samotnou realizací experimentů na zvolené datové množině byla provedena její statistická analýza za účelem získání základního přehledu o zpracovávané množině dat. Data byla analyzována samostatně po jednotlivých třídách. Nejprve byla analyzována pozitivní třída a následně negativní třída. Analýza byla realizována se zaměřením především na získání statistických informací o počtu unikátních slov a slovní délce uživatelských hodnocení v jednotlivých třídách. Následující statistické údaje byly získány pomocí nástroje Microsoft Excel na již extrahovaných hodnoceních, po odstranění duplicitních příspěvků a nepísmenných znaků (viz kapitola 5.3). Výsledky statistické analýzy jsou zobrazeny v následujících tabulkách, statistické údaje pro pozitivní třídu v tabulce 4 a pro negativní třídu v tabulce 5.

Tab. 4 Statistické informace pro pozitivní třídu uživatelských hodnocení

Celkový počet pozitivních hodnocení	15 915
Počet slov nejkratšího hodnocení	4
Počet slov nejdelšího hodnocení	3 007
Průměrná délka hodnocení	104.90644046497
Počet unikátních slov	28 130

Tab. 5 Statistické informace pro negativní třídu uživatelských hodnocení

Celkový počet negativních hodnocení	4 475
Počet slov nejkratšího hodnocení	7
Počet slov nejdelšího hodnocení	1 756
Průměrná délka hodnocení	113.72156424581
Počet unikátních slov	15 438

5 Metodika

5.1 Postup práce

V rámci této diplomové práci je zkoumána problematika částečně řízeného strojového učení a jeho porovnání s ostatními druhy strojového učení, neřízeným a řízeným strojovým učením na základě experimentů. Za tímto účelem byla realizována řada experimentů nad zvolenou datovou kolekcí. Zvolená datová kolekce obsahovala poměrně velkou množinu textových uživatelských hodnocení produktů z oblasti elektroniky napsaných v přirozeném anglickém jazyce. Jednotlivá uživatelská hodnocení vždy patřila právě do jedné ze dvou cílových tříd (buď do pozitivní, nebo negativní). V realizovaných experimentech je tedy zkoumán problém tzv. binární klasifikace, kdy existují pouze dvě cílové třídy, do kterých mohou být jednotlivé datové položky na základě klasifikace přiřazeny.

Před samotnou realizací experimentální práce byla nejprve celá datová kolekce extrahována z xml struktury, ve které byla stažena z internetu. Následně byla použita celá řada metod na přípravu a předzpracování dat (odstranění duplicit, stop slov, atd. viz kapitola 5.3), především za účelem snížení výpočetních nároků při práci s touto kolekcí. Na konci fáze přípravy dat byla data převedena do vektorové reprezentace, která se využívá pro jednotnou reprezentaci jednotlivých uživatelských hodnocení pro algoritmy strojového učení.

Následně byly provedeny experimenty na demonstrační datové sadě, která byla vytvořena náhodným výběrem podmnožiny dat z reálné datové kolekce o velikosti jednoho tisíce uživatelských hodnocení. Demonstrační datová množina měla za cíl především prozkoumat problematiku částečně řízeného učení v porovnání s ostatními druhy strojového učení (řízeného a neřízeného učení). Zároveň tyto demonstrační experimenty sloužily k nalezení a výběru vhodných nástrojů pro realizaci experimentů nad reálnou datovou kolekcí. Demonstrační experimenty v neposlední řadě sloužily pro účely částečného ověření správnosti zvolených technik předzpracování dat a získání přehledu o výpočetní náročnosti použitých algoritmů.

Jako poslední byly provedeny experimenty pro tři různě velké množiny dat z reálné datové kolekce, které měly za úkol ověřit realizovatelnost navržených postupů a chování částečně řízeného učení na reálných datech. Zároveň v této stěžejní části diplomové práce byly získány výsledky binární klasifikace pro následné porovnání všech tří zkoumaných metod strojového učení. Výsledky těchto experimentu jsou interpretovány, vyhodnoceny a diskutovány v následujících kapitolách této práce (viz kapitoly 6 a 7).

5.2 Použitá výpočetní technika

Vzhledem k tomu, že praktická část této diplomové práce je založená na experimentech, je vhodné si definovat prostředí, ve kterém byly experimenty realizová-

ny. Veškeré předzpracování dat bylo realizováno na notebooku Lenovo s operačním systémem Microsoft Windows 10. Dále na notebooku Lenovo byly realizovány demonstrační experimenty a experimenty pro datovou sadu čítající 8 000 textových hodnocení produktů. Notebook Dell s operačním systémem Windows 7 byl vypůjčen z důvodu nedostatku operační paměti u notebooku Lenovo pro zpracování datové sady obsahující 16 000 textových hodnocení produktů. Výkon použité výpočetní techniky má zásadní vliv na výsledky experimentu, především na dobu trvání jednotlivých výpočtů. Další omezení se pak týkají velikosti množin zpracovávaných dat, který bylo možné zpracovat s ohledem na použitou výpočetní techniku v reálném čase.

notebook: Lenovo Thinkpad Edge E520

- operační systém: Microsoft Windows 10 Pro 64-bit
- procesor: Intel(R) Core(TM) i5-2430M CPU @ 2.40GHz, 2401 Mhz
- operační paměť: 4 GB, DDR3 RAM 1333 MHz
- pevný disk: 500 GB SATA 7200 otáček/min.

notebook: Dell Latitude E5440

- operační systém: Microsoft Windows 7 Pro 64-bit
- procesor: Intel(R) Core(TM) i5-4310U CPU @ 2.00GHz, 2.60 GHz
- operační paměť: 8 GB, DDR3 RAM 1333 MHz
- pevný disk: 500 GB SATA 5400 otáček/min.

5.3 Příprava dat

Fáze přípravy dat je velice důležitou a zároveň nezbytnou součástí experimentů, která má zásadní vliv na výsledky experimentů. Existuje rozsáhlá řada metod a postupů, které lze na předzpracování dat použít, ale zároveň neexistují žádná přesná pravidla, která by zaručila dokonalé výsledky experimentů. Volba těchto metod se obvykle odvíjí především od charakteru zdrojových dat a požadovaných výsledných informací, které chceme pomocí metod strojového učení z těchto dat získat. Dalším rozhodovacím faktorem při volbě metod předzpracování je také velikost vstupní datové kolekce s ohledem na výpočetní výkon. S rostoucí velikostí vstupní kolekce dat jsou samozřejmě kladeny vyšší požadavky na výpočetní výkon, nutný pro jejich zpracování. Vzhledem k textovému charakteru vstupních dat, který se typicky vyznačuje velkou slovní dimenzionalitou a relativně nízkému výpočetnímu výkonu notebooku autora práce, bylo nutné aplikovat celou řadu metod za účelem snížení dimenzionality dat. V této kapitole budou popsány všechny kroky a metody předzpracování, které autor práce použil a realizoval nad zvolenou datovou kolekcí.

Extrakce dat z xml struktury

Prvním krokem přípravy dat byla extrakce důležitých informací z xml struktury ve, které byly staženy z internetu. Pro tyto účely autor implementoval ve skriptovacím jazyce python skript *extractData.py*, který extrahuje z xml struktury příslušné údaje. Zdrojový kód skriptu *extractData.py* je přiložen v sekci příloh této práce (viz příloha B). Před samotnou extrakcí vybraných atributů, bylo nutné upravit xml strukturu do validního stavu. To bylo provedeno přidáním kořenového elementu „*<reviews>*“ na začátek a přidáním ukončujícího kořenového elementu „</reviews>“ na konec datového souboru. Dále bylo nutné odstranit nevalidní značku „SUB“, která byla obsahem několika uživatelských hodnocení a z neznámého důvodu způsobovala neočekávaný konec extrakce dat. Odstranění všech výskytů této nevalidní značky z datového souboru, bylo provedeno pomocí funkce *vyhledat a nahradit* v textovém editoru Sublime Text. Po uvedení xml struktury do validního stavu, již bylo možné přistupovat k jejím jednotlivých atributům a následně provést jejich extrakci.

Samotná extrakce dat byla fixně nastavena na získání tří zvolených atributů, a to konkrétně, číselného hodnocení, názvu hodnocení a textového uživatelského hodnocení. Během extrakce názvu hodnocení a samotného textového hodnocení, již byly aplikovány první metody na snižování dimenzionality dat. Nejprve byly odebrány všechny čísla, nepísmenné a speciální znaky (jako jsou například, %, @, #, &,?, +, atd.), které obvykle nenesou žádnou užitečnou informaci z pohledu analýzy textu. V mnoha odborných studiích bylo dokázáno, že odstraněním těchto znaků nedojde k podstatné ztrátě informací důležitých pro účely dolování znalostí, ale právě naopak odstranění nevalidních znaků má pozitivní vliv na další analýzu dat (např. Sang a kol, 2016 nebo Kirk, 2014). Dalším krokem provedeným za účelem snížení dimenzionality dat bylo převedení všech velkých písmen v hodnoceních na malá písmena. Tato technika je běžně využívána, protože má pozitivní vliv na snížení celkové dimenzionality dat a to bez podstatného vlivu na ztrátu informace.

Následně na základě číselného hodnocení byla hodnocení rozdělena podle jejich příslušné třídy do dvou různých souborů. Výsledkem extrakce dat pomocí programu *extractData.py* jsou dva soubory, soubor *allpos.txt* obsahující všechny pozitivní hodnocení a soubor *allneg.txt* obsahující všechny negativní hodnocení. Zdrojový kód skriptu *extractData.py* je umístěn v sekci příloh (viz příloha B).

Odstranění duplicit

Po získání extrahovaných hodnocení bylo nutné zkontrolovat, zda zdrojová data neobsahovala nějaká duplicitní hodnocení. Pro tuto kontrolu byl využit nástroj Microsoft Excel, který nabízí zcela jednoduchou funkci na odstranění duplicit v datech. Použité zdrojové soubory s hodnocením produktů obsahovaly celou řadu duplicitních uživatelských recenzí. Konkrétně se jednalo o 2619 hodnocení, která byla následně odstraněna.

Snižování slovní dimenzionality dat

Odstraněním duplicitních uživatelských hodnocení byl získán konečný počet hodnocení, která budou dále zpracována. Výsledný počet pozitivních hodnocení výrazně převyšoval počet negativních hodnocení. Konkrétně bylo získáno ze vstupního souboru 4 475 negativních uživatelských hodnocení a 15 915 pozitivních hodnocení. Pro další zpracování byly vytvořeny skupiny hodnocení po 4 000 hodnocení, které byly vytvořeny náhodným výběrem z původních souboru. Před samotným použitím metod pro snižování slovní dimenzionality dat, za účelem snížení výpočetních nároků byl nejprve získán základní přehled o počtu unikátních slov a četnosti jejich výskytu. Tento přehled byl zásadní pro určení dalšího postupu přípravy dat a při výběru metod. Přehledy o počtu unikátních slov a absolutní četnosti jejich výskytů byly vytvořeny samostatně pro sadu 4 000 negativních hodnocení a pro sadu 12 000 pozitivních hodnocení. Pro získání statistických údajů o počtu unikátních slov a jejich absolutní četnosti byl použit volně dostupný internetový nástroj *word_count* (Bruce a kol., 2002). Tato webovou aplikaci byla zvolena především díky tomu, že jako jedna z mála volně dostupných aplikací dokázala najednou zpracovat všech 12 000 pozitivních recenzí. Aplikace se vyznačuje jednoduchým uživatelským rozhraním, u kterého stačí do textového pole nakopírovat zkoumaný text a poté ho odeslat na server. Následně je zobrazena zpracovaná odpověď ze serveru v novém okně jako tabulka unikátních slov a jejich absolutních četností výskytu. Vzhledem k tomu, že podle těchto informací byl zvolen další postup přípravy dat a metody pro snižování dimenzionality, bylo nutné získané údaje nezávisle ověřit jiným nástrojem. Konkrétně byl použit nástroj VecText, který nám ve škole prezentoval doc. Ing. František Dařena, Ph.D. a pracovali jsme s ním na cvičeních v předmětu IPI. Vzhledem k tomu, že výstupy z obou aplikací byly identické, bylo možné pokračovat v další práci.

Tab. 6 Nejčetnější slova pro pozitivní a negativní sadu hodnocení (Slova jsou již po převodu na malá písmena, například položka „i“ reprezentuje anglické zájmeno „I“).

	Pozitivní hodnocení		Negativní hodnocení	
Pořadí	**Slovo**	**Četnost**	**Slovo**	**Četnost**
1.	the	65219	the	24599
2.	i	35517	i	13865
3.	and	33321	to	12782
4.	to	32189	a	10829
5.	a	31265	and	10749
6.	it	26301	it	10560
7.	is	21021	of	6495
8.	for	17630	this	6451
9.	this	16799	is	6342
10.	of	16742	for	5269

Celkový počet unikátních slov u negativních příspěvků dosahoval počtu 14 815 slov a pro sadu pozitivních příspěvků 28 130 slov. Jednotlivé absolutní četnosti výskytu pro prvních deset nejčastěji se vyskytujících se slov v pozitivní a negativní třídě jsou uvedeny v následující tabulce (viz Tab. 6). Z předcházející tabulky je patrná skutečnost, že s největší četností se v hodnoceních vyskytovala pouze tzv. „stop slova", jako jsou například spojky, předložky nebo zájmena. Tyto obecná slova nenesou žádnou důležitou informaci z pohledu dolování znalostí v rámci této práce, a proto byla odstraněna. Pro odstranění stop slov byl vytvořen seznam stop slov na základě výběru z obecných seznamů anglických volně dostupných na internetu v projektu *stop-words* (Balucha, 2011). Vytvořený seznam obsahoval 175 obecných slov, především předložek, zájmen a spojek, výčet všech slov z vytvořeného seznamu stop slov je umístěn v přílohách této práce (viz. Tab. 22). Ze všech slov vytvořeného seznamu stop slov byl odebrán znak apostrofu „'", který byl ve všech textových hodnoceních nahrazen prázdným znakem. Vzhledem k tomuto nahrazení pak hodnocení obsahovala slova bez apostrofu (jako například *wouldnt* nebo *its)*, která by bez odstranění apostrofu ze seznamu stop slov nebyla odstraněna z textových hodnocení. Po odstranění stop slov, hodnocení obsahovala stále mnoho obecných slov, proto byl vytvořen dodatečný seznam stop slov. Dodatečný seznam byl vytvořen na základě standardního seznamu stop slov v programu Cluto, který je běžně k dispozici s tímto programem a obsahuje 427 obecných slov. Dodatečný seznam obsahoval řadu duplikátních položek z prvního seznamu obecných stop slov a také řadu slov, která podle by podle autorova názoru mohli mít vliv při rozhodování o příslušnosti hodnocení do cílových tříd (jako například, *greatest, great*, *problem* nebo *problems*). Proto nejprve byly odstraněny duplikátní položky a následně byla vybrána slova, která budou v hodnoceních ponechána pro další zpracování. Výsledný upravený seznam obsahoval nakonec 231 obecných stop slov a je přiložen v sekci příloh (viz Tab. 23).

Pro odstranění stop slov autor vytvořil jednochý skript removeStop.py implementovaný ve skriptovacím jazyce Python, který je platformě nezávislý. Zdrojový kód je taktéž přiložen v přílohách této práce. Jedná se o velice jednoduchý skript, který se ovládá pomocí příkazové řádky pomocí následujících parametrů:

python removeStop.py < seznam_stopslov > < zdroj > < výstup >

Jako první parametr si skript načte seznam nežádoucích slov, která mají být odstraněna ze zdrojového souboru, následně si načte samotný zdrojový soubor a název výstupního souboru. Poté projde zdrojový soubor a odstraní z něj všechny nežádoucí *stopslova* slova, která se v něm vyskytují, a zbylá slova vytiskne do výsledného souboru.

Dalším běžným postupem v oblasti přípravy dat pro metody strojového učení je odebrání termů s určitou nízkou četností výskytu. Tato metoda je v praxi běžně využívána za účelem snížení slovní dimenzionality a zároveň slouží pro odebrání gramaticky špatně napsaných slov, slov vzniklých díky překlepům apod. Vzhledem k tomu, že všechna uživatelská hodnocení byla napsána v přirozeném anglickém

jazyce, obsahovala celou řadu překlepů a gramaticky špatně napsaných slov nebo neexistujících slov. Ovšem přesná hranice absolutní četnosti pro odstranění slov je závislá na požadovaných výsledných informacích a použitých metodách strojového učení. Po prostudování prací týkající se této oblasti dolování znalostí (jako je například, Schakel, Wilson, 2015) a konzultaci s vedoucím práce doc. Ing. Janem Žižkou, CSc. a, byla odstraněna všechna slova s absolutní četností výskytu 3 a menší. Odstraněním stop slov a slov s nízkou absolutní četností došlo k výraznému snížení počtu unikátních slov nutných pro reprezentaci uživatelských hodnocení u obou tříd (pozitivní a negativní). U negativní třídy klesl počet unikátních položek na 5 248 slov a u pozitivní třídy na 8 823 slov.

Během procesu snižování slovní dimenzionality vstupních dat, bylo provedeno odstranění všech slov s písmennou délkou 3 a menší. Tyto slova obsahovala především spojky, předložky a zkratky, které nebyly zahrnuty do seznamu stop slov. Odstranění těchto slov je zcela běžnou a používanou metodou v případě kdy není výhodné využití třeba právě zkratek (například NHL, Dr, doc, atd.) při dolování znalostí z dat. Tyto slova je vhodné zahrnout například při řešení problému kategorizace dokumentů do různých oblastí. Nicméně z pohledu problematiky řešené v této práci, podle autorova názoru nedochází při odstranění těchto slov, k výraznější ztrátě podstatných informací. Odstranění slov s malou délkou bylo realizováno pomocí skriptu *removeStop.py* během procesu odstraňování stop slov.

Předcházející krok byl původně posledním krokem snižování slovní dimenzionality dat a po spojení souborů pozitivních a negativních hodnocení byly provedeny první experimenty na těchto datech, nicméně hned v další fázi se ukázalo, že předzpracování dat potřebuje menší vylepšení. Při výpočtu důležitých atributů pro rozhodování o příslušnosti hodnocení do pozitivní nebo negativní třídy, byly jako nejdůležitější parametry vyhodnoceny názvy počítačových komponent, elektroniky a firem (jako například, canon, windows, router, apple atd.), které výrazně zkreslovaly výsledky, protože tyto slova jsou z pohledu dané problematiky rozhodování o příslušnosti do cílových tříd zcela irelevantní. Na základě této skutečnosti byly vytvořeny další dva seznamy stop slov, jejichž účelem je alespoň částečně potlačit některé slova, která byla vyhodnocena jako důležitá při rozhodování o příslušnosti uživatelských hodnocení do cílových tříd. První seznam obsahuje názvy elektronických produktů a pojmy z této oblasti (viz Tab. 24), které byly nejčastěji vyhodnocovány jako důležité atributy. Druhý seznam tvoří jména výrobců a prodejců těchto komponent (viz Tab. 25). Seznamy byly vytvořeny podle slovníku na internetu, podle kategorií produktů na serveru *amazon.com* a především na základě vstupních dat a podle jejich vyhodnocené důležitosti. Odstranění těchto nežádoucích slov bylo realizováno pomocí skriptu *removeStop.py* během odstraňování obecných stop slov. Po odstranění názvů firem a elektronických produktů a pojmů z uživatelských hodnocení byla vyhodnocena jako důležitá slova (jako například, „terrible“ v překladu otřesný nebo „disappointed“ v překladu zklamaný, viz Tab. 7), která by měla mít správný vliv při rozhodování o příslušnosti uživatelských hodnocení do cílových tříd. Po aplikování veškerých metod na snížení slovní dimenzionality hodnocení a spojení hodnocení z obou tříd, klesl celkový počet uni-

kátních položek na 7 879 slov. Samozřejmě by šlo v této oblasti dále pokračovat a zkoumat vliv dalších uprav na celkovou úspěšnost klasifikace, ale protože zkoumání metod předzpracování a přípravy dat nebylo cílem této práce, byla v tomto stavu ukončena fáze snižování slovní dimenzionality dat.

Tab. 7 Výsledné ohodnocení atributů pomocí algoritmů pro výběr důležitých atributů před a po odebrání názvů firem a produktů

	Před odebráním názvů firem a produktů		**Po odebrání názvů firem a produktů**	
Pořadí	**Chí kvadrát**	**Gain ratio**	**Chí kvadrát**	**Gain ratio**
1.	company	company	terrible	refuses
2.	windows	windows	beware	flashing
3.	devices	support	perfect	intermittently
4.	canon	devices	disappointed	recommends
5.	flashing	terrible	unable	unreadable
6.	beware	beware	refuses	beware
7.	terrible	card	replacement	terrible
8.	unable	canon	plenty	unable
9.	support	disappointed	supposed	trash
10.	exchange	unable	crisp	beautifully

Tvorba vektorové reprezentace uživatelských hodnocení

Další částí přípravy dat před samotnou realizací experimentu byl převod všech uživatelských hodnocení na jejich vektorovou reprezentaci ve formě matice typu T-D (term – dokument). Pro převod byl použit na internetu volně dostupný skript doc2mat, který vytvořil autor programu Cluto George Karypis (Karypis, 2006). Tento skript je implementovaný v interpretovaném programovacím jazyce Perl a je tedy platformě nezávislý. Výhodou tohoto skriptu je také vysoká rychlost zpracování i poměrně velkého objemu dat. Dalším důvodem proč byl použit tento skript, je fakt, že výsledný soubor v maticovém formátu lze dále zpracovat v nástroji Cluto, který byl v práci využit na aplikaci metod neřízeného strojového učení. Skript se spouští přes příkazovou řádku, kde se mu pomocí parametrů zadá cesta ke zdrojovému a cílovému souboru a případná parametrizace skriptu v následujícím formátu:

perl doc2mat [parametry] textový_soubor maticový_soubor

Parametrizace skriptu je poměrně omezená obsahuje pouze několik možností nastavení průběhu zpracování zdrojového textového souboru, jako jsou například parametry použití stemmingu nebo vymazání stop slov. Podrobnější popis parametrizace je k dispozici přímo v návodu od autora nástroje CLUTO (Karypis, 2006). Textový soubor značí zdrojový soubor, který obsahuje všechny zpracovávaná uživatelská hodnocení, přičemž každý řádek souboru reprezentuje jedno hodnocení.

Výsledný maticový soubor vzniklý převodem zdrojového souboru obsahuje vektorovou maticovou reprezentaci T-D udávající frekvenci výskytu jednotlivých slov v daném hodnocení. Pro účely přípravy dat byl pro všechny experimenty tento skript použit s následující parametrizací:

perl doc2mat –nostem –nostop textový_soubor maticový_soubor

Kde parametr *-nostem* zakazuje skriptu použití stemmingu a parametr *-nostop* zakazuje použití standardního seznamu stop slov programu CLUTO.

Vzhledem k tomu, že nástroj CLUTO pokrýval pouze oblast neřízeného strojového učení, bylo nutné převést datovou reprezentaci do univerzálního formátu CSV, který sloužil jako vstup pro zbylé použité programové nástroje Weka a Keel. Pro převod vytvořil autor skript mat2csv.py implementovaný v multiplatformním interpretovaném jazyce Python. Zdrojový kód tohoto skriptu je přiložen v sekci příloh této práce (viz příloha H). V podstatě se jedná o velice jednoduchý skript ovládaný pomocí konzole, který vyžaduje tři následující vstupní parametry:

python mat2csv vstup.mat.label vstup.mat vystup.csv

Oba vstupní soubory jsou vytvořeny nástrojem doc2mat, který převádí textové soubory do maticové reprezentace. Soubor vstup.mat.label obsahuje jednotlivá unikátní slova, která se vyskytovala v původním textovém souboru. Soubor vstup.mat obsahuje maticovou reprezentaci jednotlivých uživatelských hodnocení pomocí řady dvojic čísel, kde první číslo slouží jako identifikátor řádku unikátního slova z předchozího souboru a druhé číslo udává počet výskytů daného slova v tomto konkrétním uživatelském hodnocení. Je vhodné zmínit, že dvojice čísel reprezentující slova v dokumentech a jejich četnost jsou zobrazeny pouze pro četnosti výskytu 1 a vyšší.

Příprava dat pro experimenty částečně řízeného učení

Poslední částí přípravy dat je příprava vstupních souboru pro experimenty částečně řízeného učení. Tyto experimenty byly realizovány v nástroji KEEL, který bohužel zaostává v možnostech hromadné úpravy dat a nabízí možnosti editace jednotlivých záznamů. Pro účely přípravy dat pro tyto experimenty bylo nutné vytvořit skript *createExpData.py*, který bude vytvářet soubory ve formátu KEEL podle požadované parametrizace. Zdrojový kód tohoto scriptu je umístěn v sekci příloh (viz příloha I). Tvorba vstupních dat probíhá ve dvou krocích z důvodu zachování stejné testovací množiny pro daný experiment. V prvním kroku jsou nejprve sestaveny ze všech uživatelských hodnocení trénovací a testovací množiny dat. Rozdělení dat probíhá v předem zvoleném poměru (66 % hodnocení pro trénování a 34 % hodnocení na testování, viz podkapitola 5.4.1), který je fixně nastaven ve zdrojovém kódu skriptu. Skript se spouští pomocí konzolového příkazu s následujícími parametry:

createExpData.py < pozitivní_hodnocení.dat > < negativní_hodnocení.dat > < celkem_pozitivních > < celkem_negativních>

Vstupní parametry jsou soubory s vektorovou reprezentací pozitivních a negativních uživatelských hodnocení bez popisných hlaviček a celkové počty zpracovávaných uživatelských hodnocení pro obě třídy. Výstupem skriptu jsou celkově tři soubory *test.dat*, *trainpos.dat* a *trainneg.dat*. Soubor *test.dat* obsahuje vybranou množinu dat, která bude použita pro testování a zbylé dva soubory obsahují množiny pozitivních a negativních příspěvků, které budou použity ve fázy trénování.

Ve druhém kroku jsou následně použity nově vygenerované soubory jako vstup pro tvorbu sady vstupních souborů pro jednotlivé experimenty. Skript se ve druhém kroku znovu spouští pomocí konzolového příkazu s rozdílem většího množství parametrů:

createExpData.py < –old > < trainpos.dat > < trainneg.dat > < test.dat > < celkem_pozitivních > < celkem_negativních> < počet_označených_prvků > < hlavička_trénovacího_souboru > < hlavička_testovacího_souboru > < prefix_souborů >

První nový parametr „-old“ označuje, že se jedná o druhý krok tvorby vstupních souborů. Dále následují cesty ke vstupních souborům z předcházejícího kroku a celkový počet pozitivních a negativních hodnocení použitých v prvním kroku. Dalším novým parametrem je „počet_označených_prvků“, který udává přesný počet označených položek, které budou použity pro tvorbu trénovacích dat. Další parametry jsou cesty k souboru s vygenerovanou hlavičkou z nastroje KEEL pro trénovací a testovací soubory. Vygenerované hlavičky KEEL souborů jsou použity z předchozích experimentů pro řízené učení s tím rozdílem, že hlavička pro trénovací soubory je doplněna o nedefinovanou hodnotu „unlabeled“. Tato hodnota je umístěna mezi nominální hodnoty atributu *document_class* a pro příslušné hodnocení značí, že není známa jeho cílová třída. Poslední parametr „prefix_souborů“ slouží pro určení prefixu jména vytvořených souborů, který musí být stejný jako jméno vytvořené datové sady v nástroji Keel. Konečným výstupem tohoto skriptu pro částečně řízené učení je 30 souborů, které jsou rozděleny do tří kategorií po 10. Celkový počet souborů je 30, protože na jedno provedení experimentu jsou třeba 3 soubory a celkový počet opakování byl stanoven na 10 pokusů. U všech vygenerovaných souborů, které budou použity pro experimenty v nástroji KEEL byl dodržen předem určený poměr počtu pozitivních a negativních uživatelských hodnocení.

Soubor obsahující testovací data (s koncovkou *tst.dat*) slouží pro otestování natrénovaného klasifikátoru. Testovací soubor vždy obsahuje předgenerovanou hlavičku z nástroje KEEL a 34 % uživatelských hodnocení z celkového počtu hodnocení použité datové množiny. Soubor obsahující trénovací data (s koncovkou *tra.dat*) obsahuje hlavičku pro trénovací soubor a zbylých 66 % uživatelských hodnocení z celkového počtu. Obsah tohoto souboru je upraven na základě parametru o počtu označených prvků, který definuje kolik označených hodnocení má obsahovat.

Z obou tříd (pozitivní a negativní) je náhodně vybrána polovina celkového počtu označených uživatelských hodnocení. Označená část trénovacích dat je tedy vždy tvořena stejným počtem pozitivních a negativních hodnocení. Zbylým hodnocením je následně změněna hodnota atributu „document_class“ na nedefinovanou hodnotu „unlabeled“. Trénovací soubor tedy obsahuje v první části stanovený počet označených hodnocení a následně zbytek neoznačených hodnocení. Třetí vygenerovaný soubor (s koncovkou *trs.dat*) slouží pro kontrolu klasifikace v trénovací fázy a obsahuje stejnou předgenerovanou hlavičku a uživatelská hodnocení jako trénovací soubor s tím rozdílem, že všechna hodnocení v tomto souboru jsou označená (u všech hodnocení je známa hodnota atributu *document_class*).

5.4 Návrh experimentů

V následující části je popsán navržený postup průběhu jednotlivých experimentů nad zvolenou datovou kolekcí. Nejprve je popsána volba velikosti datových množin pro jednotlivé experimenty s ohledem na možnosti použité výpočetní techniky. Následně jsou popsány použité postupy při realizaci experimentů pro všechny tři zkoumané druhy strojového učení, částečně řízené, řízené a neřízené učení.

5.4.1 Určení velikosti datových množin

Určení velikosti použitých datových množin je jedním z nejdůležitějších kroků před samotným započetím experimentů. Velikost a složení datové množiny může výrazně ovlivnit výsledky experimentů. Při určování velikosti zkoumané datové množiny je ovšem důležité brát ohled na možnosti použité výpočetní techniky. Vzhledem k tomu, že v autor měl k dispozici pro realizaci experimentů běžný notebook, bylo nutné tuto skutečnost zvážit při volbě velikosti množiny dat. Autor nejprve provedl několik testovacích experimentů s cílem získání představy o časové a výpočetní náročnosti jednotlivých algoritmů strojového učení a na jejich základě určil následující velikosti jednotlivých datových množin pro další zpracování (viz Tab. 8). Velikost demonstační datové sady byla stanovena na 1 000 náhodně vybraných uživatelských hodnocení produktů z celkové datové kolekce. Následně byly vytvořeny tři reálné datové sady, dvě po 8 000 uživatelských hodnocení a jedna o velikosti 16 000 užitavelských hodnocení. Složení datových množin z pohledu jednotlivých tříd udává v Tab. 8 poměr zastoupení tříd, který značí poměr pozitivních uživatelských hodnocení vůči těm negativním. U demonstrační sady a první reálné datové sady byl poměr hodnocení stanoven na vyvážený, obě sady tedy obsahují 50 % pozitivních a 50 % negativních hodnocení. Vzhledem k tomu, že v datové kolekci celkový počet pozitivních příspěvků výrazně převyšoval (přibližně 75 % pozitivních a 25 % negativních) celkový počet negativních příspěvků, další reálné sady již byly vytvořeny se zachováním tohoto poměru (75:25). Při určování velikosti datových množin bylo také zjištěno, že pro zpracování datové množiny o velikosti 16 000 hodnocení nebyl notebook Lenovo dostačující (viz kapitola 5.2) a bude potřeba více než 4 GB operační paměti RAM. Z tohoho důvodu

byl zapůjčen notebook značky Dell (viz kapitola 5.2), na kterém byly realizovány veškeré experimenty pro tuto datovou sadou.
Další důležitou částí volby velikosti použitých množin dat je určení velikosti trénovací a testovací množiny. Ze zvolené množiny dat pro daný expetiment byly vždy náhodně vyčleněny přibližně dvě třetiny uživatelských hodnocení na trénování a testování probíhalo na zbylé třetině dat. Tento typ rozdělené vstupní množiny dat v poměru použití 66 % dat pro trénování a 34 % na trénování je standardní doporučený postup v odborné literatuře (např. Marsland, 2015; Zaki, Meira, 2014). Cílem tohoto rozdělení je především zisk reálného pohledu na klasifikační problém nad rozsáhlou testovací datovou množinou a zároveň zamezení přílišnému přetrénování klasifikátoru. Obě množiny, trénovací i testovací jsou ve všech experimentech vzájemně disjunktní (neobsahují stejná uživatelská hodnocení).

Tab. 8 Zvolené velikosti datových množin pro experimentalní zpracování

Typ množiny	Velikost datové množiny	Poměř zastoupení tříd (P:N)	Trénovací množina	Testovací množina
Demonstrační	1 000	50 : 50	660	340
Reálná č. 1	8 000	50 : 50	5 280	2 720
Reálná č. 2	8 000	75 : 25	5 280	2 720
Reálná č. 3	16 000	75 : 25	10 560	5 440

5.4.2 Neřízené učení

Všechny experimenty v rámci neřízeného učení strojového učení resp. shlukování byly realizovány v nástroji Cluto. Pomocí několika experimentů nad vytvořenými datovými množinami byla zjišťována vhodná parametrizace jednotlivých shlukovacích algoritmů, které jsou v tomto programu implementovány. Následně na základě dosažených výsledků byl vybrán pro další práci algoritmus *k-means*, který měl z použitých algoritmů nejvyšší přenost v přiřazování uživatelských hodnocení do cílových tříd. Dále byla zkoumána vhodná parametrizice tohoto algoritmu pro dosažení co nejvyšší přesnosti, podstatným parametrem je především použitá metoda měření podobnosti jednotlivých textových hodnocení ve společném prostoru. Metoda měření podobnosti byla vybrána na základě několika experimentálních pokusů, ze kterých nejvyšší přesnosti dosahovala metoda měření podobnosti pomocí kosinové vzdálenosti. Pro ověření správnosti volby a vhodnosti této metody pro měření podobnosti vektorové reprezentace textových dat, byly nastudovány další odborné práce, které potvrzují skutečnost, že kosinová podobnost dosahuje v tomto případě relativně dobrých výsledků a že se jedná o hojně využívanou metodu (Žižka a kol. 2012; Sidorov a kol. 2014). Pro volbu použité metody měření podobnosti, ale i metody shlukování a její parametrizace by se jistě dala provést hlubší a důkladnější analýza této oblasti strojového učení. Nicméně cílem práce je především základní porovnání různých metod strojového učení a díky limitovaným obsahovým možnostem není možné pokrýt veškeré aspekty do hloubky. Dalším odůvodněním je také skutečnost, že neřízené učení by mělo podle očekávání

autora získaného z nastudovaných odborných zdrojů dosahovat nejhorších výsledků v porovnání s ostatními typy strojového učení. Tento předpoklad byl ve většině případů potvrzen i na základě experimentů, a proto se autor zaměřil především na experimenty v oblasti řízeného a částečně řízeného učení.

Popis provedení experimentů neřízeného učení

Vzhledem k tomu, že práce je experimentálního typu je zcela zásadní uvést přesnou konfiguraci, která byla použita pro získání výsledků. Tyto údaje jsou důležité pro ověření kvality dosažených výsledků a možnosti případného opětovného provedení experimentů. Jak již bylo zmíněno v předcházející části věkeré experimenty neřízeného učení bylo provedeny v nástroji Cluto. Konkrétně byla použita verze gCLUTO 1.0 z roku 2003 s jednoduchým grafickým uživatelským rozhraním. Spuštění aplikace gCLUTO se na platformě windows provádí pomocí spustitelného souboru *gcluto.exe*, který je volně dostupný na internetových stránkách autora tohoto programu (Karypis, 2006). Po spuštění je nutné nejprve vytvořit nový projekt a následně importovat vstupní data pomocí položky „Import Data“ v menu „Project“.

Použité vstupní soubory

- maticový soubor (Matrix File, přípona .mat)
- soubor s názvy sloupců (Column File, přípona .mat.clabel)
- soubor s identifikací tříd hodnocení (Row Class File, přípona .mat.rclass)

Vstupní maticový soubor obsahuje kompletní vektorovou reprezentaci všech uživatelských hodnocení, která byla vytvořena na základě vstupní datové kolekce pomocí skriptu doc2mat (viz kapitola 5.3). Soubor s názvy sloupců vznikl zároveň při tvorbě maticového souboru a obsahuje všechny unikátní termy, které se vyskytovali ve zvolené datové množině. Soubor s identifikací tříd pro jednotlivá uživatelská hodnocení byl vytvořen manuálně v textovém editoru Sublime Text 2. Tento soubor obsahuje na každém řádku jedno číslo buď 0, nebo 1, které identifikuje patřičnou třídu pro zařazení uživatelského hodnocení na daném řádku maticového souboru. Číslo 0 značí, že hodnocení patří do pozitivní třídy a číslo 1 je označení pro negativní hodnocení respektive negativní třídu. V dalším kroku následujícím po načtení všech potřebných vstupních souborů je nutné nastavit zvolenou parametrizaci shlukování. Ta se nastavuje přímo před samotným shlukováním v editačním okně, které lze vyvolat pomocí programového menu v záložce „Data“ a následně pod položkou „Cluster“.

Použitá parametrizace

- shlukovací metoda – *k-means* (Direct)
- počet shluků – 2
- reprezentace řádků – žádná (None)
- reprezentace sloupců – IDF (Inverse Document Frquency)

- metoda měření podobnosti – kosinová vzdálenost (Cosine)
- kriteriální funkce – l2
- počet iterací k-means - 10
- počet pokusů – 10

Nejprve bylo nutné nastavit metodu shlukování na „Direct“, pod kterou se ukrývá algoritmus *k-means* a následně provést snížení hledaného počtu shluků na dva. Zbylé parametry byly ponechány ve výchozím nastavení pro algoritmus *k-means*. Počet pokusů (provedení experimentů) byl stanoven na 10 stejně jako u experimentů řízeného a částečně řízeného strojového učení. Program gCLUTO po provedení shlukování nad vstupními daty s nastavenou parametrizací, zobrazí rozsáhlou řadu výstupních údajů, které jsou ve výsledkovém formuláři v tabulkové formě.

Výstupní data

- čistota (purity)
- přenost (accuracy)
- entropie (entropy)
- distribuce tříd (class distribution)
- matice záměn (confusion matrix)
- významné atributy (descriptive and descriminating features)

Kvalita shluků respektive přesnost shlukování lze určovat na základě několika metrik podle hledaných cílových výsledků. V této práci je kvalita shlukování určena pomocí metriky přesnosti (angl. *accuracy*, viz podkapitola 2.5.2) a to především z důvodu vhodného formátu pro srovnání s výsledky řízeného a částečně řízeného strojového učení.

5.4.3 Řízené učení

Experimenty řízeného strojového učení byly realizovány pomocí dvou různých nástrojů. V první části těchto experimentů byl použit nástroj Weka pro určení důležitých atributů pomocí algoritmu výběru rysů (angl. *feature selection*). V této části bylo realizováno několik experimentů za účelem získání pouze nejdůležitějších rysů pro snížení výpočetní náročnosti klasifikačních algoritmů ve druhé části. Výběr rysů probíhal na základě zhodnocení výsledků ze čtyř rozdílných metod výběru rysů.

- Chí kvadrát – (Chi square)
- ziskový poměr (gain ratio)
- symetrická nejistota (symmetrical uncertainty)
- informační zisk (info gain)

Druhá část v podobě použití různých klasifikačních algoritmů byla realizována v nástroji KEEL (viz kapitola 3.2) a to především z důvodu přesnějšího srovnání výsledků klasifikace s výsledky částečně řízeného učení, které budou realizovány nad stejnou implementací použitých klasifikátorů. Při experimentech řízeného učení byly zkoumány především zástupci ruzných typů klasifikátorů, které se liší z pohledu klasifikačního principu nad vstupní množinou dat. Podle očekávání autora by měly metody řízeného strojového učení dosahovat nejlepších výsledků z pohledu klasifikace. Tento předpoklad byl potvrzen i na základě experimentů a proto se autor rozhodl realizovat podstatně větší počet experimentů oproti neřízenému učení. Pro realizaci experimentu byly použity celkem čtyři klasifikátory implementované v nástroji KEEL, C4.5, k-NN , NB a SMO. Hlavní principy těchto klasifikačních algoritmů jsou popsány v teoretické části této práce (viz kapitola 2.5.1). Význam zkrácených názvů klasifikátorů je vysvětlen v následujícím odstavci a dále v práci jsou již pro popis těchto algoritmů využity vždy právě tyto zkratky.

- C4.5 – rozhodovací strom C4.5
- NB – Naivní Bayesův klasifikátor
- k-NN – *k*-nejbližších sousedů
- SMO – algoritmus podpůrných vektorů

Vzhledem k počtu vysokému počtu experimentů a omezenému obsahovému rozsahu práce byly experimenty pro řízené učení provedeny pouze se základní parametrizací použitých klasifikačních algoritmů. Dalším důvodem bylo také to, že i se základní parametrizací dosahovaly algoritmy řízeného učení nejlepších výsledků klasifikace a proto již nebylo nutné zkoumat tuto oblast podrobněji.

Popis provedení experimentů řízeného učení

V rámci řízeného učení byla realizována řada experimentů, které budou vyhodnoceny na základě dosažených výsledků v následující části (viz kapitola 6). Výsledky experimentů jsou pro finální srovnání různých typů strojového učení zcela zásadní, a proto je nutné přesně popsat postup realizovaný pro jejich získání. Jak již bylo zmíněno v předcházející části, první část experimentů pro řízené učení byla provedena v nástroji Weka (viz kapitola 3.2). V této části byly provedeny experimenty s použitím algoritmů na výběr rysů za účelem zvýšení přesnosti klasifikace a zároveň redukce výpočetní náročnosti experimentů ve druhé části. Konkrétně byl použit nástroj Weka verze 3.8 z roku 2016, který je volně dostupný ke stažení na internetových stránkách autorů z Univerzity Wakaito (Weka, 2016). Po spuštění programu Weka je nutné spustit Weka explorer pomocí položky „explorer“ v aplikačním menu a následně provést nahraní vstupního souboru pomocí kontextového menu „Open file…“.

Použité vstupní soubory

- Vektorová reprezentace uložená ve formátu *csv*

Jediným vstupním souborem do nástroje Weka je vytvořená vektorová reprezentace zvolené datové množiny uložená v souborovém formátu csv, do kterého byla převedena z maticového formátu (viz kapitola 5.3). Po nahrání souboru je nutné použít filtr na změnu posledního číselného atributu *document_class*, který označuje třídu daného uživatelského hodnocení na nominální hodnotu. Tento filtr se aplikuje výběrem filtru „NumericToNominal“, který se nachází v sekci filtrů „unsupervised“ ve složce „attribute“. Tento převod je podstatný pro algoritmy výběrů rysů, které vyžadují, aby vstupní data obsahovala atribut označení třídy nominálního typu. Poté bylo možné provést experimenty s různými algoritmy výběru rysů.

Použitá parametrizace

- metoda hledání (search metod) - Ranker
- počet vybraných rysů (numToSelect) – 10 %
- práh výběru (threshold) - 0

Realizované experimenty byly provedeny pro všechny metody výběru se shodnou parametrizací. Použitá metoda hledání byla *Ranker* s rozhodovacím prahem výběru nastaveným na hodnotu 0 a počet hledaných rysů byl stanoven na 10 % z celkového počtu rysů vstupní datové množiny. Tento krok byl zvolen na základě poznatků nastudovaných z odborných článků a po konzultaci s vedoucím práce především z důvodu snížení výpočetní náročnosti klasifikačních algoritmů bez výraznější ztráty informace. Tento postup, odebrání méně podstatných rysů, které ve své podstatě tvoří informační šum z pohledu dolování znalostí ze vstupní datové množiny, byl ověřen v celé řadě odborných prací a běžně se využívá za účelem zkvalitnění a především zrychlení procesu klasifikace (např. Bramer, Petridis 2014; Agarwal, Mittal, 2016). Výběrem rysů lze totiž výrazně snížit počet relevantních atributů pro účely klasifikace z řádu tisíců na desítky či stovky atributů bez výraznější ztráty přesnosti klasifikace.

Výstupní data

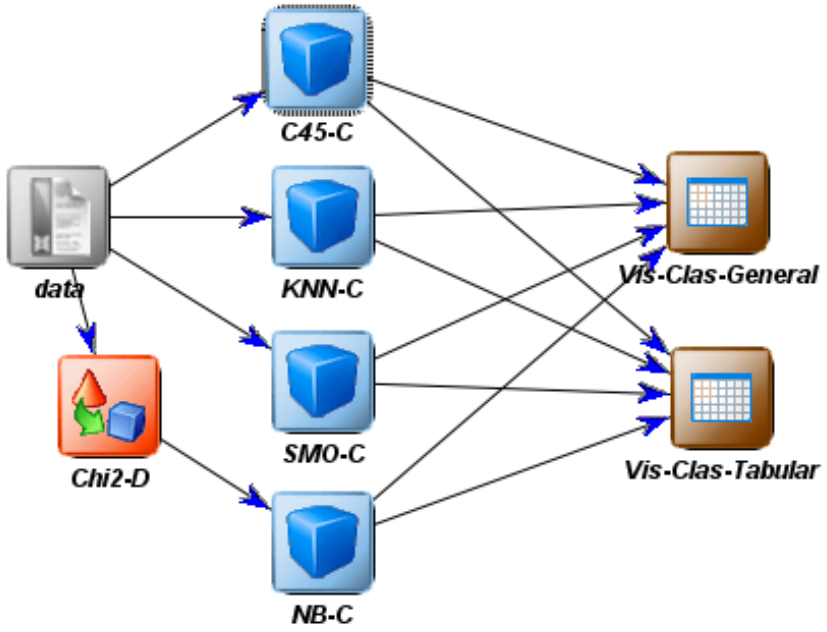
- Redukovaná vektorová reprezentace uložená ve formátu *csv*

Výstupem z této experimentální části je redukovaná vektorová reprezentace uložená ve formátu csv, která obsahuje pouze vybrané rysy na základě výsledků metod jejich výběru.
Druhá část experimentů v oblasti řízeného strojového učení se již týká samotné realizace klasifikace. Klasifikace byla realizována pomocí nástroje KEEL verze 3.0, která je rovněž volně dostupná na internetových stránkách autorů (Keel, 2005). Po spuštění tohoto nástroje pomocí spustitelného souboru v Javě *GraphInterKeel.jar* je nejprve nutné vstupní data importovat. Nahrání dat do aplikace je provedeno v modulu pro správu dat „Data Management“, kde jsou data převedenena do formátu KEEL (s příponou .dat).

Použité vstupní soubory

- Redukovaná vektorová reprezentace uložená ve formátu *csv*

Vstupním souborem do nástroje KEEL je soubor ve formátu csv, který obsahuje redukovanou vektorovou reprezentaci vzniklou na základě algoritmů výběru rysů (feature selection) v prvním kroku. Po importu dat do aplikace je možné s daty pracovat v modulu experimentů „Experiments“ pomocí jednoduchého uživatelského rozhraní. Pro sestavení experimentů pro řízené učení stačí podle přiloženého schéma (viz Obr. 11) sestavit bloky z nabídky dostupných algoritmů a vytvořit jejich propojení pomocí datových toků. Vstupní data jsou reprezentována pomocí datového bloku „data“, ze kterého jsou rozdistribuována pomocí datových toků do jednotlivých bloků reprezentujících klasifikátory. Jedinou vyjímkou je algoritmus *Naivní Bayes*, který vyžaduje diskretizaci všech vstupních hodnot, která je realizována diskretizačním blokem „Chi2-D“. V posledním kroku jsou veškeré klasifikační bloky propojeny s bloky pro zpracování a výpis výsledků.



Obr. 11 Schéma sestaveného experimentu pro řízené učení z nástroje KEEL

Po sestavení navženého experimentu je nutné pomocí tlačítka „Run Experiment“ vygenerovat spustitelný soubor pro tento experiment, který se uloží v podobě archivu zip na určené místo na disku. Pro spuštění experimentu je nutné tento archiv rozbalit a upravit ve spouštěcím konfiguračním souboru velikost používané operační paměti. Spouštěcí konfigurační soubor *RunKeel.xml* se nachází ve složce „Scripts“ a pro každou navrženou akci, která má být realizována v rámci experimentu obsahuje následující záznam v xml struktuře:

```
<sentence>
    <command>java</command>
    <option>-Xmx5120000000</option>
    <option />
    <option>-jar</option>
    <option />
    <option />
    <executableFile>../exe/SelfTraining.jar</executableFile>
    <scriptFile>./SelfTraining/all75wre/config0s0.txt</scriptFile>
</sentence>
```

Pro změnu velikosti použité operační paměti je nutné upravit všechny parametry *„-Xmx“* na požadovanou hodnotu. Základní nastavená hodnota tohoto parametru je 512 MB operační paměti, která pro realizaci většiny experimentů není dostačující. Maximální hodnota použité operační paměti je zároveň limitována nastavením operačního systému. Pro experimenty s menšími datovými množinami na notebooku Lenovo byla použitá operační paměť nastavena na 2048 MB. Pro experimenty u kterých docházelo k předčasnému ukončení z důvodů nedostatku operační paměti, byla na notebooku Dell nastavena na velikost operační paměti na 4096 MB nebo 6000 MB. Po nastavení velikosti použité operační paměti lze experiment spustit z podsložky „Scripts“ zadaním následujícího příkazu do příkazové řádky spuštěné v této složce:

java –jar RunKeel.jar

Použitá parametrizace

C4.5

- průřez (prune) – povolen (true)
- jistota (confidence) – 0.25
- minimální počet instancí k vytvoření listu (instances Per Leaf) – 2

Klasifikační algoritmus byl použit s výchozí parametrizací, která je přednastavena v programu KEEL. Parametru průřezu značí, zda algoritmus může během tvorby rozhodovacího stromu použít funkci prořezávání (prunning). Jistota značí minimální hranici jistoty, které musí dosahovat list, aby bylo zváženo jeho přiřazení do stromu. Minimální počet instancí určuje počet instatcí, které jsou potřebné k vytvoření nového listu.

k-NN

- metoda měření vzdálenosti (distance metric) – Euklidovská (Euclidean)
- počet testovaných sousedů (K Value) – 1

U klasifikačního algoritmu *k-NN* byla použita standardní parametrizace, která je tvořena Euklidovskou metodou měření vzdálenosti jednotlivých hodnocení. Další pametr počet testovaných sousedů je nastaven na hodnotu 1.

NB

- bez možnosti další parametrizace

SMO

- penalizační cena (C, cost) – 1.0
- typ předzpracování (preprocess type) – normalizace (Normalize)
- typ jádra (kernel type) – polynomiální (PolyKernel)
- transformace nominálních atributů na binární (convert nominal attributes to binary) – povolena (true)
- (fit logistic models) – zakázáno (false)

Klasifikační algoritmus SMO byl také použit se standardní parametrizací, která je tvořena typem předzpracování dat nastveným na normalizaci, typem jádra nastaveným na polynomiální. Dále je nastavena penalizační cena pro nevhodné atributy, povolená transformace nominálních atributů na binární a zakázané srovnávání výstupu pomocí logistického modelu.

Výstupní data

- přenost (accuracy)
- výsledky klasifikace pro jednotlivá hodnocení (class distribution)
- matice záměn (confusion matrix)

Výsledky experimentů jsou po dokončení spuštěného experimentu uloženy v podsložce „Results“. Výsledky se skládají z řady běžných výstupů, jako jsou například výsledky klasifikace pro jednotlivá hodnocení, matice záměn nebo přesnost. Úspěšnost klasifikace natrénovaných klasifikátorů pomocí řízeného učení je porovnávána s ostatními typy učení na základě dosažené přesnosti klasifikace.

5.4.4 Částečně řízené učení

Hlavním cílem této práce bylo především porovnání metod částečně řízeného strojového učení s řízeným a neřízeným učením, proto bylo pro tento typ učení zkoumáno podstatně větší množství metod než pro zbylé typy. Konkrétně byly použity všechny následující metody, jejichž principy byly zároveň popsány v teoretické části této práce (viz podkapitola 2.4.2):

- Self-training
- Co-training s výběrem relevantního náhodného podprostoru (Rel-RASCO)
- Co-training s výběrem náhodného podprostoru (RASCO)

- Tri-training

Veškeré experimenty provedené pro částečně řízené strojové učení byly stejně jako experimenty pro řízené učení realizovány v nástroji Keel (viz kapitola 3.2). Volba klasifikačních algoritmů z důvodu zachování stejných podmínek v rámci porovnání řízeného a neřízeného učení byla zcela stejná jako pro řízené učení. Každá z metod částečně řízeného učení byla vyzkoušena pro všechny čtyři zvolené klasifikační algoritmy (C4.5, k-NN, NB, SMO) s použitím stejné parametrizace těchto algoritmů, která byla již popsána v předcházející podkapitole. Při návhru experimentu pro částečně řízené učení bylo také nutné určit poměry označených položek, které budou použity pro natrénování jednotlivých klasifikátorů.
Cílem experimentů s částečně řízeným učením je ověřit předpoklad, že lze pomocí malého počtu označených dat a mnohonásobně většího počtu neoznačených dat natrénovat počáteční klasifikátor, který dosáhne klasifikační přesnosti blízké řízenému učení. Dalším cílem je to, že bude dosahovat lepších výsledků než algoritmy neřízeného učení. Na základě samotného konceptu částečně řízené učení bylo zcela jasné, že pro tyto účely nelze použít velký počet dat, protože jinak by jednotlivé experimenty postrádaly smysl, prakticky by se jednalo o řízené učení. Dalším faktorem při volbě velikosti označených množin je i samotný charakter textových dat v přirozeném jazyce, které obsahují velké množství unikátních termů. Díky množství termů je reprezentace jednotlivých hodnocení velice řídká, obsahuje jen velice málo nenulových hodnot, a proto je důležité zvolit velikost označené množiny tak, aby vytvořila dostatečný základ pro účely klasifikace hodnocení. Objemy použitých množin označených položek pro jednotlivé experimetny byly zvoleny na základě doporučení z odborné literatury (Abney, 2008; Søgaard, 2013) a následné konzultace s vedoucím práce. Zvolené počty označených položek, použité pro všechny experimenty částečně řízeného učení na reálných datech jsou následující:

- 50 označených položek
- 100 označených položek
- 500 označených položek
- 1000 označených položek

Nejmenší označená množina má 50 položek, velikost menší než 50 nebyla vzhledem k charakteru vektorové reprezentace textových dat v přirozeném jazyce zcela vyloučena. Další označené množiny mají velikost 100, 500 a 1000 položek. U všech těchto množin bude zkoumáno, zda má zvyšující se počet označených položek vliv na přesnost klasifikace. Horní hranice 1000 položek byla zvolena především z důvodu, aby nedošlo k porušení základní myšlenky částečně řízeného učení.

Popis provedení experimentů částečně řízeného učení
Stejně jako u experimentů pro řízené učení byla pro částečně řízené učení realizována řada experimentu, které budou vyhodnoceny na základě dosažených výsledků v následující části (viz kapitola 6). V následující části je popsán přesný postup realizace experimentů a použitá finální parametrizace. Tyto údaje jsou zcela zá-

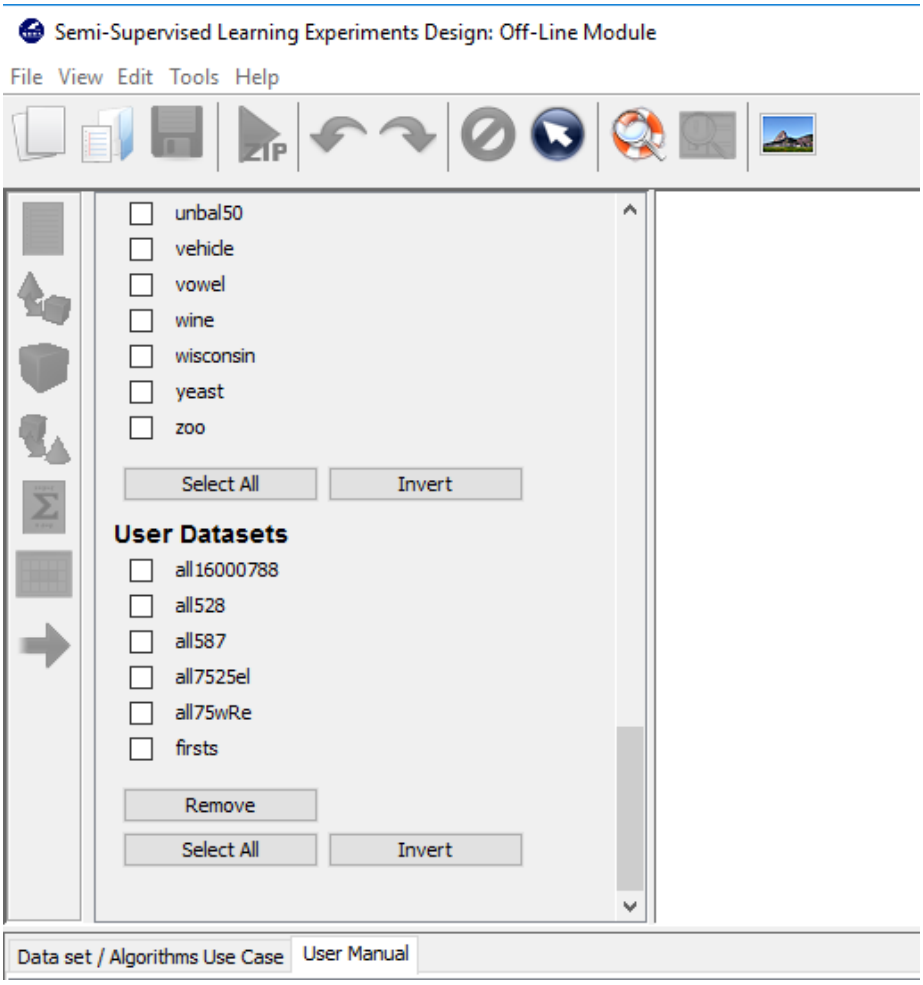
sadní pro výsledné srovnání jednotlivých typů strojového učení, aby bylo možné výsledky znovu reprodukovat. Všechny experimenty byly realizovány v nástroji Keel verze 3.0 stejně jako pro experimenty řízeného učení. Experimenty jsou realizovány v modulu pro částečně řízené učení, který byl implementován zcela samostatně a má omezené funkce. V tomto modulu nefunguje import dat pomocí uživatelského rozhraní a také zde není možnost vytvářet upravená data ve formátu Keel do tohoto modulu. Z těchto důvodů autor práce musel vytvořit skript createExpData.py, popsaný v části o přípravě dat (viz kapitola 5.3), pomocí kterého lze vytvořit modulem požadovaná data. Import dat je realizován doplněním datové sady přímo do konfiguračního souboru nástroje Keel pro částečně řízené učení. Konfigurační soubor *DatasetsSSL.xml* se nachází v hlavní složce nástroje v podsložce data. Nejlepším nalezeným způsobem importu dat je zkopírování záznamu vytvořené datové sady z konfiguračního souboru *Datasets.xml* pro klasický modul, který se nachází ve stejné složce. Tento záznam o nově importovaném datasetu byl vytvořen během experimentů řízeného učení.

Použité vstupní soubory

Vytvořená sada souborů ve formátu Keel:

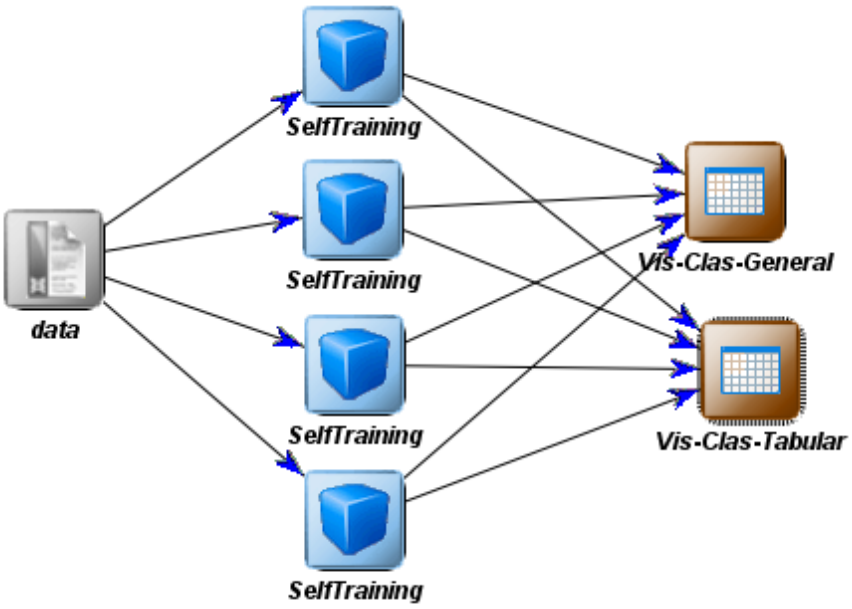
- testovací soubory (s koncovkou tst.dat)
- trénovací soubory (s koncovkou tra.dat)
- ověřovací soubory (s koncovkou trs.dat)

Na provedení jednoho experimentu částečně řízeného učení jsou vždy vyžadovány tři vstupní soubory. Testovací soubor slouží pro otestování natrénovaného klasifikátoru, který byl natrénován pomocí trénovacího soubor. Ověřovací soubor slouží pro ověření trénovací fáze a ke zjištění úspěšnosti klasifikace dosažené během trénování. Celkově se vstupní sada pro jeden experiment skládá z 30 souborů, které slouží pro deset provedení tohoto experimentu. Před návrhem struktury experimentu v designeru je potřeba všechny tyto soubory nakopírovat do složky vytvořené datové sady. Datové složky pro importovaná data se nachází v podsložce nástroje Keel „Data“.



Obr. 12 Ukázka nástroje Keel po importu uživatelských datových sad

Po importu dat do aplikace je možné s daty pracovat v modulu částečně řízeného učení (angl. *Semi-Supervised Learning, SSL*), který se nachází v hlavním menu nástroje pod položkou modulů (angl. *Modules*). Po spuštění SSL modulu jsou importovaná data dostupná v sekci uživatelských datových sad (angl. *User Datasets*, viz Obr. 12). Pro sestavení experimentů pro částečně řízeného učení stačí podle přiloženého schéma (viz Obr. 13) pro každou z metod (Self-training, Tri-training, RASCO, Rel-RASCO), sestavit bloky z nabídky dostupných algoritmů a vytvořit jejich propojení pomocí datových toků. Propojení jednotlivých bloků je zcela stejné jako pro experimenty řízeného učení. Jedinou změnou vůči řízenému učení je použití algoritmů částečně řízeného učení a nastavení používaných klasifikátorů v rámci těchto bloků.



Obr. 13 Schéma sestaveného experimentu pro metodu Self-training z nástroje KEEL

Použitá parametrizace

Společným parametrem pro všechny algoritmy částečně řízeného učení byly použité klasifikátory (*Classifier*), které byly zvoleny stejně jako u experimentů řízeného učení. Každý z algoritmů částečně řízeného učení byl tedy vyzkoušen pro klasifikační algoritmy C4.5, k-NN, NB, SMO.

Self-training

- počet vybraných příkladů (number of selected examples) - 1
- maximální počet iterací (MaxIter) - 40
- použitý klasifikátor (Classifier) – C4.5, k-NN, NB, SMO

Parametry *Self-trainingu* byly po otestování rozdílných počtů vybíraných příkladů ve finálních experimentech ponechány v původním výchozím nastavení. Počet vybíraných neoznačených položek byl nastaven na 1, což znamená, že byla vždy vybrána pouze jedna neoznačená položka s nejvyšší jistotou klasifikace, která byla zařazena do množiny označených hodnocení. Maximální počet iterací *Self-trainingu* byl nastaven na 40 iterací.

Rel-RASCO

- maximalni počet iterací (MaxIter) - 10
- počet pohledů (numbers of view) – 30

RASCO

- maximalni počet iterací (MaxIter) - 10
- počet pohledů (numbers of view) – 30

Parametry algoritmů Rel-RASCO a RASCO byly ponechány na základním, autory nástroje Keel doporučeném nastavení. Maximální počet iterací obou algoritmů byl nastaven na 10 iterací. Počet pohledů vytvořených nad vstupní datovou množinou byl nastaven na 30 pohledů.

Tri-training

- bez možnosti parametrizace

Pro spuštění navrženého experimentu v designeru je nutné dodržet stejný postup, který byl detailně popsán u experimentů řízeného učení v předcházející části (viz podkapitola 5.4.3). Nejprve je potřeba pomocí tlačítka „Run Experiment“ vygenerovat spustitelný soubor pro tento experiment, který se uloží v podobě archivu zip na určené místo na disku. Následně je třeba tento archiv rozbalit, upravit ve spouštěcím konfiguračním souboru velikost použité operační paměti a spustit experiment pomocí konzolového příkazu:

```
java –jar RunKeel.jar
```

Výstupní data

- přenost (accuracy)
- výsledky klasifikace pro jednotlivá hodnocení (class distribution)
- matice záměn (confusion matrix)

Výsledky experimentů jsou po dokončení spuštěného experimentu uloženy v podsložce „Results“. Výsledky se skládají z řady běžných výstupů strojového učení, stejně jako tomu bylo u řízeného učení. Úspěšnost klasifikace natrénovaných klasifikátorů pomocí částečně řízeného učení je porovnávána s ostatními typy učení na základě dosažené hodnoty metriky přesnosti.

6 Výsledky experimentů

V této kapitole jsou prezentovány výsledky dosažené realizací navržených experimentů nad zvolenou datovou kolekcí. Výsledky jsou představeny pomocí grafů a tabulek, které slouží pro vyjádření srovnání výsledků částečně řízeného strojového učení s řízeným a neřízeným strojovým učením. Pro přehlednost popisu výsledků jsou v grafech a tabulkách použity zkratky algoritmů a metod strojového učení. Veškeré tyto zkratky byly definovány v teoretické části (viz kapitola 2) této práce nebo v části návrhu experimentů (viz kapitola 5.4). Nejprve jsou představeny experimenty provedené na demonstrační sadě, které byly realizovány jako první v rámci praktické části. Tyto experimenty sloužily především pro výběr vhodného nástrojového aparátu a počáteční ověření realizace praktické části. Na základě demonstračních experimentů byly také zvoleny techniky přípravy dat a algoritmy strojového učení. Následně jsou v této kapitole již představeny výsledky experimentů provedené na reálných datových množinách.

6.1 Demonstrační datová sada

Jedním z cílů práce bylo provést experimenty pro vhodně zvolenou demonstrační datovou sadu. Autor práce zvolil jako demonstrační datavou sadu malou podmnožinu dat z reálné datové sady o velikosti 1000 uživatelských hodnocení. Demonstrační datová sada byla vytvořena z reálné sady především z důvodu ověření realizovatelnosti navržených postupů. Dále tato sada byla použita pro získání přehledu s existujícími nástroji a jejich následnou volbu pro realizaci experimentů všech typů strojového učení. V této části je zároveň definován jednotný formát prezentace výsledků, který bude použit pro prezentaci výsledků všech experimentů, které byly získány na reálných datových množinách.

Zavedené značení experimentů
Pro přehlednost popisu výsledků jednotlivých experimentů jsou v následujících částech práce používány pouze zkratky názvů algoritmů a autorem vytvořené značení experimentů. Běžně používané zkratky názvů všech použitých algoritmů a metod byly definovány v teoretické části práce a poté znovu rekapitulovány v kapitole návrhu experimentů (viz kapitola 5.4). Z důvodu přehlednosti popisu a orientačních důvodů bylo vytvořeno následující značení výsledných grafů a tabulek, které se snaží o co nejpřehlednější a nejstručnější popis experimentů. Značení se skládá ze dvou částí, datové množiny a vzhledem k tomu že poměr velikostí trénovací a testovací množiny byl ve všech případech stejný (na trénování bylo použito 66 % hodnocení z celkového počtu a zbylých 34 % bylo použito na testování), nejsou tyto údaje z důvodu přehlednosti u výsledků uváděny. Ve druhé části značení je v závorkách uveden poměr počtu pozitivních hodnocení vůči hodnocením negativním obsažených v datové množině. Na následujících výsledcích experimentů demonstrační množiny je představen a vysvětlen formát prezentace, která bude použita v následující kapitole srovnání metod strojového učení.

Výsledky experimentů pro demonstrační množinu

Následující výsledky v podobě grafů a tabulek průměrné naměřené přesnosti byly získány na demonstrační datové množiny. Podle zavedeného značení se tedy jednalo o případ „1000 (50:50)“, kde číslo 1 000 reprezentuje celkovou velikost použité datové množiny. Tato množina byla během experimentů rozdělena podle předem stanoveného poměru na trénovací množinu a testovací množinu. Trénovací množina obsahovala 66 % hodnocení z celkového použitého počtu a testovací množina obsahovala zbylých 34 % hodnocení. Výraz v závorkách „(50:50)“ značí, že se jednalo o vyváženou množinu, která obsahovala stejný počet pozitivních i negativních uživatelských hodnocení.

Následující tabulka a graf obsahují výsledky pro případ „1000 (50:50)“ s využitím 100 označených hodnocení. Počet označených hodnocení vždy značí velikost násady, která byla použita pro počáteční natrénování klasifikačních algoritmů částečně řízeného učení. Počáteční násada vždy obsahovala vyvážený počet označených pozitivních a negativních hodnocení.

Tab. 9 Průměrná přesnost u experimentu „1000 (50:50)“ pro 100 označených hodnocení

algoritmus	k-means	C4.5	k-NN	NB	SMO
shlukování	0.8270	-	-	-	-
klasifikace	-	0.7176	0.6088	0.8285	0.8559
Self-training	-	0.6379	0.5653	0.5965	0.7271
Rel-RASCO	-	0.6388	0.5656	0.6168	0.7412
RASCO	-	0.6494	0.5656	0.6224	0.7362
Tri-training	-	0.6379	0.5421	0.5994	0.7059

V prvním sloupci tabulky prezentující výsledky jsou vždy umístěny názvy jednotlivých použitých algoritmů strojového učení (viz Tab. 9). První řádek tabulky obsahuje konkrétní shlukovací a klasifikační algoritmy, které byly zkoumány během experimentů. Ve zbylých sloupcích tabulky jsou uvedeny dosažené hodnoty naměřené průměrné přesnosti pro deset měření u všech algoritmů na zkoumané množině. Z rozsahových důvodů není uvedena vypočítaná hodnota rozptylu naměřených přesností a také z důvodu, že v naprosté většině případů rozptyl nepřesáhl hodnotu 1 %. Všechny prezentované hodnoty průměrné přesnosti klasifikace byly získány testováním natrénovaných klasifikátorů na testovací množině a z důvodu lepší přehlednosti byly zaokrouhleny na 4 desetinná místa. Zcela přesné výsledky bez zaokrouhlení byly uloženy na přiloženém DVD spolu s ostatními výstupy této práce. Symbol polmčky „-“ v tabulce značí, že pro danou metodu strojového učení nebylo možné použít tento konkrétní shlukovací nebo klasifikační algoritmus.

V následujícím grafu jsou vyneseny hodnoty naměřené průměrné přenosti z předcházející tabulky vyjádřené v procentech (viz Tab. 9) pro případ „1000 (50:50)“ pro 100 označených položek (viz Obr. 14). Jedná se o sloupcový graf, kde jednotlivé sloupce značí použité shlukovací a klasifikační algoritmy. Na vodorovné ose grafu jsou umístěny názvy metod strojového učení, které byly při realiza-

ci experimentu využívány. Svislá osa v grafu reprezentuje úroveň dosažené průměrné hodnoty metriky přesnosti v procentech (angl. *accuracy*).

Obr. 14 Průměrná přesnost u experimentu „1000 (50:50)“ pro 100 označených hodnocení

Výsledky naměřené na demonstrační množině nelze objektivně považovat za nějak zásadní, protože byly získány na velice malé datové množině, což může výsledky výrazně zkreslovat. Typickým příkladem ovlivnění výsledků velikostí množiny byla naměřená průměrná přesnost shlukování, která v tomto případě dosáhla 82,70 %, tedy skoro stejných výsledků jako nejlepší klasifikační algoritmy u řízeného učení. Díky experimentům provedeným na demonstrační množině autor získal počáteční představu o konfiguraci algoritmů a realizovatelnosti reálných experimentů. Z výsledků demonstrační množiny byl patrný fakt, že při zvyšování počáteční násady se nezanedbatelně zvyšuje přesnost u většiny algoritmů částečně řízeného učení. Především z důvodů zkreslení výsledků a dostatečného množství získaných výsledků nad reálnými množinami se autor rozhodl výsledky naměřené na demonstrační množině nezahrnout do výsledného srovnání. Dalším důvodem je také omezený obsah diplomové práce, nicméně veškeré dosažené výsledky jsou uloženy na přiloženém DVD.

6.2 Srovnání metod strojového učení

Hlavním cílem práce bylo pomocí experimentů nad reálnou množinou dat provést srovnání tří metod strojového učení, neřízeného, řízeného a částečně řízeného učení. V následující části jsou představeny výsledky experimentů pro reálné datové množiny, které byly realizovány právě za účelem srovnání těchto metod. Srovnání

metod strojového učení bylo provedeno na základě metriky přesnosti (angl. *accuracy*), která určuje přenost klasifikace uživatelských hodnocení do cílových tříd. Postup realizace experimentů, použité nástroje, zvolené velikosti datových množin, použité algoritmy a jejich parametrizace byly popsány v předcházející části o návrhu experimentů (viz kapitola 5.4). V první podkapitole této části jsou popsány výsledky experimentů pro datovou množinu o velikosti 8 000 uživatelských hodnocení s vyrovnaným poměrem pozitivních a negativních příspěvků. Ve druhé podkapitole jsou uvedeny výsledky pro datovou množinu o stejné velikosti, tedy 8 000 uživatelských hodnocení s nevyváženým poměrem počtu pozitivních a negativních hodnocení. U druhé datové množiny byl poměr stanoven na 75 % pozitivních a 25 % negativních příspěvků. Poslední podkapitola obsahuje výsledky pro datovou množinu o velikosti 16 000 uživatelských hodnocení s nevyváženým poměrem počtu pozitivních a negativních hodnocení.

6.2.1 Reálná datová množina č. 1

Pro datovou množinu o velikosti 8 000 uživatelských hodnocení s vyváženým poměrem počtu pozitivních a negativních příspěvků bylo realizováno celkově 6 typů experimentů. První dva typy experimentů byly provedeny pro získání výsledků neřízeného a řízeného učení. Zbylé čtyři typy experimentů byly provedeny za účelem získání výsledné přesnosti algoritmů částečně řízeného učení pro 50, 100, 500 a 1 000 označených uživatelských hodnocení. Výsledné srovnání všech metod je provedeno pomocí 4 grafů a tabulek, které srovnávají naměřenou průměrnou přesnost částečně řízeného učení s naměřenou přesností neřízeného a řízeného učení. Ve všech grafech a tabulkách jsou naměřené hodnoty přesnosti pro neřízené a řízené učení uváděny opakovaně z důvodu snazšího porovnání s nově naměřenými hodnotami pro částečně řízené učení.

Tab. 10 Průměrná přesnost u experimentu „8000 (50:50)“ pro 50 označených hodnocení

algoritmus	k-means	C4.5	k-NN	NB	SMO
shlukování	0.7799	-	-	-	-
klasifikace	-	0.7739	0.5989	0.8544	0.8643
Self-training	-	0.6214	0.5551	0.5156	0.6643
Rel-RASCO	-	0.6147	0.5718	0.4953	0.6685
RASCO	-	0.6257	0.5577	0.4902	0.6582
Tri-training	-	0.6266	0.5475	0.5032	0.6605

Naměřená hodnota průměrné přesnosti pro případ použití 50 označených hodnocení (viz Obr. 15 a Tab. 10) u částečně řízeného učení dosáhla v nejlepším případě pouze necelých 67 %, konkrétně tato hodnota byla naměřena u metody Rel-RASCO pro klasifikátor podpůrných vektorů SMO. U zbylých metod strojového učení byla naměřena podstatně vyšší hodnota průměrné přesnosti. Pro shlukování byla naměřena průměrná hodnota přesnosti dosahující téměř 78 % a pro řízené učení

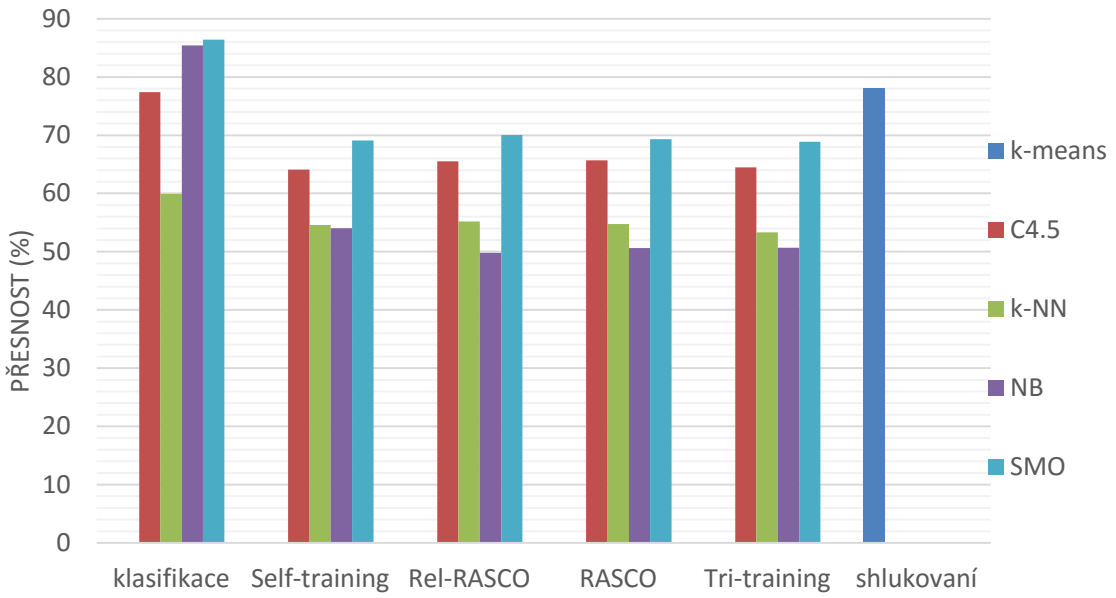
dosahoval nejlepších výsledků přesnosti algoritmus podpůrných vektorů s průměrnou hodnotou přesnosti 86 %.

Obr. 15 Průměrná přesnost u experimentu „8000 (50:50)“ pro 50 označených hodnocení

Tab. 11 Průměrná přesnost u experimentu „8000 (50:50)“ pro 100 označených hodnocení

algoritmus	k-means	C4.5	k-NN	NB	SMO
shlukování	0.7799	-	-	-	-
klasifikace	-	0.7739	0.5989	0.8544	0.8643
Self-training	-	0.6410	0.5457	0.5401	0.6911
Rel-RASCO	-	0.6551	0.5517	0.4982	0.7005
RASCO	-	0.6569	0.5475	0.5060	0.6932
Tri-training	-	0.6446	0.5333	0.5068	0.6887

Naměřené hodnoty průměrné přesnosti pro případ použití 100 označených hodnocení (viz Obr. 16 a Tab. 11) u částečně řízeného učení vzrostly přibližně o 3 až 4 % u klasifikačních algoritmů *podpůrných vektorů SMO* a *rozhodovacího stromu C4.5* oproti případu použití 50 položek. Pro klasifikační algoritmus *Naivního Bayese NB* hodnota přesnosti vzrostla pouze u algoritmu *Self-training*. Průměrná přesnost pro klasifikační algoritmus *k-nejbližších sousedů k-NN* v tomto případě klesla u všech algoritmů částečně řízeného učení o 1 až 2 %.



Obr. 16 Průměrná přesnost u experimentu „8000 (50:50)“ pro 100 označených hodnocení

Tab. 12 Průměrná přesnost u experimentu „8000 (50:50)“ pro 500 označených hodnocení

algoritmus	k-means	C4.5	k-NN	NB	SMO
shlukování	0.7799	-	-	-	-
klasifikace	-	0.7739	0.5989	0.8544	0.8643
Self-training	-	0.7020	0.5731	0.6058	0.7808
Rel-RASCO	-	0.6948	0.5755	0.6096	0.7815
RASCO	-	0.6954	0.5738	0.6072	0.7813
Tri-training	-	0.6971	0.5711	0.6037	0.7773

Průměrná přesnost pro případ použití 500 označených hodnocení u částečně řízeného učení výrazně vzrostla pro všechny klasifikační algoritmy (viz Obr. 17 a Tab. 12) v porovnání s předchozím případem. U *podpůrných vektorů SMO* a *Naivního Bayese NB* se nárůst naměřené průměrné přesnosti pohyboval přibližně mezi 6 až 10 %. U zbylých klasifikačních algoritmů *rozhodovacího stromu C4.5* a *k nejbližších sousedů KNN* hodnota přesnosti vzrostla v rozmezí 2 až 5 %. V tomto případě také poprvé průměrná přesnost částečně řízeného učení byla lepší než hodnota shlukování. Vyšší průměrné přesnosti než shlukování dosáhl klasifikační algoritmus *podpůrných vektorů SMO* pro všechny algoritmy částečně řízeného učení kromě *Tri-trainingu*, který byl o 2 desetiny procenta horší.

Obr. 17 Průměrná přesnost u experimentu „8000 (50:50)“ pro 500 označených hodnocení

Tab. 13 Průměrná přesnost u experimentu „8000 (50:50)“ pro 1000 označených hodnocení

algoritmus	k-means	C4.5	k-NN	NB	SMO
shlukování	0.7799	-	-	-	-
klasifikace	-	0.7739	0.5989	0.8544	0.8643
Self-training	-	0.7206	0.5927	0.6632	0.8073
Rel-RASCO	-	0.7197	0.5936	0.6655	0.8072
RASCO	-	0.7218	0.5928	0.6658	0.8079
Tri-training	-	0.7228	0.5918	0.6656	0.8063

Naměřená průměrná přesnost pro případ použití 1000 označených hodnocení, stejně jako v předchozím případě zaznamenala růst pro všechny použité algoritmy (viz Obr. 18 a Obr. 17). Největší zvýšení průměrné přesnosti bylo zaznamenáno u klasifikačního algoritmů *Naivního Bayese*, kdy přesnost vzrostla pro všechny algoritmy částečně řízeného učení přibližně o 6 %. Pro zbylé klasifikační algoritmy se hodnota vzrůstu průměrné přenosti pohybovala přibližně okolo 2 % u všech algoritmů částečně řízeného učení. Naměřená přesnost pro algoritmus *podpůrných vektorů SMO* překročila hranici přesnosti 80 % a přiblížila se výsledkům řízeného učení, které byly lepší přibližně o 6 %.

Obr. 18 Průměrná přesnost u experimentu „8000 (50:50)“ pro 1000 označených hodnocení

6.2.2 Reálná datová množina č. 2

Druhá reálná datová množina měla stejnou velikost jako předchozí, tedy 8 000 uživatelských hodnocení. Jediným rozdílem byla změna vyváženého poměru počtu pozitivních a negativních hodnocení, který byl stanoven na 75 % pozitivních hodnocení a 25 % negativních hodnocení. Stejně jako v předchozím případě bylo realizováno celkově 6 typů experimentů. Čtyři typy experimentů byly provedeny za účelem získání výsledné přesnosti algoritmů částečně řízeného učení pro 50, 100, 500 a 1 000 označených uživatelských hodnocení. Zbylé dva typy experimentů byly provedeny pro získání výsledků shlukování a klasifikace. Výsledné srovnání všech metod je provedeno pomocí 4 grafů a tabulek, které srovnávají naměřené průměrné přesnosti pro všechny tři metody strojového učení. Naměřené hodnoty průměrné přesnosti pro shlukování a klasifikaci uváděny opakovaně z důvodu snazšího porovnání s nově naměřenými hodnotami pro částečně řízené učení.

Tab. 14 Průměrná přesnost u experimentu „8000 (75:25)“ pro 50 označených hodnocení

algoritmus	k-means	C4.5	k-NN	NB	SMO
shlukování	0.7160	-	-	-	-
klasifikace	-	0.8301	0.7662	0.8688	0.8772
Self-training	-	0.6682	0.5677	0.7351	0.6329
Rel-RASCO	-	0.5759	0.6769	0.5969	0.6036
RASCO	-	0.6172	0.6784	0.5846	0.6539
Tri-training	-	0.5587	0.7017	0.6877	0.6555

V rámci experimentů druhé datové množiny naměřená hodnota průměrné přesnosti pro případ použití 50 označených hodnocení u částečně řízeného učení dosáhla v nejlepším případě přibližně 73,5 % (viz Obr. 19 a Tab. 14). Tato hodnota byla naměřena pro algoritmus *Self-training* s využitím klasifikačního algoritmu *Naivního Bayese NB.* Průměrná přesnost ostatních algoritmů částečně řízeného učení nepřesáhla 70 %. Průměrná přesnost neřízeného učení v podobě shlukování pomocí algoritmu *k-means* dosahovala hodnoty 71,6 %. V případě této datové množiny bylo částečně řízené učení lepší než neřízené učení již pro 50 označených položek. Výsledky řízeného učení byly výrazně lepší než pro zbylé metody strojového učení. Nejvyšší naměřená přesnost byla pro řízené učení u klasifikačního algoritmu podpůrných vektorů, kde se hodnota přenosti blížila k 88 %.

Obr. 19 Průměrná přesnost u experimentu „8000 (75:25)“ pro 50 označených hodnocení

Tab. 15 Průměrná přesnost u experimentu „8000 (75:25)“ pro 100 označených hodnocení

algoritmus	k-means	C4.5	k-NN	NB	SMO
shlukování	0.7160	-	-	-	-
klasifikace	-	0.8301	0.7662	0.8688	0.8772
Self-training	-	0.6901	0.6169	0.6037	0.7300
Rel-RASCO	-	0.6397	0.7317	0.5463	0.7071
RASCO	-	0.6699	0.7371	0.5255	0.6919
Tri-training	-	0.6344	0.7453	0.5754	0.6883

V případě experimentu pro 100 označených hodnocení došlo ke zvýšení průměrné přesnosti u všech klasifikačních algoritmů kromě *Naivního Bayese* (viz Obr. 20 a Tab. 15). Průměrná přenost pro klasifikační algoritmy *podpůrných vektorů SMO*, *rozhodovacího stromu C4.5* a *k-nejbližších sousedů KNN* ve všech případech vzrostla o 3 až 10 %. Zajímavým výsledkem je fakt, že se zvýšením počtu označených položek průměrná přesnost u *Naivního Bayese NB* výrazně klesla v případě *Self-trainingu* dokonce přibližně o 13 %.

Obr. 20 Průměrná přesnost u experimentu „8000 (75:25)“ pro 100 označených hodnocení

Tab. 16 Průměrná přesnost u experimentu „8000 (75:25)“ pro 500 označených hodnocení

algoritmus	k-means	C4.5	k-NN	NB	SMO
shlukování	0.7160	-	-	-	-
klasifikace	-	0.8301	0.7662	0.8688	0.8772
Self-training	-	0.7493	0.6881	0.3652	0.7816
Rel-RASCO	-	0.6790	0.7504	0.3481	0.7841
RASCO	-	0.6707	0.7715	0.3378	0.7870
Tri-training	-	0.6764	0.7501	0.3664	0.7757

Průměrná přesnost pro případ použití 500 označených hodnocení u částečně řízeného učení znovu vzrostla pro všechny klasifikační algoritmy, kromě *Naivního Bayese* v porovnání s předchozím případem (viz Obr. 21 a Tab. 16). Hodnota průměrné přenosti pro klasifikační algoritmy *podpůrných vektorů SMO*, *rozhodovacího stromu C4.5* a *k-nejbližších sousedů KNN* ve všech případech vzrostla o 3 až 8 %.

Zvýšením počtu označených hodnocení na 500, průměrná přesnost u *Naivního Bayese NB* výrazně klesla v případě všech algoritmů částečně řízeného učení přibližně o 20 %.

Obr. 21 Průměrná přesnost u experimentu „8000 (75:25)“ pro 500 označených hodnocení

Tab. 17 Průměrná přesnost u experimentu „8000 (75:25)“ pro 1000 označených hodnocení

algoritmus	k-means	C4.5	k-NN	NB	SMO
shlukování	0.7160	-	-	-	-
klasifikace	-	0.8301	0.7662	0.8688	0.8772
Self-training	-	0.7549	0.7028	0.3067	0.8183
Rel-RASCO	-	0.6951	0.7554	0.3062	0.8143
RASCO	-	0.6939	0.7550	0.3036	0.8182
Tri-training	-	0.7002	0.7528	0.4837	0.8161

Hodnoty naměřené průměrné přesnosti v posledním experimentu pro tuto množinu v případě použití 1 000 označených hodnocení měly podobný průběh jako pro 500 označených hodnocení (viz Obr. 22 a Tab. 17). Průměrná přenost pro klasifikační algoritmy *podpůrných vektorů SMO*, *rozhodovacího stromu C4.5* a *k-nejbližších sousedů KNN* ve všech případech vzrostla o 2 až 3 %. Pro klasifikační algoritmus *Naivního Bayese* znovu klesla přibližně o 3 až 6 % u všech metod částečně řízeného učení. Nejvyšší přenosti u algoritmů částečně řízeného učení stejně jako u předchozí datové množiny bylo dosaženo při použití klasifikačního algoritmu *podpůrných vektorů SMO*. Naměřená přesnost v tomto případě dosáhla téměř 82 %

a přiblížila se průměrné přenosti nejlepších klasifikátorů řízeného učení. Nejlepší klasifikátor řízeného učení pro tuto množinu byl algoritmus *podpůrných vektorů SMO*, u kterého průměrná přenost dosahovala hodnoty 87,72 %. Velmi dobrých výsledků přesnosti dosáhl také algoritmus *Naivního Bayese*, kde naměřená hodnota přesnosti byla 86,87 %.

Obr. 22 Průměrná přesnost u experimentu „8000 (75:25)“ pro 1000 označených hodnocení

6.2.3 Reálná datová množina č. 3

Poslední použitá reálná datová množina obsahovala 16 000 uživatelských hodnocení s nevyváženým počtem pozitivních a negativních příspěvků. Poměr hodnocení byl stejný jako u předcházející množiny, obsahovala tedy z celkového počtu 75 % pozitivních hodnocení a 25 % negativních hodnocení. Stejně jako pro předešlé množiny bylo realizováno celkově 6 typů experimentů. Čtyři typy experimentů byly provedeny za účelem získání výsledné přesnosti algoritmů částečně řízeného učení pro 50, 100, 500 a 1 000 označených uživatelských hodnocení. Zbylé dva typy experimentů byly realizovány pro získání výsledků řízeného a neřízeného učení. Výsledné srovnání všech metod je provedeno pomocí 4 grafů a tabulek, které srovnávají naměřené průměrné přesnosti pro všechny tři metody strojového učení. V následujících grafech a tabulkách jsou naměřené hodnoty průměrné přesnosti pro neřízené a řízené učení uváděny opakovaně z důvodu snazšího porovnání s nově naměřenými hodnotami pro částečně řízené učení.

Tab. 18 Průměrná přesnost u experimentu „16000 (75:25)“ pro 50 označených hodnocení

algoritmus	k-means	C4.5	k-NN	NB	SMO
shlukování	0.7251	-	-	-	-
klasifikace	-	0.8301	0.7708	0.8689	0.8848
Self-training	-	0.6811	0.6785	0.7422	0.6947
Rel-RASCO	-	0.5730	0.6648	0.6296	0.6813
RASCO	-	0.5963	0.6827	0.6315	0.6867
Tri-training	-	0.5999	0.6375	0.7094	0.6602

Průměrná hodnota přesnosti pro případ použití 50 označených hodnocení u částečně řízeného učení dosáhla v nejlepším případě 74,21 % (viz Obr. 23 a Tab. 18). Tato hodnota byla naměřena u klasifikačního algoritmu *Naivního Bayese NB* při použití algoritmu *Self-training*. Výsledná naměřená hodnota přesnosti u experimentů neřízeného učení byla přibližně 72,5 %. Stejně jako u předchozí datové množiny bylo i zde částečně řízené učení lepší již pro 50 označených položek. Naměřená průměrná přesnost pro algoritmy řízeného učení stejně jako u experimentů předchozích množin výrazně převyšovala zbylé metody strojového učení. Nejvyšší průměrné přenosti dosáhl klasifikační algoritmus *podpurných vektorů SMO*, kde byla hodnota přenosti 88,48 %.

Obr. 23 Průměrná přesnost u experimentu „16000 (75:25)“ pro 50 označených hodnocení

Tab. 19 Průměrná přesnost u experimentu „16000 (75:25)“ pro 100 označených hodnocení

algoritmus	k-means	C4.5	k-NN	NB	SMO
shlukování	0.7251	-	-	-	-
klasifikace	-	0.8301	0.7708	0.8689	0.8848
Self-training	-	0.6675	0.7168	0.6682	0.7238
Rel-RASCO	-	0.6351	0.7185	0.5715	0.7115
RASCO	-	0.6296	0.7110	0.5819	0.7101
Tri-training	-	0.6618	0.7199	0.6225	0.7159

V případě s použitím 100 označených hodnocení vzrostla hodnota průměrné přesnosti u všech algoritmů částečně řízeného učení pro všechny klasifikační algoritmy kromě *Naivního Bayese* (viz Obr. 24 a Tab. 19). U *Naivního Bayese* s přidáním označených hodnocení kleska hodnota naměřené přesnosti přibližně o 6 až 7 % pro všechny algoritmy částečně řízeného učení. Pro zbylé klasifikační algoritmy se průměrná přesnost navýšila přibližně o 3 až 8 %.

Obr. 24 Průměrná přesnost u experimentu „16000 (75:25)“ pro 100 označených hodnocení

Tab. 20 Průměrná přesnost u experimentu „16000 (75:25)“ pro 500 označených hodnocení

algoritmus	k-means	C4.5	k-NN	NB	SMO
shlukování	0.7251	-	-	-	-
klasifikace	-	0.8301	0.7708	0.8689	0.8848
Self-training	-	0.6954	0.7549	0.4165	0.7769
Rel-RASCO	-	0.6973	0.7550	0.4048	0.7752
RASCO	-	0.6963	0.7551	0.3999	0.7713
Tri-training	-	0.7014	0.7482	0.4178	0.7641

V porovnání s předchozím případem průměrná přesnost pro případ použití 500 označených hodnocení u částečně řízeného učení nezanedbatelně vzrostla pro všechny klasifikační algoritmy, kromě *Naivního Bayese* (viz Obr. 25 a Tab. 20). Naměřená přenost pro klasifikační algoritmy *podpůrných vektorů SMO*, *rozhodovacího stromu C4.5* a *k-nejbližších sousedů KNN* ve všech případech vzrostla o 3 až 4 %. V tomto případě již přenost částečně řízeného učení s využitím klasifikačního algoritmu *podpůrných vektorů SMO* byla lepší než *k-nejbližších sousedů KNN* u experimentů řízeného učení. Zvětšení množiny označených položek mělo negativní vliv na klasifikační algoritmus *Naivního Bayese*, kde průměrná přenost výrazně klesla přibližně o 20 % u všech algoritmů částečně řízeného učení.

Obr. 25 Průměrná přesnost u experimentu „16000 (75:25)“ pro 500 označených hodnocení

Tab. 21 Průměrná přesnost u experimentu „16000 (75:25)“ pro 1000 označených hodnocení

algoritmus	k-means	C4.5	k-NN	NB	SMO
shlukování	0.7251	-	-	-	-
klasifikace	-	0.8301	0.7708	0.8689	0.8848
self-training	-	0.7177	0.7669	0.3584	0.8101
Rel-RASCO	-	0.7083	0.7672	0.3574	0.8106
RASCO	-	0.7119	0.7675	0.3596	0.8072
Tri-training	-	0.7110	0.7653	0.6477	0.8102

V posledním případě byl znovu zaznamenán nárůst naměřených hodnot průměrné přesnosti u většiny zkoumaných algoritmů částečně řízeného učení (viz Obr. 26 a Tab. 21). Výjimkou byly hodnoty naměřené při použití klasifikačního algoritmu *Naivního Bayese*, kdy hodnota naměřené přesnosti po zvětšení množiny označených hodnocení klesla u algoritmů *Self-training*, *Rel-RASCO* a *RASCO* přibližně o 3 až 4 %. Zajímavostí je v tomto případě naměřená přesnost pro *Tri-training*, kde při použití *Naivního Bayese* vzrostla přibližně o 23 %. Pro ostatní klasifikační algoritmy *SMO*, *C4.5* a *k-NN* vzrostla průměrná přenost ve všech případech přibližně o 2 až 5 %. Nejvyšší přenosti u algoritmů částečně řízeného učení bylo dosaženo v tomto případě při použití klasifikačního algoritmu *podpůrných vektorů SMO*, kde naměřená průměrná přesnost byla přibližně 81 %.

Obr. 26 Průměrná přesnost u experimentu „16000 (75:25)“ pro 1000 označených hodnocení

7 Diskuze

Strojové učení je v praxi velice rozšířenou podoblastí umělé inteligence, která nachází významné uplatnění v mnoha oborech lidské činosti, kde slouží pro dolování znalostí z dat. Získané znalosti pak mohou sloužit jako podpora při strategickém rozhodování, při klasifikaci nebo jako základ pro různé rozhodovací mechanismy.
V průběhu vývoje oblasti strojového učení vzniklo obrovské množství technik, metod a algoritmů, které lze použít za účelem dolování znalostí z dat. Samozřejmostí v tomto oboru je i nezastavitelný vývoj nových technik a algoritmů, které se snaží o vylepšení nebo překonání těch, co již existují. Existence velkého množství technik a algoritmů je zároveň problematickým aspektem strojového učení, protože pro dosažení dobrých výsledků (například klasifikace dokumentů) je obvykle nutné využít jejich kombinaci. Je nutné zmínit, že použité techniky jsou závislé na charakteru zkoumaných dat a volba nevhodných technik může mít negativní vliv na výsledky.

Cílem experimentální části této práce bylo praktické srovnání metod částečně řízeného učení s metodami řízeného a neřízeného učení. Srovnání probíhalo na reálných datech textového charakteru, která byla tvořena kolekcí uživatelských hodnocení produktů z kategorie elekroniky. Vzhledem k textovému charakteru dat je nutné zdůraznit, že proces jejich zpracování a přípravy byl velice náročný. Musela být použita celá řada technik předzpracování a přípravy dat za účelem získání jejich vhodné reprezentace pro algoritmy strojového učení. Nicméně následně naměřená přenost klasifikace ukázala, že důkladnost přípravy a předzpracování se vyplatila, protože bylo dosaženo velmi dobrých výsledků z pohledu přesnosti klasifikace do cílových tříd vzhledem k charakteru dat. Na základě dosažených výsledků lze také konstatovat, že pro tento typ dat jsou prakticky použitelné všechny tři zkoumané metody strojového učení.

Během fáze předzpracování a přípravy dat byla použita řada technik, které se projevily jako přínosné pro zlepšení dosažené výsledné přesnosti klasifikace. Největší pozitivní vliv z technik předzpracování na výslednou přesnost klasifikace a časovou náročnost výpočtu jednotlivých klasifikačních algoritmů mělo odebrání termů s nízkou četností výskytu v kombinaci s odebráním nežádoucích stop slov. Průměrná přesnost klasifikace při použití těchto metod vzrostla ve většině případů přibližně o 3 %. Oblast předzpracování a přípravy dat je velice podstatnou součástí experimentů a zcela určitě by se dala provést důkladnější analýza těchto technik. Vzhledem k tomu, že tato oblast nebyla hlavním cílem práce a výsledná průměrná přenost dosáhla uspokojivých výsledků pro všechny metody strojového učení, nebyla oblast předzpracování a přípravy dat dále rozvíjena.

Celkově nejhorších výsledků z pohledu metriky přenosti dosahovalo podle očekávání autora shlukování, které bylo realizováno pomocí algoritmu *k-means*. V rámci experimentů na zvolených datových množinách bylo zjištěno, že shlukování dosahuje lepších výsledků při vyváženém poměru zastoupení cílových tříd. Výsledná průměrná přenost shlukování u vyvážené množiny byla výrazně lepší než pro zbylé dvě nevyvážené datové množiny. Rozdíl naměřené průměrné přesnosti

činil přibližně 6 %, přičemž pro vyváženou datovou množinu byla průměrná přesnost téměř 78 %. Přestože shlukování dosáhlo nejhorších výsledků z pohledu metriky přesnosti, je nutné brát v potaz fakt, že pracuje pouze s neoznačenými daty a i přes tuto skutečnost dosahovala přesnost na reálných datech vysokých hodnot. Další pozitivní stránkou neřízeného učení je fakt, že výpočetní a především časová náročnost shlukování je výrazně menší oproti zbylým metodám strojového učení. Negativní stránka shlukování spočívá v dosažení relativně nízké průměrné přesnosti v případě, kdy je zkoumaná datová množina nevyvážená z pohledu poměru zastoupení jednotlivých tříd.

Před započetím klasifikace v rámci experimentů řízeného učení byly použity metody výběru rysů za účelem omezení dimenzionality vektorové reprezentace textových hodnocení. Výběr 10 % nejvýznamnějších rysů výrazně přispěl ke snížení výpočetní i paměťové náročnosti experimentů a zároveň došlo i ke zvýšení naměřené přesnosti klasifikace v rámci řízeného a částečně řízeného učení. Tento fakt byl pravděpodobně způsoben tím, že klasifikátory byly natrénovány na slovech, které lépe rozhodují o zařazení textového hodnocení do cílové třídy.

Celkově nejlepších výsledků z pohledu hodnoty naměřené průměrné přesnosti klasifikace dosahovaly metody řízeného učení, které byly u klasifikačních algoritmů Naivního Bayese a algoritmu podpůrných vektorů ve všech případech výrazně lepší než shlukování a algoritmy částečně řízeného učení. Tyto klasifikátory u všech datových množin přesáhly hodnotu naměřené průměrné přenosti 85 %. Úplně nejlepších výsledků bylo dosaženo u experimentů s nejpočetnější datovou množinou, kde pro klasifikační algoritmus podpůrných vektorů byla naměřená průměrná hodnota přesnosti 88,48 %. Tato skutečnost byla způsobena větší velikostí datové množiny použité pro trénování, díky které byl natrénovaný klasifikátor podstatně kvalitnější. Nejhorších výsledků z pohledu výsledné přenosti u klasifikačních algoritmů řízeného učení dosahoval algoritmus *k-nejbližších sousedů*, který zaostával za nejlepším algoritmem podpůrných vektorů u nevyvážených množin přibližně o 10 %. U vyvážené množiny z pohledu počtu pozitivních a negativních hodnocení tento rozdíl dokonce činil přibližně 25 %. Na základě experimentů řízeného učení lze pro klasifikaci dat textového charakteru doporučit použití klasifikačních algoritmů *Naivního Bayese* a *algoritmu podpůrných vektorů* v případě dostatečně vysokého počtu označených položek. Negativní stránkou řízeného učení je závislost kvality klasifikace na velkém počtu označených položek, jejichž zisk je obvykle finančně i časově náročný.

V rámci experimentů částečně řízeného učení dosahovala naměřená průměrná přenost velice dobrých hodnot. Ve srovnání s řízeným učením v případě porovnání z pohledu nejlepších klasifikátorů u všech experimentů zaostávala hodnota přenosti částečně řízeného učení přibližně o 6 až 7 %. V případě použití velice malé počáteční násady (angl. *seed*) v podobě 50 a 100 označených hodnocení dosahoval rozdíl průměrné přenosti vůči řízenému učení ve většině případů téměř 20 %. Tato skutečnost byla zapříčiněna textovým charakterem dat, jejichž vektorová reprezentace je vysoce dimenzionální a velice řídká (obsahuje jen velmi malý počet nenulových hodnot). Vzhledem k charakteru použité reprezentace textových dat je

nutné pro získání kvalitního klasifikátoru vytvořit dostatečně velký počáteční slovník. Téměř u všech naměřených výsledků se potvrdila skutečnost, že se zvýšením počáteční násady označených položek dochází ke zvýšení kvality natrénovaného klasifikátoru z pohledu přenosti klasifikace. Největší skok byl zaznamenán mezi případy použití 100 a 500 označených hodnocení, kde se hodnota přesnosti pro 500 označených hodnocení zvýšila v případě nejlepšího klasifikátoru přibližně o 7 %. Zajímavým výsledkem bylo chování klasifikačního algoritmu *Naivního Bayese*, u kterého z ne zcela jasných důvodů pro nevyvážené datové množiny po zvýšení počtu označených položek klesala hodnota průměrné přenosti, přičemž v rámci řízeného učení byl tento algoritmus jeden z nejlepších z pohledu přenosti klasifikace. Tuto skutečnost by jistě bylo vhodné detailněji prozkoumat vzhledem k tomu, že při použití 1000 označených položek u algoritmu *Tri-training* následně skokově vzrostla průměrná přesnost přibližně o 23 % (viz Obr. 26). Mezi pozitivní stránky částečně řízeného učení lze zcela určitě zařadit skutečnost, že i při použití relativně malé počáteční násady lze dosáhnout velmi dobrých výsledků z pohledu klasifikace.

Volba algoritmů strojového učení určených pro použití v případě dolování znalostí ukrytých v datech je velice složitým problémem. Před samotným výběrem algoritmů je nutné zvážit celou řadu okolností, jako jsou výpočetní a paměťová náročnost algoritmů, charakter analyzovaných dat a také formát požadovaných výsledků. Doporučení vyvozená z výsledků srovnání metod strojového učení by mohla být následující: použití dostačně velké násady pro algoritmy částečně řízeného učení, které mělo podstatný vliv na zvýšení přesnosti klasifikace. Dále použití metod výběru určitého procenta nejvýznamnějších rysů, které se jevilo jako velice výhodné z hlediska snížení jak časové, tak i výpočetní náročnosti algoritmů. Zároveň se použití nejvýznamnějších rysů pozitivně projevilo na zvýšení kvality klasifikace z pohledu naměřené metriky přesnosti. Použití klasifikačních algoritmů *Naivního Bayese* a *algoritmů podpůrných vektorů*, které dosahovaly nejvyšší přesnosti pro zkoumaná data texového charakteru. Dále v případě shlukování použití algoritmu *k-means* s využitím vektorové reprezentace IDF, při jejímž použití bylo dosaženo nejlepších výsledků shlukování.

Návrh dalšího pokračování práce

V rámci této diplomové práce byla zkoumána oblast strojového učení, konkrétně především srovnání jeho různých metod v rámci řešení problému reálné klasifikace uživatelských hodnocení elektronických produktů. Vzhledem k počtu existujících algoritmů není možné v rámci omezeného obsahu diplomové práce ani zdaleka pokrýt celou tuto problematiku. Proto jednou z možností dalšího rozvoje této práce je prozkoumat větší množství existujících klasifikačních algoritmů, jako jsou například neuronové sítě nebo rozhodovací strom c5. Další z možností je hlubší prozkoumání metod předzpracování a přípravy dat, které by mohly výrazně vylepšit stávající výsledky. Vhodným pokračováním by také mohlo být například důkladnější prozkoumání dalších možností parametrizace všech použitých algoritmů.

8 Závěr

Tato diplomová práce se zabývala zkoumáním metod strojového učení, konkrétně řízeného, neřízeného a částečně řízeného strojového učení. Cílem práce bylo provést srovnání těchto metod na základě realizace experimentů v podobě dolování znalostí nad reálnými daty a následně provést vyhodnocení, interpretaci a diskuzi dosažených výsledků. Posledním cílem práce bylo základě vyhodnocení výsledků vyvodit vhodné závěry a doporučení. V rámci celé práce autor čerpal ze zahraničních odborných zdrojů a domnívá se, že celá práce může být vhodným informačním zdrojem pro další zájemce, kteří se zajímají o oblast strojového učení.

V úvodní teoretické části práce byla popsána základní problematika dolování znalostí a strojového učení především v oblasti zpracování přirozeného textu. V rámci této kapitoly byly také popsány principy jednotlivých algoritmů strojového učení, které byly použity v praktické experimentální části práce a možné metody hodnocení kvality výsledků získaných pomocí těchto algoritmů.

V kapitole Zhodnocení současného stavu byly popsány autorem zvažované nástroje v rámci praktické části práce, které se běžně využívají pro účely dolování znalostí z dat. Byly popsány komerční nástroje i jejich volně dostupné varianty, včetně pozitivních a negativních vlastností těchto nástrojů.

Cílem práce bylo také získat vhodná reálná data, která budou použita pro realizaci experimentů. Použitá datová kolekce je tvořena z textových uživatelských hodnocení produktů elektroniky na serveru Amazon, které byly volně dostupné ke stažení na internetu. Obsah a formát datové kolekce byl popsán ve čtvrté kapitole.

Popis navržených experimentů, použitých nástrojů a postupů je popsán v kapitole Metodika. Nejprve byl popsán základní postup práce, následně byla definována použitá výpočetní technika, na které byly experimenty realizovány. V další části metodiky byla popsána fáze přípravy dat a použité metody předzpracování dat. V poslední části metodiky byly popsány navržené experimenty pro jednotlivé metody strojového učení, určení velikosti použitých datových množin a přesné postupy provedení realizovaných experimentů.

Výsledky realizovaných experimentů na demonstračních i reálných datech jsou prezentovány v kapitole 6 ve formě grafů a tabulek, kde jsou vyneseny naměřené průměrné hodnoty přenosti klasifikace uživatelských hodnocení do cílových tříd. Pomocí výsledků je zde provedeno srovnání všech tří zkoumaných metod strojového učení.

Interpretace a diskuze dosažených výsledků byla provedena v rámci kapitoly Diskuze, kde byly diskutovány naměřené hodnoty průměrné přesnosti, pozitivní a negativní stránky řízeného, neřízeného a částečně řízeného učení. Z výsledků srovnání všech tří metod strojového učení byly vyvozeny příslušné závěry a doporučení. Výsledné srovnání, závěry a doporučení nabízí zajímavý přehled o jednotlivých metodách strojového učení a může sloužit jako zdroj informací pro danou problematiku, proto se autor domnívá, že práce splnila zadání z pohledu srovnání metod strojového učení na reálných datech.

Z výsledků experimentů vyplývá, že pro dolování znalostí z textových reálných dat jsou použitelné všechny tři zkoumané metody strojového učení. Byly popsány negativní a pozitivní stránky jednotlivých metod strojového učení. Na základě experimentů bylo ověřeno, že při zvyšování počáteční násady částečně řízeného učení dochází ke zvyšování přesnosti klasifikace. Dále bylo na základě experimentů ověřeno, že při dostatečné počáteční násadě označených položek, dosahuje částečně řízené učení velmi dobrých výsledků z pohledu hodnoty přesnosti klasifikace, které se blíží hodnotám přesnosti naměřené pro algoritmy řízeného učení. Pro klasifikaci dat textového charakteru by autor na základě výsledků experimentů doporučil použití klasifikačního algoritmu Naivního Bayese a algoritmy podpůrných vektorů.

V rámci praktické části práce byla vytvořena celá řada skriptů v platformě nezávislém programovacím jazyce Python, které jsou snadno znovu použitelné a mohou být uplatněny při řešení dalších problémů dolování znalostí z dat.

V diskuzi byly také uvedeny i možné směry budoucího rozšíření nebo pokračování práce, například v podobě zkoumání většího rozsahu parametrizace použitých algoritmů nebo využití zcela nových variant algoritmů strojového učení.

9 Literatura

9.1 Knižní zdroje

ABNEY, Steven. *Bootstrapping. Proceeding ACL '02 Proceedings of the 40th Annual Meeting on Association for Computational Linguistics.* Stroudsburg, PA, USA: Association for Computational Linguistics, 2002, s. 360-367. DOI: 10.3115/1073083.1073143.

ABNEY, Steven. *Semisupervised learning in computational linguistics*. Boca Raton, FL: Chapman & Hall/CRC, 2008, xi, 308 s. Series in computer science and data analysis. ISBN 1584885599.

AGARWAL, Basant a Namita MITTAL. *Machine Learning Approach for Sentiment Analysis. Prominent Feature Extraction for Sentiment Analysis*. Cham: Springer International Publishing, 2016, s. 21 - 46. DOI: 10.1007/978-3-319-25343-5_3. ISBN 978-3-319-25341-1.

AGGARWAL, Charu C.; ZHAI, ChengXiang. *Mining text data*. Springer-Verlag New York, 2012, 524 s. ISBN 978-1-4614-3222-7.

BAHARUDIN, Baharum, Lam Hong LEE a Khairullah KHAN. *A Review of Machine Learning Algorithms for Text-Documents Classification*. Journal of Advances in Information Technology, ročník 1, č. 1, 2010 DOI: 10.4304/jait.1.1.4-20.

BLITZER, John, DREDZE, Mark a PEREIRA, Fernando. *Biographies, bollywood, boom-boxes and blenders: Domain adaptation for sentiment classification*. V: *ACL*. 2007. s. 440-447.

BLUM, Avrim a Tom MITCHELL. *Combining labeled and unlabeled data with co-training*. *COLT' 98 Proceedings of the eleventh annual conference on Computational learning theory*. New York: ACM, 1998, s. 92 - 100. DOI: 10.1145/279943.279962. ISBN 1-58113-057-0.

BRAMER, Max a Miltos PETRIDIS. *Research and Development in Intelligent Systems XXXI: Incorporating Applications and Innovations in Intelligent Systems XXII*. Springer International Publishing, 2014, 344 s. ISBN 9-783319120690.

CAMPBELL, Colin a Yiming YING. *Learning with support vector machines*. San Rafael, Calif.: Morgan & Claypool, 2011. ISBN 9781608456161.

CEJPEK, Jiří. *Informace, komunikace a myšlení: úvod do informační vědy*. 2., přeprac. vyd. Praha: Karolinum, 2005. ISBN 80-246-1037-x.

CERRA, F., PACALA, J., BRANDT, B.F., LUTFIYYA, M.N. *The Application of Informatics in Delineating the Proof of Concept for Creating Knowledge of the Value Added by Interprofessional Practice and Education*. Healthcare 2015, *3*, 1158-1173.

DEVI, K. Lakshmi, P. N. KUMAR a P. SUBATHRA. *Performance Evaluation of Sentiment Classification Using Query Strategies in a Pool Based Active Learning Scenario*. *Computational Intelligence, Cyber Security and Computational Models*. Springer Singapore, 2016, s. 65 - 75. ISBN 978-981-10-0251-9.

FAKERI-TABRIZI, Ali, Massih-Reza AMINI, Cyril GOUTTE a Nicolas USUNIER. *Multi-view self-learning.* Neurocomputing. 2015, 155, s. 117-127. DOI: 10.1016/j.neucom.2014.12.041. ISSN 09252312.

GOLDMAN, Sally A. a Yan ZHOU. *Enhancing Supervised Learning with Unlabeled Data*. *Proceeding ICML '00 Proceedings of the Seventeenth International Conference on Machine Learning*. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 2000, s. 327-334. ISBN 1-55860-707-2.

GANESAN, Kavita a Michael SUBOTIN. A general supervised approach to segmentation of clinical texts. In: *2014 IEEE International Conference on Big Data (Big Data)*. IEEE, 2014, s. 33-40. DOI: 10.1109/BigData.2014.7004390. ISBN 978-1-4799-5666-1.

GARCÍA, Salvador, Julián LUENGO a Francisco HERRERA. *A Data Mining Software Package Including Data Preparation and Reduction: KEEL*. s. 285. Springer International Publishing, 2015. DOI: 10.1007/978-3-319-10247-4_10.

HYREMATH, S., P. a JYOTHI R. *Follicle Detection and Ovarian Classification in Digital Ultrasound Images of Ovaries*. *Advancements and Breakthroughs in Ultrasound Imaging*. InTech, 2013. DOI: 10.5772/56518. ISBN 978-953-51-1159-7.

HACKELING, Gavin. *Mastering Machine Learning with scikit-learn: Apply effective learning algorithms to real-world problems using scikit-learn*. UK, Birmingham: Packt Publishing Ltd., 2014, 238 s. ISBN 978-1-78398-836-5.

IKEDA, Kazushi, Tadashi YANAGIHARA, Kazunori MATSUMOTO a Yasuhiro TAKISHIMA. *Unsupervised Text Normalization Approach for Morphological Analysis of Blog Documents*. 401 s. DOI: 10.1007/978-3-642-10439-8_41.

IKONOMAKIS, M., S. KOTSIANTIS a V. TAMPAKAS. *Text Classification Using Machine Learning Techniques*. *WSEAS TRANSACTIONS on COMPUTERS*, ročník 4, č. 8, Srpen 2005. s. 966 -974.

JAIN, Anil K a Richard C DUBES. *Algorithms for clustering data*. Englewood Cliffs, New Jersey: Prentice Hall, c1988, 319 s. ISBN 013022278X.

JONES, Karel Sparck a Peter WILLETT. *Readings in information retrieval*. San Francisco: Morgan Kaufmann Publishers, c1997. Morgan Kaufmann series in multimedia information and systems. ISBN 1-55860-454-5.

JURAFSKY, Dan a James H MARTIN. *Speech and language processing: an introduction to natural language processing, computational linguistics, and speech recognition*. 2nd ed. Upper Saddle River, N.J.: Pearson Prentice Hall, c2009. ISBN 0131873210.

KIRK, Matthew. *Thoughtful machine learning*. First edition. Sebastopol, CA: O'Reilly Media, 2014. ISBN 9781449374068.

LIEBOWITZ, Jay. *Knowledge management handbook*. Boca Raton, Fla.: CRC Press, 1999, 328 s. ISBN 0849302382.

LIU, Feng, Bingquan LIU, Chengjie SUN, Ming LIU a Xiaolong WANG. *Improving Link Prediction in Social Networks by User Comments and Sentiment Lexicon*. *Chinese Computational Linguistics and Natural Language Processing Based on Naturally Annotated Big Data*. 356 s. DOI: 10.1007/978-3-319-25816-4_29.

LANTZ, Brett. *Machine learning with R*. Birmingham: Packt Publishing, 2013, 396 s. ISBN 978-1-78216-215-5.

MAIMON, Oded Z. a Lior ROKACH (eds.). *Data mining and knowledge discovery handbook*. 2nd ed. New York: Springer, c2010. ISBN 978-0-387-09822-7.

MANNING, Christopher D, Prabhakar RAGHAVAN a Hinrich SCHÜTZE. *Introduction to information retrieval*. 1. vyd. Cambridge: Cambridge University Press, 2008. ISBN 978-0-521-86571-5.

MARSLAND, Stephen. *Machine learning: an algorithmic perspective, second edition*. Boca Raton: CRC press, 2015. Chapman & Hall/CRC machine learning & pattern recognition series. 457 s. ISBN 978-1-4987-5978-6.

MARTIN, Nathaniel F.G., ENGLAND, James W. a FOREWORD BY JAMES K. BROOKS. *Mathematical theory of entropy*. Cambridge: Cambridge University Press, 2010. ISBN 978-0-521-17738-2.

MEHLER, Alexander, BIEMANN, Chris (ed.). *Text Mining: From Ontology Learning to Automated Text Processing Applications*. Švýcarsko: Springer International Publishing, 2014. ISBN 978-3-319-12655-5.

MEJOVA, Yelena, SRINIVASAN, Padmini. *Exploring Feature Definition and Selection for Sentiment Classifiers*. In: *Proceedings of the Fifth International AAAI Conference on Weblogs and Social Media*. The AAAI Press, 2011.

MOHRI, Mehryar, Afshin ROSTAMIZADEH a Ameet TALWALKAR. *Foundations of machine learning*. Cambridge, MA: MIT Press, 2012, xii, 414 s. Adaptive computation and machine learning. ISBN 978-0-262-01825-8.

NI, Yuan, Qiong Kai XU, Feng CAO, Yosi MASS, Dafna SHEINWALD, Hui Jia ZHU a Shao Sheng CAO. *Semantic Documents Relatedness using Concept Graph Representation*. In: *Proceedings of the Ninth ACM International Conference on Web Search and Data Mining - WSDM '16*. New York, New York, USA: ACM Press, 2016, s. 635-644. DOI: 10.1145/2835776.2835801.

PORTER, M.F. *An algorithm for suffix stripping*. *Program*. Cambridge, Londýn, ročník 14, č.3, 1980, s. 130 - 137.

POWERS, David M. W. *Evaluation: from precision, recall and F - measure to ROC , informedness, markedness & correlation*. *Journal of Machine Learning Technologies*, ročník 2, č. 1, 2011, s. 37 - 63.

ONDREJKA, Petr. *Porovnání nekomerčních nástrojů pro dolování znalosti z dat pomocí strojového učení*. Brno, 2013. Diplomová práce. Mendelova univerzita v Brně, Provozně ekonomická fakulta. Vedoucí práce doc. Ing. Jan Žižka, CSc.

RAJARAMAN, Anand, Jeffrey David ULLMAN. *Mining of Massive Datasets*. Version 2.0. Cambridge: Cambridge University Press, 2011. ISBN 9781139058452.

ROBERTSON, Stephen. *Understanding inverse document frequency: on theoretical arguments for IDF*. *Journal of documentation*, ročník 60, č. 5, 2004, s. 503 - 520.

SALTON, G., WONG, A., & YANG, C. S. *A vector space model for automatic indexing.* Communications of the ACM, ročník 18, č. 11, 1975, s. 613-620.

SAMMUT, Claude a Geoffrey I WEBB. *Encyclopedia of machine learning*. London: Springer, 2010. ISBN 9780387301648.

SANG, Jitao, Yue GAO, Bing-kun BAO, Cees SNOEK a Qionghai DAI. *Recent advances in social multimedia big data mining and applications. Multimedia Systems* ročník 22, č. 1, 2016, s. 1-3. DOI: 10.1007/s00530-015-0482-5. ISSN 0942-4962.

SHAH, A. R., C. S. OEHMEN a B.-J. WEBB-ROBERTSON. *SVM-HUSTLE--an iterative semi-supervised machine learning approach for pairwise protein remote homology detection*. *Bioinformatics*, ročník 24, č .6, 2008, s. 783-790. DOI: 10.1093/bioinformatics/btn028. ISSN 1367-4803.

SHANNON, C. E. *A Mathematical Theory of Communication*. *Bell System Technical Journal*, ročník 27, č. 3, 1948, s. 379-423. DOI: 10.1002/j.1538-7305.1948.tb01338.x. ISSN 00058580.

SCHAKEL, Adriaan MJ; WILSON, Benjamin J. *Measuring Word Significance using Distributed Representations of Words.* arXiv preprint arXiv:1508.02297, 2015.

SIDOROV, Grigori, Alexander GELBUKH, Helena GÓMEZ-ADORNO a David PINTO. *Soft Similarity and Soft Cosine Measure: Similarity of Features in Vector Space Model*. *Computación y Sistemas*, ročník 18, č. 3, 2014. DOI: 10.13053/cys-18-3-2043. ISSN 1405-5546.

SØGAARD, Anders. *Semi-supervised learning and domain adaptation in natural language processing*. San Rafael, Calif.: Morgan & Claypool Publishers, 2013. ISBN 9781608459858.

STONE, James V. *Information Theory: Tutorial Introduction*. Sheffield, Velká Británie: Sebtel Press, 2015, 1. vydání, 260 s. ISBN 978-0956372857.

TURNEY, Peter D., PANTEL, P. *From frequency to meaning: Vector space models of semantics*. *Journal of artificial intelligence research*, 2010, 37(1), s. 141-188.

VERMA, Naveen, Kyong Ho LEE a Ali SHOEB. *Data-Driven Approaches for Computation in Intelligent Biomedical Devices: A Case Study of EEG Monitoring for Chronic Seizure Detection*. *Journal of Low Power Electronics and Applications*. 2011, ročník 1, č. 1, s. 150 -174. DOI: 10.3390/jlpea1010150.

WANG, Jiao; LUO, Si-wei; ZENG, Xian-hua. *A random subspace method for co-training*. In: *Neural Networks, 2008. IJCNN 2008.(IEEE World Congress on Computational Intelligence). IEEE International Joint Conference on*. IEEE, 2008. s. 195-200.

WANG, Wei a Zhi-Hua ZHOU. *Analyzing Co-training Style Algorithms. Machine Learning: ECML 2007*. Springer Berlin Heidelberg, 2007, s. 454-465. DOI: 10.1007/978-3-540-74958-5_42. ISBN 978-3-540-74957-8.

WITTEN, I, Eibe FRANK a Mark A HALL. *Data mining: practical machine learning tools and techniques*. 3rd ed. Burlington: Morgan Kaufmann, 2011, xxxiii, 629 s. Morgan Kaufmann series in data management systems. ISBN 978-0-12-374856-0.

WOŁK, Krzysztof a Krzysztof MARASEK. *A Sentence Meaning Based Alignment Method for Parallel Text Corpora Preparation*. 229 s. DOI: 10.1007/978-3-319-05951-8_22.

WU, J. *Advances in K-means Clustering: A Data Mining Thinking*. Berlin: Springer,

2012. ISBN 978-364-2298-073.

YAN, Xiaohui, Jiafeng GUO, Shenghua LIU, Xueqi CHENG a Yanfeng WANG. *Learning Topics in Short Texts by Non-negative Matrix Factorization on Term Correlation Matrix*. *Proceedings of the 2013 SIAM International Conference on Data Mining*. Philadelphia, PA: Society for Industrial and Applied Mathematics, 2013, 749 s. DOI: 10.1137/1.9781611972832.83. ISBN 978-1-61197-262-7.

YAROWSKY, D. *Unsupervised Word Sense Disambiguation Rivaling Supervised Methods. Proceedings of the 33rd Annual Meeting of the Association for Computational Linguistics*. Cambridge, MA, s. 189–196, 1995. DOI: 10.1.1.122.539

YASLAN, Yusuf, CATALTEPE, Zehra. *Co-training with relevant random subspaces*. *Neurocomputing*, 2010, 73.10: 1652-1661.

ZAKI, Mohammed J a Wagner MEIRA. *Data mining and analysis: fundamental concepts and algorithms*. New York, NY: Cambridge University Press, 2014. ISBN 9780521766333.

ZHANG, Harry. *The optimality of Naive Bayes*. *AA*, 2004, 1.2: 3.

ZHANG, Wen, Taketoshi YOSHIDA a Xijin TANG. *A comparative study of TF*IDF, LSI and multi-words for text classification*. *Expert Systems with Applications*. 2011, 38(3), s. 2758-2765. DOI: 10.1016/j.eswa.2010.08.066. ISSN 09574174.

ZHAO, Ying; KARYPIS, George. *Criterion functions for document clustering: Experiments and analysis*. Minneapolis. Technical report, 2001.

ŽIŽKA, J., BURDA, K., DAŘENA, F., 2012. Mining Opinion-Clusters from Very Large Unstructured Real-World Textual Data. Lecture Notes in Computer Science, vydání 7557, s. 38–47.

ŽIŽKA, Jan a František DAŘENA. Mining Significant Words from Customer Opinions Written in Different Natural Languages. In: *Lecture Notes in Computer Science*. Springer Berlin Heidelberg, 2011, sv. 6836, s. 211-218. DOI: 10.1007/978-3-642-23538-2_27. ISBN 978-3-642-23537-5. ISSN 0302-9743.

9.2 Elektronické zdroje

BALUCHA, A. *Stop-words* [online]. 2011 [cit. 2016-04-25]. Dostupné z: https://code.google.com/archive/p/stop-words/

BELLINGER, Gene, Durval CASTRO a Anthony MILLS. *The Way of Systems: Data, Information, Knowledge, and Wisdom*. *Systems Thinking*. Gene Bellinger, 2004. Dostupné také z: http://www.systems-thinking.org/dikw/dikw.htm

BRUCE, David, Anna REYNOLDS a Richard BROWN. *Word Frequency Counter* [online]. 2002 [cit. 2016-05-17]. Dostupné z: http://www.writewords.org.uk/word_count.asp

IBM. *SPSS Modeler*. IBM [online]. USA, New York: IBM, 2016 [cit. 2016-02-10]. Dostupné z: http://www-03.ibm.com/software/products/cs/spss-modeler

JURAFSKY, Dan a Christopher MANNING. *Introduction to Natural Language Processing: Basic text processing* [online]. Stanford, 2015 [cit. 2016-02-24]. Dostupné z: https://class.coursera.org/nlp/lecture/126

KARYPIS, George. *CLUTO: Data Clustering Software*. Karypis Lab [online]. University of Minnesota, 2006 [cit. 2016-02-10]. Dostupné z: http://glaros.dtc.umn.edu/gkhome/views/cluto

KEEL. *KEEL: Knowledge Extraction based on Evolutionary Learning*. KEEL [online]. 2005 [cit. 2016-05-01]. Dostupné z: http://www.keel.es/

KNIME. KNIME Online Self-Training. *Knime.org* [online]. Německo, Kostnice: KNIME, 2016 [cit. 2016-02-11]. Dostupné z: http://www.knime.org/knime-online-self-training

PIECH, Chris. *K Means: The Basic Idea* [online]. Stanford, 2013 [cit. 2016-03-02]. Dostupné z: http://stanford.edu/~cpiech/cs221/handouts/kmeans.html

RAPIDMINER. RapidMiner Studio. *Rapidminer.com* [online]. Boston, USA, 2016 [cit. 2016-02-10]. Dostupné z: https://rapidminer.com/products/studio/

THE R FOUNDATION FOR STATISTICAL COMPUTING. *The R Project for Statistical Computing*. *R-project.org* [online]. Rakousko, Víděň: R, 2016 [cit. 2016-02-11]. Dostupné z: https://www.r-project.org/

WEKA, *Weka 3: Data Mining Software in Java*. Weka Documentation [online]. University of Waikato: WEKA, 2016 [cit. 2016-02-10]. Dostupné z: http://www.cs.waikato.ac.nz/ml/weka/

Přílohy

A Příklad xml struktury uživatelského hodnocení produktu

<unique_id>
B000093IRC:good_when_it_works:stefanie_clay_"stef1223"
</unique_id>
<unique_id>
29609
</unique_id>
<asin>
B000093IRC
</asin>
<product_name>
Memorex 4.7GB 4x DVD+R Media (25-Pack Spindle): Electronics
</product_name>
<product_type>
electronics
</product_type>
<helpful>
1 of 1
</helpful>
<rating>
2.0
</rating>
<title>
good when it works
</title>
<date>
November 3, 2005
</date>
<reviewer>
Stefanie Clay "stef1223"
</reviewer>
<reviewer_location>
Morristown, NJ
</reviewer_location>
<review_text>
The DVDs I burned successfully showed the movies in excellent quality. The only problem is that for every 3 good burns I get one bad one. I'm using the Plextor DVD burner (love it!!), but I'm going to try my luck with a different DVD brand. The failure rate for Memorex is unacceptable. It's less the cost that bothers me, but more the time wasted on burning coasters
</review_text>

B Zdrojový kód skriptu extractData.py

```
#!/usr/bin/python
import sys
reload(sys)
sys.setdefaultencoding('utf-8')
import lxml.etree
import csv
import re
from bs4 import BeautifulSoup

foutn=open("allneg.txt","w")
foutp=open("allpos.txt","w")

with open('all.review', 'r') as content_file:
   MyStr = content_file.read()

test = ''
text = ''
reviews = []
soup = BeautifulSoup(MyStr,'xml')

for review in soup.find_all('review'):

   if float(review.rating.string) > 3:
      test = review.review_text.string + review.title.string
      test = test.replace('\n', ' ')
      test = re.sub(r'[^a-zA-Z\s]+',' ',test).lower().strip()
      test = re.sub( '\s+', ' ', test ).strip()
      test += "\n"
      foutp.write(test)

   elif float(review.rating.string) < 3:
      test = review.review_text.string + review.title.string
      test = test.replace('\n', ' ')
      test = re.sub(r'[^a-zA-Z\s]+',' ',test).lower().strip()
      test = re.sub( '\s+', ' ', test ).strip()
      test += "\n"
      foutn.write(test)
```

C Vytvořený seznam obecných stop slov

Tab. 22 Vytvořený seznam obecných stop slov

a	dont	in	shed	wasnt
about	down	into	shell	we
above	during	is	shes	wed
after	each	isnt	should	well
again	few	it	shouldnt	were
against	for	its	so	weve
all	from	its	some	were
am	further	itself	such	werent
an	had	lets	than	what
and	hadnt	me	that	whats
any	has	more	thats	when
are	hasnt	most	the	whens
arent	have	mustnt	their	where
as	havent	my	theirs	wheres
at	having	myself	them	which
be	he	no	themselves	while
because	hed	nor	then	who
been	hell	not	there	whos
before	hes	of	theres	whom
being	her	off	these	why
below	here	on	they	whys
between	heres	once	theyd	with
both	hers	only	theyll	wont
but	herself	or	theyre	would
by	him	other	theyve	wouldnt
cant	himself	ought	this	you
cannot	his	our	those	youd
could	how	ours	through	youll
couldnt	hows	ourselves	to	youre
did	i	out	too	youve
didnt	id	over	under	your
do	ill	own	until	yours
does	im	same	up	yourself
doesnt	ive	shant	very	yourselves
doing	if	she	was	

D Upravený seznam obecných stop slov z programu Cluto

Tab. 23 Upravený seznam obecných stop slov z programu Cluto

across	certainly	furthering	least	opens	seem	thus
almost	clear	furthers	less	order	seemed	today
alone	clearly	gave	like	ordered	seeming	together
along	come	general	likely	ordering	seems	took
already	differ	generally	long	orders	sees	toward
also	different	gets	longer	others	several	turn
although	differently	give	longest	part	shall	turned
always	done	given	made	parted	show	turning
among	downed	gives	make	parting	showed	turns
another	downing	going	making	parts	showing	upon
anybody	downs	goods	many	perhaps	shows	used
anyone	early	group	member	place	side	uses
anything	either	grouped	members	places	sides	want
anywhere	even	grouping	might	point	since	wanted
area	evenly	groups	mostly	pointed	somebody	wanting
areas	ever	however	much	pointing	someone	wants
around	every	important	must	points	something	way
asked	everybody	interest	necessary	possible	somewhere	ways
asking	everyone	interested	need	present	state	wells
asks	everything	interesting	needed	presented	states	went
away	everywhere	interests	needing	presenting	still	whether
back	face	just	needs	presents	sure	whole
backed	faces	keep	next	puts	take	whose
backing	fact	keeps	nobody	quite	taken	within
backs	facts	knew	noone	rather	therefore	without
became	felt	know	nothing	really	thing	work
become	find	known	nowhere	right	things	worked
becomes	finds	knows	number	room	think	working
began	first	large	numbers	rooms	thinks	works
behind	four	largely	often	said	though	year
beings	full	last	open	says	thought	years
came	fully	later	opened	second	thoughts	young
certain	furthered	latest	opening	seconds	three	younger

E Vytvořený seznam elektronických a počítačových stop slov

Tab. 24 Vytvořený seznam elektronických a počítačových stop slov

adapter	cell	earphones	joystick	players	signal
adaptor	click	electrical	keyboard	plug	silicone
alarms	clock	electricity	keyboards	plugged	software
album	company	electronic	keyring	plugs	sound
antenna	component	emails	keys	port	speakers
antennas	connect	engineer	keyspan	ports	start
apple	connection	equipment	label	power	station
application	connector	file	laptop	price	stereo
audio	connectors	firmware	laser	print	stuff
background	console	frames	link	printer	subwoofer
bandwidth	controller	gamepad	machine	printers	support
barcode	controls	gigabit	manual	printing	tape
basic	copier	graphic	memory	product	tech
bass	copy	graphics	microphone	program	technical
batteries	cover	handset	microphones	programming	technology
battery	credit	hardware	midi	programs	touchpad
book	data	headphones	modem	projector	touchpads
books	database	heads	monitor	radar	trackball
broadcast	desk	headset	mount	radio	trackballs
business	desktop	http	mouse	reader	transmitter
button	detector	channel	network	recorder	tune
buttons	device	chip	networks	registry	unit
cable	devices	images	office	remotes	units
cables	digital	industries	optical	router	video
calculator	disk	industry	pen	satellite	videos
camera	disks	inkjet	phone	scan	vista
cameras	display	install	phones	scanner	visual
canon	dock	installation	photo	scanners	volume
canons	document	instruction	photos	screen	watts
card	documents	internet	picture	scroll	webcam
cards	door	ipod	pictures	sensor	website
cartridge	download	item	pixels	serial	wired
cartridges	driver	items	plan	setting	wireless
case	drivers	jack	playback	setup	wires
cassette	earphone	jacks	player	shelf	

F Vytvořený seznam výrobců elektroniky

Tab. 25 Vytvořený seznam firem vyrábějících a prodávajících elektroniku

amazon	eclipse	intel	nintendo	systems
amd	eizo	intex	notion	tcl
amkette	electric	karbonn	nvidia	technologies
amoi	electronic	kejian	olympus	telesis
analog	electronics	kenstar	onida	ten
apple	emachines	kenwood	oppo	texas
audiovox	emerson	kingston	orion	toshiba
avaya	ensign	konica	packard	unisonic
averatec	epson	konka	panasonic	unisys
baer	fuji	koryo	panda	usha
beetel	fujifilm	koss	pentax	verbatim
bharat	fujitsu	krishla	pioneer	vertex
bird	funai	kyocera	planex	victor
bose	gateway	lacs	powervolt	videocon
bosch	geepas	lava	products	viewsonic
bpl	gionee	lenovo	pulstron	vivo
brother	godrej	lg	qualcomm	vizio
buffalo	group	lloyd	radio	voltas
byd	grundig	magnavox	rca	wabash
canon	haier	magnum	renesas	western
casio	hasee	marantz	ricoh	windows
celkon	havells	maxx	samsung	wipro
cisco	hcl	meizu	seagate	xerox
clarion	hewlett	melco	seiko	xiaomi
company	hisense	micromax	sennheiser	xolo
computers	hitachi	microsoft	sgi	yaesu
corega	holdings	microsystems	sharp	zebronics
corp	huawei	minolta	siemens	zenith
corporation	changhong	mitsubishi	simmtronics	zopo
cyberpower	iball	mobile	skyworth	zte
data	ibm	moser	sony	
dell	ifb	motorola	system	

G Zdrojový kód skriptu removeStop.py

```
#!/usr/bin/env python

import sys
import re

class Word (object):
	def __init__(self, text, count):
		self.text = text
		self.count = count

def main (argv):
	stopwordsfile = open(sys.argv[1],"r")
	stopwords = stopwordsfile.read()
	datafile = open(sys.argv[2],"r")
	data = datafile.read()
	outputfile= open(sys.argv[3],"w")
	words =  (stopwords.split('\n'))

	datas = data.split('\n')
	stop_words = set(stopwords.split('\n'))

	for line in datas:
		outputfile.write(" ".join(word for word in line.split() if (word not in
stop_words and len(word) >3)) + '\n')

	outputfile.close()
	stopwordsfile.close()
	datafile.close()


if __name__ == "__main__":
   main(sys.argv)
```

H Zdrojový kód skriptu mat2csv.py

```
#!/usr/bin/env python
import sys
import re
import os

class Word(object):
	def __init__(self, text, count):
		self.text = text
		self.count = count

def main(argv):
	if len(sys.argv) > 2:
		print ("Converting following files: " + sys.argv[1] + " " + sys.argv[2] +
"\ninto: " + sys.argv[3])
		labelsfile = open(sys.argv[1],"r")
		labels = labelsfile.read()
		datafile = open(sys.argv[2],"r")
		data = datafile.read()
		#classfile =
		if os.path.isfile(sys.argv[3]):
			os.remove(sys.argv[3])
		outputfile = open(sys.argv[3],"a")

		words = labels.replace('\n', ',')
		#words = words[:-1]  #uncomment this to remove last comma cha-
racter
		text = words + "document_class" + '\n'
		wordsList = []

		datas = data.split('\n')
		count = len(re.findall(r'\w+', text))
		#print (count)

		words = set(text.split(','))

		for word in words:
			wordsList.append(Word(word,0))

		outputfile.write(text)

		a = 0;
```

```
                for line in datas[1:]:
                        i = 0
                        linewords = re.findall(r'\d+', line)
                        for word in linewords[:-1]:
                                if i%2 == 0:
                                        previous = int(word) - 1
                                        i = i + 1
                                else:
                                        wordsList[previous].count = int(word)
                                        i = i + 1
                        i = 0
                        for p in wordsList[:-1]:
                                if p.count > 1:
                                        p.count = 1
                                outputfile.write(str(p.count))
                                i = i + 1
                                if i < count:
                                        outputfile.write(",")
                                p.count = 0

                        if a < 12000:
                                outputfile.write('0')
                        else:
                                outputfile.write('1')
                        outputfile.write('\n')
                        a = a + 1

                labelsfile.close()
                datafile.close()
                outputfile.close()

if __name__ == "__main__":
   main(sys.argv)
```

I Zdrojový kód skriptu createExpData.py

```
#!/usr/bin/env python
import sys
import re
import os
import random
import fileinput
from itertools import compress

def main(argv):
		print ("Zpracovavam nasledujici soubory: " + sys.argv[1] + " " + sys.argv[2] + "\n")

		ratio = 0.66;
		tratio = 0.34;
		supervised = 0;

		if argv[1] != '-old':

			# otevreni a nacteni vstupniho souboru s positivnimi prispevky
			with open(sys.argv[1], "r") as psource:
				positive = [line for line in psource]

			# otevreni a nacteni vstupniho souboru s negativnimi prispevky
			with open(sys.argv[2], "r") as nsource:
				negative = [line for line in nsource]

			allposreviews = sys.argv[3];
			allnegreviews = sys.argv[4];
			print "Test set: " + str(tratio * 100) + "%"
			print "positive: " + str(int(float(tratio) * float(allposreviews)))
			print "negative: " + str(int(float(tratio) * float(allnegreviews))) + "\n"
			random_pchoice = random.sample(positive, int(float(tratio) * float(allposreviews)))
			random_nchoice = random.sample(negative, int(float(tratio) * float(allnegreviews)))
			output = random_pchoice + random_nchoice
			random.shuffle(output)
			testing = output
```

```
                for line in random_pchoice:
                        positive.remove(line)

                for line in random_nchoice:
                        negative.remove(line)

                print "Training set: " + str(ratio * 100) + "%"
                print "positive: " + str(int(float(ratio) * float(allposreviews)))
                print "negative: " + str(int(float(ratio) * float(allnegreviews)))
+ "\n"
                random_pchoice = random.sample(positive, int(float(ratio) *
float(allposreviews)))
                random_nchoice = random.sample(negative, int(float(ratio) *
float(allnegreviews)))
                trainpos = random_pchoice
                trainneg = random_nchoice
                trainFile = open('trainpos.dat',"w")
                trainFile.write("".join(trainpos))
                trainFile = open('trainneg.dat',"w")
                trainFile.write("".join(trainneg))
                testFile = open('test.dat',"w")
                testFile.write("".join(testing))

        else:

                # otevreni a nacteni vstupniho souboru s testovacimi
prispevky
                with open(sys.argv[4], "r") as nsource:
                        testing = [line for line in nsource]

                allposreviews = sys.argv[5];
                allnegreviews = sys.argv[6];
                labeled = int(sys.argv[7]);

                with open(sys.argv[8], "r") as headers:
                        header = [line for line in headers]

                with open(sys.argv[9], "r") as theaders:
                        theader = [line for line in theaders]

                filename = sys.argv[10];

                for i in range (1,11):
```

```
                    if os.path.isfile(filename+'-10-'+ str(i) +'tst.dat'):
                        os.remove(filename+'-10-'+ str(i) +'tst.dat')
                    tstFile = open(filename+'-10-'+ str(i) +'tst.dat',"a")

                    # vytvoreni trenovaciho keel souboru
                    if os.path.isfile(filename+'-10-'+ str(i) +'tra.dat'):
                        os.remove(filename+'-10-'+ str(i) +'tra.dat')
                    traFile = open(filename+'-10-'+ str(i) +'tra.dat',"a")

                    # vytvoreni transduktivniho keel souboru
                    if os.path.isfile(filename+'-10-'+ str(i) +'trs.dat'):
                        os.remove(filename+'-10-'+ str(i) +'trs.dat')
                    trsFile = open(filename+'-10-'+ str(i) +'trs.dat',"a")

                    # otevreni a nacteni vstupniho souboru s positivnimi
prispevky
                    with open(sys.argv[2], "r") as psource:
                        trainp = [line for line in psource]

                    # otevreni a nacteni vstupniho souboru s negativnimi
prispevky
                    with open(sys.argv[3], "r") as psource:
                        trainn = [line for line in psource]

                    tstFile.write("".join(theader))
                    traFile.write("".join(header))
                    trsFile.write("".join(header))

                    print "Generating files: "
                    print filename+'-10-'+ str(i) +'tra.dat'
                    print filename+'-10-'+ str(i) +'trs.dat'
                    print filename+'-10-'+ str(i) +'tst.dat' "\n"

                    print "Test set: " + str(tratio * 100) + "%"
                    print "positive: " + str(int(float(tratio) * flo-
at(allposreviews)))
                    print "negative: " + str(int(float(tratio) * flo-
at(allnegreviews))) + "\n"

                    tstFile.write("".join(testing))

                    random.shuffle(trainp)
                    random.shuffle(trainn)
```

```
                if (supervised != 1):

                        print "Labeled Part: "
                        print "positive: " + str(int(labeled * 0.5))
                        print "negative: " + str(int(labeled * 0.5)) + "\n"
                        random_pchoice = random.sample(trainp,
int(labeled * 0.5))
                        random_nchoice = random.sample(trainn,
int(labeled * 0.5))
                        labeledPart = random_pchoice + ran-
dom_nchoice

                        for line in labeledPart:
                                traFile.write("".join(line))
                                trsFile.write("".join(line))

                        for line in random_pchoice:
                                trainp.remove(line)

                        for line in random_nchoice:
                                trainn.remove(line)


                        print "Unlabeled Part: "
                        print "positive: " + str(int((float(ratio) * flo-
at(allposreviews)) - (labeled * 0.5)))
                        print "negative: " + str(int((float(ratio) * flo-
at(allnegreviews)) - (labeled * 0.5))) + "\n"
                        random_pchoice = random.sample(trainp,
int((float(ratio) * float(allposreviews)) - (labeled * 0.5)))
                        random_nchoice = random.sample(trainn,
int((float(ratio) * float(allnegreviews)) - (labeled * 0.5)))
                        training = random_pchoice + random_nchoice

                        for line in training:
                                trsFile.write("".join(line))
                                line = line[:-2]
                                line += ' unlabeled \n'
                                traFile.write("".join(line))
                else:

                        print "Labeled Part: " + str(ratio * 100) + "%"
                        print "positive: " + str(int(float(ratio) * flo-
at(allposreviews)))
```

```
                                print "negative: " + str(int(float(ratio) * flo-
at(allnegreviews))) + "\n"
                                random_pchoice = random.sample(trainp,
int(float(ratio) * float(allposreviews)))
                                random_nchoice = random.sample(trainn,
int(float(ratio) * float(allnegreviews)))
                                labeledPart = random_pchoice + ran-
dom_nchoice
                                for line in labeledPart:
                                        traFile.write("".join(line))
                                        trsFile.write("".join(line))

                        psource.close()
                        nsource.close()
                        theaders.close()
                        headers.close()

if __name__ == "__main__":
   main(sys.argv)
```