TECHNICKÁ UNIVERZITA V LIBERCI
Fakulta mechatroniky, informatiky
a mezioborových studií

Strojové učení a Datamining

Diplomová práce

M13000177

Studijní program: N2612 – Elektrotechnika a informatika
Studijní obor: 1802T007 – Informační technologie

Autor práce: **Bc. Patrik Drhlík**
Vedoucí práce: RNDr. Klára Císařová, Ph.D.

Liberec 2015

TECHNICAL UNIVERSITY OF LIBEREC
Faculty of Mechatronics, Informatics
and Interdisciplinary Studies

Machine learning and Datamining

Diploma thesis

M13000177

Study programme: N2612 – Electrical Engineering and Informatics
Study branch: 1802T007 – Information technology

Author: **Bc. Patrik Drhlík**
Supervisor: RNDr. Klára Císařová, Ph.D.

Liberec 2015

TECHNICKÁ UNIVERZITA V LIBERCI
Fakulta mechatroniky, informatiky a mezioborových studií
Akademický rok: **2014/2015**

ZADÁNÍ DIPLOMOVÉ PRÁCE

(PROJEKTU, UMĚLECKÉHO DÍLA, UMĚLECKÉHO VÝKONU)

Jméno a příjmení:	**Bc. Patrik Drhlík**
Osobní číslo:	**M13000177**
Studijní program:	**N2612 Elektrotechnika a informatika**
Studijní obor:	**Informační technologie**
Název tématu:	**Strojové učení a Datamining**
Zadávající katedra:	**Ústav mechatroniky a technické informatiky**

Zásady pro vypracování:

1. Analyzujte metody supervizovaného učení používané v dataminingu.
2. Bude-li to možné absolvujte kurz "Machine Learning" na MOOC portále coursera, v opačném případě udělejte studii o možnosti vzdělávání na portále coursera v aktuálním čase.
3. Vybraný algoritmus DM naprogramujte tak, aby program sloužil jako výkladový a poznávací experiment studentům na e-learningovém portále ALS.
4. Téma zpracujte pro e-learningový kurz dataminingu na ALS ve formátu kurzů MOOC - záznam, příklady, testy a poznávací program.

Rozsah grafických prací: **dle potřeby dokumentace**

Rozsah pracovní zprávy: **40–50 stran**

Forma zpracování diplomové práce: **tištěná/elektronická**

Seznam odborné literatury:

[1] **Yong Yin, Ikou Kaku, Jiafu Tang: Data Mining, Springer London Ltd, 2011.**

[2] **Steve McConnell: Dokonalý kód, Computer Press, 2006.**

[3] **Kotler Philip: Marketing management, Grada, 2003.**

[4] **Hendl J.: Přehled statistických metod zpracování dat, Portál, 2006.**

[5] **Olivia Parr Rud: Datamining, Computer Press, a.s., 2006.**

[6] **http://www.msps.cz/data-mining/**

[7] **http://www.spss.cz/pasw_modeler.htm**

[8] **Tutoriály k SAP a další materiály na WWW stránkách.**

Vedoucí diplomové práce: **RNDr. Klára Císařová, Ph.D.**
Ústav mechatroniky a technické informatiky

Datum zadání diplomové práce: **10. října 2014**

Termín odevzdání diplomové práce: **15. května 2015**

V. Kopecký

prof. Ing. Václav Kopecký, CSc.
děkan

L.S.

Technická univerzita v Liberci - 14 -

Kolář

doc. Ing. Milan Kolář, CSc.
vedoucí ústavu

V Liberci dne 10. října 2014

Prohlášení

Byl jsem seznámen s tím, že na mou diplomovou práci se plně vztahuje zákon č. 121/2000 Sb., o právu autorském, zejména § 60 – školní dílo.

Beru na vědomí, že Technická univerzita v Liberci (TUL) nezasahuje do mých autorských práv užitím mé diplomové práce pro vnitřní potřebu TUL.

Užiji-li diplomovou práci nebo poskytnu-li licenci k jejímu využití, jsem si vědom povinnosti informovat o této skutečnosti TUL; v tomto případě má TUL právo ode mne požadovat úhradu nákladů, které vynaložila na vytvoření díla, až do jejich skutečné výše.

Diplomovou práci jsem vypracoval samostatně s použitím uvedené literatury a na základě konzultací s vedoucím mé diplomové práce a konzultantem.

Současně čestně prohlašuji, že tištěná verze práce se shoduje s elektronickou verzí, vloženou do IS STAG.

Datum:

Podpis:

Poděkování

Rád bych poděkoval vedoucí práce RNDr. Kláře Císařové, Ph.D. za její ochotu při pomoci s prací, doporučování materiálů a konstruktivní připomínky. Dále bych rád poděkoval své přítelkyni, která mě vždy velmi podporovala. Stejně tak jako mým rodičům, kteří mi vůbec celé studium umožnili.

Abstrakt

Práce je příspěvkem ke studiu formátu MOOC kurzů. Popisuje existující MOOC portály a osobní zkušenost z kurzu Machine Learning na portále Coursera. Na základě tohoto kurzu jsou v práci analyzovány metody supervizovaného strojového učení – lineární regrese, logistická regrese, neuronové sítě a support vector machines (SVM).

Výsledkem práce je výukový program pro experimentování s lineárním SVM s jednou cílovou proměnnou. Program pomáhá uživatelům pochopit důsledky volby jednotlivých parametrů. Je realizován jako webová aplikace v jazyce Python za použití frameworku Django. Je k dispozici účastníkům kurzu dataminingu v e-learningovém portálu ALS na TUL. V rámci práce je také popsán postup zpracování tématu lineární regrese ve formátu MOOC.

Klíčová slova

MOOC, Strojové učení, Datamining, Support Vector Machines, Python

Abstract

The thesis is a contribution to studying the MOOC format. It describes existing MOOC portals and a personal experience from the Machine Learning course on Coursera. It contains an analysis of supervised machine learning methods based on the Machine Learning course. Namely linear regression, logistic regression, neural networks and support vector machines (SVM).

The result of the thesis is an educational program for experimenting with a linear SVM with one target variable. Program helps users understand the meaning of choosing each parameters. It is made as a web application in the programming language Python and the Django framework. It is available for the participants of the datamining course in the e-learning portal ALS at TUL. The thesis also describes how the MOOC course for linear regression was made.

Keywords

MOOC, Machine learning, Datamining, Support Vector Machines, Python

Obsah

Seznam obrázků

Úvod

Žijeme v době, ve které je dostupnost informací téměř neomezená. Problémem však někdy může být to, že nemají dostatečnou formu a často pocházejí z nedůvěryhodných zdrojů. Tento problém začaly globálně řešit tzv. MOOC portály (viz kapitolu 2) přibližně od roku 2009. Světové univerzity se chopily tvorby kurzů z různých oblastí, daly jim jednotný formát a začaly je volně poskytovat všem nadšeným zájemcům.

V České republice žádný takový jednotný portál prozatím nenalezneme. Ambicí do budoucna bude vytvoření podobného portálu, který by provozovala Technická univerzita v Liberci. Prozatím se tímto tématem na univerzitě zabývá skupina lidí a moje práce by měla přispět ke studiu formátu MOOC kurzů, který by se využil v tvorbě zmíněného portálu.

Jedním z průkopníků těchto portálů je Andrew Ng, který působí na Stanfordské univerzitě v Kalifornii. Ještě známější je ale pravděpodobně díky strojovému učení a dataminingu. To je oblast, která velmi souvisí s ohromným růstem informací a dat. Díky algoritmům z této oblasti je možné získat cenné informace z dat, které na první pohled nejsou vidět. Tyto metody se běžně využívají např. v marketingových oblastech, ve finančnictví a v podstatě kdekoliv, kde je k dispozici velké množství dat. Dokážeme díky nim odhadnout, jaké produkty zlevnit, komu udělit úvěr a mnoho dalších.

Spojením těchto dvou zajímavých oblastí vyšlo jako vhodné vytvoření výukového programu pro vybrané téma ze strojového učení. Zejména s důrazem na to, aby člověk dokázal algoritmus pochopit tak, aby věděl, proč použít právě ten. Programy pro tuto oblast samozřejmě již existují. Příkladem může být třeba IBM SPSS Modeler. Algoritmy v něm ale fungují jako black box a je v něm vyžadována jejich předchozí znalost. Druhou stránkou věci je cena programu, která je vysoká a znemožňuje jeho využívání běžným zájemcům.

2 MOOC

MOOC [16] neboli anglicky *Massive Open Online Course* je kurz, který je obvykle zdarma, nemá omezený počet účastníků a je k dispozici každému, kdo má internetové připojení. Volně by se termín dal přeložit jako *Otevřený onlinový kurz pro širokou veřejnost*. Tradiční onlinové nebo e-learningové kurzy jsou většinou omezeny počtem účastníků, aby bylo zajištěno, že se instruktor kurzu bude všem náležitě věnovat. V MOOC kurzech individuální kontakt s instruktorem většinou chybí. Studenti si pak musí pomoct sami např. sdružováním ve skupinkách ať už po síti v internetových fórech nebo osobně. Hlavní myšlenkou je, aby bylo kvalitní vzdělání nebo pracovní kvalifikace k dispozici všem, kteří o to mají zájem.

2.1 Formát MOOC kurzů

Kurzy se dají rozlišit podle mnoha kritérií. To, že jsou zdarma, není úplným pravidlem. Najdou se i takové, které jsou zpoplatněné. Většina ale nabízí možnost, která je někde mezi. Je možné absolvovat kurz zdarma, ale pokud je zaplacen poplatek, tak je účastníkovi kurzu po úspěšném absolvování zaslán ověřený certifikát, kterým se dále může prezentovat např. před svým zaměstnavatelem nebo ve škole. Tyto certifikáty zasílají i některé kurzy, které jsou zcela zdarma. Nejsou ale stoprocentně ověřitelné.

Kurzy se také liší tím, jak dlouho trvají. Existují kurzy, které jsou vázané na konkrétní datum a většinou trvají v rozmezí 4–12 týdnů. Kurz v těchto případech udává informaci o časové náročnosti za týden, která se pohybuje od 1–10 hodin. Takový kurz vydává nové materiály maximálně na týden nebo dva dopředu. Na druhou stranu jsou zde i kurzy, které nejsou časově omezené a student je může absolvovat svým vlastním tempem.

Oblasti, kterými se kurzy zabývají jsou různorodé. Převládají jednoznačně technická či přírodovědná témata jako jsou kapitoly z matematiky, programování nebo

umělá inteligence. Nechybí však ani kurzy týkající se finančních, hudebních nebo psychologických oblastí.

Hlavními poskytovateli těchto kurzů jsou přední světové vysoké školy jako MIT, Harvard, Stanford nebo Princeton. Mimo univerzity se na vývoji kurzů podílí i společnosti jako Google nebo Facebook.

Kurz bývá složen z několika hlavních bloků, kde jeden blok odpovídá vyučovanému týdnu. Každý takový blok se dále člení na dílčí témata, která jsou podpořena prezentacemi, odkazy na další možné podpůrné materiály a zejména krátké videozáznamy, které trvají 8–15 minut. To je výborný způsob, který pomáhá udržovat pozornost. Navíc je běžné, že se v průběhu videozáznam jednou nebo dvakrát zastaví a dá studentovi jednoduchou otázku z probrané oblasti. Pokud odpoví správně, může pokračovat. Pokud ne, tak může také pokračovat, ale může to brát jako indikátor toho, že téma zcela nepochopil. V takovém případě je vhodné si předchozích pár minut pustit znovu.

Po absolvování teoretické části přichází část praktická. Na celý blok odpovídá test přibližně s 5 otázkami. Ty se vybírají náhodně z připraveného balíku otázek a každá má několik možných odpovědí, které se také mohou lišit při dalším spuštění testu. Kurzy se pak liší tím, kolik má na daný test student pokusů. Velmi často pokusy nebývají omezené vůbec. Student si test může procházet kolikrát chce, dokud neuspěje a dokud si nevyjasní případné nesrovnalosti.

V oblastech, kde se využívá programování je běžná druhá praktická část. V té účastník dostane podrobně sepsané zadání o úloze a aplikuje naučené poznatky do předpřipraveného kódu. Programuje jen jádro věci a nemusí řešit mnohdy časově náročné úkoly, které by ho od samotného hlavního úkolu mohly odradit. Výsledné výstupy hotových programů jsou odesílány na server, který řešení porovnává s výsledkem a informuje studenta o úspěchu či neúspěchu.

Z testů i programovacích cvičení student získává body, které jsou na konci kurzu sečteny a je z nich vyhodnocen celkový výsledek. Výsledný již výše zmiňovaný certifikát tak mají nárok získat pouze ti, kterým se podaří překonat nastavenou bodovou hranici.

Poslední velmi důležitou součástí MOOC kurzu je diskuzní fórum, kde mohou studenti probírat svá řešení, formovat studijní skupinky či hledat pomoc se zadanými úkoly. Mimo odpovědi od studentů je také možné získat odpověď od pověřeného personálu, který se kurzu věnuje.

2.2 Příklady aktuálně dostupných MOOC portálů

Provozovateli MOOC portálů, webových stránek sdružující MOOC kurzy, jsou především univerzity, které mají kapacity v podobě odborných pracovníků. Mnohdy se jedná o světově známé odborníky v různých oblastech. Největšími průkopníky mezi těmito portály jsou americké univerzity. Svůj velký portál má ale i Velká Británie nebo Austrálie a na dalších pracují i jiné země.

2.2.1 Khan Academy

Khan Academy [12] je organizace, kterou založil Salman Khan v roce 2006. Netvoří typické MOOC kurzy, jejichž formát je popsán v kapitole 2.1, ale nastavila jednu z jejich nejdůležitějších součástí – krátká videa o délce maximálně 15 minut. Pan Khan s tímto formátem přišel ještě před založením Khan Academy, když začal své mladé sestřenici nahrávat videa, ve kterých ji doučoval matematiku. Ta nahrál na server YouTube a posílal jí je. V dnešní době má jeho kanál na tomto serveru tisíce videí, které sledují miliony lidí po celém světě.

V Česku vznikl projekt Khanova škola, který spočívá v překladu videí z Khan Academy a následnému poskytování českému uživateli. Momentálně mají přeloženo přes dva tisíce videí z matematiky, chemie, informatiky a dalších oblastí.

2.2.2 Udacity

Pánové Sebastian Thrun a Peter Norvig v roce 2011 vytvořili kurz s názvem *Introduction to Artificial Intelligence* (Úvod do umělé inteligence). Hlavním cílem bylo, aby byl kurz pro všechny zdarma a k dispozici. Kurzu se zúčastnilo přes 160 000 studentů z více jak 190 zemí a nedlouho na to bylo založeno Udacity [19].

Velkým rozdílem oproti ostatním portálům je fakt, že jsou kurzy provozovány světovými firmami a ne univerzitami. Sebastian Thrun, jako jeden ze zakladatelů, je sám ze společnosti Google, kde se podílí na vývoji auta, které dokáže jezdit bez řidiče. Google je tedy jednou z firem, která kurzy poskytuje. Mimo Google se na jejich vývoji podílí i Facebook nebo AT&T – americká firma zabývající se nejen mobilními technologiemi.

Mimo jednotlivé MOOC kurzy poskytuje Udacity také tzv. *Nanodegrees* neboli nanovzdělání. Jedná se o sdružení kurzů do jednoho balíku, který dohromady tvoří

komplexní řešení problematiky. Aktuálně jich je na portále k dispozici 5 a každý se zabývá konkrétní oblastí. Jedním z nich je např. vzdělání v oblasti vývoje front endu webových aplikací. Nejsou ale zdarma a po týdenní zkušební době účastník platí 200 $ za měsíc. Netrvá v řádech týdnů, ale měsíců, konkrétně od šesti do dvanácti. Jejich smyslem je vytvoření kvalifikace, která povede k získání práce v daném oboru.

2.2.3 edX

Dalším známým portálem je edX [9], na jehož založení se v roce 2012 podílely univerzity Harvard a MIT. To jsou také jedny z univerzit, které zde kurzy poskytují. Mimo univerzit se zde na kurzech podílí také např. firma Microsoft. Je zde možné nalézt i kurzy, které nejsou speciálně zaměřené na studenty, ale na pracující odborníky, kterým se nadále hodí vzdělávání v oboru. Dalo by se to přirovnat k Nanodegree od Udacity (viz 2.2.2), ale pouze v rovině jednotlivých kurzů. Jinak je portál klasickým představitelem posyktovatelů MOOC kurzů.

2.2.4 Canvas Network

Americký portál s názvem Canvas Network [4] byl založen v roce 2012. Kurzy jsou ve valné většině vázané na daný čas a jsou uživateli vypsány na hlavní stránce bez jakékoliv kategorizace. V provozu jich je několik desítek a tvoří je univerzity. Canvas Network se pyšní zejména tím, že poskytuje vlastní nástroj na vytváření kurzů, tzv. LMS – *Learning Management System*.

2.2.5 Open2Study

Australským zástupcem na poli MOOC portálů je Open2Study [15]. Provozovatelem je sdružení australských univerzit. Aktuálně je zde k dispozici 49 kurzů. Rozdílem oproti ostatním portálům je to, že vždy běží pouze 4 týdny (pokud nejsou k dispozici neomezeně pro učení se vlastním tempem). Zajímavou možností je určitě poskytování tzv. akreditovaných kurzů. Ty patří lehce mimo samotný portál Open2Study, ale provozuje ho stejné sdružení univerzit, které se nebojí jít cestou vyučování online. V těchto programech je proto možné získat i vysokoškolský titul.

2.2.6 FutureLearn

Tento portál [10] provozuje stejnojmenná společnost, která má základnu v britských univerzitách. Mimo ty spolupracuje i s bristkou národní knihovnou, britským muzeem a dalšími důležitými britskými vzdělávacími institucemi. První kurzy spustili v září roku 2013 a po necelém roce se k nim zaregistrovalo na půl milionu studentů. V dnešní době se pyšní číslem překračujícím jeden milion studentů. Formát kurzů a druhy jejich poskytování se jinak v zásadě neliší od popisovaného formátu v kapitole 2.1.

2.2.7 Academic Earth

Texaský portál Academic Earth [1] začal být vyvíjen v roce 2008 a o rok později již začal poskytovat online kurzy zdarma. Kurzy zde poskytují výhradně univerzity zejména z USA. Ty doplňuje např. univerzita z Oxfordu, Velké Británie. Web jako takový je ale rozcestník, kde si student může podle kategorie vyhledat téma, kterému by se chtěl věnovat a odtud je pak přesměrován na univerzitní stránky nebo např. portál YouTube.

2.2.8 Coursera

Coursera [7] by se pravděpodobně dala považovat za nejznámější a také nejoblíbenější MOOC portál. Zázemí má na Standfordské univerzitě v Kalifornii. Jejími zakladateli jsou Daphne Koller a Andrew Ng. Andrew v roce 2011 vytvořil kurz s názvem *Machine Learning* (Strojové učení), na který se mu přihlásilo přes 100 000 lidí. To byl jeden z hlavních impulzů, které ho vedly k založení portálu s podobnými kurzy.

Kurzy zde poskytuje na 80 univerzit a dalších institucí z USA, Evropy i Asie. Formát kurzů je zhruba takový jako v kapitole 2.1. Navíc zde existují tzv. *Specializations* (Specializace), které se velmi podobají Nanodegrees, které poskytuje portál Udacity (viz kapitola 2.2.2). Poskytuje jich však větší množství. Specializace je složena ze 3–10 kurzů a celková doba na získání specializace může dosahovat až jednoho roku. Po absolvování všech kurzů specializace musí student celé snažení zakončit vlastním projektem, v kterém by měl aplikovat nabyté znalosti. K dokončení je za každý dílčí kurz třeba zaplatit 49 $. Pokud se žákovi nepodaří napoprvé kurz absolvovat, tak má možnost to zkusit ještě jednou a nemusí tak znovu platit.

2.3 Moje zkušenost s portálem Coursera

Na podzim loňského roku (2014) jsem se v rámci řešení této práce a podle jejího zadání přihlásil na kurz *Machine Learning* (Strojové učení) [2] na MOOC portále Coursera (viz kapitola 2.2.8). Je to stejný kurz, po jehož zpřístupnění veřejnosti Andrew Ng založil Courseru. Jednalo se o mou první zkušenost s MOOC kurzem. Kurz trval přesně deset týdnů a za úspěšné absolvování všech cvičení a testů sliboval certifikát potvrzující znalost probraných témat. Předmět byl vyučován v angličtině a k dispozici byly titulky v dalších 4 jazycích (čeština mezi nimi nebyla). Časová náročnost každého týdne byla odhadována na 5–7 hodin.

Každý týden jsem si na postup v kurzu vyhradil jeden den. Každý týden byly z teorie k dispozici jeden až dva přednáškové bloky, které se skládaly z dílčích 5–8 krátkých videí. V náročnějších tématech tato videa dosahovala nejvíce délky 20 minut, jinak přibližně kolem 10 minut. V průběhu videa se vždy jednou nebo dvakrát sám zastavil průběh a místo něj se objevila kontrolní otázka se čtyřmi možnými odpověďmi. Vždy se to týkalo tématu, o kterém Andrew právě mluvil, takže jsem si buď ověřil, že mu rozumím, a nebo že by se mi hodilo si kus pustit ještě jednou. S tímto způsobem podání látky jsem se setkal poprvé a velmi pozitivně mě to překvapilo, protože jsem látku chápal mnohem snadněji. Dělal jsem si sám poznámky na papír a odpovídání na kontrolní otázky mi tak vcelku nedělalo žádný problém.

Po shlédnutí přednášek přišla první praktická část, která měla ověřit teoretickou znalost problému. Jednomu přednáškovému bloku odpovídal jeden krátký test, který se vždy skládal z pěti otázek, kde každá měla čtyři možné odpovědi. Některé byly stavěny tak, že byla správná pouze jedna odpověď, ale většina tak, že jich mohlo, ale nemuselo být víc správných. To test trochu komplikovalo, ale o to lépe to dokázalo znalosti prověřit. Po odeslání jsem okamžitě viděl, co bylo správně a co ne. U všech odpovědí navíc leželo vysvětlení, proč je daná odpověď správně či špatně. Když jsem neměl plný počet bodů, tak jsem test opakoval znovu. Počet pokusů nebyl omezen a tak jsem vůbec nebyl ve stresu z toho, že by se mi to nemuselo povést. Jediným omezením byla možnost absolvování dalšího pokusu pouze každých deset minut. K nahlédnutí jsem pak vždy měl všechny pokusy, s kterými se např. daly porovnávat odpovědi pro zlepšení výsledku. Otázky i odpovědi se ale poměrně dost měnily. Takto jsem vždy postupoval dokola do získání plného počtu bodů. Pouze v jednom případě jsem potřeboval čtyři pokusy, jinak se mi to dařilo napoprvé nebo

napodruhé.

Když se mi povedlo absolvovat test, tak jsem přikročil k plnění programovacích cvičení. V prvním a posledním týdnu tato cvičení chyběla. Každé cvičení spočívalo ve stažení zazipovaného archivu se zdrojovými kódy a instrukcemi v pdf souboru. Programovalo se v programu Octave, který je obdobou Matlabu. Hlavním rozdílem je to, že je zdarma. Každé cvičení se skládalo ze 4–7 dílčích úkolů. V balíku souborů bylo vždy několik skriptů. Celé to bylo postavené tak, aby se student zabýval pouze jádrem věci. Mým úkolem bylo doplnit určitý počet skriptů, které se pak postupně volaly v hlavním připraveném programu. Bylo potřeba dělat vše postupně, protože na sebe úkoly vždy navazovaly a počítaly s tím, že předchozí funkce jsou již hotové. Hlavní program vždy porovnával hodnoty, které mají vycházet s těmi, které vracely mé funkce. Pokud se lišily, tak jsem věděl, že je něco špatně a musel jsem program předělat. Po vyhotovení všech částí se přímo z konzole Octave na server Coursery odesílaly výsledky. Přesněji se odesílaly moje naprogramované skripty, pouštěly se na tamnějším serveru a porovnávaly se výstupní hodnoty. Pokud nastala chyba, byl jsem informován. Pokud ne, bylo mi poblahopřáno a měl jsem pro jeden týden splněno.

Plný počet bodů za teoretické úlohy mohl být vždy 5 a z programovacích úloh 100. Na každou úlohu byly přibližně dva týdny času a pokud jsem to nestihl, tak jsem místo plného počtu mohl získat maximálně 80 %. Žádné body by se nepřipsaly pouze v případě nevyplnění nebo odeslání po termínu ukončení celého kurzu. 80 % byla také hranice, nad kterou se člověk musel dostat, aby získal certifikát (můj certifikát k nahlédnutí v příloze B).

Neodmyslitelnou částí celého snažení byla konzultace výsledků s ostatními studenty v diskuzních fórech. Nesměly se tam publikovat žádné výsledky ani konkrétní postupy a členové týmu kurzu to pečlivě kontrolovali a takové příspěvky rychle mazali. Naopak ale také odpovídali a radili, jak dál. Ke každému programovacímu cvičení vždy vytvořili Unit Testy s ověřenými daty, na kterých jsem si mohl ověřit správnou funkčnost.

Celkově mě celý svět kolem MOOC velmi nadchl, zejména pak v oblasti strojového učení. Kurz mě bavil od začátku do konce a na učení nových informací jsem se vždy velmi těšil. Nemohu jinak než tento kurz doporučit i ostatním nadšencům. Informace z kurzu jsou k dispozici i po jeho skončení a kdykoliv potřebuji sehnat informace o strojovém učení, tak vím, kde hledat.

3 Metody supervizovaného učení

Strojové učení [2, lekce 1, s. 15], [3, s. 60–61] je oblast, která se zabývá algoritmy, jež jsou schopné zlepšovat své výsledky na základě rostoucí zkušenosti. Dvě největší kapitoly této oblasti jsou supervizované a nesupervizované učení. Tato práce se zabývá pouze supervizovaným učením. Tyto algoritmy tvoří model nad známými daty. Jeden vzorek dat se skládá z páru vstupního a výstupního objektu. To, že data známe, znamená že pro každý vstupní objekt je známá hodnota výstupního objektu. V nesupervizovaném učení o vzorku tuto informaci nemáme. Vstupní objekt bývá vektor hodnot (atributů) a výstupní objekt pak hodnota. Pokud se jedná o data, na která se aplikuje regresní model, tak je výstupní hodnota spojitá. Pokud se aplikuje klasifikační model, tak je výstupní hodnota diskrétní (kategorická).

3.1 Popis použitého značení

Trénovací set – data, na kterých se model učí
Testovací set – data, na kterých se model testuje
$\mathbf{x}$ – vstupní vektor vzorku
$\mathbf{y}$ – výstupní hodnota vzorku
$\mathbf{m}$ – počet vzorků v trénovacím setu
$\mathbf{n}$ – délka vstupního vektoru (počet atributů)
θ – vektor parametrů ve funkci *hypotézy*
$(\mathbf{x}, \mathbf{y})$ – jeden vzorek
$(\mathbf{x^{(i)}}, \mathbf{y^{(i)}})$ – i-tý vzorek
$\mathbf{a_i^{(j)}}$ – aktivační funkce neuronové sítě jednotky i ve vrstvě j
$\mathbf{\Theta^{(j)}}$ – matice vah zajišťující funkční mapování z vrstvy j do vstvy $j+1$

3.2 Lineární regrese

Lineární regrese [18, s. 10], [3, s. 49–51] je statistická metoda, která se snaží najít závislost mezi vstupní a výstupní proměnnou. Na základě známých vzorků a vstupní hodnoty nových dat predikuje jejich výstupní hodnotu. Účelem je nalezení přímky, pro kterou platí, že součet kvadrátů reziduí je minimální. Funkce, která přímku reprezentuje se nazývá *hypotéza* (angl. *hypothesis*) [2, lekce 2, s. 4] Název volím podle absolvovaného anglického kurzu Machine learning. V české literatuře se místo pojmu hypotéza používá název regresní funkce. Je popsána následujícím vzorcem:

$$h_\theta(x) = \theta_0 + \theta_1 x. \tag{1}$$

Hlavní otázkou je nalezení všech parametrů θ_i. K tomu se využívá *hodnotová funkce* (angl. *cost function*) [2, lekce 2, s. 8–21], která je popsána následovně:

$$J(\theta_0, \theta_1) = \frac{1}{2m} \sum_{i=1}^{m} (h_\theta(x^{(i)}) - y^{(i)})^2. \tag{2}$$

Tato funkce je konvexní a má miskovitý tvar (angl. *bowl-shaped*). To tedy znamená, že pro jednu dvojici hodnot θ_0 a θ_1 je její hodnota minimální. Tyto konkrétní parametry vytvoří přímku, která minimalizuje součet kvadrátů odchylek. Jednou z možností minimalizování $J(\theta_0, \theta_1)$ je využití *metody postupného klesání* (angl. *gradient descent*) [2, lekce 2, s. 23–48]. Tato metoda postupně mění parametry θ_0 a θ_1 tak, že snižuje hodnotu $J(\theta_0, \theta_1)$. Využívá k tomu konstantu nazývanou *míra učení* (angl. *learning rate*) a značenou α. Ta určuje velikost kroku, kterým se tato metoda přibližuje k minimu $J(\theta_0, \theta_1)$. Je potřeba inicializovat parametry θ_j na náhodnou hodnotu. Nejčastěji se oba parametry ale inicializují na 0. Dokud nebude výsledek konvergovat, bude se upravovat podle vzorce 3:

$$\begin{aligned} &\text{opakovat dokud nekonverguje } \{ \\ &\qquad \theta_j := \theta_j - \alpha \frac{\partial}{\partial \theta_j} J(\theta_0, \theta_1) \\ &\qquad\qquad (\text{pro } j \ = 1 \text{ a } j \ = 0) \\ &\}. \end{aligned} \tag{3}$$

Existuje také analytické řešení nalezení parametrů θ [2, lekce 4, s. 23–28], které

se nazývá *normální rovnice* (angl. *normal equation*). Výhodou je, že není potřeba nastavovat míru učení α. Nevýhodou může být použitelnost pouze do určité velikosti vstupního vektoru (počtu atributů) – maximálně $n = 1000$. Jak je vidět ve vzorci 4, výpočet při větším počtu atributů může být výpočetně náročný kvůli hledání inverzní matice po násobení dvou matic.

$$\theta = (X^T X)^{-1} X^T y \tag{4}$$

Důležitým krokem je výběr míry učení α tak, aby výsledek konvergoval. Pokud je hodnota příliš malá, tak může algoritmus trvat dlouho, ale o to blíže k minimu se dostane. Při zvolení příliš vysoké hodnoty se může stát, že algoritmus nebude vůbec konvergovat nebo dokonce divergovat. Po zjištění parametrů regresní funkce lze jednoduše predikovat nové hodnoty dosazením do funkce hypotézy (viz vzorec 1). Je tak možné si jednoduše ověřit úspěšnost modelu aplikací na testovacím setu.

Některá data se ale nechovají lineárně. Řešením je aplikace nelineární hypotézy, která může být exponenciální nebo polynomiální a mít libovolně vhodný stupeň. Např. kvadratická nebo druhá odmocnina:

$$\begin{aligned} h_\theta(x) &= \theta_0 + \theta_1 x^2, \\ h_\theta(x) &= \theta_0 + \theta_1 \sqrt{x}. \end{aligned} \tag{5}$$

3.3 Logistická regrese

Logistická regrese [18, s. 12] je velmi podobná lineární regresi. Výstupní hodnota však není spojitá, ale diskrétní. Může nabývat libovolného množství hodnot, ale nejčastěji se využívají hodnoty dvě. Negativní (0) a pozitivní (1) třída. Model logistické regrese určuje pravděpodobnost, se kterou je možné určit, jestli vzorek patří do pozitivní třídy. Obor hodnot funkce *hypotézy* musí být v intervalu $0 <= h_\theta(x) <= 1$ [2, lekce 6, s. 4]. Predikujeme, že vzorek patří do pozitivní třídy, pokud $h_\theta(x) >= 0.5$. Nemůže být lineární, aby lépe odrážela pravděpodobnostní rozložení. Využívá se proto tzv. sigmoidální funkce [3, s. 52] ($g(z)$ ve vzorci 6). Hypotéza [2, lekce 6, s. 6] vypadá následovně:

$$h_\theta(x) = g(\theta^T x),\ kde$$
$$g(z) = \frac{1}{1+e^{-z}}. \tag{6}$$

Kdyby byla hodnotová funkce stejná jako v lineární regresi (viz vzorec 2), tak by za použití sigmoidální funkce v *hypotéze* nebyla konvexní a proto bychom v ní nedokázali nalézt globální minimum. Celková hodnotová funkce má tvar popsaný ve vzorci 8. Je spojením dvou logaritmických funkcí (viz vzorec 7), které jsou obě konvexní.

$$Cost(h_\theta(x), y) = \begin{cases} -\log h_\theta(x) & \text{if } y\ = 1 \\ -\log 1 - h_\theta(x) & \text{if } y\ = 0 \end{cases} \tag{7}$$

$$\begin{aligned} J(\theta) &= \frac{1}{m}\sum_{i=1}^{m} y^{(i)} Cost(h_\theta(x^{(i)}, y^{(i)})) \\ &= -\frac{1}{m}[\sum_{i=1}^{m} y^{(i)} \log h_\theta(x^{(i)}) + (1-y^{(i)}) \log (1 - h_\theta(x^{(i)}))] \end{aligned} \tag{8}$$

Nalezení parametrů θ je stejně jako v lineární regresi (kapitola 3.2) možné pomocí *metody postupného klesání* (viz vzorec 3). Algoritmus funguje totožně. Rozdílem je pouze použití funkce *hypotézy* pro logistickou regresi. Existují však ještě sofistikovanější algoritmy pro nelineární optimalizaci. Jsou jimi např. algoritmy BFGS nebo L-BFGS [2, lekce 6, s. 24]. Jsou mnohdy rychlejší než *metoda postupného klesání* a nepotřebují ke své funkci nastavení hodnoty míry učení α. Nevýhodou může být to, že jsou celkově složitější.

3.4 Neuronové sítě

Další supervizovanou metodou strojového učení jsou neuronové sítě [18, s. 13–14]. Na rozdíl od obou zmíněných regresí nevychází ze žádného statistického modelu. Inspiruje se funkcí lidského mozku a celý proces se podobá rozpoznávání vzorů a minimalizaci chyby. Metoda se poučí ze zkušenosti z každé přijaté informace.

Neuronovou síť tvoří skupina vrstev, kde v každé je sada uzlů (viz obrázek 3.1). První vrstva se nazývá vstupní a je totožná se vstupním vektorem x. Každý uzel zde odpovídá jednomu atributu a je jich tedy vždy n. Poslední vrstva (na obrázku

3.1 třetí) se nazývá výstupní. Může obsahovat libovolný počet uzlů v závislosti na řešeném problému. Pokud je výstupem neuronové sítě pouze binární hodnota, tak je uzel jen jeden. Mezi vstupní a výstupní vrstvou může být libovolný počet tzv. skrytých vrstev o libovolných počtech uzlů. Tyto počty závisí na složitosti problému a jsou předmětem experimentování. Nejběžnější je však použití jedné skryté vrstvy. V případě použití více vrstev se doporučuje nastavení stejného počtu uzlů pro každou z nich [2, lekce 9, s. 28]. Dobrým zvykem je využití tzv. nultých uzlů (viz uzly označené +1 na obrázku 3.1) do všech vrstev, kromě té poslední. Hodnoty těchto uzlů jsou vždy rovny jedné.

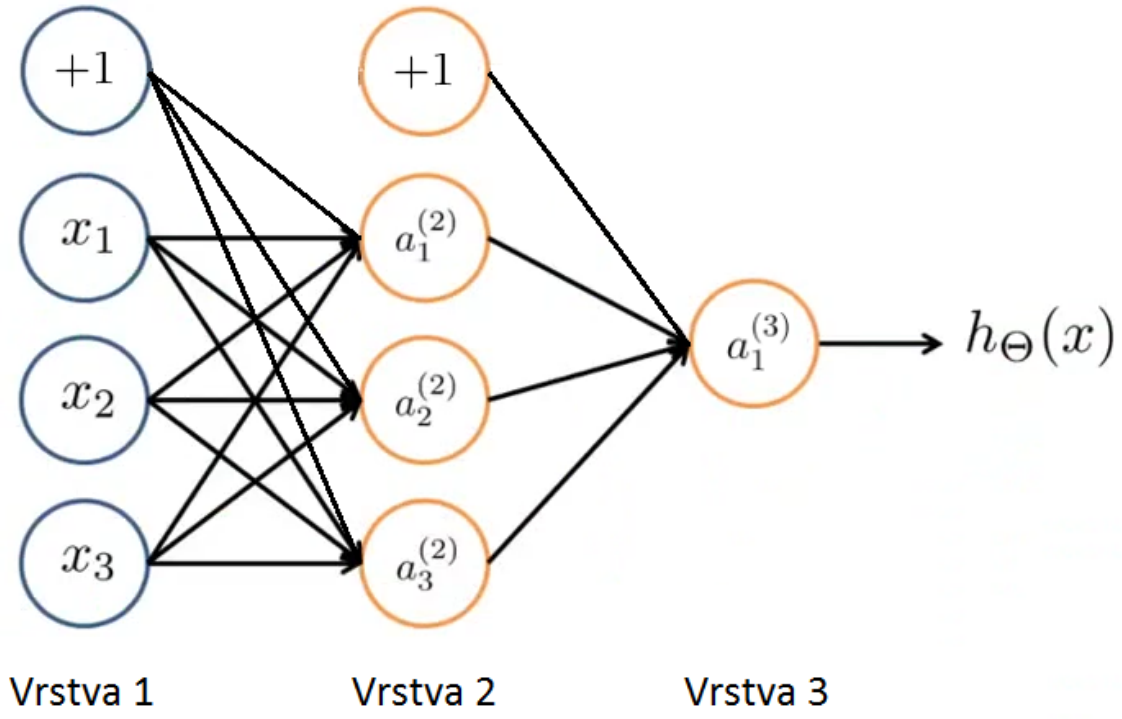
Každý uzel v každé vrstvě kromě první je definován tzv. *aktivační funkcí*. Pokud by neuronová síť neměla žádnou skrytou vrstvu, tak by se jednalo o logistickou regresi. Jako *aktivační funkce* se totiž používá sigmoidální funkce (viz $g(z)$ ve vzorci 6). Poslední uzel je zároveň funkce *hypotézy*. Celkově se tedy na neuronovou síť dá pohlížet jako na skupinu vnořených logistických regresí.

K vypočtení celkové *hypotézy* neuronové sítě je zapotřebí o něco víc informací než v logistické regresi. Namísto použití vektoru θ se zde používají matice $\Theta^{(j)}$ a je jich o jednu vrstvu méně než je jejich celkový počet. Je možné je považovat za takové mezivrstvy. Rozměr každé z nich závisí na rozměrech sousedních vrstev. Pokud je s_j počet uzlů ve vrstvě j a s_{j+1} počet uzlů ve vrstvě $j+1$, tak má matice $\Theta^{(j)}$ rozměr $s_{j+1} \times (s_j + 1)$ [2, lekce 8, s. 20] (bez vložených nultých uzlů).

Jak se ke konečnému výsledku dojde by mělo být patrné z předpisů pro všechny aktivační funkce z následujících vzorců (konkrétní předpis pro neuronovou síť z obrázku 3.1):

$$\begin{aligned} a_1^{(2)} &= g(\Theta_{10}^{(1)} x_0 + \Theta_{11}^{(1)} x_1 + \Theta_{12}^{(1)} x_2 + \Theta_{13}^{(1)} x_3), \\ a_2^{(2)} &= g(\Theta_{20}^{(1)} x_0 + \Theta_{21}^{(1)} x_1 + \Theta_{22}^{(1)} x_2 + \Theta_{23}^{(1)} x_3), \\ a_3^{(2)} &= g(\Theta_{30}^{(1)} x_0 + \Theta_{31}^{(1)} x_1 + \Theta_{32}^{(1)} x_2 + \Theta_{33}^{(1)} x_3), \\ h_\Theta(x) = a_1^{(3)} &= g(\Theta_{10}^{(2)} a_0^{(2)} + \Theta_{11}^{(2)} a_1^{(2)} + \Theta_{12}^{(2)} a_2^{(2)} + \Theta_{13}^{(2)} a_3^{(2)}). \end{aligned} \tag{9}$$

Výpočet hodnotové funkce je prakticky totožný s hodnotovou funkcí logistické regrese (viz vzorec 8). V neuronové síti je ale nutné počítat s tím, že může mít právě K výstupních uzlů a je tedy tak nutné přes všechny tyto výstupy iterovat. Jednoduchou úpravou a přeznačením z vektoru θ na matici Θ vzniká tento vztah:



Obrázek 3.1: Příklad neuronové sítě se třemi vrstvami [2, lekce 8, s. 19].

$$J(\Theta) = -\frac{1}{m}[\sum_{i=1}^{m}\sum_{k=1}^{K} y_k^{(i)} \log\left((h_\Theta(x^{(i)}))_k\right) + (1 - y_k^{(i)}) \log\left(1 - (h_\Theta(x^{(i)}))_k\right)]. \quad (10)$$

Výsledný záměr s neuronovou sítí je stejný jako s lineární i logistickou regresí. Potřebujeme najít takové parametry Θ, pro které bude $J(\Theta)$ minimální. Postup je principiálně stejný jako u logistické regrese. Opět se využívá *metoda postupného klesání*, ke které jsou potřeba všechny parciální derivace $\frac{\partial}{\partial \Theta_{ij}^{(l)}} J(\Theta)$, kde l je číslo mezivrstvy (nebo vrstvy, ke které se konkrétní matice parametrů Θ váže) a příslušné indexy i a j označující jednotlivé prvky této matice [2, lekce 9, s. 5–13]. Pokročilejší optimalizační metodou je spojení algoritmů dopředné a zpětné propagace (angl. *forward and back propagation*) [3, s. 167].

3.5 Support Vector Machines

Autory algoritmu Support Vector Machines (SVM) jsou Vapnik a Cortes, kteří ho publikovali v roce 1995 [6]. Česky se dá volně přeložit jako metoda podpůrných vektorů. Může být využit jak pro regresní, tak i klasifikační analýzu. Stejně jako předchozí algoritmy může být klasifikační SVM použit pro rozlišování více než dvou tříd. V případě klasifikátoru, který rozděluje data pouze na dvě množiny, výstupní objekt nabývá hodnot $y \in \{-1, 1\}$.

SVM se používá jak pro separaci lineárních, tak i nelineárních dat. V obou případech se snaží najít takovou křivku (v lineárním případě přímku), která množiny rozdělí co největší mezerou (angl. *margin*). Takovou křivku nelze vždy nalézt. Zabývat se ale budu pouze lineárním případem. Vapnik a spol. v [6] pro tuto příležitost zavádí pojem *soft margin*. Taková mezera stále rozděluje obě množiny v co největším měřítku, ale dovoluje špatnou klasifikaci některých vzorků (např. outlierů[1]).

Místo parametrů θ se v SVM pracuje s parametry w, b a *hypotéza* vypadá jako

$$h_{w,b}(x) = g(w^T x \ + b). \tag{11}$$

Parametr b zde nahrazuje θ_0 a parametr w nahrazuje zbylý vektor $[\theta_1, \ldots, \theta_n]$ [13, s. 3]. K nalezení parametru w lze dospět vyřešením [13, s. 13]

$$w = \sum_{i=1}^{m} \alpha_i y^{(i)} x^{(i)}, \tag{12}$$

kde je úkolem nalezení parametrů α_i. Většina z těchto parametrů bude nulových a ty, které budou nenulové, určují podpůrné vektory (angl. *support vectors*). To jsou takové vzorky, které jsou poblíž přímky rozdělující obě skupiny [13, s. 11]. Po nalezení parametrů α_i a w již jen stačí zjistit poslední parametr b z rovnice

$$w^T x + b = 0, \tag{13}$$

která označuje body na přímce rozdělující obě množiny bodů [13, s. 5]. V samotném algoritmu se vyskytuje velmi důležitý parametr C, který ovlivňuje chování rohodující přímky. Umožňuje využití zmíněného *soft marginu*. Při malých hodnotách

[1] Outlier je vzorek v datech, který se vlastnostmi blíží jedné třídě, ale patří do druhé.

v řádech desetin jednotek až jednotek se chová tak, že zanedbává outliery. Jakmile se hodnota C zvyšuje, tak se snaží najít množiny rozdělit co nejpřesněji, včetně outlierů.

V případě nelineárního SVM se dá uplatnit tzv. *kernel trick*. Spočívá v tom, že se nevyužijí původní vstupní hodnoty x, ale zobrazí se do jiného prostoru (s vysokou dimenzí), díky kterému se ulehčí výpočetní náročnost. Zobrazení provádí tzv. *kernel* funkce, která může být definována různě. Při nepoužití kernelu v případě lineárního SVM se mluví o *lineárním kernelu*. Dalšími příklady jsou např. *RBF* nebo *Gaussiánský* kernel [13, s. 16].

4 Výukový program pro experimentování s algoritmem SVM

Když jsem se poprvé setkal s dataminingem, tak jsme zkoumali různé případové studie. Jedním z úkolů, který nás v kurzu čekal, bylo vytvoření vlastní případové studie na námi zvoleném datasetu. Vyučovaným a doporučovaným nástrojem na vytvoření studie byl program IBM SPSS Modeler. Ten obsahuje sadu nástrojů pro celkovou analýzu dat včetně algoritmů strojového učení. Každý takový algoritmus ale funguje jako *black box*[1]. Dělalo mi problém pochopit, jaký algoritmus použít a především s jakými parametry. Z tohoto důvodu jsme došli k závěru, že by bylo vhodné vytvořit výukový program, který by pomohl studentům pochopit, jak tyto algoritmy fungují. V rámci této práce jsme zvolili implementaci lineárního SVM.

4.1 Použité technologie

Při tvorbě výukového programu se rozhodovalo mezi implementací desktopové a webové aplikace. Nakonec byla kvůli snadné přenositelnosti a dostupnosti i na mobilních platformách zvolena webová aplikace. Velmi oblíbený jazyk pro programování webových aplikací je PHP. Ten ale nebyl shledán jako vhodný pro implementaci těchto výpočetně náročnějších algoritmů. V kurzu *Machine learning* (viz 2.3) jsme používali program Octave a programovali jsme tam několik algoritmů strojového učení včetně lineárního SVM. Byl proto zvolen jazyk, který umí velmi dobře zpracovávat vektory a matice a navíc je schopný vytvořit webovou aplikaci – Python. Má k dispozici knihovny numpy a matplotlib, které ho velmi přibližují k funkčnosti matlabu. Díky výborně popsaným tutoriálům webového frameworku Django byl vývoj aplikace bezproblémový.

[1] Není známá jeho implementace ani způsob, jakým funguje.

4.1.1 Python Django framework

Django framework [8] umožňuje rychlou a snadnou tvorbu webových aplikací. Volně se drží návrhového vzoru *MVC*[2]. Oproti ostatním frameworkům není Django o nic ochuzen. Zajímavostí navrch je např. automatické vygenerování administračního rozhraní, které se vytvoří z definovaných modelů. To ale nebylo v aplikaci nevyužil, protože nebylo potřeba pracovat s databází ani vytvářet datové modely.

Z bezpečnostního hlediska je framework také velmi dobře vybaven. Součástí je systém autentifikace uživatelů. Jsou ošetřeny nejběžnější bezpečnostní slabiny jako je SQL injection, cross-site scripting, cross-site request forgery nebo clickjacking. Díky tomu, že je Django open source projekt, tak do něj existuje celá řada již vytvořených komponent, pomůcek a dalších věcí, které dokáží urychlit a usnadnit vývoj webu. Volba Djanga se ukázala jako velmi vhodná pro daný problém a také přinesla znalost nové technologie.

4.1.2 NumPy

NumPy je knihovna jazyka Python [14]. NumPy se využívá ve vědecké oblasti k sofistikovanějím výpočtům. Součástí jsou všechny možné matematické funkce, které jsou potřeba k počítání libovolných problémů. Hlavní výsadou je však podobnost s matlabem. Většina funkcí, které jsou v matlabu, jsou i v NumPy se stejnými názvy. Hlavní výhodou je zpracování N–rozměrných polí, které se velmi jednoduše používá. Pro vektory i matice jsou implementovány všechny základní matematické operace. Stejně jako v matlabu zde funguje maticové násobení i násobení po prvcích (i ostatní operace jako sčítání apod.). Knihovna také dovoluje využití zdrojového kódu v jazyce C/C++ nebo Fortran. Nakonec nechybí ani další nástroje z lineární algebry, Fourierova transformace nebo práce s pseudonáhodnými čísly.

4.1.3 Matplotlib

Další použitou knihovnou je Matplotlib [11]. Je to silný nástroj, který slouží k vykreslování 2D grafů. Její síla a oblíbenost spočívá v tom, že obsahuje nástroj *pyplot*, který má syntax velmi podobný matlabu. Uživatel má možnost nastavení veškerých vykreslovaných informací. Jako datové vstupy pro grafy je možné využívat i objekty

[2]Model View Controller je architektura, která rozděluje aplikaci na tři části – data (model), zobrazení pomocí šablon (view) a propojovací vrstvu (controller).

polí z knihovny NumPy 4.1.2. Grafy lze exportovat do souborů, zobrazovat skrze konzoli a také odesílat ve správném formátu prohlížeči. Díky tomu se dá zobrazení vypočtených grafů implementovat do webové aplikace. Např. využitím zmiňovaného Django frameworku 4.1.1. Export je možný do rastrových, ale i vektorových formátů (např. SVG). Použití je velmi snadné a ovládání základních funkcí je téměř totožné s matlabem.

4.2 Struktura aplikace

Celá aplikace respektuje rozložení složek a souborů vycházející z Django frameworku:

- project – kořenový adresář s názvem aplikace,
  - project – adresář s nastavením celého projektu,
  - static – adresář se statickými soubory (obrázky, CSS[3], JS[4]),
  - svm – adresář s výukovým programem SVM,
    - datasets – adresář s datasety,
    - models – adresář pro cachování modelů,
    - templates – adresář pro šablony aplikace SVM,
    - templatetags – adresář pro pomocné filtery v šablonách,
    - svm_train.py – soubor s třídami pro zpracování SVM,
  - templates – adresář se šablonami pro celý projekt,
  - manage.py – soubor, který obsahuje užitečné programy pro obsluhu Django projektu.

V adresáři s nastavením celého projektu jsou tři velmi důležité soubory – *settings.py*, *urls.py* a *views.py*. V prvním zmíněném dochází k registraci aplikací používaných v projektu, nastavení časového pásma, databáze apod. V *urls.py* se nastavuje globální routování[5] a v třetím se řeší předávání parametrů šablonám z jednotlivých pohledů.

[3]Kaskádové styly nastavující vzhled stránky.
[4]Soubory skriptovacího jazyka JavaScript.
[5]Podoba URL odkazu stránek.

Za zmínku z kořenového adresáře stojí ještě soubor *manage.py*. Byl využíván zejména pro spouštění webového serveru, na kterém aplikace běžela. Jedná se o jednoduchý server, který je součástí Django instalace. Tvůrci Djanga ale pro nasazení do produkce doporučují využití lepších serverů jako je např. Apache. Druhou důležitou součástí souboru je tzv. *Django shell* neboli konzole. Díky ní bylo dosaženo jednoduchého testování aplikace i z příkazové řádky bez potřeby zapínání webového serveru.

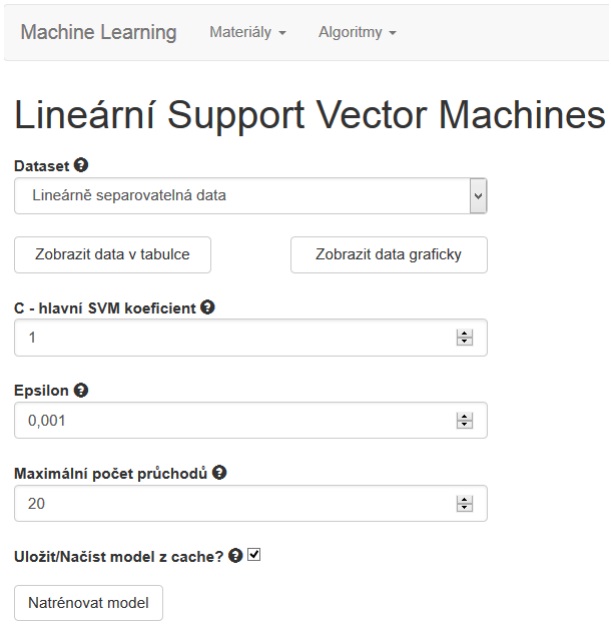
Konkrétní aplikace je v adresáři *svm*. Všechny výpočetní prvky používané k reprezentaci dat, vytváření modelů SVM a jejich testování jsou v souboru *svm_train.py*. Existují zde i další soubory, které nejsou v adresářové struktuře uvedeny. Názvy mají stejné jako v adresáři s nastavením celého projektu a doplňují jeho pravidla pro svoji konkrétní potřebu. Datasety, nad kterými algoritmus pracuje, jsou ve složce *datasets*. Protože trénování modelů za použití určitých parametrů může být časově náročnější, existuje složka *models*, do které se natrénované modely ukládají.

4.3 Popis uživatelského prostředí aplikace

Aplikace byla rozdělena do třech základních sekcí. Tou první je domovská stránka, na které jsou pouze statické informace o celém projektu, který je věnován strojovému učení. Do budoucna se plánuje implementace dalších algoritmů z této oblasti, o kterých už bylo psáno dříve jako např. lineární a logistickou regresi nebo neuronové sítě. Tato práce se zaměřuje pouze na lineární SVM. Celá aplikace je vytvořena tak, aby byla velmi jednoduše pochopitelná a ovladatelná. V horním menu jsou odkazy na hlavní stránku, materiály k algoritmům a sekce pro samotnou demonstraci a experimentování s algoritmy. V sekci s materiály jsou umístěny odkazy na literaturu, z které se v této práci čerpá. Ve zmíněné literatuře je detailně vysvětlená funkčnost implementovaného algoritmu. Dále také seznam používaných pojmů a popis aplikace. Konkrétně popis jednotlivých datasetů, vstupních parametrů a způsobu vyhodnocení natrénovaného modelu. Nechybí tam ani odkaz na kurz *Machine Learning* na portále Coursera, který by měl vyzkoušet každý, kdo se zajímá o tuto problematiku.

Tou hlavní částí je ale sekce algoritmy. Při prvním příchodu se uživateli zobrazí jednoduchý formulář na levé straně obrazovky, kterým se nastavují parametry k trénování modelu (viz obrázek 4.1). Pro demonstraci jsou pro uživatele připraveny tři datasety (viz kapitola 4.4), na kterých může být algoritmus natrénován. Po

vybrání datasetu si uživatel může data prohlédnout a něco o nich na první pohled zjistit. K dispozici jsou dvě možnosti – zobrazení dat v tabulce a grafické zobrazení dat. V tabulkovém zobrazení se lze také např. dozvědět, kolik vzorků daný dataset obsahuje, kolik z nich je negativních a kolik pozitivních. V grafickém zobrazení uživatel vidí všechny vozrku v grafu. Pozitivní jsou označeny černým křížkem a negativní žlutým kolečkem. V případě, že je k datasetu k dispozici i tzv. testovací (validační) dataset, uvidí je oba.

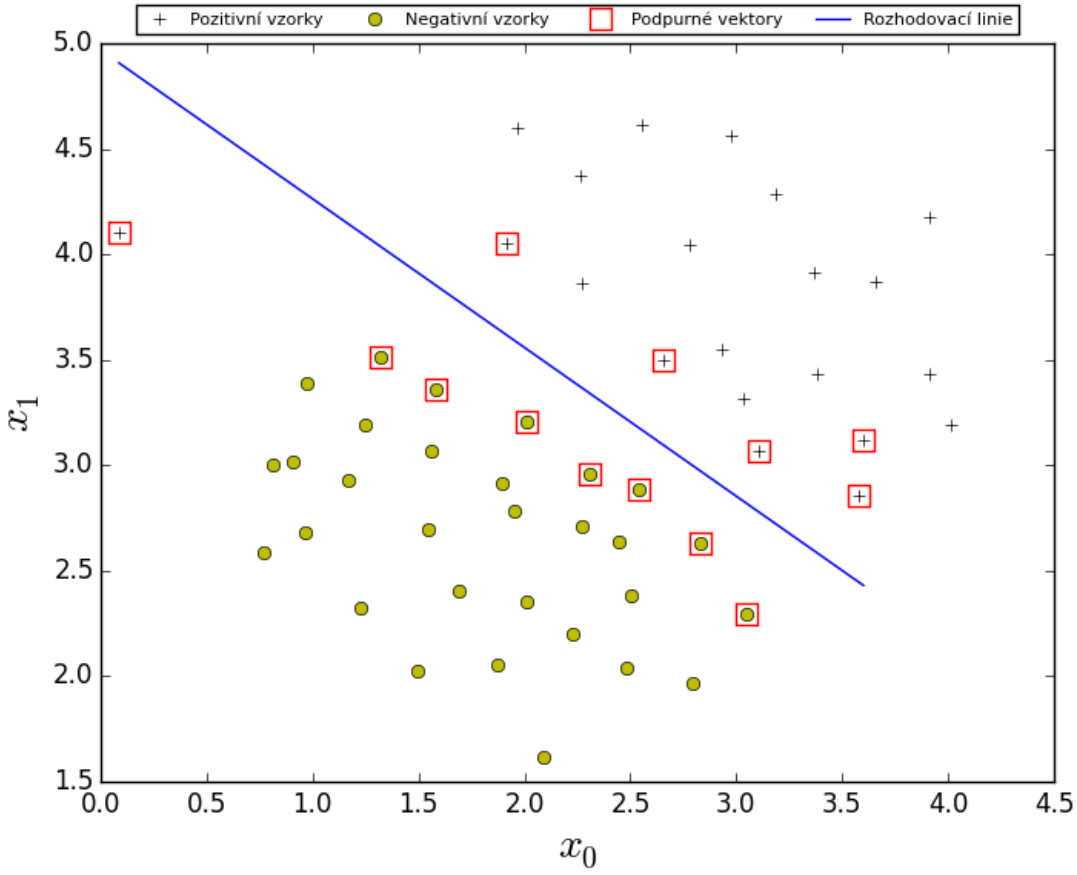


Obrázek 4.1: Screenshot formuláře pro natrénování modelu SVM.

V použité implementaci algoritmu je mimo datasetu potřeba zadat další tři parametry SVM. Jsou jimi parametr *C*, *Epsilon* a *Maximální počet průchodů*. O tom, co který parametr znamená se píše v kapitole 4.5. Před odesláním má uživatel ještě možnost nastavení cachování modelů. Pokud je políčko zakškrtlé, tak se natrénovaný model uloží do souboru. Pokud už byl model se stejnými parametry natrénován, tak se ze souboru načte. V nezaškrtlém případě se model vždy trénuje znovu a odnikud se nenačítá.

K úspěšnému odeslání formuláře a zahájení trénování modelu je potřeba správně vyplnit všechny údaje. Pokud některý z údajů chybí, tak je uživatel musí doplnit. Po odeslání vyplněného formuláře se natrénuje model a zobrazí se výsledky. Hlavním výsledkem je vykreslení grafu, v kterém jsou zakresleny všechny vzorky, přímka (rozhodující linie) a označené podpůrné vektory. Konkrétní výsledek je vidět na obrázku 4.2. Podpůrné vektory jsou ohraničeny červenými čtverečky. To jsou takové

vzorky, které mají nenulové koeficienty α (viz 3.5).



Obrázek 4.2: Screenshot výsledku natrénovaného modelu SVM podle parametrů z obrázku 4.1.

Druhým výsledkem je jednoduchá tabulka, která procentuálně hodnotí úspěšnost modelu na testovacích datech. Ta se počítá z podílu počtu všech správně predikovaných vzorků a počtu všech vzorků v datasetu. Zobrazuje také počet podpůrných vektorů (zkratka SV), celkový počet vzorků v datasetu a počty vzorků, v kterých se algoritmus zmýlil nebo byl naopak úspěšný. To jsou čtyři čísla určující falešně pozitivní (angl. *false positive*), falešně negativní (angl. *false negative*), pravdivě pozitivní (angl. *true positive*) a pravdivě negativní (angl. *true negative*) vzorky (viz tabulku 4.1).

Tabulka 4.1: Výsledky testování modelu podle parametrů z obrázku 4.1.

	Nepravdivé	Pravdivé
Pozitivní	0	20
Negativní	1	30
Celkový počet vzorků	51	
Úspěšnost	98.039 %	
Počet SV	13	

4.4 Použité datasety

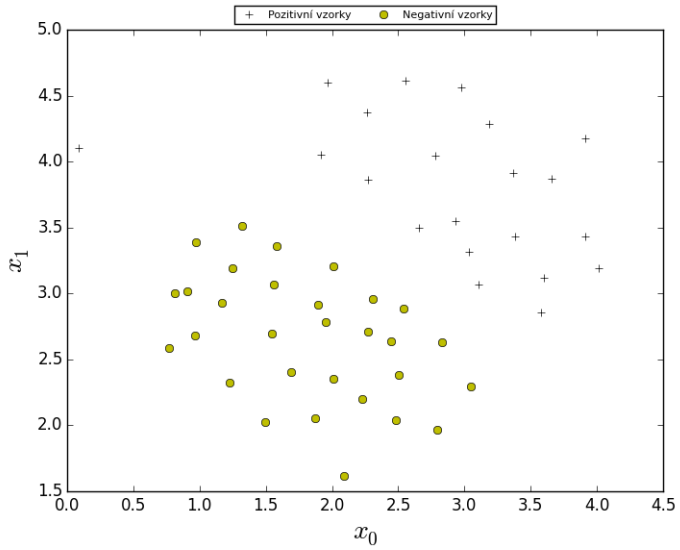
Podle [13] byly připraveny tři datasety, na kterých si uživatel může vyzkoušet funkčnost algoritmu. Každý z datasetů má různé vlastnosti, které jsou popsány v následujících kapitolách. V tabulce 4.2 je zaznamenáno, kolik a jakých vzorků jednotlivé datasety obsahují. Poslední dataset v tabulce je označen slovem *val*. To značí, že se jedná o validační dataset, který se používá k testování modelu natrénovaného nad datasetem označeným stejným číslem.

Tabulka 4.2: Porovnání informací o jednotlivých datasetech.

Dataset	Počet vzorků	Počet atributů	Počet y=1	Počet y=0	Lineárně separovatelný
4.4.1	51	3	21	30	ano
4.4.2	863	3	480	383	ne
4.4.3	211	3	106	105	částečně
4.4.3 val	200	3	87	113	–

4.4.1 Lineárně separovatelná data

Na tomto datasetu (viz obrázek 4.3) je jasně vidět, že existuje přímka, která dokáže lineárně rozdělit prvky do dvou skupin. Na tomto příkladu algoritmus lineární SVM funguje nejlépe. Hlavní zajímavostí je však to, že je zde přibližně na souřadnicích $(x_0, x_1) = (0.1, 4.1)$ outlier. Očividně je blíže k druhé skupině, než ke které ve skutečnosti patří. Uživatel si na tomto datasetu může jednoduše vyzkoušet důležitost jednotlivých parametrů a to, jak je potřeba je nastavit, aby např. outlier obsahovali nebo ne.



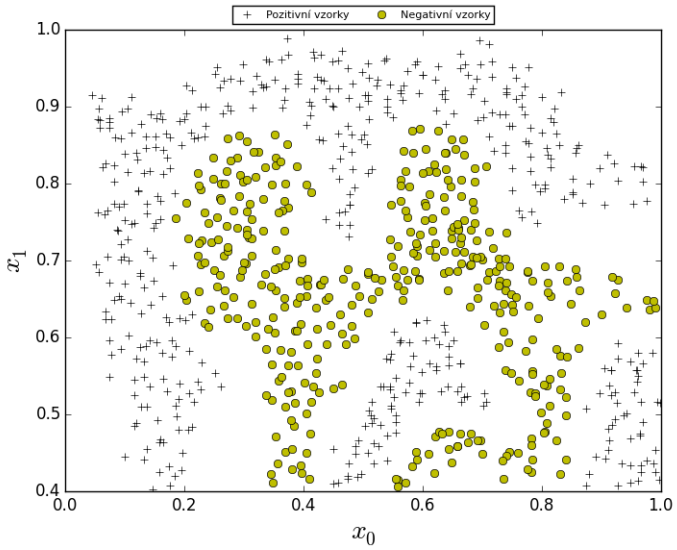
Obrázek 4.3: Lineárně separovatelný dataset.

4.4.2 Lineárně neseparovatelná data

Tento dataset (viz obr. 4.4) není vhodný ke zpracování lineárním SVM. Začlenil do výběru byl proto, aby si to uživatel uvědomil a např. navrhl možné řešení. Tím by v tomto případě bylo použití SVM s jiným kernelem (např. RBF), který by dokázal najít nelineární křivku, která data dokáže rozdělit. Linie, kterou by křivka měla vézt je na první pohled také zřejmá jako u předchozího datasetu. Uživateli ale nic nebrání ve vyzkoušení natrénování lineárního modelu a vykreslení separující křivky.

4.4.3 Lineárně částečně separovatelná data s validačním datasetem

Poslední dataset (viz obr. 4.5 vlevo) se oproti dvěma předchozím liší zejména dvěma aspekty. Tím prvním je, že sice neexistuje přímka, která by obě skupiny bodů dokázala perfektně rozdělit, ale dá se najít taková, která dataset rozděluje s velmi vysokou úspěšností. Není tedy problém na tento dataset využít lineární SVM. Jiný kernel by pravděpodobně úspěšnost modelu dokázal ještě zvýšit, ale ne tak výrazně. Druhým aspektem je, že testování modelu neprobíhá na stejných natrénovaných datech, ale na zvláštním validačním datasetu (viz obr. 4.5 vpravo). To je v praxi také preferovaný způsob. Pokud k dispozici takový dataset není, tak se data buď testují na trénovacím setu a nebo se trénovací dataset v určitém poměru rozdělí.

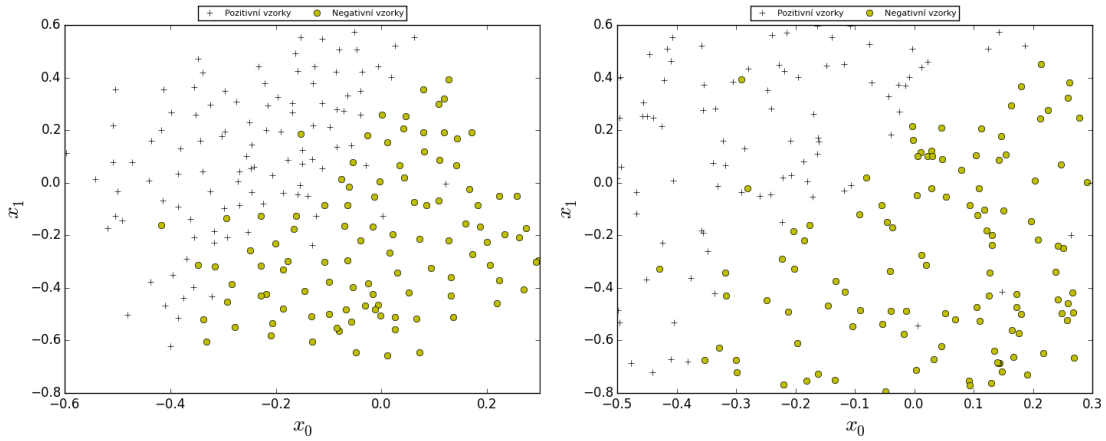


Obrázek 4.4: Nelineárně separovatelný dataset nevhodný pro zpracování lineárním SVM.

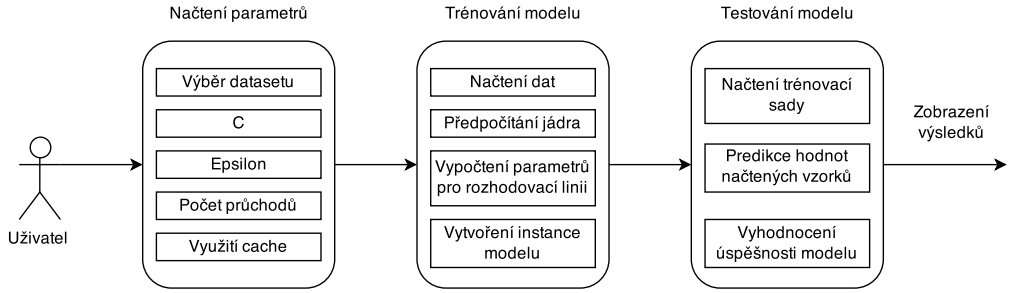
4.5 Implementace SVM

Pro lepší pochopení algoritmu je uvedeno grafické schéma (viz obrázek 4.6) průběhu celého algoritmu. Celý průběh je rozdělen do tří hlavních částí. Tou první je načtení parametrů z formuláře, které zadá uživatel. V druhé části přichází na řadu trénování modelu na základě připravených parametrů. Po úspěšném vytvoření modelu se ve třetí části průběhu otestuje jeho kvalita. Konečné výsledky jsou uživateli zobrazeny ve formě vykreslení rozhodující linie a tabulky správně a špatně predikovaných vzorků (viz 4.3).

Vedle grafického schématu je navíc uveden diagram tříd, který popisuje strukturu tříd v aplikaci. Jak je vidět na obrázku 4.7, tak je hlavní třídou `SvmTrain`. Ta zajišťuje načtení dat ze souboru a vytvoření modelu. Data se načítají pomocí třídy `LoadDataFromTxt` a samotný SVM model je reprezentován třídou `SvmModel`. Tato třída využívá libovolnou třídu, která dědí od abstraktní třídy `KernelFunction`. Ta zajišťuje způsob výpočtu kernel matice, kterou algoritmus využívá. Protože se v této práci využívá pouze lineární SVM, tak byla pro tento účel vytvořena třída `LinearKernelFunction`, která nemá oproti abstraktní třídě žádný parametr navíc a pouze implementuje základní metodu `compute_kernel()`. Na diagramu je znázorněna možnost případné rozšířitelnosti o další kernely. Poslední třídou, která v diagramu disponuje je `SvmPredict`. Ta využívá instanci třídy `SvmModel` k otes-



Obrázek 4.5: Lineárně částečně separovatelný dataset (vlevo) a validační dataset (vpravo), který se využívá k testování modelu.



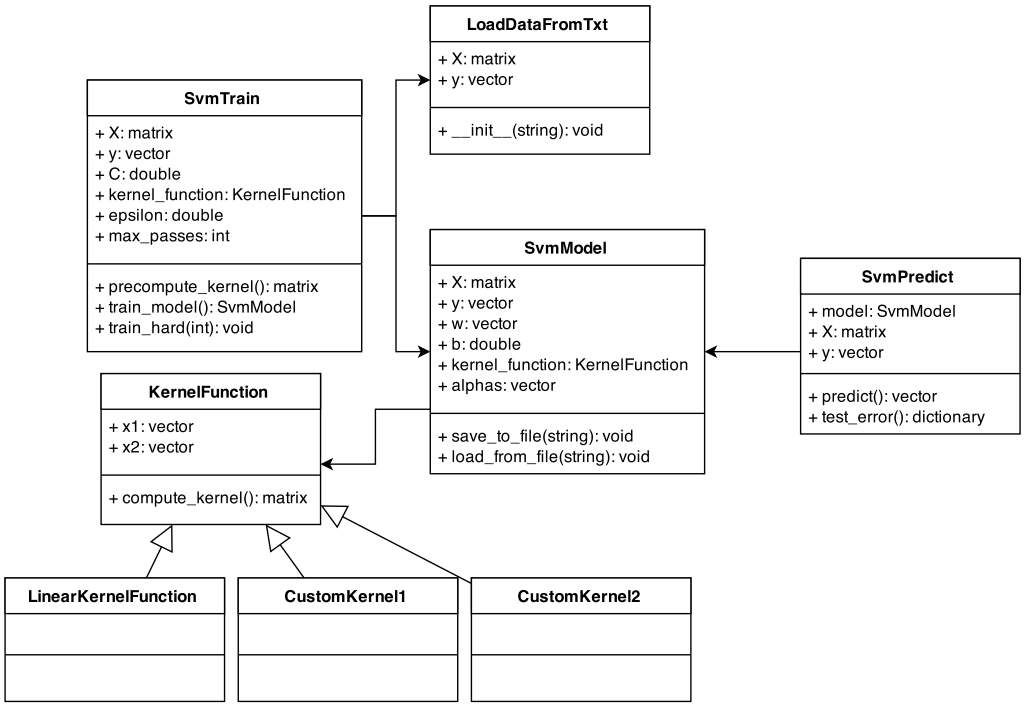
Obrázek 4.6: Grafický popis průběhu algoritmu SVM.

tování úspěšnosti modelu.

4.5.1 Formát dat

Datasety jsou uloženy v souborech ve formátu `.txt`. Každý řádek odpovídá jednomu vzorku. Všechny atributy jsou odděleny čárkou. Jako desetinná čárka se využívá tečka. Protože se pracuje pouze s datasety, které se dají zobrazit ve dvojrozměrném prostoru, tak obsahují pouze dva atributy. Třetím atributem je vždy cílová proměnná, která nabývá pouze hodnot 0 nebo 1.

Pro načtení dat se využívá třída `LoadDataFromTxt`. Při inicializaci se jí poskytne parametr s názvem souboru, který má načíst. Ze souboru se pak následně vytvoří dvě proměnné – `X` a `y`. Matice reprezentující atributy jednotlivých vzorků, resp. cílová proměnná určující třídu, do které vzorek spadá. Data se ukládají jako datový



Obrázek 4.7: Diagram tříd mé implementace algoritmu SVM.

typ `numpy.ndarray`, který dokáže reprezentovat n-rozměrné pole.

4.5.2 Nastavení modelu a popis vstupních parametrů algoritmu

Nastavování parametrů modelu probíhá ve formuláři popisovaném v kapitole 4.3. Kromě výběru datasetu a možné volby cachování je k natrénování modelu potřeba nastavit tři základní parametry.

Hlavním parametrem algoritmu je C. Ten dokáže model nastavit tak, aby více či méně reflektoval postavení outlierů v datasetu. Při použití nízkých hodnot v řádu jednotek algoritmus outliery ignoruje. Čím vyšší je pak jeho hodnota, tím více se snaží nalézt přímku, která by dokázala dataset rozdělit co nejlépe. S nižší hodnotou tedy algoritmus využívá tzv. *soft margin* (viz kapitolu 3.5). Pokud je ale hodnota příliš nízká (řádově setiny až tisíciny jednotek), tak začíná být algoritmus nestabilní a mnohdy je výsledná přímka nesmyslná. Při vyšších hodnotách (v řádek stovek až tisíců) naopak už nejsou rozdíly tak znatelné a záleží zde především na počáteční inicializaci algoritmu (viz kapitolu 4.5.3). Uživatel má v programu možnost vyzkoušet C v rozsahu 10^{-3} až 10^{4}.

Druhým zadávaným parametrem je *epsilon*, který je na diagramu tříd i v sa-

motném kódu nazván *float_tolerance*. Uživatel ho může zadat v rozsahu 10^{-3} až 10^{-2}. Parametr se využívá pro porovnávání α parametrů v algoritmu kolem nuly a C. Parametr ovlivňuje především přesnost zasazení přímky (kvůli porovnávání). Čím je menší, tím je výsledek přesnější. Druhou stránkou věci je ale větší časová náročnost, kdy musí algoritmus podniknout víc kroků k optimalizaci při zadané toleranci.

Posledním parametrem je *maximální počet průchodů*, v kódu nazvaný jako *max_passes*. Uživatel má možnost zadat hodnotu z rozsahu 1–50. Je to číslo, které ovlivňuje potřebný počet průchodů algoritmu, než bude moct skončit a vyhodnotit výsledky. Zde platí, že při větším počtu průchodů má algoritmus větší pravděpodobnost na lepší výsledek, ale je výpočetně náročnější.

4.5.3 Trénování modelu s využitím algoritmu SMO

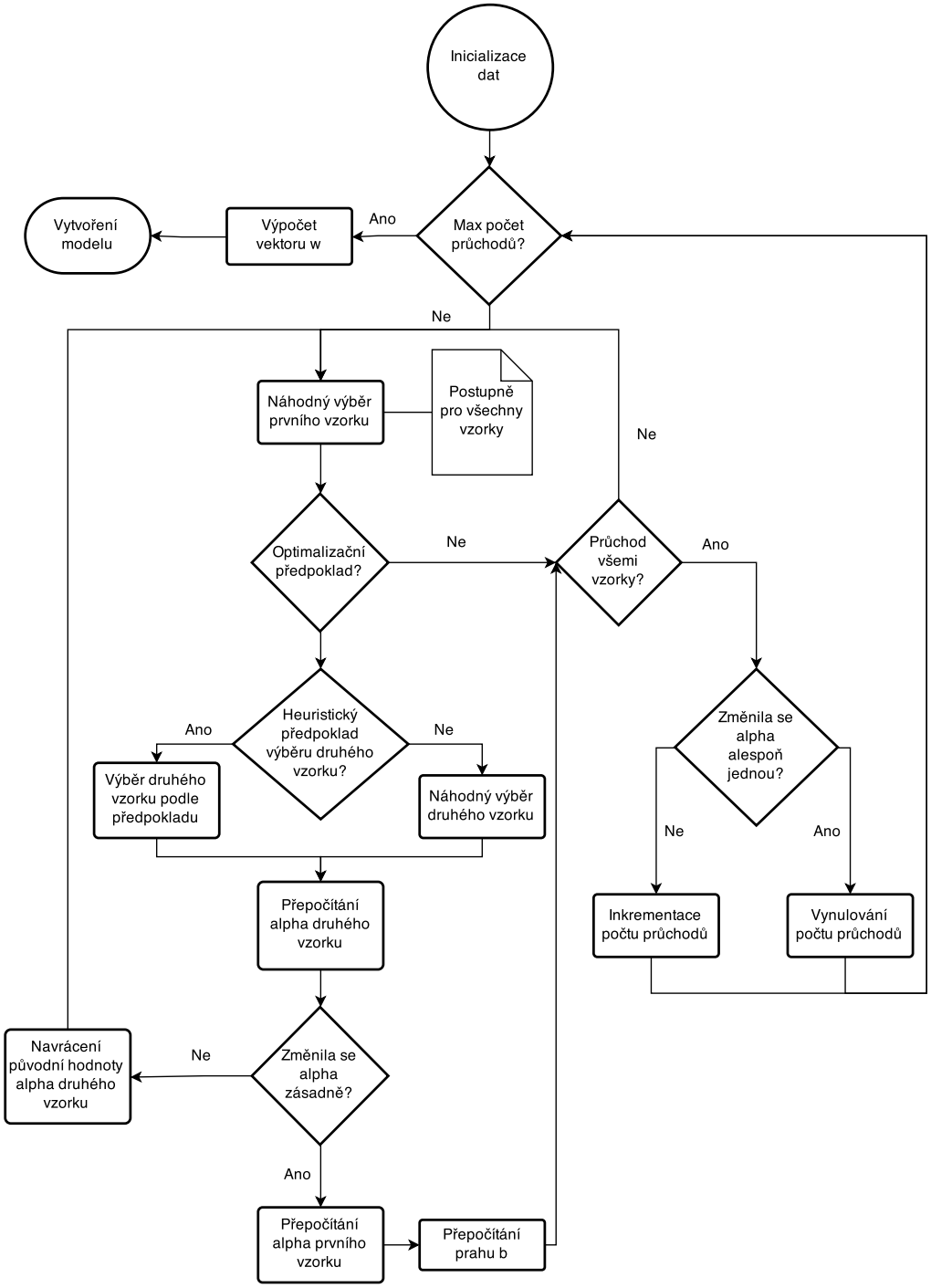
Když s SVM poprvé přišel Vapnik [6], využíval k výpočtům algoritmus PCG. V roce 1999 přišel s novým algoritmem John Platt [17]. Nazval ho SMO (angl. *Sequential minimal optimization*) a nahradil jím využití PCG. Také bylo rozhodnuto zvolení této implementace zejména díky zkušenosti z kurzu strojového učení [2]. SMO je oproti PCG výrazně rychlejší díky tomu, že vždy vzájemně optimalizuje pouze dvojice vzorků.

Trénování modelu zajišťuje třída `SvmTrain`. Průběh celého algoritmu je vidět na vývojovém diagramu na obrázku 4.8. Hlavní vnější cyklus ovlivňuje proměnná *maximální počet průchodů*, kterou zadává uživatel. Dokud je počet průchodů menší, tak se následující optimalizace opakuje.

Vnitřní cyklus běží přes všechny vzorky v datasetu v náhodném pořadí. SMO vždy optimalizuje dvojici vzorků. Konkrétně dochází k optimalizaci podpůrných vektorů – parametrů α (viz kapitolu 3.5). Pro každý vzorek se udržuje tzv. chybová cache v proměnné `E`. Je to hodnota, která je rozdílem hypotézy (viz vzorec 11) a hodnoty cílového atributu `y`. Po uložení nové hodnoty do chybové cache se kontroluje, jestli je aktuální prvek vhodný k optimalizaci nebo ne. Aby byl vybraný vzorek vhodný, tak musí splňovat následující podmínku:

```
(y[i] * E[i] <- float_tolerance and alpha[i] < C) or
(y[i] * E[i] > float_tolerance and alpha[i] > 0)
```

Situaci, kdy tato podmínka není splněna je popsána až na konci průběhu. Když je splněna, tak dochází k výběru druhého vzorku pro optimalizaci. Je zde využit



Obrázek 4.8: Vývojový diagram trénování SVM modelu s využitím algoritmu SMO.

heuristický předpoklad, který ve svém algoritmu popisuje John Platt [17]. Ve verzi SVM programované v rámci kurzu *Machine Learning*, je druhý vzorek automaticky

vybírán náhodně. V 15 % případů se v tomto případě stalo, že se hodnoty inicializovali tak, že prvky v chybové cache začaly divergovat. Dopadlo to vždy tak, že se program zastavil z důvodu nedostatku paměti. Zpočátku to bylo řešeno dalším parametrem ve formuláři pro nastavení modelu. Ve chvíli, kdy hodnota v cache přesáhla zadanou hodnotu, se celý algoritmus zastavil a začal počítat znovu.

Po aplikaci výběru podle Johna Platta už se to nikdy nestalo a v každém případě algoritmus skončil bez nutnosti restartu. Předpoklad spočívá v nalezení všech parametrů α, které jsou v intervalu $0 < \alpha < C$. Pokud takové parametry existují, tak se zvolí ten, který má největší rozdíl chyby od chyby prvně vybraného vzorku. Pokud žádný parametr této podmínce nevyhovuje, tak se zvolí jakýkoliv náhodný. U obou případů je nutné, aby oba vzorky byly různé.

Po výběru druhého vzorku se uloží jeho chyba do cache stejně jako u prvního vzorku. V dalším kroku se vypočítá nová hodnota α druhého vzorku a následně se porovnává se starou hodnotou. Pokud se hodnota téměř nezměnila (rozdíl nové a staré hodnoty je menší než `float_tolerance`), tak se nová hodnota zahodí, zachová se stará a algoritmus pokračuje v průchodu opětovným výběrem prvního vzorku. Jestliže se tato hodnota liší výrazněji, tak se přepočítá i α prvního vzorku. Navíc se přepočítá výsledný práh `b`, který je zpočátku inicializován na nulu. Nakonec inkrementuje počet změn parametrů α, ke kterému v této fázi došlo.

Na tento krok už navazuje i situace, kdy zpočátku nebyla splněna optimalizační podmínka. Pokud se ještě neprošly všechny vzorky, tak se pokračuje dalším výběrem prvního vzorku. V druhém případě algoritmus kontroluje, jestli se parametry α alespoň jednou změnily. Pokud ne, tak se inkrementuje celkový počet průchodů nejvnějšího cyklu a v opačném případě se tento počet vynuluje. Algoritmus tedy končí, pokud se již dvojice α nemění, neboli nejdou vzájemně optimalizovat. Patrné je, že každý běh algoritmu vytvoří trochu jiný model i se stejnými parametry. Občas se díky náhodné inicializace stane, že model vůbec neodpovídá datasetu ani parametrům. V tomto případě je potřeba model natrénovat znovu a vybrat si takový, který data nejlépe rozděluje.

Popsaný průběh zajišťuje metoda `train_model()`. V diagramu tříd na obrázku 4.7 je ve třídě ještě metoda `train_hard()`, která se v aplikaci sama nevyužívá. Ta byla vytvořena jen pro testovací účely. Přijímá pouze jeden parametr, kterým je celé číslo. To nastavuje kolikrát v řadě se model trénuje za sebou. Tímto testováním bylo zjištěno, že před implementací heuristiky výběru druhého vzorku algoritmus občas divergoval. Byly počítány časy jednotlivých průběhů a také počty těch úspěšných

a neúspěšných. Z toho bylo zjištěno výše zmiňované procento průběhů, které se nedokončilo.

4.5.4 Testování modelu

O testování modelu se stará třída `SvmPredict`. Výsledek tohoto testování již byl popsán v ukázce aplikace na obrázku 4.2 a v tabulce 4.1. O první se stará metoda `predict()`, která závisí na kernelu, na kterém je model natrénovaný. Nezávisle na kernelu je výsledkem pole s hodnotami jedna nebo nula. Jedna znamená, že je daný vzorek predikován jako pozitivní a nula jako negativní. V případě lineárního kernelu se k predikci využívá následující výpočet, který vychází z rovnice rozhodovací linie (viz rovnice 13).

$$p = X \ * w \ + b, \tag{14}$$

kde p je výsledné pole. Jeho hodnoty zatím nejsou 0 nebo 1. Nakonec je ještě potřeba nahradit všechny hodnoty větší nebo rovné nule za jedničku a záporné na nulu.

Z hodnot tohoto pole vychází výpočet úspěšnosti a kvality modelu. Počítá ho metoda `test_error()`. Jejím výsledkem je 5 čísel. Jak je vidět na tabulce 4.1, tak první čtveřice čísel říká, jestli byly pozitivní i negativní vzorky predikovány pravdivě či nepravdivě. Z toho vychází i úspěšnost celého modelu, která se počítá jednoduše jako počet všech správně predikovaných vzorků dělený celkovým počtem a v tabulce se zobrazuje v procentech.

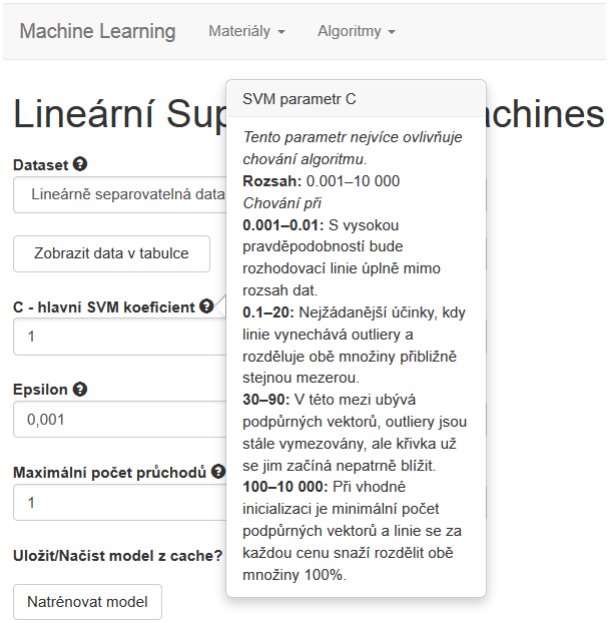
4.6 Implementace SVM v praxi

V tomto výukovém programu jsem si vyzkoušel vlastní implementaci algoritmu zejména i kvůli tomu, že jsem toto programování absolvoval i v kurzu Machine Learning. Pro praktické a nevýukové využití je doporučeno zvolit některou z již hotových knihoven. Nejslibněji vypadá knihovna *libsvm* [5], která je implementována a vyvíjena v řadě programovacích jazyků včetně jazyka Python, který byl zvolen i v této práci. Rozdílem oproti vlastní implementaci je zřejmě celková efektivita výpočtu a možnost využití dalších implementovaných kernelů, díky kterým lze vyhledávat i složitější nelineární rozhodovací linie.

4.7 Ukázka výukových prvků aplikace

K tomu, aby uživatel programu zjistil, jak ho má používat a aby pochopil, jak ho používat správně, slouží několik jednoduchých pomocníků. V sekci materiály má k dispozici seznam používaných pojmů, které mu pomohou zorientovat se v problematice. S nimi pak popis celé aplikace s představením všech parametrů a náhledů na připravené datasety.

K těmto materiálům má ve formuláři k dispozici tzv. vyskakovací okna (angl. *popovers*). Tyto okna obsahují nápovědy k prvkům, u kterých jsou umístěny. Jsou symbolizovány malou ikonou otazníku. Po kliknutí se zobrazí těsně vedle otazníku, aby uživatel neztratil přehled o tom, k čemu nápověda patří. V nápovědě jsou vždy stručně popsány vysvětlované pojmy a pokud se např. jednalo o vstupní parametry algoritmu, tak je uveden i jejich rozsah. Pro snadné pochopení je také popsáno, jak se chování algoritmu při různých volbách rozsahů liší. Vidět je to na obrázku 4.9.



Obrázek 4.9: Screenshot z aplikace, kde je vyskakovací nápověda k parametru C.

5 Téma zpracované ve formátu MOOC

Dlouho jsem přemýšlel, které téma v tomto formátu sám zpracovat. Logickou volbou by se zdálo zpracovat support vector machines, když jsem pro něj vytvořil výukový program. Program je zařazen do e-learningového kurzu předmětu Datamining, kde jej mají studenti k dispozici. Testovací kurz typu MOOC, který jsem vytvářel, má ambice být k dispozici širší cílové skupině studentů, než je skupina studentů povinně volitelného předmětu Datamining. Spolu s vedoucí práce jsme hledali vhodné téma pro větší cílovou skupinu studentů. Zároveň téma vhodné pro porovnání se slavným kurzem Machine Learning. Lineární regrese byla součástí absolvovaného kurzu na portálu Coursera a pro naše studenty mnoha oborů je linerání regrese potřebnou znalostí. Také je to typická státnicová otázka. To bylo dost argumentů pro výběr obsahu našeho pokusu o kurz typu MOOC. V kurzu jde především o testování formátu kurzu MOOC jako celku a hledání formátu streamovaných výkladových záznamů. Celé téma jsem zpracoval do pěti krátkých instruktážních videí. V každém se nahrává záběr na přednášejícího, kterým jsem v tomto případě sám. Střídají se v nich prostřihy z připravených prezentací a videa z vizualizéru. Po shlédnutí videí je pro studenta připraven jednoduchý test o pěti otázkách z probrané látky a možnost vyzkoušení naprogramování jednoduché úlohy.

5.1 Použité nástroje

K nahrávání přednášek jsem využíval obyčejný přenosný počítač s přední kamerou. K tomu vizualizér, který mi posloužil k záznamu poznámek na papír. Nejdůležitějším pomocníkem mi byl systém EduArt, který přednášky nahrál na internet, celkově je zpracoval a umožnil mi na ně jednoduše odkazovat.

5.1.1 EduArt

K tomuto systému jsem se dostal díky panu profesoru Janu Slovákovi z Ústavu matematiky a statistiky z Masarykovi Univerzity v Brně. Umožnil mi k němu bezplatně přístup pro využití v rámci mé diplomové práce. Program je jinak zpoplatněn podle ceníku uvedeného na vlastních internetových stránkách. Jedná se o program, který zpracovává nahrávané přednášky a nahrává je na webový server. Funguje v několika módech. Mód, který mi byl zpřístupněn využívá tzv. serverovou multilicenci. Pro mě to znamená, že se mi materiály z nahraných videí v počítači ukládají pouze v zakódované formě a terpve při nahrání na server se dekódují a jsou k dispozici pro zobrazení dalšími uživateli.

O každém videu je potřeba nastavit základní informace jako je název nebo téma přednášky. Videa se nahrávají na server `www.polymedia.cz`, odkud jsou k dispozici komukoliv, kdo zná jejich odkaz. Ten je tvořen z názvu zadané přednášky. Pro nahrávání je k dispozici klient, který vše obsluhuje. Pouhým kliknutím začíná nahrávání obsahu a po dokončení je možné přednášku rovnou nahrát na web. K systému také patří webové rozhraní, v kterém je možné nahrané přednášky upravovat.

Poslední součástí celého systému je konečný přehrávač nahraných přednášek. Je rozdělen do dvou hlavních částí, kde v jedné z nich je záběr na nahrávanou osobu a v druhé ostatní materiály ve formě prezentací, záběrů z vizualizéru apod. Program dokáže podle nastavení přednášky rozdělovat do jednotlivých kapitol. Přednáškou je tedy možné proskakovat posuny po jednotlivých kapitolách bez potřeby prohlížení celého obsahovat.

5.1.2 Vizualizér

Občas se při vytváření přednášek vyskytla potřeba pro nahrání ručně psaného výkladu. Např. kvůli složitějším vzorcům a případným malůvkám, které dokázaly lépe vysvětlit problém než pouhý text na prezentaci.

K tomu jsem si vypůjčil nahrávací vizualizér značky Elmo. Jedná se o zařízení, které disponuje kamerou na výsuvném polohovatelném rameni s třemi klouby. Z nejvyšší pozice nastavení ramena dokáže nahrávat poznámky z papíru přibližně formátu A4. K vizualizéru patří i klientská aplikace, která umožňuje i další ovládání přes počítač. Jejím hlavním přínosem je možnost zobrazení záběrů z vizualizéru

a okamžité zobrazení v okně, které se dá pohodlně přesunout na externí obrazovku připojenou k počítači.

5.1.3 ALS

ALS (zkratka pro *Advanced Learning Space*) je e-learningový portál provozovaný na Technické univerzitě v Liberci. Slouží zejména k podpoře studia. Student v něm má k dispozici přístup do kurzů (předmětů), kterých se účastní. V nich se nachází podpůrné materiály v podobě prezentací, záznamů z přednášek nebo odkazů na externí zdroje. Portál také umožňuje vytváření jednoduchých testů, v kterých se náhodně generují otázky i odpovědi.

5.2 Popis zpracovaného tématu v rámci portálu ALS

Celý kurz je zasazen do e-learningového portálu ALS (viz předchozí kapitolu 5.1.3). V úvodní části je připravené krátké povídání, které shrnuje a uvádí celý kurz. Je v ní zmínka o tom, že je součástí této diplomové práce, a že je to experiment ve vytvoření kurzu, který respektuje popisovaný MOOC formát. Přispívá tímto způsobem k analýze tohoto formátu.

Tento kurz je zaměřen zejména na vysokoškolské studenty. Další kurzy v podobném duchu by měly sloužit jako doplněk a podpora k samotnému studiu. Např. pro zlepšení a objasnění jeho problémových partií. Hlavní myšlenkou je ale poskytování zajímavých témat napříč různými obory i pro širší veřejnost.

Po tomto úvodu následují čtyři sekce, do kterých je rozdělené povídání o lineární regresi. Postupně se v něm mluví o reprezentaci modelu a tvorbě *hodnotové funkce*. Její využití v *metodě postupného klesání* pro získání optimálních parametrů regresní funkce a následný návod pro naprogramování celého algoritmu v libovolném programovacím jazyce.

Každá sekce obsahuje nahrané video (odkaz na portál *polymedia*, na který videa nahrává systém EduArt 5.1.1), přiloženou prezentaci, kontrolní otázku z probrané látky a odkaz na externí materiály.

Po shlédnutí všech materiálů má student možnost ověření svých nabytých znalostí. První z možností je test o pěti otázkách, které se generují z připraveného balíku otázek. Student není omezen počtem pokusů a test může opakovat kolikrát

chce. Pro zvídavější je zde druhá možnost, v kterém je zadání pro jednoduché programovací cvičení. Doporučené je zpracování v jazyce Python. Pro tuto možnost je popsán i způsob instalace potřebného softwaru a jsou uvedeny odkazy na návody pro začátečníky v tomto jazyce. Není to ale omezení a uživatel si tak úlohu může naprogramovat v libovolném jazyce dle své volby.

Závěr

Díky této práci jsem si prohloubil znalosti z oblasti strojového učení, které díky tomu budu moct aplikovat v dataminingových úlohách. Seznámil jsem se s problematikou MOOC kurzů a sám jsem si jeden kurz vyzkoušel. Nadchl mě způsob, kterým dokáže být látka vyučována. Velkým přínosem byla možnost okamžité aplikace nabytých znalostí v programovacích cvičení. Do budoucna jsem velmi namotivován pro vyzkoušení některého z dalších kurzů.

Po získání potřebných znalostí jsem vytvořil výukový program, který studentům dovolí pečlivě prozkoumat chování algoritmu Support Vector Machines. Velkým přínosem pro mě bylo hlubší proniknutí do programovacího jazyka Python, zejména do webového frameworku Django, v kterém jsem aplikaci programoval. Takto připravený program s demonstrací algoritmu by mohl být výborným doplňkem MOOC kurzů. Zejména pro účastníky, kteří se nezajímají o konkrétní implementaci, ale o použití algoritmu. V praxi by bylo vhodnější na reálných datech využít již existující a optimalizované knihovny. Pro použité nasazení ale má implementace bez problému postačuje.

Jako téma pro zpracování ve formátu MOOC jsem zvolil lineární regresi. Stalo se tak především proto, aby případné použití našeho MOOC pokusu, bylo užitečné pro větší cílovou skupinu studentů. SVM je program pro předmět Datamining oboru Informační technologie, který byl zařazen do kurzu na portále ALS a je studenty testován. Téma linární regrese je společné mnoha oborům mnoha fakult TUL. Patří k základním znalostem požadovaným u státnic, proto se zdálo být užitečné vytvoření právě tohoto tématu jako MOOC kurz. Téma jsem zasadil do e-learningového portálu ALS, který dokázal pokrýt poměrně značnou část potřeb pro vytvoření kurzu v požadovaném formátu.

Pro další tvorbu kurzů doporučuji nahrávání krátkých videí, které nebudou studenta nudit. Pro podporu jeho pozornosti bych na portál doplnil možnost zastavení videa kvůli jednoduché kontrolní otázce, na kterou by musel odpovědět. Příjemným

přínosem může být předpřipravení diskuzních kruhů, v kterých by studenti sdíleli své poznatky. Hlavní vlastností všech kurzů by měla být jejich co největší samoobslužnost a možnost zpětné vazby pro další vylepšování.

Literatura

[1] Academic Earth. *750+ Free Online Courses From Top Universities — AE.org* [online]. 2015. [cit. 7. 4. 2015]. Dostupné z: <http://academicearth.org/>.

[2] Andrew Ng. *Machine Learning – Stanford University — Coursera* [online]. 2014. [cit. 8. 4. 2015]. Dostupné z: <https://www.coursera.org/course/ml>.

[3] Berka, P. *Dobývání znalostí z databází*. Academia, 2003. Dostupné z: <http://books.google.cz/books?id=tGvFAAAACAAJ>. ISBN 9788020010629.

[4] Canvas Network. *Canvas Network — Free online courses — MOOCs* [online]. 2015. [cit. 7. 4. 2015]. Dostupné z: <https://www.canvas.net/>.

[5] Chang, C.-C. – Lin, C.-J. LIBSVM: A library for support vector machines. *ACM Transactions on Intelligent Systems and Technology*. 2011, 2, s. 27:1–27:27. Software available at http://www.csie.ntu.edu.tw/~cjlin/libsvm.

[6] Cortes, C. – Vapnik, V. Support-Vector Networks. *Machine Learning*. 1995, 20, 3, s. 273–297. ISSN 0885-6125. Dostupné z: <http://dx.doi.org/10.1023/A:1022627411411>.

[7] Coursera Inc. *Coursera – Free Online Courses From Top Universities* [online]. 2015. [cit. 7. 4. 2015]. Dostupné z: <https://www.coursera.org/>.

[8] Django Software Foundation. *The Web framework for perfectionists with deadlines — Django* [online]. 2015. [cit. 16. 4. 2015]. Dostupné z: <https://www.djangoproject.com/>.

[9] edX Inc. *edX — Free online courses from the world's best universities* [online]. 2015. [cit. 7. 4. 2015]. Dostupné z: <https://www.edx.org/>.

[10] FutureLearn. *FutureLearn – Free online courses* [online]. 2015. [cit. 7. 4. 2015]. Dostupné z: <https://www.futurelearn.com/>.

[11] Hunter, J. D. Matplotlib: A 2D graphics environment. *Computing In Science & Engineering*. 2007, 9, 3, s. 90–95.

[12] Khan Academy. *Khan Academy* [online]. 2015. [cit. 7. 4. 2015]. Dostupné z: <https://www.khanacademy.org/>.

[13] Ng, A. Support Vector Machines. *CS229 Machine Learning Lecture Notes*. 2014. Dostupné z: <http://cs229.stanford.edu/notes/cs229-notes3.pdf>.

[14] Numpy developers. *NumPy* [online]. 2013. [cit. 16. 4. 2015]. Dostupné z: <http://www.numpy.org/>.

[15] Open Universities Australia Pty Ltd. *OPEN2STUDY - FREE Online Study For Everyone* [online]. 2015. [cit. 8. 4. 2015]. Dostupné z: <https://www.open2study.com/>.

[16] Pappano, L. The Year of the MOOC. *The New York Times*. 2012, 2, 12.

[17] Platt, J. – others. Fast training of support vector machines using sequential minimal optimization. *Advances in kernel methods—support vector learning*. 1999, 3. Dostupné z: <http://research.microsoft.com/pubs/68391/smo-book.pdf>.

[18] Rud, O. *Data mining: praktický průvodce dolováním dat pro efektivní prodej, cílený marketing a podporu zákazníků (CRM)*. Rychle a jistě. Computer Press, 2001. Dostupné z: <https://books.google.cz/books?id=cimeAAAACAAJ>. ISBN 9788072265770.

[19] Udacity, Inc. *Advance Your Career Through Project-Based Online Classes - Udacity* [online]. 2011-2015. [cit. 7. 4. 2015]. Dostupné z: <https://www.udacity.com/>.

A Obsah přiloženého CD

Přiložené CD obsahuje zdrojové kódy k výsledné aplikaci a diplomovou práci v elektronické podobě.

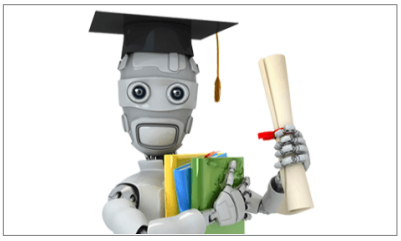
B Certifikát z kurzu Machine Learning

DECEMBER 17, 2014

Online Course
Statement of Accomplishment

PATRIK DRHLÍK

HAS SUCCESSFULLY COMPLETED A FREE ONLINE OFFERING OF THE FOLLOWING COURSE PROVIDED BY STANFORD UNIVERSITY THROUGH COURSERA INC.



Machine Learning

Congratulations! You have successfully completed the online Machine Learning course (ml-class.org). To successfully complete the course, students were required to watch lectures, review questions and complete programming assignments.

ASSOCIATE PROFESSOR ANDREW NG
COMPUTER SCIENCE DEPARTMENT
STANFORD UNIVERSITY

PLEASE NOTE: SOME ONLINE COURSES MAY DRAW ON MATERIAL FROM COURSES TAUGHT ON CAMPUS BUT THEY ARE NOT EQUIVALENT TO ON-CAMPUS COURSES. THIS STATEMENT DOES NOT AFFIRM THAT THIS PARTICIPANT WAS ENROLLED AS A STUDENT AT STANFORD UNIVERSITY IN ANY WAY. IT DOES NOT CONFER A STANFORD UNIVERSITY GRADE, COURSE CREDIT OR DEGREE, AND IT DOES NOT VERIFY THE IDENTITY OF THE PARTICIPANT.