ZÁPADOČESKÁ UNIVERZITA V PLZNI

FAKULTA EKONOMICKÁ

Diplomová práce

Hodnocení spotřebitelských úvěrů pomocí SVM techniky s využitím sw nástroje Mathematica

The evaluation of consumer loans using SVM technique using sw Mathematica

Bc. Jan Harmady

Plzeň 2015

zadání

Prohlašuji, že jsem diplomovou práci na téma

Hodnocení spotřebitelských úvěrů pomocí SVM techniky s využitím sw nástroje Mathematica

vypracoval samostatně pod odborným dohledem vedoucího diplomové práce
za použití pramenů uvedených v přiložené bibliografii.

V Plzni, dne 16.4.2015 ……

podpis autora

Poděkování

Na tomto místě bych rád poděkoval doc. RNDr. Ing. Ladislavu Lukášovi, CSc. za cenné rady a odborné vedení při zpracování této diplomové práce.

Obsah

Úvod

Přestože bankovní instituce v současné době začínají hledat nové zdroje svých příjmů, zůstává získávání finančních prostředků od veřejnosti jejich hlavním zdrojem příjmu. Svá finanční aktiva nakupuje banka v podobě cenných papírů, dluhopisů, vkladů klientů a úvěrů.

Problém, který nejen bankovní instituce musí řešit, je dlužníkova schopnost udělený úvěr včas a v plné výši splatit. Dlužníkova schopnost dostát svým závazkům se nazývá bonita klienta a banka ji vyhodnocuje v rámci hodnocení kreditního rizika.

Přestože by se banky měly snažit toto riziko minimalizovat tak, aby poskytovaly úvěry jen bonitním klientům, v rámci konkurenčního boje a vidiny vyšších zisků bankovní instituce toto riziko v minulosti podceňovaly. Toto podceňování vyústilo v 80. letech v akutní dluhové krize, což vedlo ke vzniku Basilejského výboru pro bankovní dohled. Toto fórum vydalo v roce 2006 důležitý dokument, který významně ovlivnil způsob měření rizik v bankovních institucích, jednalo se o Novou basilejskou kapitálovou dohodu (New Basel Capital Accord) označovanou jako Basel II.

Basel II přinesl flexibilitu v možnostech měření rizik, když umožnil rizika měřit pomocí interních hodnocení pro zjištění požadované minimální kapitálové úrovně. Čím spolehlivější nástroj pro měření rizik bankovní instituce vlastní, tím nižší je jí povolena úroveň kapitálu, čímž dochází ke snižování nákladů na kapitál.

Přístupy k měření kreditního rizika je možné rozdělit na dva typy. Prvním typem je expertní posuzování a druhým statistické modelování. Expertní posudky jsou silně vázané na implicitní znalosti finančních expertů. Zároveň je tento přístup časově náročný, jelikož expert je nucen posuzovat jednotlivě každou žádost, což je pracné.

Druhý přístup je schopen pracnost a časovou náročnost vyhodnocení žádosti o úvěr značně zredukovat. Zároveň si také zachovává objektivitu a na rozdíl od finančních expertů při vyšším zatížení nepodléhá únavě, která může vést k chybám v klasifikaci a pomalému vyhodnocování žádostí.

Tradiční statistické metody v oblasti klasifikace dat byly překonány, a to především nástupem umělých neuronových sítí a technikami na těchto sítích založených. I neuronové sítě ale mají nedostatky, které se snaží odstranit metoda Vladimira Vapnika Support Vector Machine.

Tato práce se zabývá právě metodou Support Vector Machine (zkráceně SVM), často překládanou jako metoda podpůrných vektorů, a možností jejího využití ke klasifikaci klientských žádostí o spotřebitelské úvěry.

První část práce se zabývá spotřebitelskými úvěry a důvody potřeby klasifikace žádostí podle klientovy schopnosti splácet, což souvisí s úvěrovým rizikem a bonitou klienta, ale také Basilejským výborem pro bankovní dohled, který, ačkoli neformální autorita, vydává nařízení o řízení a měření rizik pro bankovní instituce.

Následně jsou popsány metody strojového učení, které by mohly být alternativou podpůrným vektorům, ovšem největší pozornost je poté věnována právě metodě Support Vector Machine.

Zbytek práce se pak věnuje implementaci vybraných algoritmů metody Support Vector Machine. S tím souvisí návrh struktury vstupních dat pro vytvoření modelu a ověření jejich fungování na různých variantách vstupních dat.

Cíle práce

Cíle této diplomové práce byly vymezeny následovně:

1. *Uveďte do problematiky klasifikace spotřebitelských úvěrů*

Úvod do problematiky bankovních úvěrových produktů se zaměřením na spotřebitelské úvěry a nutnost jejich klasifikace podle bonity klientů je zpracován v první kapitole této diplomové práce. Zabývá se tedy především úvěrovým rizikem a potřebou jej v bankovních institucích efektivně měřit. Informace o tomto tématu byly čerpány z několika publikací, mezi které patří například Řízení obchodních bank od V. Kašparovské (2006), Bankovnictví pro bankéře a klienty od P. Dvořáka (2005), které byly zdrojem především pro kapitoly týkající se spotřebitelských úvěrů a jejich rizik. Dále byly použity například Bankovní regulace a dohled od L. Juroškové (2012) nebo internetové stránky Basilejského výboru pro bankovní dohled - Bank for international settlements sekce Basel committee on banking supervision (www.bis.org/bcbs). Tyto dva zdroje sloužily především jako podkladové materiály pro kapitoly o Basilejských dohodách. Jedná se tedy o rešeršní část týkající se spotřebitelských úvěrů.

2. *Charakterizuje vybrané metody strojového učení*

Druhá a třetí kapitola je zaměřena na strojové učení a jeho vybrané metody vhodné pro klasifikaci dat, a to především na techniku Support Vector Machine. V kapitole 2 je obsažen obecný úvod do strojového učení a popis některých z metod strojového učení. Jako zdroje informací zde posloužily například Umělá inteligence od V. Mařík a (1993), Úvod do umělých neuronových sítí od Miloše Křivana (2008) nebo Pattern recognition and machine learning od Christophera Bishopa (2009). V kapitole 3 je detailněji popsána metoda podpůrných vektorů. Jako zdroje informací byly v tomto případě použity Computational inteligence and it´s applications od H. K. Lama (2012), Pattern Classification (Duda, Hart a Stork, 2001), ale také řada zahraničních odborných článků a tezí, které se metodou SVM a především její aplikací nebo způsoby implementace zabývají.

3. *Implementujte vybrané algoritmy SVM v SW Mathematica*

Cíl implementace vybraných algoritmů metody Support Vector Machine se pokouší naplnit třetí část této diplomové práce. Nejprve je řešen návrh struktury a možná

zakódování atributů popisujících žadatele o spotřebitelský úvěr tak, aby byla metodou zpracovatelná. Dále jsou v této části práce stručně popsány softwarové nástroje a programovací jazyky, ve kterých implementace probíhala. Těmito jazyky byla zvolena Java pro zpracování vstupních dat a vytvoření trénovacího modelu a Wolfram Mathematica, ve kterém dochází k tvorbě příslušných rovnic a vizualizaci jejich výstupů.

4. *Proveďte výpočty pomocí implementovaných algoritmů nad vstupními daty*

Výpočty pomocí implementovaných programů jsou provedeny v kapitole 8. Zde je řešeno několik variant vstupních dat, na kterých byly ověřeny některé teoretické poznatky získané v rešeršní části této diplomové práce. K provedení výpočtů je kromě zdrojových kódů vytvořených v rámci této práce využita externí knihovna LIBSVM v jazyce Java, která slouží k řešení výpočtů kvadratického programování. Výpočty jsou provedeny nad třemi sadami vstupních dat, kdy na první sadu je aplikováno lineární jádro a jedná se o dokonale oddělitelná data, ve druhém případě je opět využito lineární jádro, ovšem data již nelze jednoznačně oddělit, a je tak demonstrována klasifikace s chybou a ve třetí sadě je využita Gaussova jádrová transformace na lineárně neoddělitelná data.

5. *Zhodnoťte výsledky provedených výpočtů a formulujte závěr*

Devátá kapitola se pokusí shrnout poznatky ověřené provedením výpočtů pomocí implementovaných algoritmů strojového učení. Také je nastíněno několik variant dalšího možného pokračování v této práci. Závěr obsahuje zhodnocení dosažených cílů vytyčených v rámci této diplomové práce.

1 Úvěrové produkty

Úvěrové produkty tvoří značnou část aktiv bankovních institucí, jsou tedy významným zdrojem příjmů. Poskytování úvěrů sebou ale přináší řadu rizik, která musí každá bankovní instituce před poskytnutím úvěru vyhodnotit. Jinak při nesprávně nastaveném a příliš rizikovém portfoliu aktiv, které sice může přinášet vyšší možný zisk, hrozí bankovní instituci propad do insolvence.

Podle Dvořáka (2005) se členění bankovních úvěrových produktů liší jak mezi jednotlivými státy, tak ale i mezi bankami v rámci jednoho státu, proto není vhodné zavádět podrobnou systematizaci a členění bankovních úvěrových produktů. Je však možné definovat parametry popisující úvěr:

- Příjemce úvěru, kterým může být stát, podnikatelské osoby nebo fyzické osoby.
- Forma poskytnutí úvěru, kde jsou rozlišovány dva základní typy. Peněžní úvěr a závazkové úvěry.
- Doba splatnosti úvěru. Úvěry v tomto ohledu lze rozdělit na krátkodobé, střednědobé a dlouhodobé.
- Účel použití. U účelových úvěrů lze poskytnuté finanční prostředky použít pouze na účel vymezený v rámci smlouvy o poskytnutí úvěru.
- Měna, ve které je úvěr poskytnut. Úvěry poskytované v zahraniční měně se nazývají devizové.
- Způsob zajištění. Zajištění úvěru může probíhat různými způsoby, například zajištění zástavou nemovitosti, zástavou movité věci nebo je také možné zajištění ručitelem.

Úvěrové produkty jsou jednou z nejžádanějších forem financování jak u podnikatelů, tak i u spotřebitelů. Mezi významné bankovní úvěrové produkty patří hypoteční úvěry, investiční úvěry, ale také spotřebitelské úvěry.

Právě na problematiku spotřebitelských úvěrů, respektive na ohodnocování schopnosti žadatele splácet tento typ úvěrového produktu je tato práce zaměřena.

1.1 Spotřebitelský úvěr

Spotřebitelské úvěry jsou poskytovány fyzickým osobám nejednajícím v rámci své podnikatelské činnosti. Od roku 2011 vstoupil v platnost nový zákon o spotřebitelském úvěru, který zvyšuje povinnosti poskytovatelů spotřebitelských úvěrů a zároveň posiluje postavení spotřebitele. Jedná se o zákon č.145/2010 Sb..

Podle tohoto zákona je spotřebitelský úvěr definován jako odložená platba, půjčka, úvěr nebo jiná finanční služba poskytovaná nebo přislíbená spotřebiteli věřitelem, nebo zprostředkovatelem.

Zákon č.145/2010 Sb. vymezuje důležité pojmy úvěrového vztahu. Spotřebitele zákon definuje jako fyzickou osobu, která nejedná v rámci své podnikatelské činnosti. Věřitelem je osoba, které nabízí nebo poskytuje spotřebitelský úvěr v rámci své podnikatelské činnosti. V rámci spotřebně úvěrového vztahu může vystupovat také zprostředkovatel, což je osoba, která není věřitelem, ale za odměnu nabízí spotřebiteli možnost uzavření smlouvy o spotřebitelském úvěru. Pro spotřebitele je důležitým údajem roční procentní sazba nákladů (RPSN), což jsou celkové náklady spotřebitelského úvěru vyjádřené jako roční procentní podíl z celkové výše úvěru. Tyto náklady jsou včetně úroků, provizí a daní s výjimkou nákladů na notáře.

Tento zákon byl vydán s cílem posílit postavení spotřebitele, který je v spotřebitelsko-úvěrovém vztahu slabší stranou. Toho se snaží dosáhnout zvýšenou informační povinností věřitele tak, aby spotřebitel obdržel všechny informace vždy a včas. Zákon ošetřuje například i fázi před uzavřením úvěru (reklamu). Spotřebitel musí obdržet informace o věřiteli, o úrokové sazbě, o počtu a výši splátek, o právu úvěr předčasně splatit nebo o právu odstoupit od smlouvy o spotřebitelském úvěru.

Kromě zákona o spotřebitelském úvěru, který platí pro bankovní i nebankovní věřitele nebo zprostředkovatele, je vytvářen na bankovní subjekty tlak ze strany Basilejského výboru pro bankovní dohled (dále jen Basilejský výbor), který vydávanými standardy usiluje o regularizaci norem a stabilitu bankovního sektoru, protože stabilní a rizikům odolný bankovní sektor pozitivně ovlivňuje ekonomický vývoj.

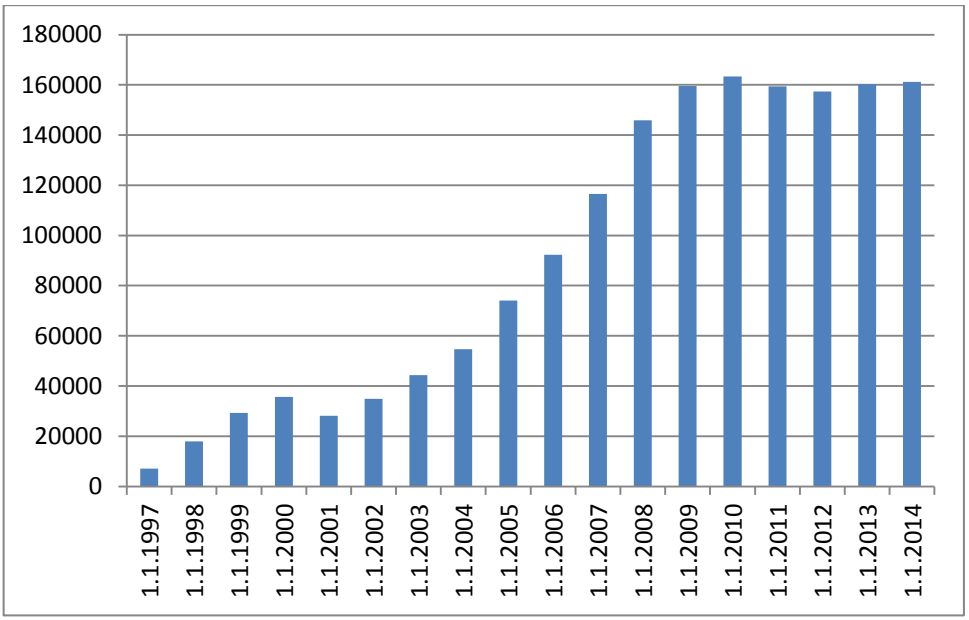
Poskytnutí úvěru sebou pro věřitele nese řadu rizik. Kašparovská (2006) tato rizika rozděluje do pěti základních kategorií:

1. Úvěrové riziko představuje situaci, kdy dlužník nebude schopen dostát svým závazkům. Toto riziko bude důkladněji popsáno v následující samostatné podkapitole.
2. Tržní riziko definuje Česká národní banka jako potenciální ztrátu způsobenou změnami cen, kurzů a sazeb na finančních trzích. Tržní riziko je tvořeno rizikem úrokovým, měnovým, akciovým, komoditním a dluhopisovým. Rizikem je tedy fluktuace úrokových měr, měnových kurzů a fluktuace, která vyplývá z vývoje cen akcií, dluhopisů a dalších komodit.
3. Riziko likvidity je riziko, že banka nebude schopná dostát svým závazkům v době jejich splatnosti nebo že banka nebude schopna financovat své aktiva. Toto riziko lze rozlišit na dva typy. Prvním je, že banka nebude schopna uzavřít pozici daného finančního nástroje z důvodu mělkosti trhu nebo rozvratu na něm, což se nazývá tržním likviditním rizikem. Druhý typ je likviditní riziko financování, tedy že v bance dojde k nesouladu splatnosti aktiv a pasiv.
4. Operační riziko. O operační riziko se jedná, pokud dojde ke ztrátě vlivem selhání vnitřních procesů banky, absencí těchto procesů, selháním lidského faktoru, vlivem vnějších událostí nebo porušením či nenaplněním právní normy.
5. Ostatní rizika. Do této skupiny Kašparovská (2006) zahrnuje ztrátu reputace, změnu podmínek regulátorem, nedostatečnou velikost kapitálu vzhledem k rizikům nebo riziko ztráty dobrého jména.

U bankovních úvěrů se s největší intenzitou posuzuje úvěrové riziko, které se banka snaží minimalizovat, protože, a to především u retailových bank, tento typ úvěru zaujímá významnou pozici mezi ostatními produkty, jelikož je významným zdrojem příjmů.

Komárková ve své studii (2012) rozšiřuje výše uvedenou kategorizaci rizik o riziko teritoriální a svrchované. Teritoriální riziko je riziko, které souvisí s politickými událostmi a administrativními opatřeními, ale také například s možností přírodní katastrofy. Globální finanční krize znovu odhalila svrchované riziko, které se v minulosti již příliš neuvažovalo. Toto riziko souvisí s neschopností vlády dostát svým závazkům, což vede ke státnímu bankrotu.

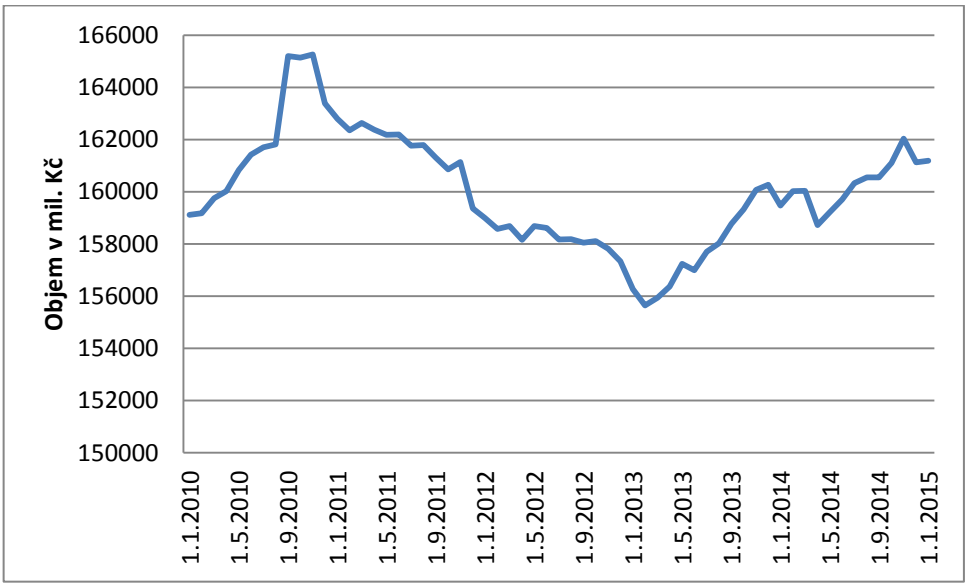
Graf 1: Vývoj spotřebitelských úvěrů v mil. Kč



Zdroj: ČNB, systém ARAD, 2015

Z grafu 1 sestaveného z dat systému časových řad ARAD České národní banky je zřejmé, že obliba spotřebitelských úvěrů od roku 1997 prudce rostla až do roku 2010, kdy svět postihla globální finanční krize. Po mírném poklesu objemu spotřebitelských úvěrů nastává opět období růstu jejich popularity, což je vidět na detailnějším grafu 2, kde od ledna 2013 začal objem spotřebitelských úvěrů opět stoupat.

Graf 2: Měsíční vývoj spotřebitelský úvěrů v mil. Kč



Zdroj: ČNB, systém ARAD, 2015

1.1 Úvěrové riziko

Úvěrové riziko, nebo také jinak kreditní riziko, je riziko vyplývající ze selhání klienta tím, že nedostojí svým závazkům podle podmínek smlouvy, na základě které se banka stala věřitelem klienta.

Českou národní bankou je toto riziko definováno jako „riziko, že protistrana nedostojí plně svému finančnímu závazku“.

Kreditní riziko je považováno za nejvýznamnější bankovní riziko, a i proto je mu věnována největší pozornost.

Podle tohoto rizika se banka rozhoduje, zda obchod realizuje, s jakou úrokovou sazbou nebo jaký zvolí způsob zajištění. Proto je potřeba toto riziko kvantifikovat. Snahou banky je úvěrové riziko minimalizovat. O to se snaží již na úrovni obchodů s klienty zjišťováním bonity klienta.

Bonita klienta je složena z právních, finančních a ekonomických charakteristik, které jsou důležitými indikátory jeho schopnosti splnit závazek vůči bance. Bonita klienta se zjišťuje pomocí ratingu, což je proces stanovení bonity a její vyjádření pomocí pevně stanovené stupnice. Stanovení bonity klienta může provádět sama banka, pak se jedná o interní rating, v případě, že jej provádí externí ratingová agentura, jej pak nazýváme externím ratingem. (Kašparovská, 2006)

Interní rating obvykle vychází z úvěrové analýzy. Ta využívá metodické postupy, k nimž patří finanční poměrová analýza nebo scoring (metoda zjednodušeného bodového hodnocení).

Pohledávky vzniklé z udělených spotřebitelských úvěrů lze rozdělit do dvou základních kategorií. Pohledávky se selháním jsou takové pohledávky, kdy dlužník nesplatí svůj závazek podle smlouvy o spotřebitelském úvěru, nebo když alespoň jedna ze splátek je více než 90 dnů po její splatnosti.

Při hodnocení bonity klienta mohou nastat dvě chyby. Chyba I. druhu je situace, kdy bonitní klient je vyhodnocen jako neschopen úvěr splatit, chyba II. druhu je chyba, kdy klient je vyhodnocen jako kvalitní, schopen dostát svým závazků, přestože to tak není. (Kašparovská, 2006)

U chyby I. druhu vznikají bance náklady ušlé příležitosti a náklady na analýzu klienta, chyba II. druhu představuje pro banku reálnou ztrátu v podobě nesplacení poskytnutého úvěru.

Banka se snaží minimalizovat pravděpodobnost chyby v hodnocení žádostí o úvěry, čehož může docílit zapojením kvalitních expertů s dostatečnými tacitními znalostmi, což vyžaduje vysoké náklady jak na čas, jelikož export posuzuje každý případ jednotlivě, ale také vysoké náklady finanční, protože experti s dostatkem znalostí a zkušeností schopných objektivně případ posoudit jsou náležitě finančně oceňováni.

Z těchto důvodů může být pro banku výhodné do hodnocení bonity klienta zapojit metody statistického modelování, které při správné konfiguraci zajistí rychlé a objektivní zhodnocení klientovy žádosti o spotřebitelský úvěr.

1.2 Basilejský výbor pro bankovní dohled

Basilejský výbor pro bankovní dohled (dále jen Basilejská komise) byl založen v roce 1974 centrálními bankami zemí G-10. Přestože Basilejská komise je relativně neformální fórum, členské autority se zavazují vydané standardy a doporučení implementovat. Od roku 2012 Basilejská komise s cílem posílit bankovní systém a zvýšit důvěru veřejnosti monitoruje zavádění vydaných doporučení. Požadavky Basilejská komise vydává pod názvem Dohoda o kapitálové přiměřenosti a většinou se zkráceně označují jako Basel. Tyto požadavky výrazným způsobem ovlivňují proces řízení rizik v bankách, jelikož těmto institucím ukládají povinnost udržovat minimální výši regulačního kapitálu. (BCBS, 2014)

Basilejský výbor pro bankovní dohled se také zabývá zvyšováním efektivity spolupráce a výměny informací v oblasti bankovního dohledu. Dohled v České republice vykonává Česká národní banka, která je členem Skupiny bankovních dohledů pro střední a východní Evropu. Tato skupina mimo výkon dohledu také pomáhá s přípravami zemím mimo Evropskou unii na vstup do Evropské unie. (ČNB, 2015)

1.2.1 Kapitálová přiměřenost

Po pádu Brettonwoodského systému v roce 1971 byly banky nuceny hledat nové zdroje příjmů, jelikož jejich klienti především kvůli vysoké inflaci přestávali spořit a více se zaměřovali na investování za pomoci investičních společností. Banky tak půjčovaly

více prostředků než v minulosti, čímž se snižovala úroveň jejich udržovaného množství kapitálu.

Především díky dluhové krizi z 80. let se pozornost bankovních regulátorů obrátila na řízení rizik bank. Guvernéři zemí G-10 pověřili předsedu Basilejského výboru vytvořením a konsolidací pravidel kapitálové přiměřenosti. Na prosazení těchto pravidel měly zájem především Velká Británie a USA, které se obávaly rychlého růstu a expanze japonských bank, které žádné kapitálové požadavky nedodržovaly, což bylo jejich konkurenční výhodou.

Jedním z hlavních důvodů existence pravidel kapitálové přiměřenosti je ochrana spotřebitele před bankovním selháním. Banky přirozeně inklinují ke snižování svého kapitálu, protože náklady na tento držený kapitál jsou vysoké.

Stanovení výše minimálního kapitálu není postaveno na podílu vlastního kapitálu a dluhu, protože to by vedlo banku k provádění výnosnějších, avšak výrazně rizikovějších transakcí. Tento koncept by tedy nezabránil možné insolventnosti banky. Basilejský výbor tedy požadavky kapitálové přiměřenosti postavil na rizikové vyváženosti. (Jurošková, 2012)

1.2.2 Basel I

Basel I je první z basilejských dohod z roku 1988, kterou sice podepsaly pouze země G-10, ale která se stala celosvětovým standardem řízení rizik. Tato dohoda nařizovala bankám držet kapitál ve výši alespoň 8% ze součtu rizikově vyvážených aktiv. Toto číslo nebylo vědecky podloženo, avšak znamenalo výrazné zvýšení kapitálového nárazníku, protože před dluhovou krizí se výše kapitálu významných bank pohybovala okolo 4%. Na implementaci pravidel dostaly banky 4 roky a každý stát si mohl určit způsob, jakým nová pravidla zavede.

Basel I byl velkým úspěchem vzhledem k tomu, že se jednalo o nezávazný dokument vydaný neformální autoritou. Přesto obsahoval řadu nedostatků, které byly známy již při vydání dokumentu (Jurošková, 2012):

- rizikové váhy byly jednoduše definovány a neuvažovaly postavení protistrany, což bylo výhodné pro snadnou aplikaci, ovšem riziko neodpovídalo realitě,
- riziková váha soukromého sektoru byla jednotně definována jako 100%,

- nebyla dostatečně ošetřena problematika opravných položek k úvěrům,
- nepodařilo se vytvořit stejné podmínky v jednotlivých státech,
- zohledňováno bylo pouze úvěrové riziko.

I přes své nedostatky byla první basilejská dohoda velkým posunem v oblasti regulace a bankovního dohledu. Díky svojí jednoduché struktuře a zapojení mnoha států i mimo G-10, do které patřily Belgie, Kanada, Francie, Německo, Itálie, Japonsko, Nizozemí, Spojené státy americké, Švédsko a Velká Británie, zajistila tato dohoda významný růst minimální úrovně kapitálu bank.

1.2.3 Basel II

Druhá basilejská dohoda označovaná jako Basel II vyšla ve svojí první verzi již v roce 1999, ovšem v platnost vešla až její třetí verze v roce 2004, která odstraňovala přílišnou obecnost první verze a nechtěné povinné navýšení kapitálu pro banky se sofistikovaným přístupem k řízení rizik. (Dvořák, 2005)

Druhá basilejská dohoda vyšla pod názvem New Basel Capital Accord (Nová basilejská kapitálová dohoda) a jedním jejím z hlavních cílů bylo umožnit bankám používání vlastních modelů pro měření rizik.

Zásadní rozdíl mezi Basel I a Basel II je přechod od jednoduchých metod vyjádření potřeby minimální úrovně kapitálu k pokročilejším postupům. Basel II je tedy výrazně komplikovanější dokument než Basel I, ale v zásadě lze jeho strukturu rozdělit do tří pilířů.

První pilíř vychází z první basilejské dohody, ovšem je flexibilnější v možnostech měření rizik. Navíc jsou minimální kapitálové požadavky rozšířeny o operační riziko. Banka si může zvolit, zda bude postupovat podle definovaných norem, nebo zda u kontrolního orgánu zažádá o schválení speciálních přístupů k měření rizik. Využití speciálních přístupů může, především u kreditního rizika, přinést značnou úsporu nákladů na kapitál. (Jurošková, 2012)

Tento první pilíř odstraňuje nedostatek první basilejské dohody, kterým bylo hrubé stanovování rizikových vah dle obecného charakteru dlužníka, a přechází se na individuální vyhodnocování bonity jednotlivých klientů.

Standardizovaná metoda určená pro ohodnocování rizikovosti pohledávky vycházela ze stanovení rizikových vah pohledávek za centrálními bankami a vládami externími

ratingovými agenturami. Regulátor pak mohl takto určené ohodnocení rizikovosti upravit u pohledávek za centrálními bankami a vládami. Rizikovost pohledávek za bankami pak mohla být stanovena jako o stupeň horší než rizikovost centrální banky země nebo na základě externího ratingu banky. Riziková váha pohledávek za fyzickými osobami pak byla stanovena na 75%.

Metoda interních ratingů byla alternativou ke standardizované metodě. Rozdíl spočíval v tom, že při výpočtu rizikovosti výpočet kombinoval charakteristiky vycházející z interního hodnocení banky se stanovenými postupy.

Druhý pilíř upravuje aktivity týkající se bankovního dohledu, kterým je v případě České republiky Česká národní banka. Jedná se o definici práv a povinností národních regulátorů. Důležitým úkolem je kontrola spolehlivosti metody měření rizika. Efektivní proces by se podle Basilejského výboru měl řídit následujícími čtyřmi principy (Dvořák, 2005):

1. Existence postupu pro ohodnocení své kapitálové přiměřenosti vzhledem ke svému rizikovému portfoliu.
2. Regulátor hodnotí interně stanovenou úroveň kapitálové přiměřenosti a její schopnost monitorování ukazatelů.
3. Regulátor požaduje zvyšování stanovené kapitálové přiměřenosti nad minimální stanovenou úroveň.
4. Regulátor včas preventivně intervenuje, aby zabránil poklesu kapitálu pod minimální stanovenou úroveň.

Regulátor má práva informační, regulační, ale také sankční a jeho úkolem je kontrola spolehlivosti a prediktivní účinnosti interních metody měření rizika v jednotlivých bankách.

Třetí pilíř upravuje tržní disciplínu, tedy zveřejňování relevantních ukazatelů rizik. Banky podle tohoto pilíře budou nuceny implementovat moderní metody řízení rizik a dále je zpřesňovat a zároveň informovat o svých postupech a detailu svého rizikového profilu. Třetí pilíř druhé basilejské dohody klade důraz na transparentnost a dokumentaci metod a výsledků měření.

1.2.4 Basel III

V roce 2010 zareagoval Basilejský výbor na globální finanční krizi vydáním souboru nových nařízení s označením Basel III. Tato basilejské dohoda se snaží odstranit nedostatky jejího předchůdce Basel II, které se projevily především během krize. Basel III se snaží jasně definovat pravidla a parametry a zároveň zavádí leverage ratio jako doplněk ke kapitálové přiměřenosti, aby se zamezilo podceňování rizik v době ekonomického růstů.

Cílem dohody Basel III je vytvoření odolného bankovního systému, který podpoří trvale udržitelný ekonomický růst. Snaží se upravit bankovní prostředí tak, aby dokázalo absorbovat šoky a zmenšovat dopad selhání trhů.

Změny proti dohodě Basel II spočívají především v posílení kapitálu bank, přičemž důraz je kladen nejen na jeho objem, ale také na dostatečnou kvalitu tohoto kapitálu. Dále Basel III podporuje vytváření kapitálových rezerv v době ekonomického blahobytu tak, aby mohl zmírnit dopady budoucí krize, tyto rezervy se nazývají proticyklický polštář. Zaměřuje se také na komplexnější řízení rizik. Zavádí také globální standardy likvidity. (Jurošková, 2012)

Zaměření se na pravidla likvidity jsou výraznou změnou proti předchozím dohodám, které řešily výlučně kapitálovou přiměřenost. K této změně přispěla především globální dluhová krize a problémy s likviditou během ní. Basilejský výbor ve třetí dohodě zavádí dva nové pojmy. Jsou jimi ukazatel krytí likviditou a ukazatel čistého stabilního financování.

Česká národní banka v roce 2011 vydala svůj postoj k nařízením Basilejského výboru pro bankovní dohled. Souhlasila například se zkvalitněním kapitálu, naopak výhrady měla k tvorbě proticyklického polštáře, jehož kalibrace se jevila jako složitá.

Pravidla definovaná v dokumentu Basel III se začala implementovat v roce 2013 a jejich zavádění by mělo probíhat až do roku 2018 tak, aby jejich aplikace nebrzdila růst ekonomik po finanční krizi.

2 Strojové učení

Strojové učení je oblastí umělé inteligence, která slouží k odvozování znalostí z předložených příkladů. Algoritmy strojového učení jsou schopny provádět klasifikaci dat, a proto nacházejí uplatnění při řešení úloh v mnoha oblastech výzkumu jako je lékařství, robotika nebo tvorba bází pro expertní systémy.

Schopnost učit se, neboli adaptovat se měnícím se podmínkám, je vlastností živých organismů, kterou se v minulosti odlišovali od strojů. Disciplíny umělé inteligence a strojového učení se snaží integrovat tuto vlastnost i do technických systémů.

Základním předpokladem pro úspěšnou klasifikaci pomocí algoritmů strojového učení je existence diskriminačních predikátů, což jsou atributy, které odlišují jednu třídu dat od ostatních. Množina diskriminačních predikátů může být značně obsáhlá, jejího zjednodušení lze dosáhnout generalizací. Opakem generalizace je specializace, která přidává nové prvky do skupiny predikátů. Specializace se provádí, pokud predikáty jednoznačně nepopisují danou třídu dat. (Mařík, 1993)

Dalšími operacemi strojového učení jsou abstrakce a konkretizace, dedukce a indukce. Pokud je ubrána část informace, jedná se o abstrakci, v opačném případě se jedná o konkretizaci, tedy zpřesnění výroku přidáním informace. Dedukce je způsob odvozování, který zachovává pravdivost, tedy je vždy jistý správný výsledek. Indukce je typ inference, kdy není zaručen správný výsledek.

Problémem učení z příkladů je prohledávání stavového prostoru. Při prohledávání lze uplatnit výše zmíněné operátory. Znalosti obvykle mají tvar generalizačních stromů, což je pro strojové učení typické.

Přesnost výsledného popisu je vhodným kritériem pro posuzování a porovnávání přístupů k učení. Přesnost udává procentuální úspěšnost při klasifikaci nových objektů, při aplikaci popisu.

Strojové učení lze rozdělit do dvou základních skupin - učení s učitelem a učení bez učitele.

Učení s učitelem lze využít v situaci, kdy je znám dostatečný počet příkladů včetně jejich výstupní hodnoty. To si můžeme představit jako situaci v bance, kde pracuje zkušený expert na spotřebitelské úvěry, který je schopen s velkou přesností hodnotit klientovu schopnost splácet. Při velkém zájmu o spotřebitelské úvěry budeme chtít

zvýšit počet našich expertů a najmeme absolventy vysoké školy, ovšem zjistíme, že nedosahují kvalit našeho experta. Rozdíl je ve zkušenostech, které stávající pracovník získal z dříve zpracovaných žádostí o úvěr a následné platební morálky klientů. Tyto jeho informace o v minulosti zpracovaných žádostech o úvěr jsou trénovacími daty, podle kterých formuluje rozhodnutí o nově příchozích žadatelích. Pokud by tato historická data byla sepsána, jednalo by se o koncept, podle kterého mohou nově najatí pracovníci vyhodnocovat své klienty. Stávající expert banky by tedy reprezentoval učitele. Pokud bychom nahradili tyto pracovníky algoritmem, který by na základě trénovacích dat vyhodnotil, zda klient je schopen splatit úvěr, o který žádá, či nikoli, jednalo by se o strojové učení s učitelem.

U učení s učitelem jsou tedy příklady, neboli trénovací data, již rozřazená do příslušných tříd a systém tuto informaci explicitně poskytuje.

Učení bez učitele má proti předchozí situaci jeden zásadní rozdíl, a tím je absence klasifikované množiny trénovacích dat. Vrátíme-li se opět k příkladu s bankou, můžeme si představit situaci, kdy počet zaměstnanců nerozšiřujeme, ale jsme nuceni zkušeného experta nahradit novým, méně zkušeným pracovníkem. Nový pracovník nemá šanci těžit z příkladů úvěrů z minulosti. Může však nově příchozí klienty roztřídit do skupin nebo mezi nimi hledat nové vazby a následně aplikovat samoučení.

Tato práce se zabývá metodou Supprort Vector Machine, která sestavuje model na základě trénovací množiny a spadá tedy do kategorie učení s učitelem.

V následujících podkapitolách jsou stručně popsány vybrané metody strojového učení, alternativy metody podpůrných vektorů.

2.1 Top-down induction of decision trees

Algoritmus TDIDT je jednoduchý příklad učení formou induktivního rozhodovacího stromu. Řadí se mezi algoritmy umělé inteligence určené pro klasifikaci dat, přestože postrádá některé vlastnosti typické pro umělou inteligenci, jako je například využití dříve získaných znalostí. (Mařík, 1993)

Výsledný popis má tvar rozhodovacího stromu. Při klasifikaci je procházen strom shora dolů a postupně jsou prováděny testy předepsané v uzlech, v listech se nachází výsledné ohodnocení.

Při vytváření rozhodovacího stromu je nejprve vyhledán atribut, který v sobě nese největší množství informace, tedy atribut, který nejlépe diskriminuje mezi kladnými a zápornými příklady. Tento atribut se stává kořenem.

Dalším krokem je rozdělení množiny příkladů na tolik podmnožin, kolik je hodnot kořenového atributu. V každé z podmnožin jsou příklady s jedinou hodnotou tohoto atributu. Poté je v každé z podmnožin vyhledán další nejvýznamnější atribut a takto se rekurzivně pokračuje, dokud není dosaženo kritérium ukončení.

Pro výběr nejvýznačnějšího atributu existují různá kritéria. Nejběžnějším je měření množství informace pomocí Shannovy entropie, která je definována vztahem 2.1

$$H_j = -p_1\, log_2 p_1 - p_2 log_2 p_2\,, \tag{2.1}$$

kde jednotlivé proměnné popíšeme jako:

p_1 … je poměr pozitivních příkladů v *j*-té podmnožině k celkovému počtu prvků této podmnožiny

p_2 … je poměr negativních příkladů v *j*-té podmnožině k celkovému počtu prvků této podmnožiny.

H_j je kladné číslo, neboť p_1 a p_2 jsou z intervalu $\langle 0; 1\rangle$. Celková velikost entropie je dána váženým součtem entropií jednotlivých podmnožin

$$H = \sum_{j=1}^{K} P_j H_j\,, \tag{2.2}$$

kde: *K*... je počet podmnožin indukovaných atributem

H_j ... je entropie podmnožiny *j*

P_j ... je poměr velikosti *j*-té podmnožiny k množině všech příkladů.

Nejvýznačnějším atributem je takový atribut, pro který je hodnota *H* minimální.

Podívejme se na jednoduchý modelový příklad. V tabulce 1 jsou znázorněna data o věku klienta a jeho měsíčním příjmu. Podle těchto dat se pokusíme vytvořit rozhodovací strom.

Tabulka 1: Data pro znázornění TDIDT

Úvěr řádně splacen	Věk pod 30 let	Měsíční příjem pod 20 000 Kč
ANO	NE	NE
ANO	NE	NE
NE	NE	ANO
NE	ANO	ANO
NE	ANO	NE
NE	ANO	NE

Zdroj: vlastní zpracování, 2015

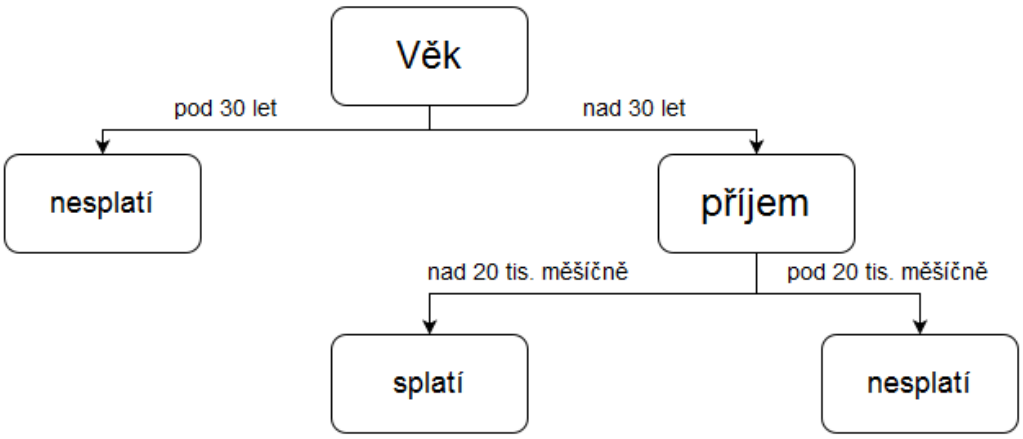
Abychom správně vybrali kořen rozhodovacího stromu, musíme vypočítat entropii atributů, tedy věku a příjmu.

$$H_{věk} = \frac{3}{6}\left(-\frac{2}{3}\ log_2\frac{2}{3} - \frac{1}{3}log_2\frac{1}{3}\right) \approx 0{,}46$$

$$H_{plat} = \frac{4}{6}\left(-\frac{2}{4}\ log_2\frac{2}{4} - \frac{2}{4}log_2\frac{2}{4}\right) \approx 0{,}67$$

Nižší entropii má atribut věk, který vybereme jako kořen stromu. Výsledný strom pro tento příklad je znázorněn na obrázku 1.

Obrázek 1: Rozhodovací strom algoritmu TDIDT



Zdroj: vlastní zpracování, 2015

2.2 AQ algoritmus

Typičtější metodou umělé inteligence, než výše uvedený TDIDT, je algoritmus AQ (poprvé publikován Michalskim v roce 1969), který je určen pro učení z příkladů popsaných pomocí hodnot atributů. Jeho výstupem jsou produkční pravidla.

Jedná se o učení z klasifikovaných příkladů.

Postup algoritmu lze, podle Mařík (1993), shrnout následujícím způsobem:

1. Rozdělit množinu příkladů na dvě podmnožiny: množinu *PE* obsahující pouze pozitivní příklady a množinu *NE* obsahující pouze negativní příklady.
2. Vybrat z množiny *PE* náhodně jeden příklad a označit jej *s* (tento příklad nazýváme jádro).
3. Naleznout všechny maximální generalizace popisu jádra, přičemž generalizace nesmí pokrýt žádný z příkladů množiny *NE*.
4. Podle vhodného popisu preferenčního kritéria vybrat nejlepší z těchto popisů a zařadit jej do množiny popisů.
5. Pokud množina popisů pokrývá všechny prvky z *PE*, ukončit práci.

Výsledným popisem je pak disjunkce všech nalezených popisů. Jinak je nutné vybrat nové jádro z dosud nepokrytých pozitivních příkladů.

Preferenční kritérium může být založeno na nákladové funkci tak, že přednost je dávána atributům se snadněji dostupnými hodnotami (například změření teploty těla je dostupnější než tomografické vyšetření).

Postup algoritmu znázorníme na modelovém příkladu. Data o klientech žádajících o úvěr pro tento případ jsou zobrazena v tabulce 2.

Tabulka 2: Data pro znázornění AQ algoritmu

Číslo klienta	Úvěr řádně splacen	Věk pod 30 let	Měsíční příjem pod 20 000 Kč
1	**ANO**	NE	ANO
2	**ANO**	NE	NE
3	**ANO**	ANO	NE
4	**NE**	ANO	ANO
5	**NE**	ANO	ANO

Zdroj: vlastní zpracování, 2015

Množinu dat rozdělíme na pozitivní (PE) a negativní příklady (NE). Do množiny PE spadají klienti 1, 2 a 3 a z nich náhodně vybereme jednoho, který bude představovat jádro. Zvolíme klienta 3 a vytvoříme pro něj co neobecnější popis, takový, aby nepopisoval ani jeden prvek z množiny NE. Klient číslo 3 je mladší třiceti let a jeho měsíční příjem přesahuje dvacet tisíc Kč. Tento popis můžeme zobecnit vynecháním informace o věku, jelikož tak nepopíšeme žádného klienta z množiny NE, zároveň však popíšeme klienta číslo 2 z množiny PE. Z množiny PE zbývá popsat klient 1, který je starší třiceti let a jeho měsíční příjem je pod 20 000 Kč. I jeho popis můžeme zobecnit, a to vynecháním informace o příjmu.

Výsledný popis klientů, kteří jsou schopni splatit úvěr je následující:

Je starší 30 let NEBO jeho měsíční příjem přesahuje 20 000 Kč.

2.3 Neuronové sítě

Další z technik vhodných pro klasifikaci dat jsou neuronové sítě. Jedná se sítě mnoha jednoduchých procesorů, které jsou navzájem propojeny. Graf těchto propojení nazýváme topologie sítě. Procesory nazýváme neurony podle centrální nervové soustavy člověka, jelikož ji zjednodušeně modelují.

Původ názvu neuronových sítí je v době vzniku prvních počítačů, kdy se vědci snažili o matematickou reprezentaci biologických systémů. Bylo vytvořeno mnoho modelů,

ovšem mnoho z nich selhávalo kvůli jejich příliš vysoké biologické věrohodnosti. Zájem o neuronové sítě poklesl a růst opět začal až v 70. letech, kdy byl vytvořen algoritmus zpětného šíření. (Bishop, 2009)

2.3.1 Perceptron

Model neuronové sítě sestávající se z jednoho neuronu se nazývá perceptron. Perceptron je základní stavební jednotnou neuronových sítí a poprvé byl popsán v roce 1957.

Perceptrony jsou schopny klasifikovat pouze lineárně separovatelné množiny dat, a to do dvou tříd. Ke klasifikaci do více tříd je nutné zvýšit počet vrstev perceptronu, vzniká tak vícevrstvý perceptron.

Do neuronu se sbíhá *n* spojů (axonů), které reprezentují výstupy z jiných neuronů nebo podněty z okolí. Každý z axonů vede do neuronu příznak, který je vyjádřen číselnou hodnotou a pochází ze vstupního prostoru. Vektor $\boldsymbol{x} = [x_1, \dots, x_n]$ jsou vstupy, které charakterizují zkoumaný objekt. Každý spoj je navíc popsán reálným číslem *w*, které udává jeho (synaptickou) váhu, a každý neuron prahem θ. Potenciál neuronu lze vyjádřit jako

$$\xi = \sum_{i=1}^{n} w_i x_i - \vartheta \,. \tag{2.3}$$

kde: ξ … udává potenciál neuronu,

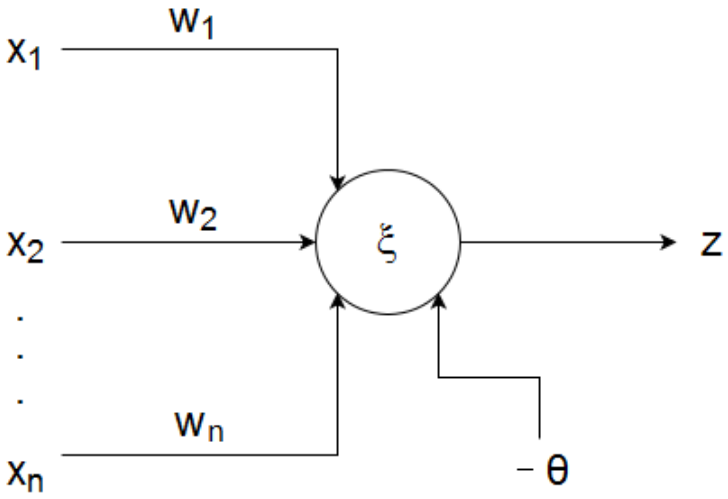
w_i … je synaptická váha spoje,

ϑ … je prahem neuronu,

x_i … je vstupní signál.

Na tento potenciál reaguje neuron svou odezvou $z = S(\xi)$, kde S je nelineární přenosová funkce.

Obrázek 2: Perceptron



Zdroj: vlastní zpracovnání, 2015

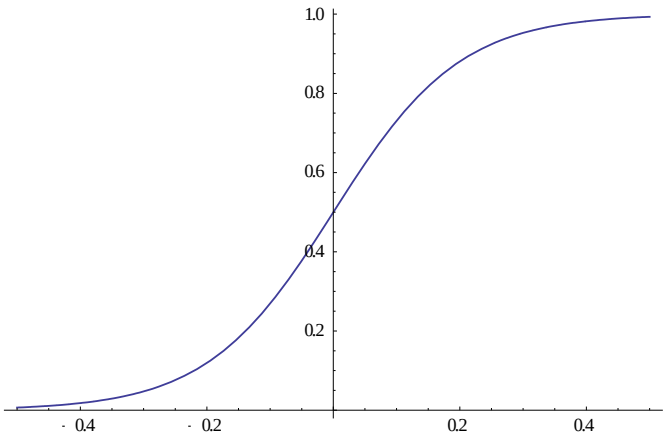
Přenosová funkce má obvykle tvar logistické funkce (sigmoidy), což je monotónně rostoucí funkce mezi dvěma asymptotickými hodnotami.

$$S(\xi) = \frac{1}{1 + \exp(-\lambda\xi)}, \tag{2.4}$$

kde λ udává strmost funkce a ξ je potenciál neuronu. (Mařík, 1993)

Příklad logistické funkce se strmostí 10 je uveden v grafu 3, vyvedeném v softwaru Wolfram Mathematica.

Graf 3: Logistická funkce (λ = 10)



Zdroj: vlastní zpracování, 2015

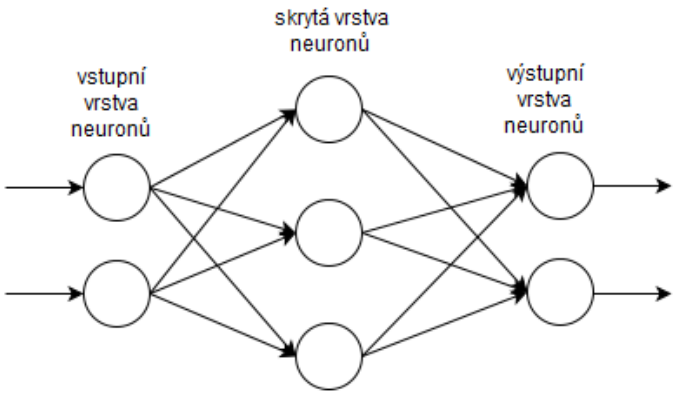
Je-li $\boldsymbol{w}.\boldsymbol{x} > 0$, tedy překročí-li celkový podnět práh, pak zkoumaný objekt leží v kladném nadprostoru vymezeném separující nadrovinou $\boldsymbol{w}.\boldsymbol{x} = 0$, v opačném případě leří v nadprostoru doplňkovém.

2.3.2 Vícevrstvé sítě

Vícevrstvá síť typu $m - k_1 - k_2 - k_r - n$ je síť se vstupní vrstvou dimenze m přijímající vstupní faktory v podobě vektoru $\mathbf{x} = [x_1, \dots, x_n]$, výstupní vrstvou dimenze n s odezvou $\mathbf{y} = [y_1, \dots, y_n]$ a r skrytými vrstvami, kde s-tá vrstva má k_s neuronů.

Jako vícevrstvou síť lze označit takovou síť neuronů, kde alespoň jedna vrstva neuronů nemá externí vstup nebo výstup. Na obrázku 3 je naznačen případ vícevrstvé sítě s jednou skrytou vrstvou neuronů.

Obrázek 3: Vícevrstvá neuronová síť



Zdroj: vlastní zpracování, 2015

Neurony dvou sousedních vrstev jsou propojeny každý s každým, čímž vzniká tzv. konekcionistický systém.

Aktivní neboli pracovní mód sítě je podle Mařík (1993) následující:

1. Vektor **x** je vstupem pro neurony první, vstupní vrstvy. Tato vrstva přenáší příslušné vstupní hodnoty vektoru ovlivněné váhami spojů do neuronů první skryté vrstvy.
2. Stimuly jsou v první skryté vrstvě zpracovány a slouží jako podněty do další vrstvy.
3. Bod dva se opakuje, dokud z výstupů nevytvoříme vektory **y** výsledné odezvy celé sítě na stimul **x**.

Pro klasifikaci vstupů {x} do dvou tříd lze použít síť $m - k - 1$ s trénovací množinou $\{[\boldsymbol{x}, \boldsymbol{y}], ...\}$, kde y může nabývat hodnot 0 nebo -1 v závislosti na třídě, do které patří.

Důležitou podmínkou pro úspěšnou klasifikaci je správný návrh vstupů a třídění odpovědí.

Neuronové sítě jsou v oblasti klasifikace dat úspěšnější než většina konkurenčních metod, především ve smyslu generalizační schopnosti. Dolní vrstvy neuronů provedou takovou kombinaci lineárních a nelineárních transformací vstupů, že ve výstupní vrstvě lze aplikovat separující nadrovinu i u množin, které nejsou lineárně separovatelné.

2.4 Support Vector Machine

Support Vector Machine je jedna z modernějších technologií pro klasifikaci dat. Tato metoda se snaží odstranit nedostatky neuronových sítí a v posledních letech se objevila v mnoha studiích zabývajících se klasifikací dat nebo regresní analýzou. (Li, 2006)

Cílem této práce je využít Support Vector Machine pro klasifikaci klientů podle jejich schopnosti splácet při žádosti o spotřebitelský úvěr, proto se touto metodou podrobněji zabývá následující kapitola.

3 Support Vector Machine

Support Vector Machine (SVM) je metoda strojového učení s učitelem pro klasifikaci podle vzoru. Metoda formuluje závěr podle trénovacích dat, na jejichž základě sestavuje vyhodnocovací model. Učením s učitelem je myšleno, že u trénovacích dat lze určit, do jaké třídy daný prvek patří.

Podle složitosti rozložení trénovacích dat a jejich dimenze pak můžeme využít jeden z typů SVM, kterými jsou maximum margin classifier a soft margin classifier. Ty můžeme následně rozšířit pomocí jádrových transformací.

Principem SVM klasifikátoru je najít takovou nadrovinu, která by dokázala optimálně oddělit prvky pařící do definovaných tříd dat v trénovacím souboru. Metoda se snaží minimalizovat strukturální riziko, což je jeden z hlavních rozdílů v porovnání například s metodou umělých neuronových sítí, které minimalizují riziko empirické.

Důležitou vlastností metody Support Vector Machine je, že úloha určení parametrů modelu je konvexním optimalizačním problémem, a tak jakékoli lokální řešení úlohy je zároveň globálním optimem. (Bishop, 2009)

Metoda podpůrných vektorů, díky svým dobrým výsledkům při klasifikaci a predikci, našla uplatnění v řadě vědních disciplín. Výzkumná skupina okolo H. K. Lama (2012) popsala její využití a aplikaci pro klasifikaci pacientů podle rizika úmrtí na srdeční chorobu nebo pro rozpoznávání ručně psaných číslic.

Dále lze najít studie o využití SVM pro rozpoznávání otisků prstů, rozpoznávání tváří z obrázků, rozpoznávání textu, pro potvrzení rakovinové tkáně nebo třeba pro předpověď vývoje akcií.

Cílem metody je proložit trénovací data nadrovinou tak, aby co nejoptimálněji oddělovala prvky spadající do různých tříd.

3.1 Rozpoznávání podle vzoru

Support Vector Machine patří do skupiny klasifikačních metod, které se snaží data x_i rozdělit do dvou tříd, definovaných hodnotami $y_i \in \{+1, -1\}$. Proto je v následujících odstavcích nejprve stručně popsána myšlenka klasifikace podle vzorových dat a jednodušší alternativa k Support Vector Machine.

Rozhodnutí o tom, do které skupiny vyhodnocovaný prvek patří, formulují metody podle vzorových dat, často nazývaných jako trénovací data, což je množina dříve naměřených hodnot, k nimž známe hodnotu výstupu a podle kterých se formuluje rozhodovací model. Tento typ učení je aplikován v řadě algoritmů strojového učení včetně SVM.

Jednodušší aplikace rozpoznávání podle vzoru je testování statistických hypotéz v jednodimenzionálním prostoru (Nilsson, 2006). Příkladem takového testu může být Studentův *t*-test. V Studentově testu máme hypotézy H_0 a H_1, které odpovídají třídám +1, -1, neboli

$$g(x) = \begin{cases} H_0, když\ x < \bar{x} \\ H_1, když\ x > \bar{x} \end{cases} \tag{3.1}$$

, kde $\bar{x}$ je průměrem průměrů jednotlivých tříd

$$\bar{x} = \frac{1}{2}\left(\frac{1}{l_1}\sum_{i:y_1=+1} x_i + \frac{1}{l_{-1}}\sum_{i:y_1=-1} x_i\right) \tag{3.2}$$

a kde $l_c = |i: y_1 = c|$ a $H_0 < H_1$ se nazývá rozhodovacím pravidlem nebo také klasifikátorem. Říkáme tedy, že klasifikátor g je vyvozen z trénovacích dat x_i, v tomto případě výpočtem průměru $\bar{x}$.

Nicméně úlohy rozpoznávání obvykle zahrnují vícerozměrná data a neznámá základní rozdělení. Proto je prakticky nemožné vytvořit statistický test podle výše zmíněného postupu. Takové úlohy jsou obvykle řešeny pomocí algoritmů zmíněných výše, jako jsou umělé neuronové sítě, rozhodovací stromy a v neposlední řadě také využitím metody SVM.

3.2 Maximum margin klasifikátor

Ačkoli metoda podpůrných vektorů je schopna klasifikovat data do více než dvou tříd, pro účely hodnocení schopnosti klientů žádajících o úvěr splácet se budeme věnovat případu binární klasifikace dat, tedy rozdělování trénovacích dat do právě dvou tříd. Tato teorie však může být rozšířena pro klasifikace do více než dvou tříd, a to bez ztráty obecnosti.

Uvažujme tedy základní případ lineárně separovatelných dat s cílem roztřídit tato trénovací data do právě dvou tříd. Soubor trénovacích dat lze zapsat jako

$$S = \{(\mathbf{x}_1, y_1) \dots (\mathbf{x}_m, y_m)\} \tag{3.3}$$

, kde $\mathbf{x} \in R^d$, neboli $\mathbf{x}$ náleží d-dimenzionálnímu vstupnímu prostoru a y_i je identifikátor třídy a nabývá hodnot -1 a 1. Vektor $\mathbf{x}$ reprezentuje trénovací data a nese v sobě atributy popisující jednotlivé entity, hodnota y_i vyjadřuje příslušnost vektoru $\mathbf{x_i}$ do příslušné třídy +1 nebo -1.

Těmito trénovacími daty se metoda snaží proložit nadrovinu

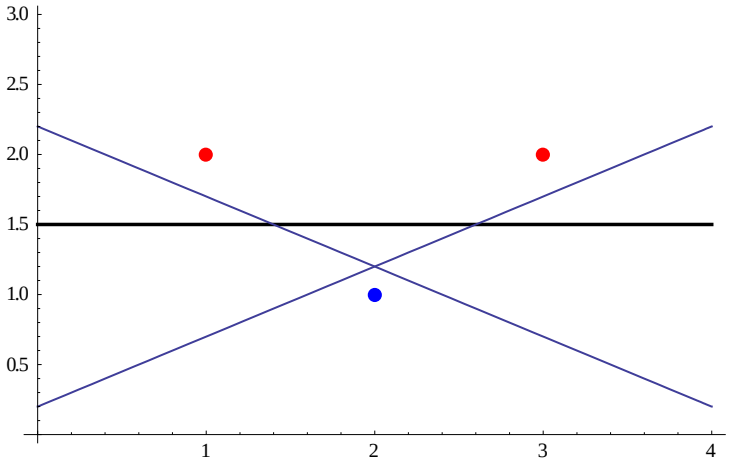
$$\begin{gathered} \langle \mathbf{w}, \mathbf{x} \rangle + b = 0 \\ \text{nebo jinak zapsáno} \\ \sum_{j=1}^{n} x_j w_j + b = 0 \end{gathered} \tag{3.4}$$

a právě na této nadrovině a hledání její optimální polohy je metoda Support Vector Machine postavena. S využitím nadroviny (3.4) pak rozhodnovací funkce f může být definována jako

$$f(x) = sgn(\langle \mathbf{w}, \mathbf{x} \rangle + b) \tag{3.5}$$

, kde vektor $\mathbf{w}$ vyjadřuje orientaci roviny a jedná se o vektor vah, který je kolmý na separující nadrovinu, $\langle \mathbf{w}, \mathbf{x} \rangle$ je skalární součin vektorů $\mathbf{w}$ a $\mathbf{x}$. Proměnná b je tzv. „bias“ neboli posun od počátku. Je zřejmé, že existuje nekonečně mnoho rovin splňujících rovnici (3.5), což je vidět na grafu 4, kde jsou body [1,2] a [3,2] odděleny od bodu [2,1] několika přímkami splňujícími rovnici (3.5). (Lam, 2012)

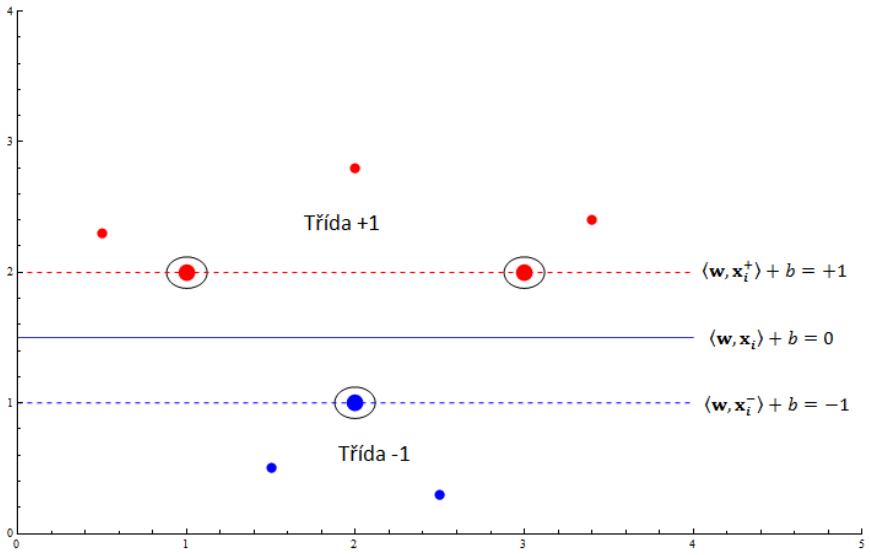
Graf 4: Možnosti polohy diskriminační nadroviny



Zdroj: vlastní zpracovnání, 2015

Nevýhodnější je však rozdělit data nadrovinou tak, aby se separační nadrovina nacházela uprostřed mezi oběma třídami (v grafu 4 se jedná o přímku y = 1,5). Tento koncept se snaží vytvořit obecné rozdělení dvou tříd takové, abychom s co nejvyšší pravděpodobností zařadili nová data do správné třídy. Proto optimální klasifikátor je takový, který najde zobecňující rovinu, která maximalizuje rozpětí mezi klasifikovanými třídami. Poloha výsledné roviny tedy závisí pouze na podpůrných vektorech (viz graf 5).

Graf 5: Podpůrné vektory, oddělující rovina s lineárním jádrem



Zdroj: vlastní zpracovnání, 2015

Cílem je maximalizovat vzdálenost těchto nejbližších, podpůrných bodů od separující nadroviny. Vyjdeme-li z rovnic přímek v grafu 5, kde je zobrazena kladná nadrovina

$\boldsymbol{x}^+$, respektive přímka, $\langle \mathbf{w}, \mathbf{x}_i^+ \rangle + b = +1$ a záporná ($\boldsymbol{x}^-$) nadrovina $\langle \mathbf{w}, \mathbf{x}_i^- \rangle + b = -1$, pak rozpětí rovnice (3.4) můžeme vyjádřit jako rozdíl kladné a záporné roviny. Každý bod kladné roviny můžeme vyjádřit jako bod záporné roviny, k němuž přičteme konstantu *k* ve směru **w**

$$\boldsymbol{x}^- + k\mathbf{w} = \boldsymbol{x}^+,$$

dosazením do rovnice kladné roviny zjistíme hodnotu konstanty *k*

$$(\boldsymbol{x}^- + k\mathbf{w}) \cdot \mathbf{w} + b = 1$$

$$k = \frac{2}{\mathbf{w} \cdot \mathbf{w}}.$$

Nyní můžeme po dosazení hodnoty *k*, vyjádřit velikost rozpětí

$$M = |\boldsymbol{x}^- + k\mathbf{w} - \boldsymbol{x}^-| = k|\mathbf{w}| = \frac{2}{\mathbf{w} \cdot \mathbf{w}} \sqrt{\mathbf{w} \cdot \mathbf{w}} = \frac{2}{\sqrt{\mathbf{w} \cdot \mathbf{w}}}.$$

Úlohou je maximalizovat toto rozpětí za podmínek

$$\mathbf{wx} + b \geq \mathbf{1}$$

a

$$\mathbf{wx} + b \leq \mathbf{-1}.$$

Tyto dvě podmiňující rovnice lze spojit do jedné souhrnné rovnice

$$y_i(\mathbf{x}_i \cdot \mathbf{w} + b) \geq 1, kde\ i = 1, 2, \dots, \ell\ .$$

Využitím euklidovské normy a faktu, že se zvětšujícím se rozpětím se zmenšuje velikost vektoru **w,** dostáváme výslednou optimalizační úlohu.

$$\min \frac{1}{2} \|\mathbf{w}\|^2$$
$$s.t.: y_i(\mathbf{x}_i \cdot \mathbf{w} + b) \geq 1, i = 1, 2, \dots, \ell \qquad (3.6)$$

Rovnosti v podmiňující rovnici dosahuje obvykle jen několik bodů, které se nazývají podpůrné vektory, a řešení celé úlohy je závislé právě na těchto bodech (Nilsson, 2006).

Jelikož řešené úlohy jsou obvykle multidimenzionální, stává se úloha 3.6 ve své primární formě prakticky neřešitelnou, proto je nutné využít principu duality a primární úlohu převést do její duální formy, která je následující

$$\max \sum_{i=1}^{\ell} \alpha_i - \frac{1}{2} \sum_{i=1}^{\ell} \sum_{j=1}^{\ell} \alpha_i \alpha_j y_i y_j \mathbf{x}_i \mathbf{x}_j$$
$$s.t.: \sum_{j=1}^{\ell} \alpha_i y_i = 0 \ a \ \alpha_i \geq 0, kde \ i = 1, 2, \dots, \ell \tag{3.7}$$

Vyřešením duální úlohy získáme optimální váhový vektor **w** a posunutí *b*.

$$\mathbf{w} = \sum_{i=1}^{\ell} \alpha_i y_i \mathbf{x}_i$$
$$b = -\frac{1}{2}(w^T x_+ + w^T x_-) \tag{3.8}$$

, kde x_+ a x_- jsou podpůrné vektory z jednotlivých tříd (Nilsson, 2006).

3.3 Soft margin klasifikátor

Jsou-li data lineárně separovatelná, je možná najít nadrovinu, která oddělí jednotlivé třídy trénovacích dat bez empirické chyby. Ale reálná data není obvykle možné jednoznačně lineárně rozdělit do tříd. Metoda maximum margin tedy není uplatnitelná a k problému musíme přistoupit jinak. Řešení je možné uvolněním (relaxací) podmínek a tolerováním malé chyby v klasifikaci. Metoda zavádí proměnnou pro vyjádření možné chyby. Optimalizace polohy dělící nadroviny je touto metodou dosaženo maximalizací rozpětí, ovšem s uvažováním porušení podmínky, což je vyjádřeno proměnou ξ_i. Toto vede k následující minimalizaci

$$\min \frac{1}{2} \|\mathbf{w}\|^2 + C \sum_{i=1}^{m} \xi_i$$
$$s.t.: \ y_i(\langle \mathbf{w}, \mathbf{x_i} \rangle + b) \geq 1 - \xi_i \text{ a } \xi_i \geq 0 \ pro \ i = 1, \dots, m \ , \tag{3.9}$$

parametr *C* vyjadřuje, do jaké míry chyba v klasifikaci ovlivní polohu nadroviny. Hodnota konstanty *C* je volena před samotnou optimalizací. Pokud je nastaveno $C = \infty$, pak je hledána taková nadrovina, aby nedošlo k chybné klasifikaci v trénovacích datech. (Lam, 2012).

Stejně jako u maximum margin klasifikátoru je výhodné využít euklidovské normy a následně řešit tuto úlohu s využitím Lagrangeovy teorie duality.

Primární úlohu formulujeme jako

$$L(\mathbf{w}, b, \boldsymbol{\xi}, \boldsymbol{\alpha}, \boldsymbol{\beta}) = \frac{1}{2}\|\mathbf{w}\|^2 + C\sum_{i=1}^{m}\xi_i - \sum_{i=1}^{m}\beta_i\xi_i - \sum_{i=1}^{m}\alpha_i[y_i(\langle \mathbf{w}, \mathbf{x_i}\rangle + b) - 1 + \xi_i] \tag{3.10}$$

, kde α_ia β_i jsou nezávislé Lagrangeovy multiplikátory. Duální úloha může být řešena výpočtem všech parciálních derivací primární úlohy při součtu vah rovnajících se nule, neboli

$$\mathbf{w} = \sum_{i=1}^{m}\alpha_i y_i \mathbf{x_i} \tag{3.11}$$

a

$$0 = \sum_{i=1}^{m}\alpha_i y_i \tag{3.12}$$

To jest následně dosazeno do primární úlohy, z čehož pak vyplývá

$$L(\mathbf{w}, b, \boldsymbol{\xi}, \boldsymbol{\alpha}, \boldsymbol{\beta}) = \sum_{i=1}^{m}\alpha_i - \frac{1}{2}\sum_{j,i=1}^{m} y_i y_j \alpha_i \alpha_j \langle \mathbf{x_i}, \mathbf{x_j}\rangle \tag{3.13}$$

Rozdíl proti maximum margin metodě je v podmínce $\boldsymbol{\alpha} + \boldsymbol{\beta} = C$ **,** kde $\boldsymbol{\alpha}, \boldsymbol{\beta} \geq 0$, takže $0 \leq \boldsymbol{\alpha}, \boldsymbol{\beta} \leq C$.

Hodnota *C* tedy určuje horní hranici velikosti Langrangeových optimalizačních proměnných α_i a β_i, tato hranice je občas označována jako „box constraint“. Na volbě hodnoty *C* je závislý poměr mezi přesností a regularizací. Volba hodnoty *C* závisí na podkladových datech, podstatě problému a obvykle ji hledáme pomocí experimentálního křížového ověřování. To se nejlépe provádí na více sadách (foldech) testovacích dat. Celý proces se označuje jako „multi-folded cross-validation“. K výpočtu Lagrangeových multiplikátorů se nejčastěji používají algoritmy kvadratického programování (Lam, 2012).

3.4 Jádrové funkce

Zatím jsme předpokládali lineárně separovatelná data. Takový předpoklad je ovšem velkým omezením. Proto zavedeme transformaci vstupních, trénovacích dat do prostoru s vyšší dimenzí, tzv. „feature space“ (Li, 2006).

V případě, že data nejsou lineárně separovatelná, ale vykazují možnost nelineárního rozdělení do tříd, je možné je velmi efektivně rozdělit data do tříd pomocí jádrových funkcí tak, aby zobrazení původních, lineárně neseparovatelných dat byla lineárně separovatelná.

$$\Phi: \mathrm{R}^{\mathrm{n}} \rightarrow \mathrm{F} \tag{3.14}$$

Jádrovou funkci zavedeme tedy tak, že aplikujeme vhodnou funkci na skalární součin

$$K(u, v) = \langle \Phi(u), \Phi(v) \rangle. \tag{3.15}$$

Díky zavedení jádrové funkce do trénovacích dat převedeme původní data do tzv. „feature space“, ve kterém jsou separovatelná. Jádrové funkce se řídí Mercerovým teorémem a nabízejí implicitní mapování do „feature space“. Explicitní mapování nemusí být vypočítáno, což výrazně usnadňuje výpočty a v kombinaci s obecností SVM zmírňuje tzv. „prokletí dimensionality“ (Lam, 2006).

Skalární součin ve vzorci může být tedy jednoduše nahrazen příslušnou jádrovou funkcí bez vlivu na Lagrangeovu optimalizační metodu. Výsledný vzorec rozhodovací funkce pak má podobu

$$f(\mathbf{x_j}) = sgn\left[\sum_{i=1}^{nSVs} y_i \alpha_i K\langle \mathbf{x_i}, \mathbf{x_j} \rangle + b\right] \tag{3.16}$$

, kde *nSVs* značí počet podpůrných vektorů, y_i je označení tříd, α_i jsou Lagrangeovy multiplikátory a b je „bias“, $\mathbf{x_i}$ podpůrné vektory, které se určují pomocí trénovacího procesu, $\mathbf{x_j}$ testovaný vektor.

Myšlenku mapování pomocí jádrové funkce je možné ukázat na případu polynomiálního jádra stupně 2.

$$\langle u, v \rangle^2 = \left(\langle \begin{pmatrix} u_1 \\ u_2 \end{pmatrix} \cdot \begin{pmatrix} v_1 \\ v_2 \end{pmatrix} \rangle\right)^2 = \langle \begin{pmatrix} u_1^2 \\ \sqrt{u_1 u_2} \\ u_2^2 \end{pmatrix} \begin{pmatrix} v_1^2 \\ \sqrt{v_1 v_2} \\ v_2^2 \end{pmatrix} \rangle = \langle \Phi(\mathrm{u}), \Phi(\mathrm{v}) \rangle, \tag{3.17}$$

, z čehož získáme $\Phi(\mathrm{x}) = \left(\mathrm{x}_1^2, \sqrt{2}\mathrm{x}_1\mathrm{x}_2, \mathrm{x}_2^2\right)$.

Obecně lze dokázat, že pro každé jádro, které generuje pozitivní jádrovou matici $M_{ij} = K(x_i, x_j)$, platí, že $K(u, v) = \langle \Phi(u), \Phi(v) \rangle$. (Li, 2006)

Využitím jádrových funkcí transformujeme jednoduchý lineární klasifikátor na obecný a dobře použitelný nelineární klasifikátor. Jádrových funkcí existuje velké množství. V tabulce 3 je několik typů často používaných jádrových funkcí.

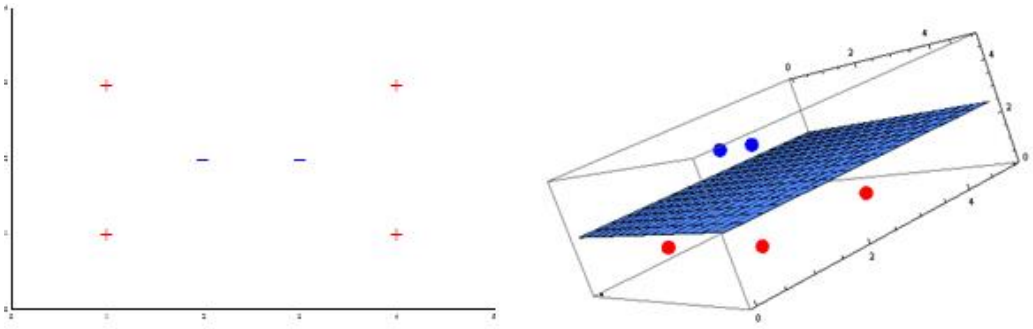
Tabulka 3: Vybrané jádrové funkce

Typ funkce	**Tvar jádrové funkce**
Lineární	$K(\mathbf{x_i}, \mathbf{x_j}) = \langle \mathbf{x_i}, \mathbf{x_j} \rangle + c$
Polynomiální	$K(\mathbf{x_i}, \mathbf{x_j}) = \left(\langle \mathbf{x_i}, \mathbf{x_j} \rangle + c\right)^d$
Gaussova (RBF)	$K(\mathbf{x_i}, \mathbf{x_j}) = exp\left[\frac{-\lVert \mathbf{x_i} - \mathbf{x_j} \rVert^2}{2\sigma^2}\right]$
Křivková	$K(\mathbf{x_i}, \mathbf{x_j}) = \prod_{k=1}^{10}\left(1 + x_{ik} * x_{jk} + \frac{1}{2} x_{ik} * x_{jk} \min(x_{ik}, x_{jk}) - \frac{1}{6}\min(x_{ik}, x_{jk})^3\right)$

Zdroj: Lam, 2012

V jádrových funkcích je *c* volitelná konstanta (často je volena 0), *d* je stupeň polynomu, σ je parametr Gaussova rozdělení.

Graf 6: Ukázka principu trasformace do „feature space“



Zdroj: vlastní zpracování, 2015

Zajímavým důsledkem jádrových transformací je skutečnost, že časová náročnost SVM algoritmu není závislá na dimenzi vstupního prostoru X. Řešení se vypočítá pouze ze skalárních součinů testovacích dat. (Nilsson, 2006)

4 Data a datové zdroje

Provedení experimentu, který by ověřil přesnost metody strojového učení Support Vector Machine, vyžaduje velké množství historických dat banky o udělených úvěrech spolu s informací, zda byly tyto úvěry řádně splaceny.

Jako kandidáti, kteří by mohli poskytnout podkladová data pro tuto práci, byly zvoleny tři tuzemské banky, které poskytují spotřebitelské úvěry.

Komerční banka, a.s., se sídlem Praha 1 poskytuje spotřebitelské úvěry bez zajištění až do výše 2 500 000 Kč. Banka vznikla v roce 1990 vyčleněním obchodní činnosti ze Státní banky československé. Komerční banka se mateřskou společností Skupiny KB a je členem mezinárodní skupiny Société Générale.

Komerční banka nabízí služby v oblasti retailového, podnikového a investičního bankovnictví. V roce 2013 měla Komerční banka 1,6 miliónu klientů a téměř 400 poboček. Služeb spotřebitelského financování, které nabízí společnost Essox, jakožto dcera společností Komerční banka, využívalo v téměř 300 000 klientů.

Komerční banka získala několikrát prestižní titul banka roku a nabízí spolupráci pro studenty pracující na bakalářských a diplomových pracích. Bohužel informace o klientech a jejich platební morálce jsou interními informacemi, které bance poskytují konkurenční výhodu, a proto požadovaná data nemohla pro účely této studie poskytnout.

Druhý pokus o získání potřebných dat byl proveden v Air Bank, která na tuzemském bankovním trhu působí od roku 2011. Podle testu serveru bankovnipoplatky.com poskytuje jeden z nejvýhodnějších spotřebitelských úvěrů na trhu. Air bank nabízí spotřebitelský úvěr bez poplatků.

Air bank je členem skupiny PPF, což je investiční skupina působící ve střední a východní Evropě. Tato relativně inovativní banka za první dva roky svého působení na českém trhu přilákala 200 000 klientů a na konci roku 2014 jejich počet přesáhl 300 000 a jejich počet pravděpodobně stále poroste, jelikož podle průzkumu agentury STEM/MARK by si 26 % dotázaných vybralo Air bank, pokud by si mělo znovu otevřít běžný účet.

Požadavek na poskytnutí trénovacích dat byl z kontaktního centra Air banky předán na příslušné oddělení, které požadavku ovšem odmítlo vyhovět.

Stejný požadavek jako na předchozí dvě bankovní instituce byl zaslán i do České spořitelny, která nabízí spotřebitelský úvěr bez zajištění až do výše 700 000 Kč, ke kterému nabízí pevnou úrokovou sazbu. Minimální výše poskytovaných spotřebitelských úvěrů je 100 000 Kč. V případě úvěru se zajištěním není jeho výše dopředu omezena, ale odvíjí se od požadavků klienta a jeho schopnosti úvěr splatit.

Česká spořitelna je největší bankou v České republice a byla založena již v roce 1825, to znamená, že je také nejstarší spořitelní institucí na českém území. Česká spořitelna je členem skupiny Erste Group a jejích služeb využívá více než 5 milionů klientů, mezi které patří fyzické osoby, ale i firmy nebo obce a města.

I zde však byl požadavek zamítnut se stejným vyjádřením jako u Komerční banky, že se jedná o citlivá interní data, která není možné poskytnout.

Jelikož žádná z vybraných bank nebyla ochotna poskytnout reálná historická data, budou v následujících kapitolách použita a zpracována pseudonáhodně vygenerovaná data.

Pokud by data byla bankou poskytnuta, pak by se pro provedení experimentu rozdělila na dvě sady. První sada dat by sloužila jako trénovací data pro vytvoření modelu a druhá sada dat by pak sloužila jako testovací data k ověření správnosti nastavení modelu.

4.1 Návrh struktury trénovacích dat

Banka se při poskytnutí spotřebitelského úvěru vystavuje úvěrovému riziku. Toto riziko se banka snaží snížit co nejkomplexnější analýzou klienta při dodržení přijatelných nákladů na tuto analýzu.

Lia, Shiue a Huang ve své práci (2006) sestavili seznam parametrů popisujících klienta, které zaujímají důležité postavení při hodnocení žádosti o úvěr. Většina z nich se shoduje se závěry z konzultace v České spořitelně, která sice nemohla poskytnout samotná data, ale byl diskutován úvěrový proces a hodnocení klienta z pohledu expertního pracovníka, z čehož vyplynuly nejdůležitější atributy z hlediska hodnocení klientovy schopnosti úvěr splatit.

Pro samotné vytvoření modelu na základě trénovacích dat je vhodné vzorová data předzpracovat, zakódovat a případně seskupit.

Do hodnocení klientovy schopnosti splácet by měl vstoupit jeho průměrný měsíční příjem za posledních šest měsíců, proti příjmům je nutné započítat průměrné pravidelné výdaje (splátky nájemného, splátky jiných úvěrů, pojistné, atd.). Dále je nutné posuzovat osobní údaje žadatele, tedy věk, pohlaví, vzdělání, rodinný stav, počet vyživovaných dětí a také místo bydliště.

Mezi osobními údaji stojí za pozornost především věk, kdy zvláště absolventi a obecně lidé do třiceti let mají velký problém se splácením spotřebitelských úvěrů, a proto Česká spořitelna velmi omezila udělování těchto úvěrů.

Velmi důležitým atributem popisujícím žadatele je jeho historie čerpaných bankovních produktů věřitelské instituce, kdy dobrá platební morálka v minulosti může být rozhodující z hlediska schválení žádosti.

Tabulka 4: Možné seskupení a zakódování informací o žadateli

průměrná výše příjmů	< 10 000 Kč	0	10 000 - 25 000 Kč	1	> 25 000 Kč	2		
výše měsíčních nákladů vzhledem k průměrným měsíčním příjmům	> 80 %	0	40 - 80 %	1	< 40 %	2		
věk	< 25 let	1	25 - 35 let	2	35 - 60 let	3	> 60 let	4
pohlaví	muž	0	žena	1				
rodinný stav	svobodný/á	0	ženatý/vdaná	1	rozvedený/á	2		
počet vyživovaných dětí	1	1	2	2	atd.			
bydliště	Zákonem stanovené číslo kraje.							
historická platební morálka klienta u věřitelské banky	žádná	0	špatná	-3	dobrá	3		
počet let v současném zaměstnání	< 1	0	1 až 3	1	3 až 10	2	> 10	3
vzdělání	ZŠ	0	SŠ	1	SŠ s maturitou	2	VŠ	3

Zdroj: vlastní zpracování, 2015

V předcházející tabulce (tabulka 4) je sestaveno možné zakódování informací o žadateli o spotřebitelský úvěr. V levém sloupci je popis atributu a za ním následuje reálná hodnota a její kódová hodnota. Například příjem 8 000 Kč bude mít v trénovacích datech hodnotu 0, stejně jako příjem 5 000 Kč. U atributu bydliště je kódovou hodnotou zákonem stanovené číslo kraje, tedy například Hlavní město Praha má hodnotu 1, Plzeňský kraj má hodnotu 4.

Aby data a výstupy metody byly zobrazitelné v dvourozměrném, popřípadě trojrozměrném prostoru, budeme v dalších částech práce používat upravený návrh struktury dat z tabulky 4. Budeme předpokládat, že banka, která data poskytla, hodnoty zakódovala do dvou skupin a výsledné ohodnocení klienta v každé skupině vyjádřila procentním podílem z maximálního možného počtu bodů ve skupině. Osobní údaje popisující klienta budou tvořit jednu hodnotu a finanční atributy popisující klienta druhou. Budeme předpokládat, že tento způsob popisu bude dostačující.

5 Wolfram Mathematica

Wolfram Mathematica je softwarový nástroj založený na symbolických výpočtech, vyvíjený společností Wolfram Research. Wolfram Research byla založena v roce 1987 Stephenem Wolframem, jenž začal pracovat na vývoji programu, který tvořil s týmem matematiků a programátorů. Společnost Wolfram Research je nyní celosvětově uznávanou společností, která je považována za průkopníky v oblasti softwaru určeného pro výpočty.

Software Mathematica je vlajkovým produktem společnosti Wolfram Research a jeho první verze byla vydána v roce 1988 a od té doby si získal miliony uživatelů, především díky svému uživatelsky příjemnému uživatelskému rozhraní a výkonnému jádru, které umožňuje provádět náročné, především symbolické, výpočty. Wolfram Mathematica kromě symbolických výpočtů umožňuje samozřejmě také programovat procedurálně a funkcionálně. (wolfram.com, 2015)

Wolfram Mathematica nalézá uplatnění při vědecko-technických výpočtech, čímž se stává využitelným pro matematiky, statistiky, analytiky a pro mnoho dalších vědecko-technických pracovníků. Wolfram Mathamatica není pouze softwarový nástroj určený pro tvorbu programů, je to zároveň jedna z největších sbírek algoritmů, ke kterým má její uživatel přístup a jejichž použití je detailně i s názornými příklady popsané v nápovědě softwaru.

Výhodné je již zmíněné rozdělení na uživatelské rozhraní (front end) a jádro systému Kernel. Front end slouží uživateli nejen k tvorbě programů a skriptů, ale také toto uživatelské rozhraní Wolframu Mathematica podporuje stylování a formátování textu a mnoho dalšího, takže lze Notebook (formát souboru) využít i jako podklad například pro prezentace. Notebooky je možné exportovat do jiných formátů (např. XML).

Jak již bylo zmíněno, první verze softwaru vyšla v roce 1988 a po updatu v roce 1989 tato verze podporovala řešení diferenciální rovnic, přesnou interpolaci polynomů, úlohy lineárního programování, grafický balík pro včetně trojrozměrného zobrazení grafů a mnoho dalšího.

První větší vylepšení systému přišlo v roce 1991, kdy byla zvýšena efektivita výpočtů lineární algebry, přibyla možnost detekce chyb a ladění vytvořeného kódu nebo třeba podpora zvukových efektů. Vydání verze 3 v roce 1996 přineslo zvýšení rychlosti

a snížení paměťové náročnosti pro běžnější operace. Následující verze, která vyšla v roce 1999, umožnila publikovat dokumenty ve volitelných formátech, přinesla možnost přímého importu dat, obrázků a zvuků nebo třeba lepší funkce pro analýzu dat. V roce 2006 verze 6 zvýšila rychlost importu a exportu nebo funkcí ListPlot, ListPlot3D a Plot3D pro velké soubory dat. Verze 8 z roku 2010 přinesla podporu pro výpočty finančních indikátorů. Verze 9, která byla využita v rámci této práce, přinesla zvýšení uživatelského komfortu díky Predictive Interface, které umožňuje automatické dokončování, zvýrazňování, rychlou nabídku možností pro práci s výstupy výsledů a další. Verze 10 z roku 2014 přinesla integraci s Wolfram cloud, podporu strojového učení nebo funkce pro konstrukce map.

Kód vytvořený v rámci této diplomové práce je napsán v programovacím jazyce Wolfram, respektive ve Wolfram Mathematica 9. Tato verze, která byla pro vypracování této práce k dispozici, jak vyplývá z přechozího odstavce s rychlým shrnutím jednotlivých verzí softwaru Mathematica, ještě neobsahuje podporu pro klasifikaci dat. Tato programová podpora je dostupná až od verze 10, proto pro předzpracování zdrojových trénovacích dat byla využita externí knihovna napsaná v jazyce Java. Její popis a použití je popsáno v následující kapitole.

Wolfram Mathematica obsahuje přes deset tisíc funkcí a některé z nich jsou přestaveny v následujících odstavcích nejen v jejich syntaktické podobě.

5.1 Export

Wolfram Mathematica je schopen automaticky exportovat do velkého množství datových souborů. Všechny formáty souborů, do kterých je možné export provést, lze zjistit výpisem `$ExportFormats` v prostředí Wolframu Mathematica, který vrátí ve verzi 9 následující seznam

```
{3DS,ACO,AIFF,AU,AVI,Base64,Binary,Bit,BMP,Byte,BYU,BZIP2,C,Character1
6,Character8,Complex128,Complex256,Complex64,CSV,DICOM,DIF,DIMACS,DOT,
DXF,EMF,EPS,ExpressionML,FASTA,FITS,FLAC,FLV,GIF,Graph6,Graphlet,Graph
ML,GXL,GZIP,HarwellBoeing,HDF,HDF5,HTML,Integer128,Integer16,Integer24
,Integer32,Integer64,Integer8,JPEG,JPEG2000,JSON,JVX,KML,LEDA,List,LWO
,MAT,MathML,Maya,MGF,MIDI,MOL,MOL2,MTX,MX,NASACDF,NB,NetCDF,NEXUS,NOFF
,OBJ,OFF,Package,Pajek,PBM,PCX,PDB,PDF,PGM,PICT,PLY,PNG,PNM,POV,PPM,PX
R,QuickTime,RawBitmap,Real128,Real32,Real64,RIB,RTF,SCT,SDF,SND,Sparse
6,STL,String,SurferGrid,SVG,SWF,Table,TAR,TerminatedString,TeX,Text,TG
A,TGF,TIFF,TSV,UnsignedInteger128,UnsignedInteger16,UnsignedInteger24,
UnsignedInteger32,UnsignedInteger64,UnsignedInteger8,UUE,VideoFrames,V
```

```
RML,VTK,WAV,Wave64,WDX,WMF,X3D,XBM,XHTML,XHTMLMathML,XLS,XLSX,XML,XYZ,
ZIP,ZPR}
```

Ze seznamu formátů, které je možné použít jako třetí parametr funkce `Export`, je vidět, že mezi exportními formáty se nacházejí například HTML, XLS, XML nebo v této práci používané PNG a PDF.

Samotná funkce `Export` je snadno použitelná. Často se používá formulace Export["file.ext",expr], kde prvním parametrem je název souboru s příponou *ext* a druhým je objekt určený k exportování.

Exportovat je možné také celý obsah Notebooku. To je možné příkazem Export["file.pdf", EvaluationNotebook[]], který provede výpočet celého Notebooku a následně jej vyexportuje do PDF souboru.

5.2 ListPointPlot3D

ListPointPlot3D je funkce Wolframu Mathematica, která slouží ke generování trojrozměrných grafů. První parametrem této grafické funkce je množina bodů, které je cílem zobrazit. Tyto body mohou být ve formátu seznamů reprezentujících souřadnice jednotlivých bodů ListPointPlot3D[{x_1, y_1, z_1},{x_2, y_2, z_2}, … , {x_n, y_n, z_n}] nebo také jako seznam několika datových kolekcí ListPointPlot3D[{$data_1$, $data_2$, … , $data_n$}]. Praktickým způsobem, jak vykreslit body dané funkce, je data uložit do struktury `Table` a následně tuto strukturu seznamů vykreslit pomocí funkce ListPointPlot3D.

```
Table[{Cos[i]Sin[i],Sin[i],0}+{0,0,i},{i,0,180Pi,.9}]
```

Tento příkaz vygeneruje jednotlivé body jako seznamy, jejichž jednotlivé hodnoty se budou řídit definovanými předpisy rovnic. Parametr na poslední pozici definuje parametry generování, v tomto případě budou do předpisů funkcí za proměnnou *i* dosazovány hodnoty od nuly do 180π se skokem o velikosti 0.9.

Vygenerovaná data v tomto případě mají následující podobu

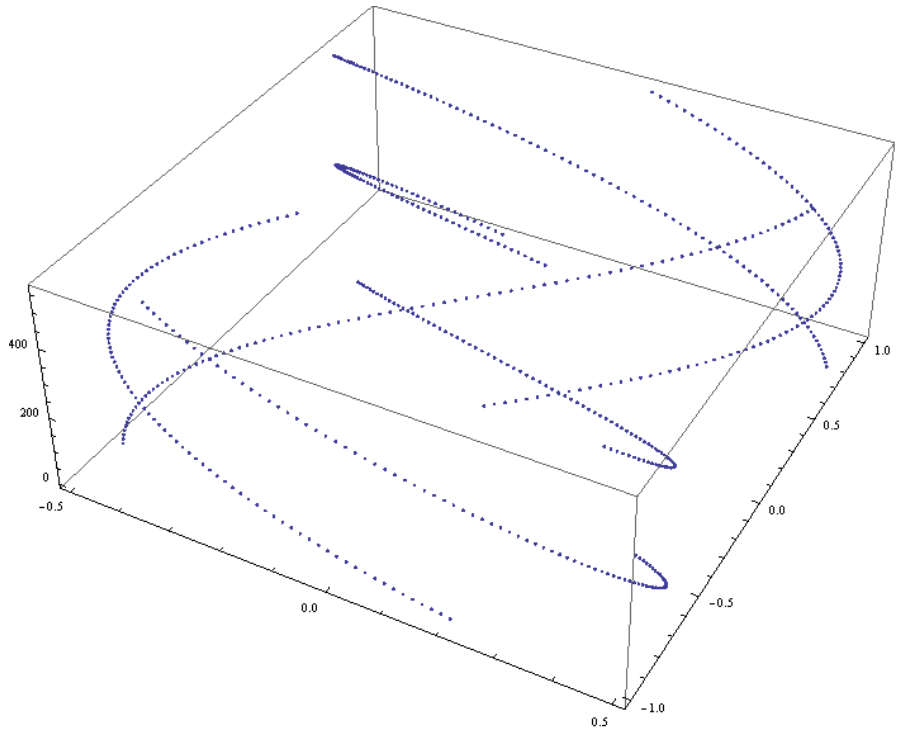
```
{{0.,0.,0.},{0.486924,0.783327,0.9},{-0.22126,0.973848,1.8},{-
0.386382,0.42738,2.7},{0.396834,-0.44252,3.6},{0.206059,-
0.97753,4.5},{-0.490468
…
{0.0740156,0.997242,560.7},{-0.498376,0.678034,561.6},
{0.152448,-0.154296,562.5},{0.429103,-0.869858,563.4},{-0.347434,-
0.927129,564.3},{-0.271227,-0.282767,565.2}}
```

Tato data jsou pak dosazena jako argument do funkce ListPointPlot3D

```
ListPointPlot3D[Table[{Cos[i]Sin[i],Sin[i],0}+{0,0,i},{i,0,180
Pi,.9}]]
```

Po spuštění tohoto zápisu jsou vykresleny vygenerované body v trojrozměrném grafu, který byl pomocí exportní funkce uložen do formátu PNG a je zobrazen na obrázku 4.

Obrázek 4: Příklad bodového grafu v softwaru Mathematica



Zdroj: Vlastní zpracování, 2015

5.3 Plot3D

Funkce Wolframu Mathematica Plot3D vykresluje předepsanou funkci jako třídimenzionální plochu. Povinnými parametry jsou předpis funkce a rozsahy proměnných *x* a *y*.

Po vykreslení plochy v Notebooku Wolframu Mathematica je možné se zobrazeným trojrozměrným grafem dále manipulovat. Rotovat plochou je možné pohyby kurzoru myši za současného přidržení levého tlačítka myši nad příslušným grafem. Také je možné graf přiblížit nebo naopak oddálit, to je možné přidržení klávesy „Ctrl“ a pohybem myši nahoru pro zvětšení a dolů pro zmenšení. Klávesa „Ctrl“ platí pro systémy Windows, pro Linux je to klávesa „Cmd“.

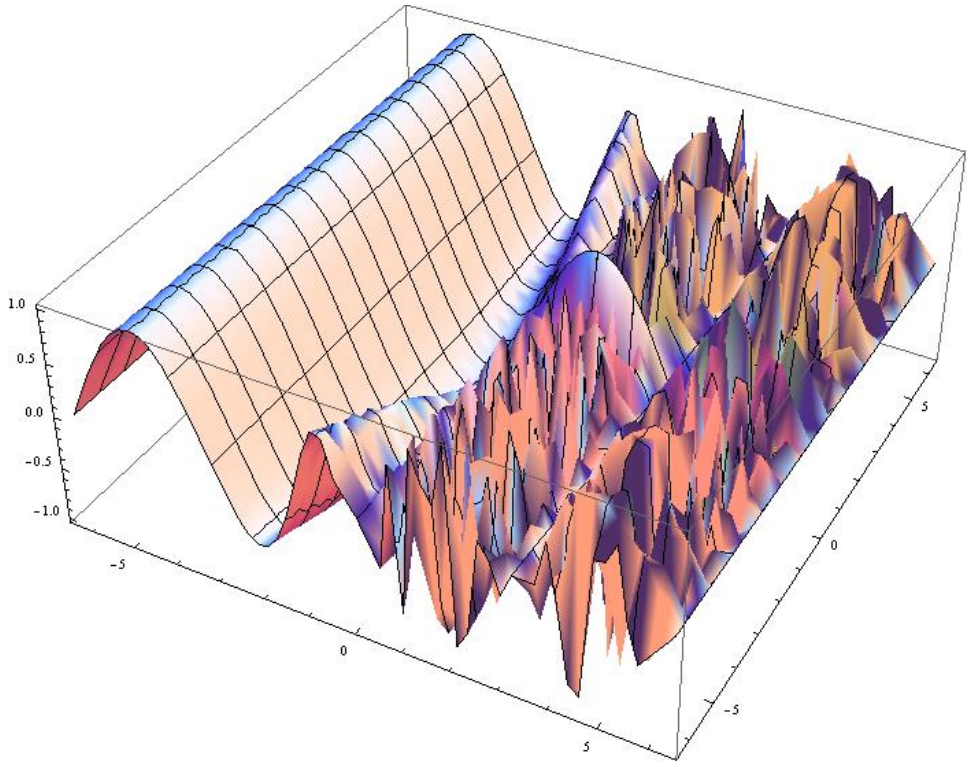
Wolfram Mathematica nabízí pro vizualizaci grafů řadu možností rozšíření základní podoby grafu. Pro Plot3D jsou těmito možnostmi například zobrazit nebo skrýt osy, definovat barvu povrchu plochy, zobrazit explicitní body nebo křivky, definovat rozsah zobrazení grafu a mnoho dalších.

Plochu ve Wolframu je tedy možné vykreslit příkazem Plot3D[*f*, {*x*, x_{min},x_{max}}, {*y*, y_{min}, y_{max}}].

```
Plot3D[Sin[x]*Cos[y*Exp[x]], {x, -2 Pi, 2 Pi}, {y, -2 Pi, 2 Pi}]
```

Tento případ plochy v trojrozměrném prostoru v softwaru Mathematica je zobrazen na obrázku 5.

Obrázek 5: Příklad funkce Plot3D ve Wolframu Mathematica



Zdroj: vlastní zpracování, 2015

U trojrozměrných grafů, které můžou být náročné z hlediska představení si jejich tvaru ve třech dimenzích na dvourozměrném monitoru, poskytuje Wofram Mathematica automatickou funkcionalitu, díky které ve vykreslené ploše dynamicky při rotování plochou mění barevné zobrazení a stínování tak, aby si uživatel co nejsnadněji představil reálný tvar vizualizované funkce.

5.4 Volby grafiky

Kromě funkcí nabízí Wolfram Mathematica také možnost upravovat konečnou podobou grafických objektů pomocí voleb. Jedná se vlastně o argumenty funkcí, které nejsou povinné.

Jednou z nich je například volba `PlotStyle`. Pomocí této volby lze nastavovat styl jednotlivých objektů a upravovat tak výchozí nastavení grafických objektů v softwaru Mathematica.

Syntaktický zápis je `PlotStyle -> g` nebo `PlotStyle -> {gᵢ}`, pokud je potřeba nastavit více příkazů stylování.

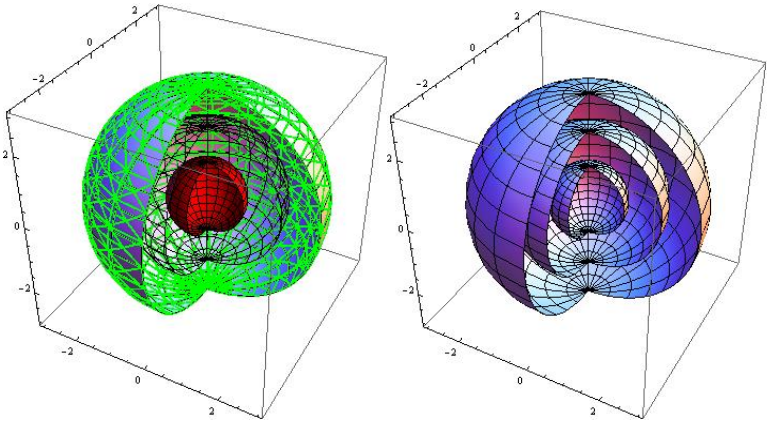
Mezi příkazy pro stylování vizualizací patří například:

- Dashing - definuje způsob přerušování přímek.
- EdgeForm – nastavuje styl okrajů obrazců.
- FaceForm – nastavuje styl výplně obrazců.
- Opacity – definuje úroveň průhlednosti objektu.
- PointSize – slouží k volbě velikosti zobrazených bodů.
- RGBColor – nastavuje barvu objektu pomocí modelu red-green-blue.
- Specularity – definuje úroveň zrcadlení obrazce.
- Thickness – nastavuje šířku křivek a čar.

```
SphericalPlot3D[{1, 2, 3}, {ϕ, 0, Pi}, {ϕ, 0, 3 Pi/2}
 , PlotStyle -> {Red, Opacity[0.5], EdgeForm[{Thick, Blue}]}]
```

Na předchozím úseku kódu ve Wolframu Mathematica jsou vyvedeny 3 kružnice s poloměry 1, 2 a 3, jejichž vzhled je upraven pomocí možností PlotStyle. Výstup je na obrázku 6, kde vlevo je výchozí nastavení Wolframu Mathematica a vpravo zobrazení ovlivněné parametry.

Obrázek 6: Vliv vybraných parametrů PlotStyle



Zdroj: vlastní zpracování, 2015

Volba `Mesh` slouží k zobrazení explicitních bodů vypočtených při generování dvoudimenzionálního grafu, respektive křivek v případě trojdimenzionálního. Tyto vypočtené body nebo křivky je možné stylovat pomocí nastavení `MeshStyle`.

Volba `Filling` přidává výplň pod křivky nebo body v grafech. V případě více překrývajících se křivek v jednom grafu dojde k automatickému zvýšení průhlednosti výplně tak, aby byly rozeznatelné jednotlivé křivky.

Pomocí voleb lze také explicitně určovat rozsah zobrazovaných dat pomocí `DataRange` nebo rozsah zobrazovaných os (`PlotRange`).

Voleb je v softwaru Mathematica velké množství a umožňují uživateli dobře pracovat s daty, respektive s jejich přehledným zobrazením.

6 Java a LIBSVM

Pro zpracování vstupních a zpracování vstupních dat bylo využito programovacího jazyka Java, tato data jsou dále zpracována pomocí knihovny externí LIBSVM, která vytváří model a poskytuje hodnoty koeficientů.

Java je objektově orientovaný programovací jazyk, který vychází z programovacího jazyka C. Java jako technologie, zahrnuje programovací jazyk a platformu. Java byla vyvinuta společností Sun Microsystems v roce 1995. Současným vlastníkem je Oracle Corporation, poté co proběhla akvizice těchto dvou společností.

Java je vyšší programovací jazyk, který je oblíbený především díky svojí přenositelnosti a nezávislosti na architektuře, což zajišťuje jednotný běh programu na různých operačních systémech.

LIBSVM je open source knihovna pro podporu výpočtů strojového učení. Tato knihovna byla vytvořena na Nation Taiwan University, autorská práva knihovny a zřeknutí se odpovědnosti za tuto knihovnu jsou uvedena v příloze D.

Pro běh této externí knihovny je potřeba Java verze 1.5 nebo vyšší.

Aktuální verze této knihovny je verze 3.20, která vyšla v listopadu roku 2014, a právě tato verze byla použita v programovém řešení při zpracování vstupních dat.

LIBSVM využívá pro vytvoření modelu metodu Sequential minimal oprimalization (SMO), což je algoritmus pro řešení úloh kvadratického programování.

Kromě klasifikace dat lze tuto volně šiřitelnou knihovnu využít pro regresní analýzu nebo hledání rozdělení vstupních dat.

Zdrojový kód v jazyce Java, který využívá LIBSVM knihovnu, byl implementován ve vývojovém prostředí Eclipse Java EE IDE for Web Developers ve verzi Mars Milestone 2. Verze jazyka Java byla použita Java SE 7.

7 Implementace algoritmů

Implementace algoritmů je rozdělena do dvou celků, které na sebe navazují. V první části, která je vytvořena v jazyce Java, dochází k načtení zdrojových dat, vytvoření modelu knihovnou LIBSVM a vyexportování podkladových souborů pro druhou programovou část. Ve druhé části pak dochází k reprezentaci výsledků, vizualizaci trénovacích dat modelu a vizualizaci rozhodovací funkce. Druhá část je vytvořena v jazyce Wolfram.

Pro zpracování vstupních dat a vytvoření modelu slouží Java třída CreateModel.java (příloha A). K vizualizaci výsledků byl v softwaru Mathematica vytvořen Notebook ShowModel.nb (příloha B).

Popis implementace algoritmů bude doprovázen modelovým příkladem, který bude využíván pro demonstraci fungování jednotlivý programových úseků, případně pro ukázky výstupů jednotlivých funkcí ve Wolframu Mathematica.

7.1 Zpracování vstupních dat

Pro zpracování vstupních dat a pro využití externí knihovny LIBSVM pro trénování modelu byla vytvořena třída CreateModel.java. Ta pracuje s trénovacími daty, která jsou uložena v souboru dataInput.csv.

Třída CreateModel.java obsahuje pouze jednu metodu určenou ke zpracování vstupních dat a vytvoření modelu. Metoda pracuje s polem vstupních argumentů, kde na prvním místě v pořadí je typ jádrové transformace. Metoda umožňuje tři transformace, a to lineární, polynomiální a RBF (Radial basis function, někdy také známa jako Gaussova). Pokud typ transformace není uveden, pak se defaultně uvažuje lineární typ transformace.

```
String kernelMetodType = args[0];
if (kernelMetodType == null) {
	kernelMetodType = "linear";
}
```

Po zjištění typu transformace se nastavují parametry modelu knihovny LIBSVM. Toto nastavení se provádí na instanci třídy svm_parameter. Mezi důležité parametry, které ovlivňují polohu separující nadroviny, patří `C`, což je konstanta ovlivňující váhu

chyby (viz rovnice 3.9) a pro Gaussovu transformaci je to parametr `gamma`, který ovlivňuje tvar křivky, respektive nadroviny.

```
svm_parameter param = new svm_parameter();
param.svm_type = svm_parameter.C_SVC;
param.C = 1000;

if (kernelMetodType.equals("linear")) {
       param.kernel_type=svm_parameter.LINEAR;
} else if (kernelMetodType.equals("polynomial")) {
       param.kernel_type=svm_parameter.POLY;
       param.coef0 = 0;
       param.gamma= 0.5;
       param.degree = 2;
} else if (kernelMetodType.equals("rbf")) {
       param.kernel_type=svm_parameter.RBF;
       param.gamma= 0.5;
       param.degree = 2;
       param.coef0 = 0;
}
```

Pro generování jednotlivých variant vstupních dat, které budou zpracovány v dalších částech této práce, jsou tyto parametry měněny přímo ve zdrojovém kódu. Proto je nutné program po změně vždy zkompilovat, o což se stará vývojové prostředí, ze kterého byl kód programu spouštěn.

V dalším pokračování programu je vytvořena instance třídy `svm_problem` (třída z knihovny LIBSVM), do které se vkládají trénovací data.

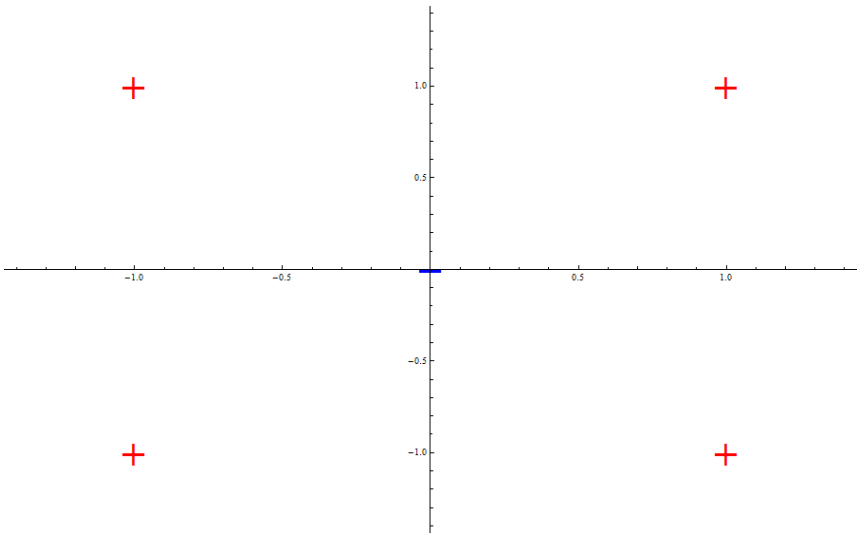
```
svm_problem p = new svm_problem();
```

Aby bylo možné trénovací data do této instance vložit, je nutné je nejprve zpracovat ze vstupního souboru. Soubor dataInput.csv s hodnotami oddělenými čárkou obsahuje řádky, kde každý z nich popisuje jeden vstupní bod modelu. Na poslední pozici v každém řádku se nachází hodnota 1 nebo -1, která popisuje příslušnost bodu do třídy. Tato data jsou množinou popsanou zápisem 3.3, poslední hodnota v řádku je y_i a předchozí hodnoty jsou vektorem **x**. Vstupní soubor může vypadat následovně

```
0,0,-1
-1,-1,1
1,-1,1
-1,1,1
1,1,1
```

Z hlediska možnosti vizualizace dat budeme pracovat se vstupními daty o dimenzí 2. Vstupní data modelového příkladu jsou vyobrazena v grafu 7.

Graf 7: Vizualizace vstupních dat modelového příkladu



Zdroj: vlastní zpracování, 2015

Tato data jsou v programu uložena do matice `pointList`, která reprezentuje tuto množinu vstupních vektorů a která je vytvořena z ArrayListů, aby byla možné dynamicky přidávat nové prvky. Statická pole totiž vyžadují definování jejich délky před samotným plněním data. V tomto případě je obtížné definovat počet prvků, jelikož počet řádků, které reprezentují jednotlivé vstupní entity (žadatele o úvěry), se může pro jednotlivé varianty trénování lišit.

Pro přečtení souboru je vytvořena instance `BufferReader`, což je standartní třída Javy sloužící ke čtení souborů. Při jejím vytvoření je v konstruktoru využita třída `FileReader`, která definuje cestu k souboru.

```
ArrayList<ArrayList<svm_node>> pointList = new
ArrayList<ArrayList<svm_node>>();
ArrayList<Double> classes = new ArrayList<Double>();
String csvFile = "dataInput.csv";
BufferedReader br = null;       String line = "";
String cvsSplitBy = ","; //separator
int pointCount = 0; //temporary number of data points
int dim = 0; //dimension of training data
 p.l = 0;
br = new BufferedReader(new FileReader(csvFile));
int iter = 0;
while ((line = br.readLine()) != null) {
      pointCount += 1;
      System.out.println("radka "+line);
```

```
            String[] dataLine = line.split(cvsSplitBy);
            for (int i = 0; i < dataLine.length;i++) {
                  if (i>dim) dim = i; }
                  if (i+1 < dataLine.length) { //point
                        pointList.add(new ArrayList<svm_node>());
                        svm_node node = new svm_node();
                        node.index = i;
                        node.value = new Double(dataLine[i
                        pointList.get(iter).add(node);
                  }else{ //class
                  classes.add(new Double(dataLine[i]));
            }
}
p.l += 1;
iter += 1;
}
```

V cyklu, který pokračuje, dokud zbývá v souboru další nepřečtený řádek, jsou data zpracována a uložena do matice `pointList`, respektive do pole `classes`, pokud se jedná o poslední hodnotu na řádku, která udává příslušnost do třídy. Načtení dalšího řádku ze souboru se vstupními daty zajišťuje metoda instance BufferReaderu `readLine`. Cyklus končí ve chvíli, kdy tato metoda vrátí „null“, což znamená, že nenalezla žádný další řádek ke čtení.

Načtená řádka je pomocí funkce split uložena do pole řetězců. Split je funkce, která podle předloženého parametru (v tomto případě čárka) rozdělí textový řetězec do pole.

Poté, co jsou data z jedné řádky rozdělena na jednotlivé hodnoty, započíná další cyklus, tentokrát cyklus `for`, ve kterém jsou data z konkrétní řádky zpracována. V tomto cyklu je zjištěna dimenze vstupních dat, poslední hodnota v řádce uložena do seznamu tříd (`classes`) a jednotlivé části vektoru popisující klienta jsou vloženy do seznamu `pointList`. Body jsou do tohoto hlavního seznamu vloženy také jako seznamy.

Takto zpracovaná vstupní data jsou následně přiřazena do již vytvořené instance `svm_problem`.

```
      p.l = pointCount;
      p.x = new svm_node[pointCount][dim];
      p.y = new double[pointCount];

      for (int i = 0; i < pointCount; i++) {
            p.y[i] = classes.get(i);
            for (int j = 0; j < pointList.get(i).size(); j++) {
```

```
                p.x[i][j] = new svm_node();
                        p.x[i][j].index = pointList.get(i).get(j).index;
                        p.x[i][j].value = pointList.get(i).get(j).value;
            }
        }
```

Instance třídy `svm_problem` vyžaduje ke správnému trénování modelu nastavit počet vstupních bodů (`p.l`), do matice `p.x` vložit vstupní body a do pole `p.y` přiřadit třídy těchto vložených bodů. Nastavení počtu trénovacích bodů a nastavení velikosti matice proběhne před systémem cyklů, jejich účelem je průchod dynamicky naplněné seznamů a vložení těchto dat do datových struktur knihovny LIBSVM.

Spouštění trénování modelu je provedeno pomocí

```
svm_model model = svm.svm_train(p, param);.
```

Tímto trénováním se vytvoří model, ze kterého lze čerpat informace o vypočtených atributech nebo vyhledaných podpůrných vektorech

Výsledky trénování modelu jsou uloženy do výstupního souboru javaData.dat, který je vstupem pro vizualizaci, která je popsána v následující kapitole. Výstupy jsou do souboru zapsány pomocí třídy `PrintWriter`.

```
PrintWriter writer = new PrintWriter("javaData.dat", "UTF-8");
```

Informace obsažené v souboru jsou vzorec pro výpočet jádrové transformace v závislosti na volbě z úvodu programu

```
if (kernelMetodType.equals("linear")) {
      writer.println("kernel[x_,y_]:=Sum[x[[i]]*y[[i]],{i,1,Length[x]}];");
else if (kernelMetodType.equals("polynomial")) {
      writer.println("kernel[x_,y_]:=Sum[(x[[i]]*y[[i]])^2,{i,1,Length[x]}];
      ");
} else if (kernelMetodType.equals("rbf")) {
      writer.println("kernel[x_,y_]:=Exp[-"+param.gamma+"*Sum[(x[[i]]-
      y[[i]])^2,{i,1,Length[x]}]];");
}
```

, hodnota koeficientu (biasu, posunutí) *b* ze vztahu 3.4

```
writer.println("b="+model.rho[0]*-1+";");
```

, seznam Lagrangeových koeficientů alfa vynásobených příslušnou hodnotou y_i

```
      writer.print("alphay={");
            for (int i = 0; i < model.SV.length; i++) {
                  if (i == 0) {
```

```
                        writer.print(model.sv_coef[0][i]);
                    }else {
                        writer.print(","+model.sv_coef[0][i]);
                    }
                }
        writer.println("};");
```

a seznam podpůrných vektorů, které byly při trénování dohledány metodou SMO.

```
writer.print("SVlist={");
  for (int i = 0; i < model.SV.length; i++) {
    if (i == 0) {
      writer.print("{");
        for (int j = 0; j < model.SV[i].length ; j++) {
           if (j == 0) {
             writer.print(model.SV[i][j].value);
           }else {
              writer.print(","+model.SV[i][j].value);
           }
         }
        writer.print("}");
     }else {
       writer.print(",{");
          for (int j = 0; j < model.SV[i].length ; j++) {
            if (j == 0) {
               writer.print(model.SV[i][j].value);
            }else {
       writer.print(","+model.SV[i][j].value);
      }}
       writer.print("}");
}}
writer.println("}");
```

Na závěr programu dojde k uzavření souboru javaData.dat příkazem

```
Writer.close();
```

7.2 Vizualizace modelu

Data vygenerovaná pomocí programu popsaného v přechozí kapitole 7.1 jsou vstupem pro skript v jazyce Wolfram. Kód je uložen souboru ShowModel.nb. Přípona *.nb je standardním datovým formátem softwaru Wolfram Mathematica.

Skript začíná příkazem `Clear["Global`*"];`, který vyprázdní veškeré proměnné v prostředí softwaru Mathematica. Proměnné ve Wolframu Mathematica totiž svojí platností přesahují Notebook, ve kterém jsou použity, proto bývá výhodné používané proměnné před spuštěním programu vyprázdnit.

Druhým krokem je nastavení cesty k souborům s původními vstupními daty. Cesta k souborům byla nastavena do adresáře, kde se nachází právě používaný Notebook Mathematica, z čehož vyplývá požadavek, aby tento adresář obsahoval dva požadované soubory ke správnému běhu Notebooku.

Soubory potřebné k běhu programu jsou vstupní data uložená v CSV souboru dataInput.csv, tedy stejná data, jaká jsou používána v Java třídě CreateModel.java. Druhým souborem je předzpracovaný model v souboru javaData.dat, který je výstupem třídy CreateModel.java.

Nastavení cesty do adresáře aktuálně používaného Notebooku se provede příkazem `SetDirectory[NotebookDirectory[]];`.

Po správném nastavení cesty k datovým souborům jsou načtena data z prvního souboru dataInput.csv. Tato data jsou uložena do proměnné `DataSet` příkazem `Import["dataInput.csv", "CSV"]`. V programovacím jazyce Wolfram není potřeba určovat typ proměnné, software Mathematica si ho určí za běhu sám. V tomto případě vytvoří seznam, jehož složky jsou další seznamy s daty z jednotlivých řádků souboru. Navážeme-li na příklad vstupních dat z kapitoly 7.1, pak by data načtená v softwaru Wolfram Mathematica měla následující podobu

```
In[1]:= DataSet
Out[1]= {{0, 0, -1}, {-1, -1, 1}, {1, -1, 1}, {-1, 1, 1}, {1, 1, 1}}.
```

Načtená data jsou pro lepší možnosti vizualizace zpracována do dvou seznamů `positiveData` a `negativeData`, do kterých jsou vstupní data roztříděna podle jejich příslušnosti do kladné nebo záporné třídy.

```
positiveData={};
negativeData={};
For[i=1,i< Length[DataSet]+1,i++,
  line = DataSet[[i]];
  point = Delete[line,3];

  If[line[[3]] > 0,
   positiveData = Append[positiveData,point]
   ,negativeData = Append[negativeData,point]
   ];
  ];
```

Oba seznamy jsou nejprve inicializovány jako prázdné, aby bylo možné do nich v cyklu přidávat prvky. Následně jsou procházeny jednotlivé prvky seznamu, ze kterého je

odstraněn poslední prvek, který značí zařazení do příslušné třídy, a zároveň je podle této hodnoty zbytek řádku zařazen do jednoho ze seznamů.

Seznamy obsažené v seznamech `positiveData` a `negativeData` jsou vektory popisující jednotlivá trénovací data. Seznamy v tomto modelovém případě vypadají takto

```
In[1]:= positiveData
In[2]:= negativeData

Out[1]= {{-1, -1}, {1, -1}, {-1, 1}, {1, 1}}
Out[2]= {{0, 0}}
```

Ze zpracovaných vstupních dat je vytvořena jejich grafická vizualizace. K tomuto účelu byl využit `ListPlot`, který je standardním grafickým nástrojem zabudovaným v softwaru Mathematica. Tento typ grafu uvažuje seznamy, které jsou pro něj vstupem, jako souřadnice bodů v rovinné kartézské soustavě souřadnic.

```
positiveDataGraph =
ListPlot[positiveData,PlotMarkers→{"+",25},PlotStyle→{Red}];
negativeDataGraph = ListPlot[negativeData,PlotMarkers→{"-
",25},PlotStyle→{Blue}];
```

Mathematica nabízí řadu nastavení, pomocí kterých lze upravit zobrazení dat. Pro zobrazení seznamů `positiveData` a `negativeData` bylo použito `PlotMarkers` a `PlotStyle`. `PlotMarkers` umožňují definovat vzhled značek v grafu, seznam `positiveData` je zobrazen značkami „+“ a `negativeData` značkami „-“. `PlotStyle` umožňuje obecné úpravy vzhledu grafu jako je barva vykreslených hodnot, jejich velikost, šířka čar a podobné.

Po zpracování vstupních dat je naimportován druhý soubor javaData.dat, který byl vygenerován třídou CreateModel.java a obsahuje informace potřebné k vizualizaci modelu. Tentokrát se nejedná o import data jako v případě souboru inputData.csv, ale je potřeba soubor přečíst jako kód napsaný ve Wolframu. K tomuto účelu slouží příkaz `<<javaData.dat;`. Tímto příkazem získáme seznam podpůrných vektorů (`SVlist`), seznam Lagrangeových multiplikátorů vynásobených klasifikací třídy (proměnná `alphay`), bias (proměnná `b`) a také jádrovou transformaci (proměnná `kernel`).

V případě modelových zdrojových dat, na kterých byly zatím jednotlivé příkazy demonstrovány, vypadá obsah souboru javaData.dat následovně

```
kernel[x_,y_]:=Sum[(x[[i]]*y[[i]])^2,{i,1,Length[x]}];
```

```
b=-1.74674125255883;
alphay={4.370023254081888,14.235105359588056,-8.38725935711482,-
10.217869256555124};
SVlist={{1.0,-1.0},{0.8,0.6},{0.25,0.5},{-0.6,0.2}}
```

V softwaru Mathematica je možné do grafů kromě funkcí a bodů přidávat také specifické tvary pomocí metody `Graphics`, která slouží k reprezentaci obrázků a práci s nimi. Jednou z funkcí je `Graphics` je `Circle`, která vykreslí na zadaných souřadnicích kruh o definované velikosti. Této vlastnosti softwaru Mathematica je využito pro vyznačení podpůrných vektorů. Jednotlivé definované kruhy jsou uloženy ve struktuře tabulky.

```
SVmarker =
  Table[Graphics[Circle[SVlist[[i]], 0.01]], {i, 1, Length[SVlist]}];
```

V dalším kroku programu je vytvořena funkce definující separující nadrovinu. Pro snadnější naprogramování funkce byl vztah 3.4 upraven dosazením za proměnnou vektoru $\mathbf{w}$ a následně vytknuto $\sum_{i=1}^{\ell} \alpha_i y_i$ tak, aby bylo možné dosadit do rovnice výpočet jádrové transformace. Výsledný tvar nadroviny pak v softwaru Mathematica je definován jako

```
separablePlane[x_] :=
  Sum[(alphay[[i]])*kernel[SVlist[[i]], x], {i, 1, Length[alphay]}] +
   b;,
```

kde `x_` je vektor, který do rovnice vstupuje jako neznámá. Jádro `kernel[SVlist[[i]], x]` je ze souboru javaData.dat. Výstup této rovnice pro polynomiální jádro stupně 2 z dat používaných v předchozím textu je následující

```
In[1]:= separablePlane[{x1, x2}]

Out[1]= -1.74674 - 10.2179 (0.36 x1^2 + 0.04 x2^2) -
 8.38726 (0.0625 x1^2 + 0.25 x2^2) +
 14.2351 (0.64 x1^2 + 0.36 x2^2) + 4.37002 (1. x1^2 + 1. x2^2)
```

Podle vztahu 3.4 se rovnice separující nadroviny musí rovnat nule. K vizualizaci této skutečnosti lze využít `ContourPlot`, která při správné konfiguraci zobrazí vrstevnici funkce, kde hodnota na ose *z* je nulová.

```
separablePlaneGraph =
  ContourPlot[
```

```
separablePlane[{x, y}], {x, -1.08, 1.08}, {y, -1.08, 1.08},
Contours -> {0}, ContourShading -> none,
ContourStyle -> {Green, Thickness[0.005]}];
```

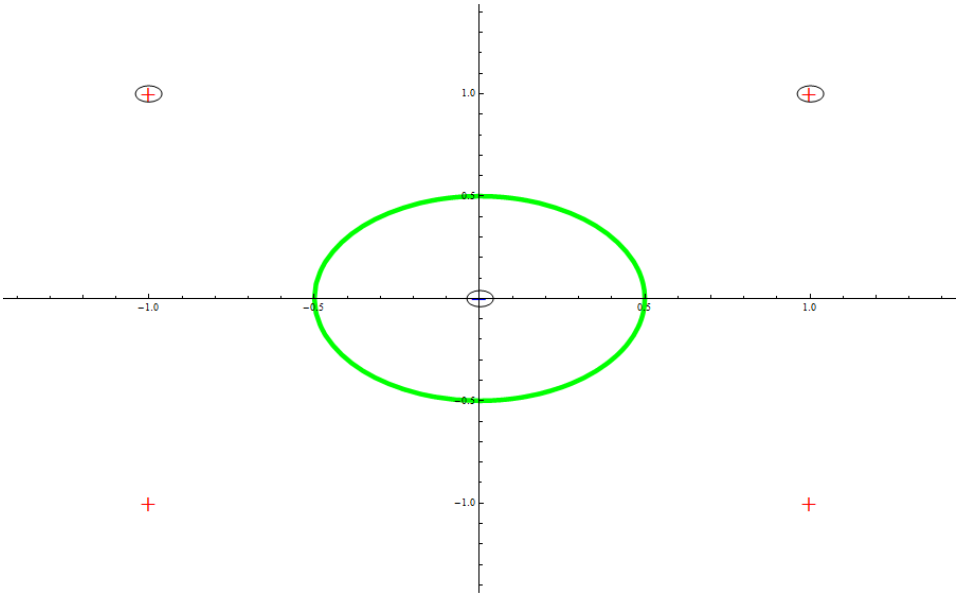
Parametry `ContourPlot` jsou funkce, jejichž vrstevnice mají být zobrazeny (`separablePlane[{x, y}]`), rozsah os *x* a *y* (`{x, -1.08, 1.08}` a `{y, -1.08, 1.08}`), důležitý parametr `Contours -> {0}`, který zajistí zobrazení pouze vrstevnice na hladině. Následují parametry, které ovlivňují vzhled vrstevnic. `ContourShading` pro nastavení stínování, a `ContourStyle` pro nastavení například barvy vrstevnic nebo jejich šířky.

Nyní máme k dispozici všechny součásti, které jsou potřebné k sestavení modelu. Jednotlivé dříve vytvořené grafy je v softwaru Wolfram Mathematica možné zobrazit do jedné kartézské soustavy souřadnic. K tomuto účelu je v softwaru Mathematica připravena funkce `Show`, do které lze jako parametry vložit jednotlivé grafické prvky, které byly v předchozích částech kódu vytvořeny, a zobrazit je všechny najednou. Také je možné ve výsledném, souhrnném, grafu provést formátování.

```
Show[positiveDataGraph, negativeDataGraph, separablePlaneGraph,
SVmarker, PlotRange -> {{-1.1, 1.1}, {-1.1, 1.1}},
 ImageSize -> {700, Automatic}, AxesOrigin -> {0, 0}]
```

`PlotRange` nastaví rozsah zobrazované oblasti grafu, ImageSize slouží pro konfiguraci velikosti obrázku grafu a `AxesOrigin` nastaví požadovaný průnik os. Pokud toto nastavení není provedeno, software Mathematica nastaví osy tak, aby byly co nejblíže zobrazovaným datům, a proto nemusí vždy procházet počátkem.

Graf 8: Souhrnný graf modelového příkladu



Zdroj: vlastní zpracování, 2015

Graf 8 je výstup modelu, jehož trénovací data obsahovala pět prvků, čtyři patřící do kladné třídy ([1,1], [-1,-1], [-1,1], [1,-1]) a jeden do záporné třídy (bod [0,0]). Trénování bylo spuštěno s polynomiálním jádrem stupně 2 a jako podpůrné vektory byly vyhodnoceny body [0,0], [1,1] a [-1,1].

```
In[1]:= SVlist
Out[1]= {{-1., 1.}, {1., 1.}, {0., 0.}}
```

Koeficienty modelu $\alpha_i y_i$, které jsou uloženy v poli `alphay`, nabyly hodnot

```
In[84]:= alphay
Out[84]= {2., 2., -4.}
```

Pro polynomiální jádrovou transformaci byla vytvořena trojrozměrná vizualizace trasformace dat a oddělovací nadroviny. Vstupní body jsou nejprve přepočteny podle vzorce 3.17 do prostoru o dimenzi 3.

```
positiveData3D = {};
negativeData3D = {};
For[i=1,i< Length[positiveData]+1,i++,
  line = positiveData[[i]];
  point3D = {line[[1]]^2,Sqrt[2]*line[[1]]*line[[2]],line[[2]]^2};
  positiveData3D = Append[positiveData3D,point3D];
  ];
For[i=1,i< Length[negativeData]+1,i++,
  line = negativeData[[i]];
  point3D = {line[[1]]^2,Sqrt[2]*line[[1]]*line[[2]],line[[2]]^2};
```

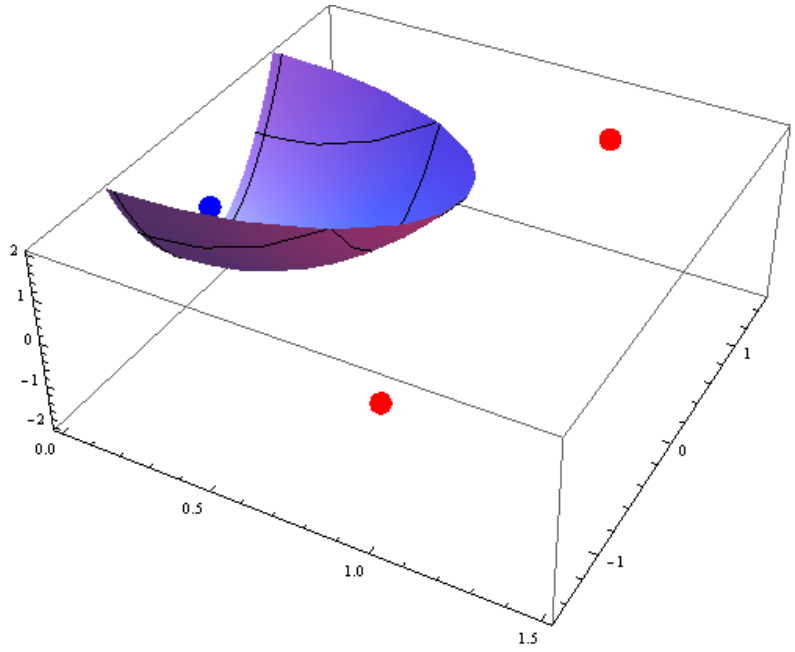
```
negativeData3D = Append[negativeData3D,point3D];
];
```

V cyklu `For` jsou z původních seznamů `positiveData` a `negativeData` vytvořeny seznamy `positiveData3D` a `negativeData3D` podle již zmíněného vzorce 3.17. Při výpočtu je použita funkce `Sqrt`, která vrací odmocninu předloženého čísla, a nový vektor je jako seznam uložen do proměnné `point3D`, který je následně přidán do seznamu, který byl inicializován před cyklem. Přidání je provedeno pomocí funkce `Append`, která na konec seznamu přidá novou hodnotu. Seznam, do kterého je hodnota přidávána, je definován jako první parametr funkce `Append`. Přidávaná hodnota je druhým parametrem této funkce.

Z takto upravených dat je následně sestaven `Module`. Výhodou `Module` je, že proměnné použité uvnitř tohoto prvku jsou lokální na rozdíl od proměnných v softwaru Mathematica obecně, které jsou vždy globální, a jejich platnost přesahuje rozsah Notebooku. V rámci modulu je možné definovat libovolný kód. Zde jsou v modulu vytvořeny trojrozměrné grafy positivních a negativních bodů. Zobrazit bod v trojrozměrném prostoru je možné pomocí `ListPointPlot3D`, kterému se jako parametr předkládá seznam bodů, a také je možné tento graf formátovat využítím `PlotStyle`. Kromě transformovaných bodů je také zobrazena oddělující nadrovina. Ta je zobrazena jako `Plot3D`, kterému se předloží funkce požadovaná k zobrazení, nastaví se rozsah osy x a osy y. Jednotlivé vytvořené grafy v `Module` jsou na závěr zobrazeny již známou funkcí `Show`. Celý tento modul je uložen do proměnné `MyPlot3D`.

Zobrazíme-li proměnnou `MyPlot3D`, získáme pohled na data ve feature space.

Graf 9: Data ve feature space



Zdroj: vlastní zpracování, 2015

Pro ostatní varianty je připraveno vykreslení separující nadroviny bez vizualizace dat, ze kterých byl model sestaven.

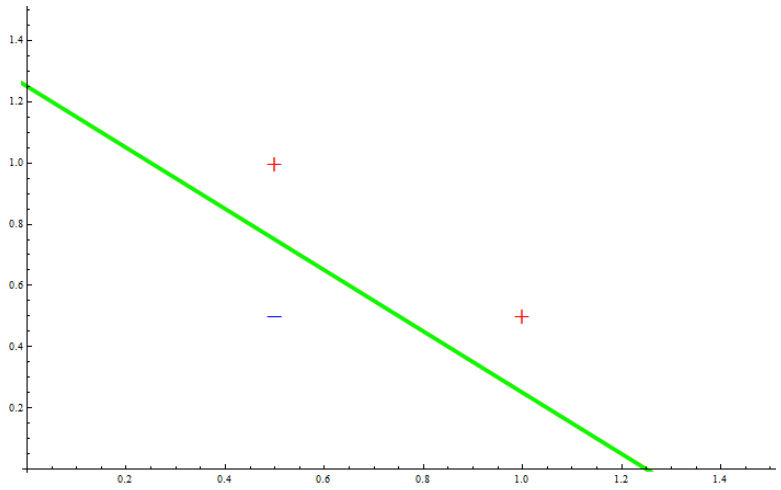
```
Plot3D[separablePlane[{x, y}], {x, -1.08, 1.08}, {y, -1.08, 1.08},
 PlotPoints -> 51, ImageSize -> {500, Automatic}]
```

Pro lineární variantu jádra je v Notebooku ShowModel.nb připravena alternativní varianta sestavení a výpočtu rovnice oddělující nadroviny. Tento postup je procedurální a je funkční pro lineární jádrovou transformaci s dvourozměrnými vstupními daty. Slouží k potvrzení správnosti prvního postupu sestavení rovnice pro separující nadrovinu.

```
w1 = 0;
w2 = 0;
For[i=1,i< Length[alphay]+1,i++,
  w1 = w1+ alphay[[i]]*SVlist[[i]][[1]];
  w2 = w2 + alphay[[i]]*SVlist[[i]][[2]];
  ];
separablePlane2[x_,y_] :=(w1*x + w2*y )+ b;
separablePlaneGraph2=ContourPlot[separablePlane2[x,y],{x,-10,20},{y,-
10,20},Contours->{0},ContourShading→none, ContourStyle→{Orange,
Thickness[0.005]}];
```

V cyklu `For` je vytvořen vektor **w**, respektive jeho složky w_1 a w_2 podle vztahu 3.8. Následně je vektor **w** dosazen do rovnice 3.4 a rovnice je uložena do proměnné `separablePlane2`, ze které je v dalším kroku vytvořen graf `separablePlaceGraph2`, stejně jako tomu bylo u původního postupu. Celý systém grafů je zobrazen pomocí `Show`, a to včetně obou rovnic separujících nadrovinu. Obě separující roviny se vzájemně překrývají.

Graf 10: Lineární oddělovací rovina (2. typ výpočtu)



Zdroj: vlastní zpracování, 2015

Vzájemný překryv obou přímek lze ověřit i porovnáním jejich rovnic, které nabízí software Wolfram Mathematica.

```
In[1]:= separablePlane2[x1, x2]
In[2]:= separablePlane[{x1, x2}]

Out[1]= -5. + 4. x1 + 4. x2
Out[2]= -5. - 16. (0.5 x1 + 0.5 x2) + 8. (1. x1 + 0.5 x2) +
 8. (0.5 x1 + 1. x2) = -5. + 4. x1 + 4. x2.
```

Výstupy grafické vizualizace jsou ve Wolframu Mathematica ukládány do samostatných souborů *.png. K tomu je použita funkce Export. Tato funkce vyžaduje minimálně dva parametry. Prvním je název souboru včetně příslušné přípony a druhým exportovaný objekt.

```
Export["graph1.png", MyPlot3D];
SystemOpen["graph1.png"]
```

Tyto dva příkazy uloží vytvořenou soustavu grafů do souboru graph1.png a následně tento soubor otevřou.

8 Zpracování a vizualizace variant

Kapitola zpracování a vizualizace jednotlivých variant vstupních dat využívá algoritmy popsané v přecházející kapitole.

Jelikož nebylo možné získat reálná data některé z bankovních institucí, jak bylo vysvětleno v kapitole 4, budou v následujících podkapitolách zpracována modelová data popisující klienty.

Zpracovány budou tři datové soubory, které reprezentují trénovací data banky A, B a C. Tato data byla pseudonáhodně vygenerována tak, aby bylo možné ověřit klasifikační možnosti metody Support Vector Machine. Vstupní data pro jednotlivé případy jsou vypsána v příloze C.

Kvůli absenci dostatečně velkého souboru reálných dat nebude možné provést testování zvolené jádrové transformace a nastavení parametrů modelu pomocí množiny testovacích dat. Obsahem této kapitoly bude zpracování vstupních dat, vytvoření modelu nad těmito daty a následná vizualizace v softwaru Wolfram Mathematica.

Demonstrován bude vliv nastavení parametrů modelu na výslednou polohu separující nadroviny, a to především konstanty *C*, která ovlivňuje poměr mezi přesností separace trénovacích dat a generalizací modelu.

8.1 Klasifikace klientů banky A

Banka A upravila návrh dat tak, že zakódovala atributy popisující žadatele o úvěr do dvou skupin. Do první skupiny zařadila osobní údaje klienta (věk, pohlaví, rodinný stav, bydliště, vzdělání) a do druhé jeho příjmy, výdaje, počet dětí, platební morálku a počet let v současném zaměstnání. Výslednou hodnotu popisující skupinu atributů nakonec vyjádřila její procentní částí z maximální hodnoty skupiny. Celý datový soubor je v příloze C, zde je pro znázornění struktury dat ukázka z tohoto souboru:

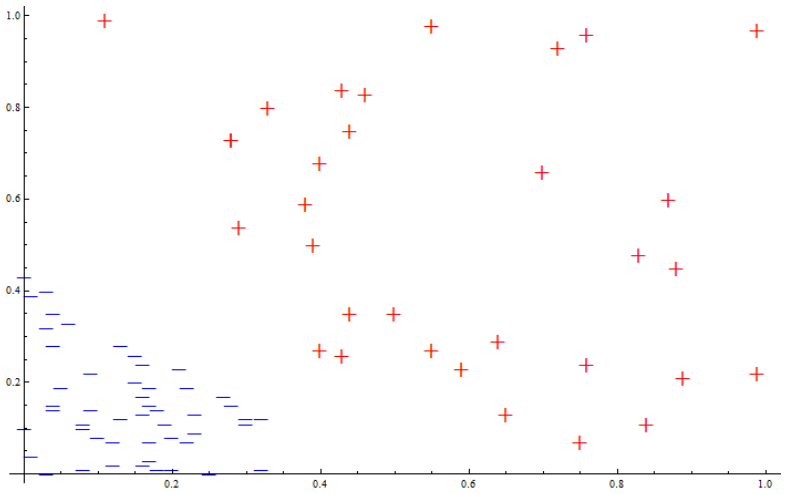
```
0.87,0.6,1
0.5,0.35,1
0.89,0.21,1

…

0.01,0.39,-1
0.05,0.19,-1
0.03,0.32,-1.
```

Výše zmíněná vstupní data, která jsou uložena v souboru dataInput.csv, zaneseme pomocí softwaru Mathematica do grafu 11. Klienti, kteří včas a v plné výši splatili svůj úvěr, jsou vyznačeni jako „+“. Klienti, kteří se při splácení svého spotřebitelského úvěru dostali do problému se svou schopností úvěr splácet, jsou označení jako „-“. Přiřazení do skupiny, jak již bylo popsáno dříve, zajišťuje třetí hodnota ze vstupního datového souboru.

Graf 11: Vstupní data, banka A



Zdroj: vlastní zpracování, 2015

Z grafu 11 je zřejmé, že data jsou lineárně separovatelná. K sestavení modelu byla využita metoda soft margin klasifikátoru, ovšem s vysoko nastavenou hodnotou parametru $C = 1000$, což znamená minimalizaci empirické chyby. Jak vyplývá z teoretické části této práce, pokud bychom parametr C položili roven nekonečnu, pak by se jednalo o maximum margin klasifikátor. Zvolená hodnota bude v tomto případě dostatečná k tomu, abychom dosáhli přesnosti separace blížící se maximum margin klasifikátoru.

Data byla zpracována třídou CreateModel.java, která vyhodnotila, že vstupní data obsahují čtyři podpůrné vektory, které můžeme zjistit výpisem v softwaru Mathematica z proměnné `SVlist`

```
{{0.4, 0.27}, {0.21, 0.23}, {0.32, 0.12}, {0.27, 0.17}}
```

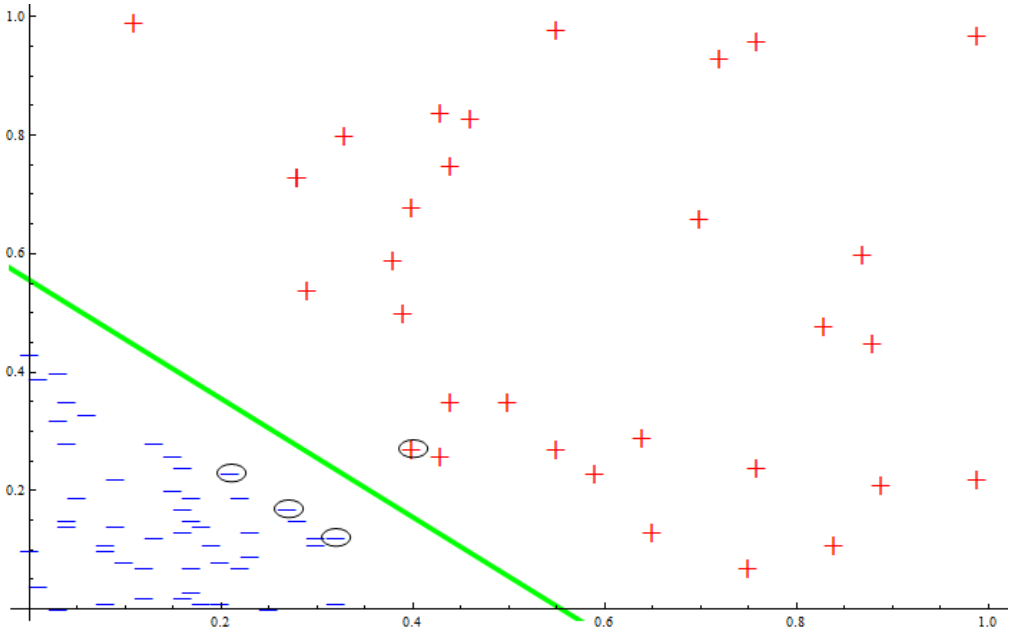
Separující nadrovina má podle výpisu následující podobu

```
-4.82616 - 45.4183 (0.32 x1 + 0.12 x2) - 11.2179 (0.27 x1 + 0.17 x2) -
18.9761 (0.21 x1 + 0.23 x2) + 75.6123 (0.4 x1 + 0.27 x2)
```

Hodnota -4,82616 je hodnota tzv. biasu (posunutí), hodnoty před závorkami jsou hodnotami součinu Lagrangeových multiplikátorů a příslušné třídy, do které patří podpůrný vektor. Tento podpůrný vektor je v závorce roznásobený neznámým vektorem. Neznámý vektor reprezentuje parametry nového žadatele o spotřebitelský úvěr. Pokud do této rovnice dosadíme za neznámý vektor hodnoty tohoto nového žadatele a vyjde nám výsledná hodnota kladná, pak je klient vyhodnocen jako bonitní, schopen úvěr splatit.

Výsledná poloha separující roviny je v grafu 12, včetně vyznačení podpůrných vektorů, které jsou zakroužkovány.

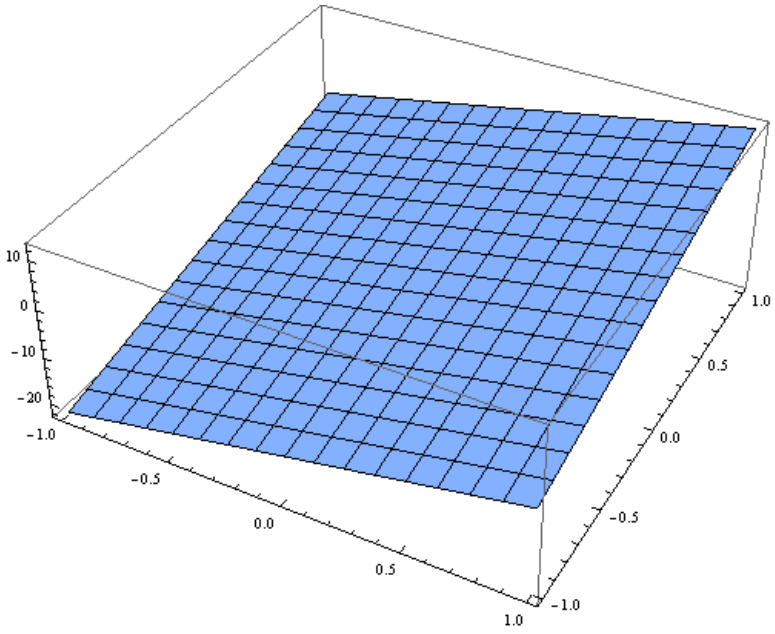
Graf 12: SVM model, banka A



Zdroj: vlastní zpracování, 2015

V grafu 13 je znázorněna poloha separující nadroviny, tentokrát v trojrozměrném prostoru.

Graf 13: Separující nadrovina, banka A



Zdroj: vlastní zpracování, 2015

8.2 Klasifikace klientů banky B

Také banka B zakódovala svá data do dvou skupin stejně jako banka A v předchozím případě.

Vstupní data jsou opět ve formátu CSV, první hodnota je ohodnocení první skupiny atributů (osobní údaje), druhá hodnota je hodnocení finančního hlediska žadatele a třetí je informace o tom, zda svůj spotřebitelský úvěr splatil, či nikoli.

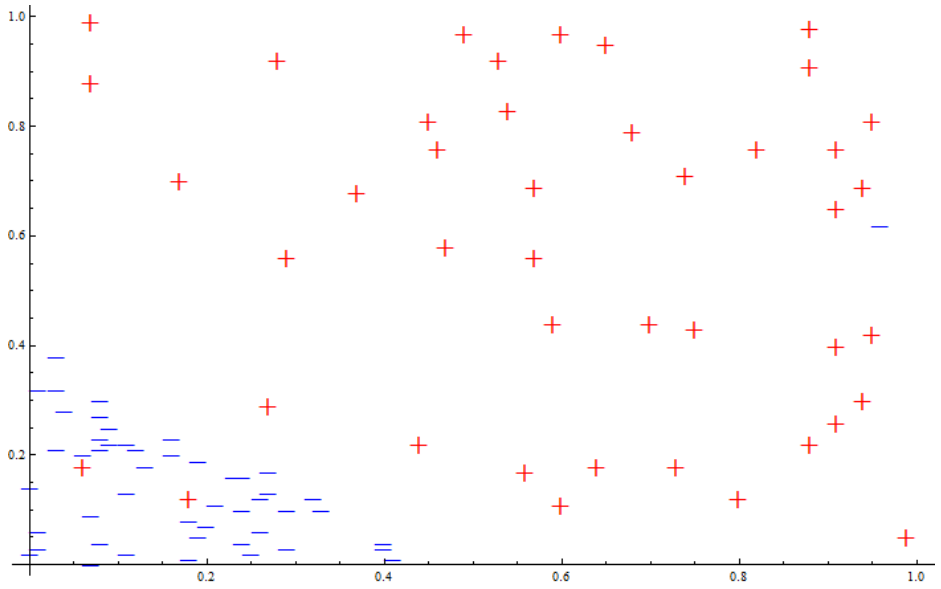
```
0.6,0.97,1
0.54,0.83,1
0.91,0.76,1

…

0.07,0.0,-1
0.27,0.17,-1
0.11,0.13,-1
```

Na grafu 14, kde jsou vstupní data žadatelů zobrazena, je možné vidět tři klienty, kteří opticky spadají do nesprávného prostoru.

Graf 14: Vstupní data, banka B

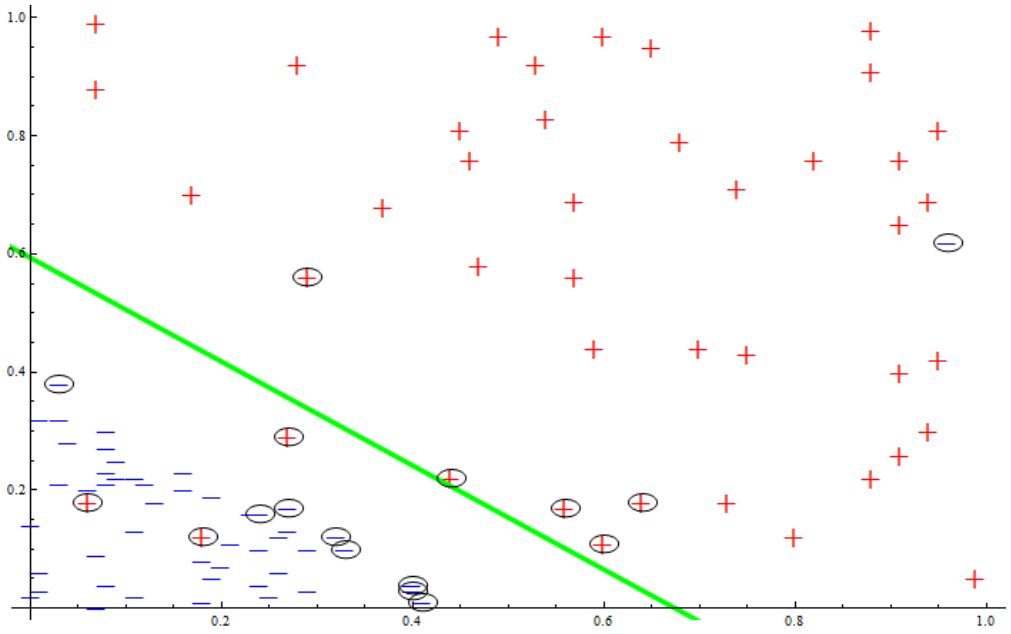


Zdroj: vlastní zpracování, 2015

Při klasifikaci tohoto typu vstupních dat je nutné správně zvolit parametr C, který určuje poměr mezi přesností a obecností výsledného modelu. Čím vyšší je zvolená hodnota parametru *C*, tím větší je přesnost klasifikace trénovacích dat, ovšem s nižší možností zobecnění výsledků.

Nejprve tedy zvolíme hodnotu $C = 100\,000$, což je vysoká hodnota, cílem je tedy vysoká přesnost klasifikace trénovacích dat. Výsledný model je zobrazen v grafu 15, kde vidíme, že pouze 3 klienti, kteří svůj úvěr řádně splatili, byli zařazeni do druhé skupiny klientů a jen jeden klient, který nesplatil, byl zařazen do skupiny řádně splácejících.

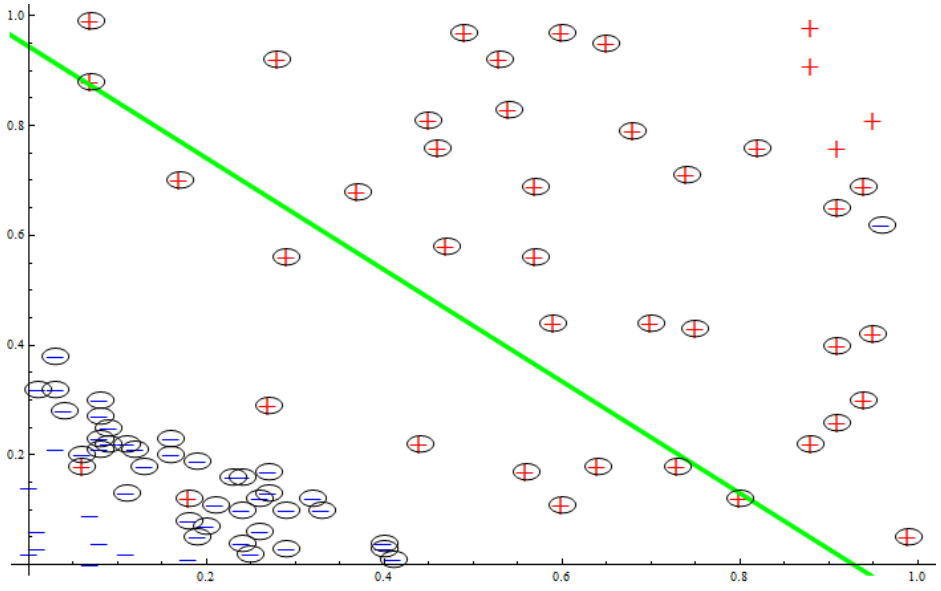
Graf 15: SVM model, C = 100000, banka B



Zdroj: vlastní zpracování, 2015

Na grafu 15 je ale také vidět, že hustota klientů, kteří svůj úvěr splatili bez potíží, je vyšší ve větší vzdálenosti od separující přímky. Dalo by se tedy předpokládat, že klienti patřící do kladné skupiny a nacházející se poblíž separující přímky, jsou výjimečné, extrémní, hodnoty. Proto nyní zvolíme parametr C na nízkou hodnotu 0,1, což by mělo přinést vyšší generalizaci výsledného modelu s nižší přesností klasifikace trénovacích dat.

Graf 16: SVM model, C = 0,1, banka B



Zdroj: vlastní zpracování, 2015

Při pohledu na graf 16, kde je vyobrazen model pro C = 0,1, je opravdu počet chybně klasifikovaných klientů vyšší, ovšem separující nadrovina modelu je položena tak, aby maximalizovala rozpětí mezi hustěji seskupenými body.

Určit, zda je výhodnější použít varianty C = 0,1, nebo C = 100000, by bylo možné pomocí testovacích dat. Testovací data je skupina klientů, o kterých již víme, zda svému závazku dostáli, či nikoli. Výslednou separující nadrovinu bychom aplikovali na tato testovací data a ověřili přesnost obou modelů. Testovací data by simulovala příchod nových klientů. O těchto klientech však již víme, zda spotřebitelský úvěr splatili, či nikoli, a mohli bychom proto porovnat rozhodnutí vzešlé z modelu sestaveného z trénovacích dat s reálnou hodnotou.

V případě nastavení parametru C na vysokou hodnotu bylo vyhodnoceno, že model obsahuje 17 podpůrných vektorů. To se promítne do délky rovnice separující nadroviny, která má podle exportu ze softwaru Mathematica tvar z obrázku 4.

Obrázek 7: Rovnice separují nadroviny, C=100000, banka B

```
-2.66533 - 33756.9 (0.41 x1 + 0.01 x2) - 100000. (0.4 x1 + 0.03 x2) -
 100000. (0.4 x1 + 0.04 x2) - 100000. (0.33 x1 + 0.1 x2) + 100000. (0.6 x1 + 0.11 x2) +
 100000. (0.18 x1 + 0.12 x2) - 100000. (0.32 x1 + 0.12 x2) - 1095.53 (0.24 x1 + 0.16 x2) -
 100000. (0.27 x1 + 0.17 x2) + 100000. (0.56 x1 + 0.17 x2) + 100000. (0.06 x1 + 0.18 x2) +
 100000. (0.64 x1 + 0.18 x2) + 100000. (0.44 x1 + 0.22 x2) + 100000. (0.27 x1 + 0.29 x2) -
 100000. (0.03 x1 + 0.38 x2) + 34852.4 (0.29 x1 + 0.56 x2) - 100000. (0.96 x1 + 0.62 x2)
```

Zdroj: vlastní zpracování, 2015

V druhém případě bylo vyhodnocení podpůrných vektorů ještě komplikovanější a výstupem je 74 podpůrných vektorů, výsledná podoba nadroviny je opět zobrazena exportem na obrázku 5.

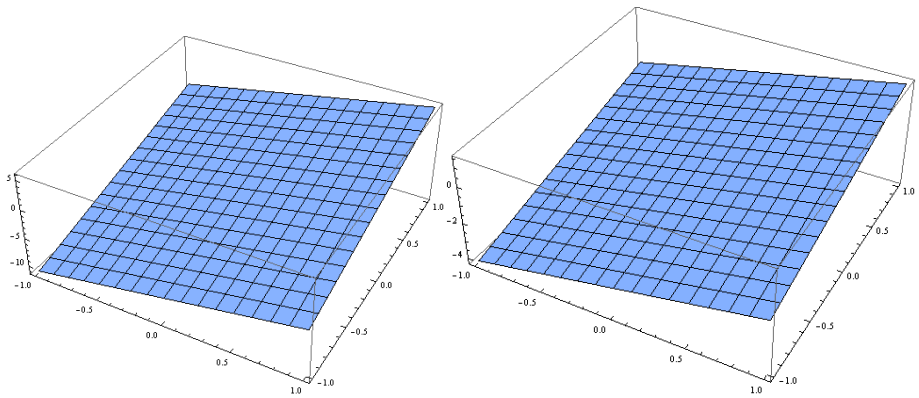
Obrázek 8: Rovnice separují nadroviny, C=0.1, banka C

$$
\begin{aligned}
&-1.34404 - 0.1\,(0.41\,x1 + 0.01\,x2) - 0.1\,(0.25\,x1 + 0.02\,x2) - 0.1\,(0.29\,x1 + 0.03\,x2) - \\
&0.1\,(0.4\,x1 + 0.03\,x2) - 0.1\,(0.24\,x1 + 0.04\,x2) - 0.1\,(0.4\,x1 + 0.04\,x2) - 0.1\,(0.19\,x1 + 0.05\,x2) + \\
&0.1\,(0.99\,x1 + 0.05\,x2) - 0.1\,(0.26\,x1 + 0.06\,x2) - 0.1\,(0.2\,x1 + 0.07\,x2) - 0.1\,(0.18\,x1 + 0.08\,x2) - \\
&0.1\,(0.24\,x1 + 0.1\,x2) - 0.1\,(0.29\,x1 + 0.1\,x2) - 0.1\,(0.33\,x1 + 0.1\,x2) - 0.1\,(0.21\,x1 + 0.11\,x2) + \\
&0.1\,(0.6\,x1 + 0.11\,x2) + 0.1\,(0.18\,x1 + 0.12\,x2) - 0.1\,(0.26\,x1 + 0.12\,x2) - 0.1\,(0.32\,x1 + 0.12\,x2) + \\
&0.1\,(0.8\,x1 + 0.12\,x2) - 0.1\,(0.11\,x1 + 0.13\,x2) - 0.1\,(0.27\,x1 + 0.13\,x2) - 0.1\,(0.23\,x1 + 0.16\,x2) - \\
&0.1\,(0.24\,x1 + 0.16\,x2) - 0.1\,(0.27\,x1 + 0.17\,x2) + 0.1\,(0.56\,x1 + 0.17\,x2) + \\
&0.1\,(0.06\,x1 + 0.18\,x2) - 0.1\,(0.13\,x1 + 0.18\,x2) + 0.1\,(0.64\,x1 + 0.18\,x2) + 0.1\,(0.73\,x1 + 0.18\,x2) - \\
&0.1\,(0.19\,x1 + 0.19\,x2) - 0.1\,(0.06\,x1 + 0.2\,x2) - 0.1\,(0.16\,x1 + 0.2\,x2) - 0.1\,(0.08\,x1 + 0.21\,x2) - \\
&0.1\,(0.12\,x1 + 0.21\,x2) - 0.1\,(0.09\,x1 + 0.22\,x2) - 0.1\,(0.11\,x1 + 0.22\,x2) + 0.1\,(0.44\,x1 + 0.22\,x2) + \\
&0.1\,(0.88\,x1 + 0.22\,x2) - 0.1\,(0.08\,x1 + 0.23\,x2) - 0.1\,(0.16\,x1 + 0.23\,x2) - 0.1\,(0.09\,x1 + 0.25\,x2) + \\
&0.1\,(0.91\,x1 + 0.26\,x2) - 0.1\,(0.08\,x1 + 0.27\,x2) - 0.1\,(0.04\,x1 + 0.28\,x2) + 0.1\,(0.27\,x1 + 0.29\,x2) - \\
&0.1\,(0.08\,x1 + 0.3\,x2) + 0.1\,(0.94\,x1 + 0.3\,x2) - 0.1\,(0.01\,x1 + 0.32\,x2) - 0.1\,(0.03\,x1 + 0.32\,x2) - \\
&0.1\,(0.03\,x1 + 0.38\,x2) + 0.1\,(0.91\,x1 + 0.4\,x2) + 0.1\,(0.95\,x1 + 0.42\,x2) + 0.1\,(0.75\,x1 + 0.43\,x2) + \\
&0.1\,(0.59\,x1 + 0.44\,x2) + 0.1\,(0.7\,x1 + 0.44\,x2) + 0.1\,(0.29\,x1 + 0.56\,x2) + 0.1\,(0.57\,x1 + 0.56\,x2) + \\
&0.1\,(0.47\,x1 + 0.58\,x2) - 0.1\,(0.96\,x1 + 0.62\,x2) + 0.1\,(0.91\,x1 + 0.65\,x2) + 0.1\,(0.37\,x1 + 0.68\,x2) + \\
&0.1\,(0.57\,x1 + 0.69\,x2) + 0.1\,(0.94\,x1 + 0.69\,x2) + 0.1\,(0.17\,x1 + 0.7\,x2) + 0.1\,(0.74\,x1 + 0.71\,x2) + \\
&0.1\,(0.46\,x1 + 0.76\,x2) + 0.1\,(0.82\,x1 + 0.76\,x2) + 0.1\,(0.68\,x1 + 0.79\,x2) + 0.1\,(0.45\,x1 + 0.81\,x2) + \\
&0.1\,(0.54\,x1 + 0.83\,x2) + 0.1\,(0.07\,x1 + 0.88\,x2) + 0.1\,(0.28\,x1 + 0.92\,x2) + 0.1\,(0.53\,x1 + 0.92\,x2) + \\
&0.1\,(0.65\,x1 + 0.95\,x2) + 0.1\,(0.49\,x1 + 0.97\,x2) + 0.1\,(0.6\,x1 + 0.97\,x2) + 0.1\,(0.07\,x1 + 0.99\,x2)
\end{aligned}
$$

Zdroj: vlastní zpracování, 2015

Podíváme-li se na trojrozměrné zobrazení těchto nadrovin, které je prakticky stejné, nadroviny se ale liší v poloze nulové vrstevnice, což je z hlediska modelu a vyhodnocování nových žadatelů o spotřebitelské úvěry rozhodující. Obě nadroviny jsou zobrazeny na obrázku 6.

Obrázek 9: Rovnice separující nadroviny, banka C, pro C=100000 (vlevo) a C=0.1 (vpravo)



Zdroj: vlastní zpracování, 2015

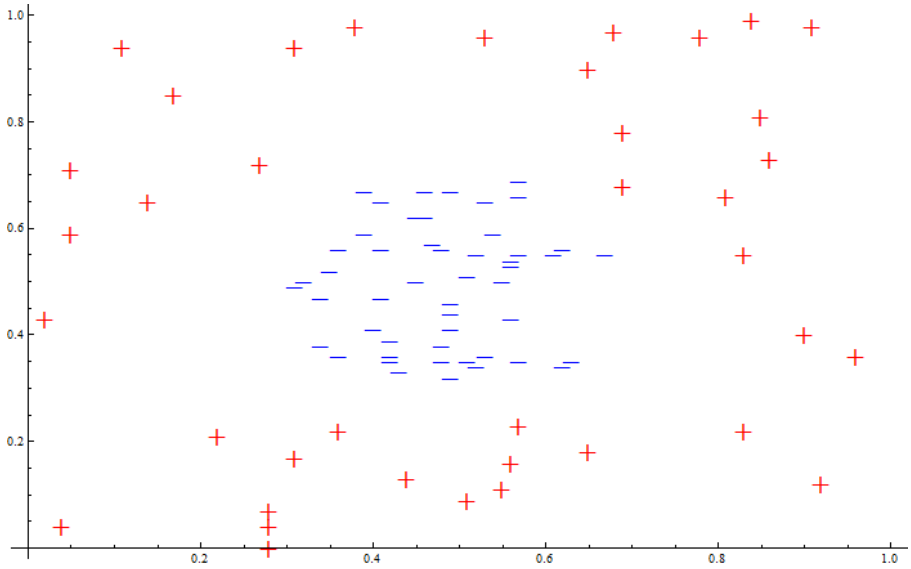
8.3 Klasifikace klientů banky C

I banka C seskupila své data o klientech, kteří již svůj poskytnutý úvěr dokázali splatit, nebo naopak splatit nedokázali, podle vzoru předchozích dvou bank. Na ose x se

nacházejí seskupené a zakódované osobní informace (věk, vzdělání, …) o již ukončených úvěrových vztazích a na ose y zakódovaný finanční popis tohoto klienta. Informace jsou uloženy do souboru dataInput.csv, včetně přiřazení do odpovídající skupiny podle toho, zda úvěr splatili, či nikoli.

Vstupní soubor zobrazený v kartézské soustavě souřadnice je v grafu 17.

Graf 17: Vstupní data, banka C



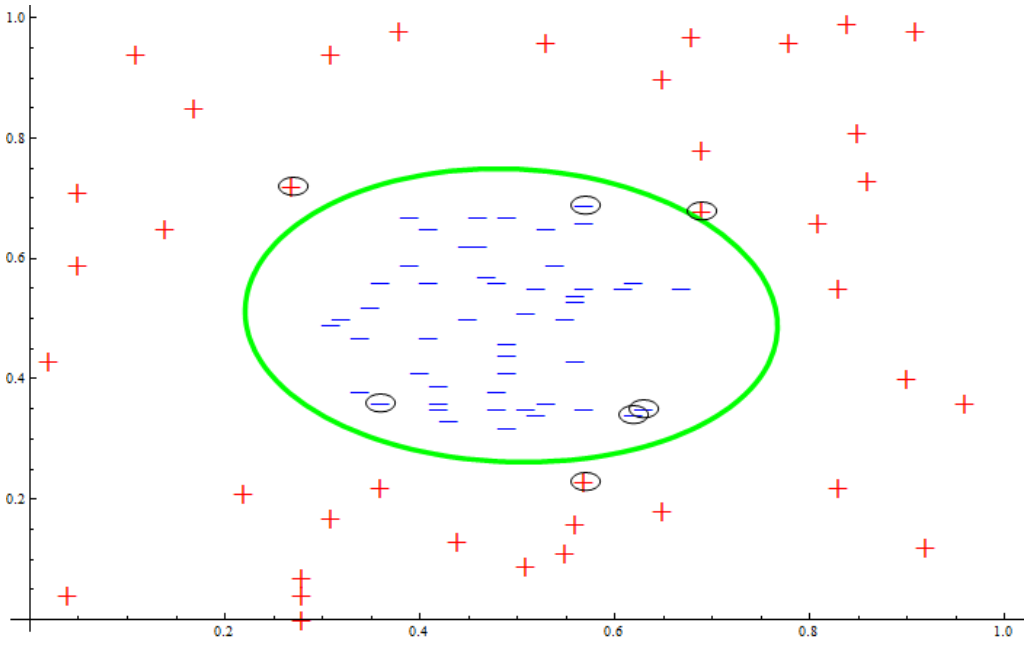
Zdroj: vlastní zpracování, 2015

Z předcházejícího grafu je patrné, že data nelze lineárně oddělit. Zároveň je ale také zřejmé, že data tvoří dva samostatné celky, kdy klienti, kteří nebyli schopni svůj úvěr splatit, jsou seskupeni uprostřed grafu a klienti, kteří byli schopni úvěr splatit, se nacházejí okolo nich.

K vytvoření modelu bude zvolena Gaussova jádrová transformace, tedy radial basis kernel (RBF). Tato metoda by měla být vhodná k separaci dat z grafu 17.

Pro vytvoření modelu nejprve zvolíme parametry, které významně ovlivňují polohu separující nadroviny. Výpočet trénování bude spuštěn s váhou chyby v klasifikaci C = 1000, model by tedy měl dvě skupiny dat oddělit s minimální chybou v klasifikaci. Dále nastavíme stupeň funkce a parametr gamma, který ovlivňuje tvar funkce. Stupeň bude nastaven na hodnotu 2 a gamma na hodnotu 0,5.

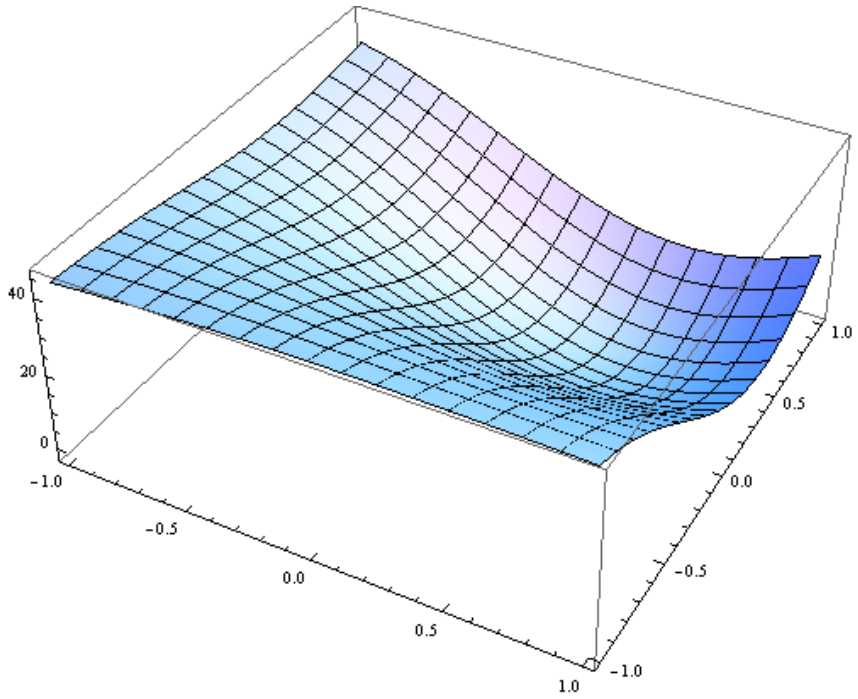
Graf 18: SVM model, banka C (C=1000, γ=0.5)



Zdroj: vlastní zpracování, 2015

S výše popsaným nastavením pro RBF transformaci jsou data oddělena bez chyby v klasifikaci (graf 18). Separující nadrovina v trojrozměrném prostoru je vyobrazena na grafu 19.

Graf 19: Separující nadrovina, banka C (C=1000, γ=0.5)



Zdroj: vlastní zpracování, 2015

Numerická podoba nadroviny z grafu 19 je ze softwaru Mathematica vyexportována do obrazku 10.

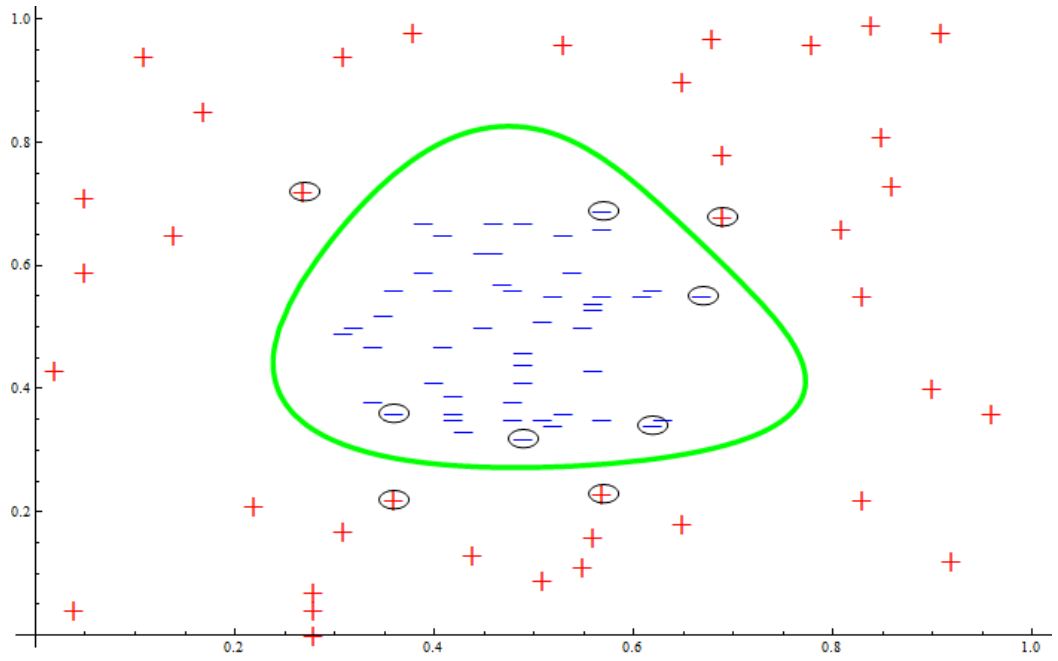
Obrázek 10: Rovnice separující nadroviny (C=1000, γ=0.5)

$$38.8875 + 1000.\, e^{-0.5\left((0.57-x1)^2+(0.23-x2)^2\right)} - 1000.\, e^{-0.5\left((0.62-x1)^2+(0.34-x2)^2\right)} - 142.362\, e^{-0.5\left((0.63-x1)^2+(0.35-x2)^2\right)} - 184.777\, e^{-0.5\left((0.36-x1)^2+(0.36-x2)^2\right)} + 1000.\, e^{-0.5\left((0.69-x1)^2+(0.68-x2)^2\right)} - 1000.\, e^{-0.5\left((0.57-x1)^2+(0.69-x2)^2\right)} + 327.139\, e^{-0.5\left((0.27-x1)^2+(0.72-x2)^2\right)}$$

Zdroj: vlastní zpracování, 2015

Při druhém trénování modelu nastavíme parametr γ na hodnotu 10. Vyšší hodnota parametru gamma by v kombinaci s vysokou hodnotou C (C = 1000) měla vést k vysoké přesnosti modelu, což by nemuselo být žádoucí, jelikož klesá generalizace modelu. Model s tímto nastavením je zobrazen v grafu 20.

Graf 20: SVM model, banka C (C=1000, γ=10)



Zdroj: vlastní zpracování, 2015

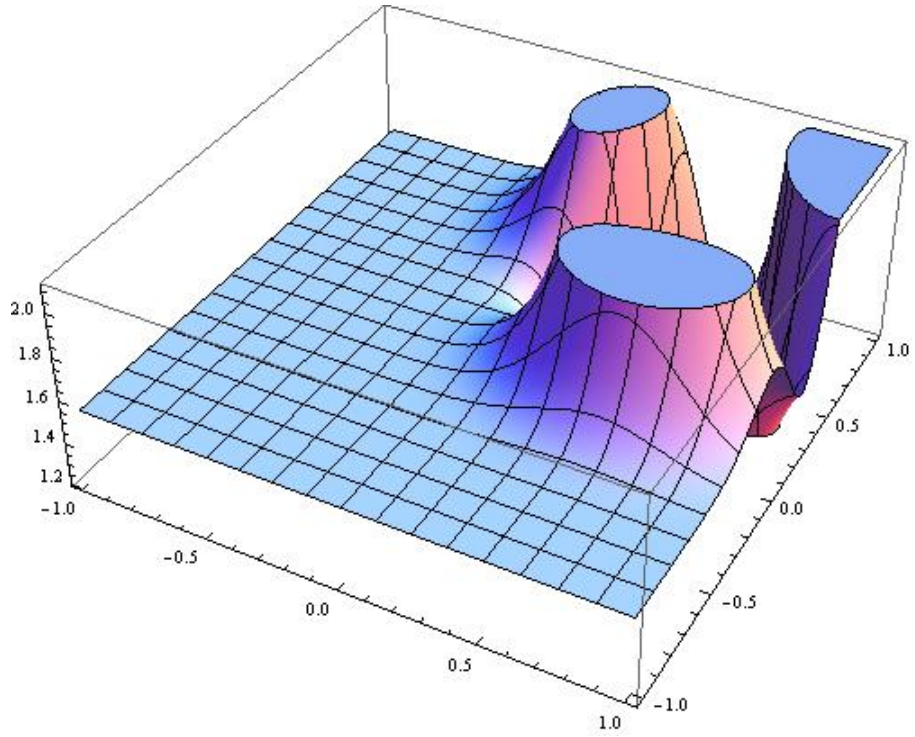
Nadrovina v číselné podobě byla vyexportována do obrázku 12 a její grafická podoba je zobrazena v grafu 21.

Obrázek 11: Rovnice separující nadroviny (C=1000, γ=10)

$$1.54759 + 0.532257\, e^{-10.\left((0.36-x1)^2+(0.22-x2)^2\right)} + 7.79228\, e^{-10.\left((0.57-x1)^2+(0.23-x2)^2\right)} - 3.92002\, e^{-10.\left((0.49-x1)^2+(0.32-x2)^2\right)} - 4.94565\, e^{-10.\left((0.62-x1)^2+(0.34-x2)^2\right)} - 0.658035\, e^{-10.\left((0.36-x1)^2+(0.36-x2)^2\right)} - 2.60868\, e^{-10.\left((0.67-x1)^2+(0.55-x2)^2\right)} + 10.6027\, e^{-10.\left((0.69-x1)^2+(0.68-x2)^2\right)} - 9.24977\, e^{-10.\left((0.57-x1)^2+(0.69-x2)^2\right)} + 2.45489\, e^{-10.\left((0.27-x1)^2+(0.72-x2)^2\right)}$$

Zdroj: vlastní zpracování, 2015

Graf 21: Separující nadrovina, banka C (C=1000, γ=10)



Zdroj: vlastní zpracování, 2015

9 Zhodnocení výsledků a možnosti další vývoje

Algoritmy, které byly vytvořeny podle teoretických poznatků popsaných v kapitole 3 a jejichž implementace byla objasněna v kapitole 7, byly aplikovány v kapitole 8 na tři sady vstupních dat. Pro jednotlivé případy byla zvolena vhodná jádrová transformace a případně testován vliv měnících se parametrů modelu na výslednou polohu oddělovací nadroviny.

Metoda Support Vector Machine ve všech případech dokázala nalézt separující nadrovinu, která měla oddělit bonitní klienty od těch, kteří patřili do skupiny problémových dlužníků.

Vzhledem k absenci reálných dat poskytnutých některou z bank nebylo možné provést pomocí křížové validace, tedy využití rozdělení poskytnutých dat na data trénovací a data testovací, dostatečné otestování algoritmů tohoto typu strojového učení.

Na základě kapitoly 8, ve které byla data zpracována a vizualizována, lze konstatovat, že metoda je schopna účinně separovat prvky patřící do různých tříd a při správné volbě jádrové transformace a vhodném nastavení parametrů modelu tak, aby byl vytvořen správný poměr přesnosti klasifikace v trénovacích datech a generalizaci modelu, by mohla být využita ke klasifikaci žadatelů o spotřebitelský úvěr.

Na tuto diplomovou práci by bylo možné navázat v několika směrech. Zde jsou stručně nastíněny tři základní.

V této práci bylo využito Wolframu Mathematica verze 9. Nejnovější verze tohoto softwaru, verze 10, přináší řadu rozšíření z hlediska integrovaných funkcí. Jednou z nových funkcionalit jsou možnosti strojového učení. Mathematica ve verzi 10 tedy umožní predikci podle trénovacích dat, ale také jejich klasifikaci. Nové funkce by měly být schopné klasifikovat numerické hodnoty, text nebo obrázky. Z toho vyplývá, že by ze současného řešení bylo možné vynechat programovou část implementovanou v jazyce Java včetně knihovny LIBSVM a celé řešení problému klasifikace žadatelů o spotřebitelský úvěr podle jejich schopnosti svým závazků dostát přenést do Wolframu Mathematica.

Wolfram Mathamatica je sice kvalitním a silným nástrojem pro technickovýzkumné výpočty, ale je nástrojem placeným. Proto v případě nedostupnosti novější verze Wolframu Mathematica by bylo možné zaměřit se na úlohy kvadratického

programování a případně vytvoření vlastní knihovny pro řešení úloh kvadratického programování. I tento postup by vedl k nahrazení externí volně šiřitelné knihovny LIBSVM. Vhodné by bylo zaměřit se na metodu Sequential Minimal Optimalization, která je v literatuře popisována jako rychlá a účinná metoda pro řešení rozsáhlých problémů kvadratického programování.

Třetím možným navázáním na tuto práci, které nevylučuje přechozí dvě varianty, je navázání užší spolupráce s některou z bankovních institucí a sestavení vstupních dat a trénování modelu podle reálných údajů o již uzavřených spotřebitelských úvěrech. V tomto případě by po sestavení modelu bylo možné provést testování pomocí sady testovacích dat a křížové validace. Toto by mohlo vést k jistému potvrzení, nebo vyvrácení možnosti využít metodu Support Vector Machine pro účely klasifikace žadatelů o úvěr podle jejich bonity.

10 Závěr

Trh spotřebitelských úvěrů utrpěl v posledních letech citelnou ránu v podobě globální finanční krize. Přesto je tento typ úvěru jedním z nejpopulárnějších a to i proto, že je prakticky nejjednodušším typem půjčky, která je poskytována na nepodnikatelské účely. Tento typ úvěru poskytují bankovní i nebankovní instituce. Pro nebankovní instituce ovšem není ohodnocování kreditního rizika takovým tématem, jakým je pro instituce bankovní. Ty jsou totiž zákony a nařízeními Basilejského výboru pro bankovní dohled tlačeny k minimalizaci tohoto kreditního rizika, a to především z důvodu udržení stability bankovního sektoru, které je jednou z důležitých podmínek udržitelného ekonomického rozvoje.

Jak bylo vysvětleno v textu práce, druhá z Basilejských dohod umožnila bankám interní měření úvěrového rizika. To umožňuje bankovním institucím sestavovat vlastní modely, založené na interních postupech a datech, k ohodnocení bonity žadatele o spotřebitelský úvěr. Výkonnost těchto interních metod kontroluje a vyhodnocuje stanovená národní autorita, v případě České republiky se jedná o Českou národní banku.

Cílem této práce bylo využití jedné z metod strojového učení, kterou vyvinul Vladimir Vapnik a která se nazývá Support Vector Machine, v překladu známá jako metoda podpůrných vektorů, k ohodnocení bonity žadatelů o spotřebitelský úvěr. Jedná se o klasifikační metodu, která se snaží minimalizovat strukturální riziko, a i proto má předpoklady pro dobré využití v hodnocení žadatelovy schopnosti splatit požadovaný spotřebitelský úvěr.

Práce je rozdělena do dvou částí. V první teoretické části je nastíněna problematika spotřebitelských úvěrů a důvody pro efektivní měření kreditní rizika. Těmito důvody jsou především nižší možná úroveň drženého kapitálu, která je bance povolena při kvalitním měření rizik, ale také vyšší objektivita a rychlost vyhodnocování ve srovnání s expertním posuzováním žádostí o spotřebitelské úvěry.

Po problematice spotřebitelských úvěrů je v teoretické části přikročeno ke strojovému učení a popisu jeho vybraných metod. Mezi tyto metody byla vybrána například metoda induktivních rozhodovacích stromů nebo neuronové sítě.

Hlavní pozornost teoretické části je však věnována metodě Support Vector Machine. U metody podpůrných vektorů je popsána varianta tzv. tvrdé klasifikace pomocí

maximum margin klasifikátoru, varianta uvažování chyby v klasifikaci (soft margin klasifikátor) a také využití jádrových transformací pro zefektivnění klasifikace.

Druhá část práce se pak věnuje implementaci algoritmů. Implementace byla provedena v jazyce Java, který zpracovává vstupní data a vytváří model, a následně ve Wolframu Mathematica, kde jsou vizualizovány jednotlivé funkce a výstupy modelu.

S implementací algoritmů souvisí také návrh struktury vstupních dat, kterému je věnována samostatné kapitola.

Výpočty pomocí implementovaných algoritmů byly provedeny nad pseudonáhodně vygenerovanými daty, jelikož poskytnutí reálných dat bankovní institucí pro vytvoření modelu nebylo možné. Banky potřebná data neposkytly se zdůvodněním, že se jedná o interní data, která bance poskytují konkurenční výhodu.

Na základě provedených pokusů s vygenerovanými daty lze však říci, že metoda Support Vector Machine je vhodným nástrojem pro klasifikaci dat a při správném nastavení trénování modelu by mohla být metoda využita pro klasifikaci žadatelů o spotřebitelský úvěr podle jejich schopnosti poskytnutý úvěr splatit.

11 Seznam tabulek

12 Seznam obrázků

13 Seznam grafů

14 Seznam použité literatury

Bank for International Settlements. *BCBS* [online]. [cit. 2015-01-22]. Dostupné z: http://www.bis.org/

BISHOP, Christopher M. *Pattern recognition and machine learning*. New York: Springer Science Business Media, 2009, xx, 738 s. ISBN 03-873-1073-8.

DUDA, Richard O. *Pattern classification*. 2nd ed. New York: J. Wiley, 2001, xx, 654 s. ISBN 04-710-5669-3.

DVOŘÁK, Petr. *Bankovnictví pro bankéře a klienty*. 3. přeprac. a rozš. vyd. Praha: Linde, 2005, 681 s. ISBN 80-720-1515-X.

HASTIE, Trevor, Robert TIBSHIRANI a J FRIEDMAN. *The elements of statistical learning: data mining, inference, and prediction*. 2nd ed. New York: Springer, 2009, xxii, 745 s. Springer series in statistics. ISBN 978-0-387-84857-0.

JOACHIMS, Thorsten. *Learning to classify text using support vector machines*. Boston: Kluwer Academic Publishers, 2002, xvi, 205 p. ISBN 9780792376798.

JUROŠKOVÁ, Lenka. *Bankovní regulace a dohled*. Praha, 2012, 174 pages. ISBN 80-872-8426-7.

KAŠPAROVSKÁ, Vlasta. *Řízení obchodních bank: vybrané kapitoly*. Vyd. 1. Praha: C. H. Beck, 2006, xix, 339 s. ISBN 80-717-9381-7.

KOMÁRKOVÁ, Zlatuše, Luboš KOMÁREK a Petr JAKUBÍK. *Zranitelnost českého bankovního sektoru*. Praha, 2012, 103 s. Studie (Národohospodářský ústav Josefa Hlávky), 10/2012. ISBN 978-808-6729-800.

KŘIVAN, Miloš. *Úvod do umělých neuronových sítí*. Vyd. 2., přeprac. Praha: Oeconomica, 2008. ISBN 978-80-245-1321-8.

LAM, Hak-Keung, S LING a Hung T NGUYEN. *Computational intelligence and its applications: evolutionary computation, fuzzy logic, neural network and support vector machine techniques*. Hackensack, NJ: Distributed by World Scientific Pub., c2012, x, 307 p. ISBN 18-481-6691-5.

LI, S, W SHIUE a M HUANG. The evaluation of consumer loans using support vector machines. *Expert Systems with Applications* [online]. 2006, vol. 30, issue 4, s. 772-782

[cit. 2015-04-12]. DOI: 10.1016/j.eswa.2005.07.041. Dostupné z: http://linkinghub.elsevier.com/retrieve/pii/S0957417405001739

LIBSVM: A Library for Support Vector Machines. CHANG, Chih-Chung. *Http://www.csie.ntu.edu.tw* [online]. [cit. 2015-01-22]. Dostupné z: http://www.csie.ntu.edu.tw/~cjlin/libsvm/index.html?js=1

MAŘÍK, Vladimír. *Umělá inteligence*. 1. vyd. Praha: Academia, 1993, 264 s. ISBN 80-200-0496-3.

NILSSON, Roland.A Flexible Implementation for Support Vector Machines. *The Mathematica Journal*. 2006, roč. 2006, č. 10 [cit. 2015-01-22]. Dostupné z: http://www.mathematica-journal.com/issue/v10i1/SupportVectorMachines.html

OSUNA, Edgar, Robert FREUND a Federico GIROSI. Support Vector Machines:Training and Applications. In: [online]. 1997 [cit. 2015-04-12]. Dostupné z: ftp://publications.ai.mit.edu/ai-publications/pdf/AIM-1602.pdf

PALÁNCZ, Béla a Lajos VÖLGYESI. Support Vector Classifier via Mathematica. [online]. 2005 [cit. 2015-04-12]. Dostupné z: http://www.agt.bme.hu/volgyesi/kiegyen/pp_class.pdf

STEEB, Willi-Hans. *The nonlinear workbook: chaos, fractals, cellular automata, neural networks, genetic algorithms, gene expression programming, support vector machine, wavelets, hidden Markov models, fuzzy logic with C , Java and SymbolicC programs*. 3rd ed. Singapore: World Scientific, 2005, xvii, 588 s. ISBN 98-125-6278-8.

TESKOVÁ, Libuše. *Lineární algebra*. 3. vyd. Plzeň: Západočeská univerzita, 2010, 243 s. ISBN 978-80-7043-966-1.

Wolfram Mathematica Documentation Center. *Wolfram* [online]. 2015 [cit. 2015-02-

15]. Dostupné na www: <http://reference.wolfram.com/mathematica/guide/Mathematica.html>

15 Seznam příloh

Příloha A : Třída CreateModel.java

Příloha B : Notebook ShowModel.nb

Příloha C : Trénovací data

Příloha D : Autorská práva LIBSVM

Příloha E : CD disk (CreateModel.java, LIBSVM, ShowModel.nb, zpracované varianty)

Příloha A : Třída CreateModel.java

```
package svm;

import java.io.BufferedReader;
import java.io.FileReader;
import java.io.IOException;
import java.io.PrintWriter;
import java.util.ArrayList;

import libsvm.svm;
import libsvm.svm_model;
import libsvm.svm_node;
import libsvm.svm_parameter;
import libsvm.svm_problem;

public class CreateModel {
	public static void main(String[] args) throws Exception{
		//take kernel type from input parameter or explicitly define
this parameter
		String kernelMetodType = args[0];
		if (kernelMetodType == null) {
			kernelMetodType = "linear";
		}
		// Preparing the SVM parameters
		svm_parameter param = new svm_parameter();
		param.svm_type = svm_parameter.C_SVC;
		param.nu = 0.5;
		param.cache_size = 40;
		param.C = 100;
		param.eps = 1e-3;
		param.p = 0.1;
		param.shrinking = 1;
		param.probability = 0;
		param.nr_weight = 0;
		param.weight_label = new int[0];
		param.weight = new double[0];
		if (kernelMetodType.equals("linear")) {
			param.kernel_type=svm_parameter.LINEAR;
		} else if (kernelMetodType.equals("polynomial")) {
			param.kernel_type=svm_parameter.POLY;
			param.coef0 = 0;
			param.gamma= 0.2;
			param.degree = 2;
		} else if (kernelMetodType.equals("rbf")) {
			param.kernel_type=svm_parameter.RBF;
			param.gamma= 0.5;
			param.degree = 2;
			param.coef0 = 0;
		}
```

```
            svm_problem p = new svm_problem();
            //parse input csv file
            ArrayList<ArrayList<svm_node>> pointList = new
ArrayList<ArrayList<svm_node>>();
            ArrayList<Double> classes = new ArrayList<Double>();
            String csvFile = "dataInput.csv";
            BufferedReader br = null;
            String line = "";
            String cvsSplitBy = ","; //separator
            int pointCount = 0; //temporary number of data points
            int dim = 0; //dimension of training data
            try {
                  //p.l = number of training data
                  p.l = 0;

                  br = new BufferedReader(new FileReader(csvFile));
                  int iter = 0;
                  while ((line = br.readLine()) != null) {
                        pointCount += 1;
                        System.out.println("radka "+line);
                        String[] dataLine = line.split(cvsSplitBy);

                        //read and save points
                        for (int i = 0; i < dataLine.length;i++) {
                              if (i>dim) {
                                    dim = i;
                              }
                              if (i+1 < dataLine.length) { //point
                                    pointList.add(new
ArrayList<svm_node>());
                                    svm_node node = new svm_node();
                                    node.index = i;
                                    node.value = new Double(dataLine[i]);

                                    pointList.get(iter).add(node);
                              }else{ //class
                                    classes.add(new Double(dataLine[i]));
                              }

                        }
                        p.l += 1;iter += 1;
                  }
            } catch (Exception e) {
                  e.printStackTrace();
            } finally {
                  if (br != null) {
                        try {
                              br.close();
                        } catch (IOException e) {
                              e.printStackTrace();
                        }
```

```
                }
            }

            p.l = pointCount;
            p.x = new svm_node[pointCount][dim];
            p.y = new double[pointCount];
            //init data for model
            for (int i = 0; i < pointCount; i++) {
                  p.y[i] = classes.get(i);
                  for (int j = 0; j < pointList.get(i).size(); j++) {
                        p.x[i][j] = new svm_node();
                        p.x[i][j].index = pointList.get(i).get(j).index;
                        p.x[i][j].value = pointList.get(i).get(j).value;
                        System.out.print(pointList.get(i).get(j).value+"
"+i+"/"+j+" ");
                  }
                  System.out.println();
            }
            svm_model model = svm.svm_train(p, param);

            PrintWriter writer = new PrintWriter("javaData.dat", "UTF-8");
            if (kernelMetodType.equals("linear")) {

      writer.println("kernel[x_,y_]:=Sum[x[[i]]*y[[i]],{i,1,Length[x]}];");
            } else if (kernelMetodType.equals("polynomial")) {

      writer.println("kernel[x_,y_]:=Sum[(x[[i]]*y[[i]])^2,{i,1,Length[x]}];
");
            } else if (kernelMetodType.equals("rbf")) {
                  writer.println("kernel[x_,y_]:=Exp[-
"+param.gamma+"*Sum[(x[[i]]-y[[i]])^2,{i,1,Length[x]}]];");
            }
            writer.println("b="+model.rho[0]*-1+";");
            writer.print("alphay={");
            for (int i = 0; i < model.SV.length; i++) {
                  if (i == 0) {
                        writer.print(model.sv_coef[0][i]);
                  }else {
                        writer.print(","+model.sv_coef[0][i]);
                  }
            }
            writer.println("};");
            writer.print("SVlist={");
            for (int i = 0; i < model.SV.length; i++) {
                  if (i == 0) {
                        writer.print("{");
                        for (int j = 0; j < model.SV[i].length ; j++) {
                              if (j == 0) {
                                    writer.print(model.SV[i][j].value);
                              }else {
```

```
		writer.print(","+model.SV[i][j].value);
					}
				}
				writer.print("}");
			}else {
				writer.print(",{");
				for (int j = 0; j < model.SV[i].length ; j++) {
					if (j == 0) {
						writer.print(model.SV[i][j].value);
					}else {

		writer.print(","+model.SV[i][j].value);
					}
				}
				writer.print("}");
			}
		}
		writer.println("}");
		writer.close();
	}
}
```

Příloha B : Notebook ShowModel.nb

```
(*clear all variables*)
Clear["Global`*"];
(*import data*)
SetDirectory[NotebookDirectory[]];
DataSet=Import["dataInput.csv","CSV"];
(*separate data according class*)
positiveData={};
negativeData={};
For[i=1,i< Length[DataSet]+1,i++,
  line = DataSet[[i]];
  point = Delete[line,3];

  If[line[[3]] > 0,
   positiveData = Append[positiveData,point]
   ,negativeData = Append[negativeData,point]
   ];
  ];
(*create graph of all input data*)
positiveDataGraph =
ListPlot[positiveData,PlotMarkers→{"+",25},PlotStyle→{Red}];
negativeDataGraph = ListPlot[negativeData,PlotMarkers→{"-
",25},PlotStyle→{Blue}];
(*import data generated from java code*)
<<javaData.dat;

SVmarker = Table[Graphics[Circle[SVlist[[i]],0.015]],{i,1,Length[SVlist]}];
separablePlane[x_]:=Sum[(alphay[[i]])*kernel[SVlist[[i]],x],{i,1,Length[alpha
y]}]+b;


separablePlaneGraph=ContourPlot[separablePlane[{x,y}],{x,0,1},{y,0,1},Contour
s->{0},ContourShading→none, ContourStyle→{Green, Thickness[0.005]}];

model2D =
Show[positiveDataGraph,negativeDataGraph,separablePlaneGraph,SVmarker,PlotRan
ge→{{0,1},{0,1}},ImageSize→{700,Automatic},AxesOrigin→{0,0}]
model3D=Plot3D[separablePlane[{x,y}],{x,-1,1},{y,-1,1},PlotPoints-
>51,ImageSize→{500,Automatic}]



(*kernelize data - just for polynomial case*)

positiveData3D = {};
negativeData3D = {};
For[i=1,i< Length[positiveData]+1,i++,
  line = positiveData[[i]];
  point3D = {line[[1]]^2,Sqrt[2]*line[[1]]*line[[2]],line[[2]]^2};
```

```
  positiveData3D = Append[positiveData3D,point3D];
  ];
For[i=1,i< Length[negativeData]+1,i++,
  line = negativeData[[i]];
  point3D = {line[[1]]^2,Sqrt[2]*line[[1]]*line[[2]],line[[2]]^2};

  negativeData3D = Append[negativeData3D,point3D];
  ];

MyPlot3D :=Module[{x1,x2,x3},
   x1 = Plot3D[separablePlane[{x,y}],{x,-5,5},{y,-5,5},PlotPoints-
>51,ImageSize→{500,Automatic}];
   x2
=ListPointPlot3D[positiveData3D,PlotStyle→{Directive[Red,PointSize[.03]]}];
   x3 =
ListPointPlot3D[negativeData3D,PlotStyle→{Directive[Blue,PointSize[.03]]}];
   Show[x1,x2,x3, PlotRange→ {{0,1.5},{-1.5,1.5},{-2,2}}]
   ];
MyPlot3D
(*second way to get svm equotation - just for linear case*)
w1 = 0;
w2 = 0;
For[i=1,i< Length[alphay]+1,i++,
  w1 = w1+ alphay[[i]]*SVlist[[i]][[1]];
  w2 = w2 + alphay[[i]]*SVlist[[i]][[2]];
  ];
separablePlane2[x_,y_] :=(w1*x + w2*y )+ b;
separablePlaneGraph2=ContourPlot[separablePlane2[x,y],{x,-10,20},{y,-
10,20},Contours->{0},ContourShading→none, ContourStyle→{Orange,
Thickness[0.005]}];
Show[positiveDataGraph,negativeDataGraph,separablePlaneGraph2,separablePlaneG
raph,PlotRange→{{0,1.5},{0,1.5}},ImageSize→{700,Automatic},AxesOrigin→{0,0}]


separablePlane[{x,y}]
```

Příloha C : Trénovací data

Banka A	Banka B	Banka C
0.87,0.6,1	0.6,0.97,1	0.91,0.98,1
0.5,0.35,1	0.54,0.83,1	0.65,0.9,1
0.89,0.21,1	0.91,0.76,1	0.11,0.94,1
0.83,0.48,1	0.45,0.81,1	0.69,0.78,1
0.55,0.27,1	0.8,0.12,1	0.31,0.17,1
0.99,0.97,1	0.88,0.22,1	0.86,0.73,1
0.28,0.73,1	0.99,0.05,1	0.31,0.94,1
0.44,0.35,1	0.56,0.17,1	0.05,0.71,1
0.39,0.5,1	0.49,0.97,1	0.22,0.21,1
0.7,0.66,1	0.91,0.65,1	0.17,0.85,1
0.59,0.23,1	0.91,0.26,1	0.9,0.4,1
0.64,0.29,1	0.28,0.92,1	0.55,0.11,1
0.72,0.93,1	0.57,0.69,1	0.83,0.55,1
0.99,0.22,1	0.07,0.88,1	0.78,0.96,1
0.76,0.96,1	0.37,0.68,1	0.02,0.43,1
0.33,0.8,1	0.82,0.76,1	0.57,0.23,1
0.46,0.83,1	0.57,0.56,1	0.56,0.16,1
0.55,0.98,1	0.27,0.29,1	0.83,0.22,1
0.43,0.84,1	0.96,0.62,-1	0.84,0.99,1
0.29,0.54,1	0.88,0.91,1	0.05,0.59,1
0.28,0.73,1	0.6,0.11,1	0.85,0.81,1
0.44,0.75,1	0.91,0.4,1	0.53,0.96,1
0.75,0.07,1	0.46,0.76,1	0.36,0.22,1
0.84,0.11,1	0.94,0.3,1	0.51,0.09,1
0.43,0.26,1	0.7,0.44,1	0.69,0.68,1
0.38,0.59,1	0.73,0.18,1	0.04,0.04,1
0.88,0.45,1	0.68,0.79,1	0.44,0.13,1
0.11,0.99,1	0.65,0.95,1	0.96,0.36,1
0.4,0.68,1	0.44,0.22,1	0.28,0.0,1
0.4,0.27,1	0.95,0.81,1	0.27,0.72,1
0.76,0.24,1	0.88,0.98,1	0.28,0.07,1
0.65,0.13,1	0.17,0.7,1	0.92,0.12,1
0.09,0.14,-1	0.59,0.44,1	0.14,0.65,1
0.21,0.23,-1	0.94,0.69,1	0.28,0.04,1
0.25,0.0,-1	0.47,0.58,1	0.65,0.18,1
0.01,0.39,-1	0.29,0.56,1	0.68,0.97,1
0.05,0.19,-1	0.75,0.43,1	0.81,0.66,1
0.03,0.32,-1	0.53,0.92,1	0.38,0.98,1
0.03,0.0,-1	0.07,0.99,1	0.42,0.36,-1
0.3,0.12,-1	0.74,0.71,1	0.41,0.56,-1
0.06,0.33,-1	0.64,0.18,1	0.49,0.41,-1
0.16,0.13,-1	0.95,0.42,1	0.45,0.62,-1
0.18,0.01,-1	0.04,0.28,-1	0.4,0.41,-1
0.0,0.1,-1	0.07,0.0,-1	0.54,0.59,-1
0.12,0.07,-1	0.27,0.17,-1	0.48,0.35,-1

0.12,0.02,-1	0.11,0.13,-1	0.49,0.44,-1
0.16,0.17,-1	0.12,0.21,-1	0.34,0.47,-1
0.16,0.24,-1	0.07,0.09,-1	0.42,0.39,-1
0.08,0.01,-1	0.01,0.32,-1	0.39,0.59,-1
0.32,0.01,-1	0.0,0.02,-1	0.35,0.52,-1
0.22,0.07,-1	0.11,0.02,-1	0.56,0.54,-1
0.08,0.1,-1	0.26,0.06,-1	0.56,0.43,-1
0.04,0.14,-1	0.08,0.3,-1	0.41,0.65,-1
0.13,0.12,-1	0.0,0.14,-1	0.49,0.46,-1
0.03,0.4,-1	0.4,0.03,-1	0.52,0.34,-1
0.04,0.15,-1	0.29,0.1,-1	0.57,0.66,-1
0.04,0.35,-1	0.23,0.16,-1	0.32,0.5,-1
0.2,0.08,-1	0.03,0.32,-1	0.46,0.67,-1
0.17,0.07,-1	0.29,0.03,-1	0.67,0.55,-1
0.1,0.08,-1	0.16,0.23,-1	0.62,0.56,-1
0.0,0.43,-1	0.4,0.04,-1	0.46,0.62,-1
0.32,0.12,-1	0.27,0.13,-1	0.31,0.49,-1
0.04,0.28,-1	0.08,0.04,-1	0.53,0.65,-1
0.18,0.14,-1	0.19,0.05,-1	0.48,0.38,-1
0.28,0.15,-1	0.21,0.11,-1	0.45,0.5,-1
0.16,0.02,-1	0.06,0.18,1	0.47,0.57,-1
0.19,0.11,-1	0.08,0.23,-1	0.57,0.69,-1
0.17,0.19,-1	0.18,0.08,-1	0.51,0.51,-1
0.13,0.28,-1	0.09,0.25,-1	0.48,0.56,-1
0.17,0.15,-1	0.33,0.1,-1	0.55,0.5,-1
0.09,0.22,-1	0.26,0.12,-1	0.36,0.56,-1
0.15,0.26,-1	0.24,0.1,-1	0.51,0.35,-1
0.22,0.19,-1	0.03,0.21,-1	0.62,0.34,-1
0.15,0.2,-1	0.18,0.01,-1	0.57,0.35,-1
0.17,0.03,-1	0.25,0.02,-1	0.39,0.67,-1
0.23,0.13,-1	0.16,0.2,-1	0.43,0.33,-1
0.08,0.11,-1	0.18,0.12,1	0.41,0.47,-1
0.01,0.04,-1	0.08,0.21,-1	0.36,0.36,-1
0.2,0.01,-1	0.08,0.27,-1	0.49,0.32,-1
0.27,0.17,-1	0.06,0.2,-1	0.49,0.67,-1
0.3,0.11,-1	0.03,0.38,-1	0.56,0.53,-1
0.23,0.09,-1	0.13,0.18,-1	0.42,0.35,-1
	0.32,0.12,-1	0.53,0.36,-1
	0.2,0.07,-1	0.63,0.35,-1
	0.19,0.19,-1	0.34,0.38,-1
	0.41,0.01,-1	0.52,0.55,-1
	0.01,0.03,-1	0.57,0.55,-1
	0.09,0.22,-1	0.61,0.55,-1
	0.01,0.06,-1	
	0.24,0.04,-1	
	0.24,0.16,-1	
	0.11,0.22,-1	

Příloha D : Autorská práva LIBSVM

Abstrakt

HARMADY, Jan. *Hodnocení spotřebitelských úvěrů pomocí SVM technika s využitím sw nástroje Mathematica.* Diplomová práce. Plzeň: Fakulta ekonomická ZČU v Plzni, 87 s., 2015

Klíčová slova: Support Vector Machine, SVM, bonita, strojové učení, spotřebitelský uvěr, klasifikace dat, Wolfram Mathematica

Prezentovaná diplomová práce se zabývá problematikou klasifikace žadatelů o spotřebitelský úvěr podle jejich schopnosti požadovaný úvěr splatit. V teoretické části je nastíněna problematika spotřebitelských úvěrů spolu s důvody pro efektivní měření úvěrového rizika. Dále jsou v práci popsány vybrané metody strojového učení, ze kterých je největší pozornost věnována technice podpůrných vektorů (Support Vector Machine). V praktické části jsou v softwaru Wolfram Mathematica a v programovacím jazyce Java implementovány algoritmy pro zpracování trénovacích dat a sestavení modelu s jeho následnou vizualizací. Pomocí implementovaných algoritmů jsou zpracovány různé tréninkové sady dat. V závěru práce jsou dosažené výsledky zhodnoceny a jsou doporučeny možnosti dalšího pokračování práce.

Abstract

HARMADY, Jan. *The evaluation of consumer loans using SVM technique using sw Mathematica.* Diploma thesis. Pilsen: Faculty of economics, University of West Bohemia, 87 p., 2015

Key words : Support Vector Machine, SVM, creditworthiness, machine learning, consumer loans, data classification, Wolfram Mathematica

The presented Diploma thesis deals with the classification of applicants for consumer loans according their creditworthiness. Theoretical part of presented work is focused on description of consumer loans, credit risk and reason for managing this risk. In the theoretical part, there are also described methods of machine learning with special attention on Support Vector Machine techniques. In the practical part of work, there are implemented algorithms using Wolfram Mathematica and Java programing language for processing training data, model creation and visualization of model, in particular. Implemented algorithms are used to classification of different data training sets. Conclusion contains evaluation of results along with recommendations for future work.