Univerzita Hradec Králové
Fakulta informatiky a managementu
KIKM

Geometrické základy vybraných dataminingových metod

Bakalářská práce

Autor: Eliška Rejmanová
Studijní obor: Aplikovaná informatika

Vedoucí práce: Mgr. Jiří Haviger, Ph.D.

Hradec Králové duben 2015

Prohlášení:

Prohlašuji, že jsem bakalářskou práci zpracovala samostatně a s použitím uvedené literatury.

V Hradci Králové dne 16.4.2015 Eliška Rejmanová

Poděkování:

Děkuji vedoucímu bakalářské práce Mgr. Jiřímu Havigerovi, Ph.D. za metodické vedení práce a cenné rady a připomínky při jejím zpracovávání.

Anotace

Tato práce se zabývá popisem geometrických základů vybraných dataminingových metod. Pro tuto práci byly vybrány tyto metody: nejbližší sousedi, K-nejbližších sousedů, support vector machines a logistická regrese.
První část práce je zaměřená na vysvětlení principu všech čtyř klasifikačních metod a také je zde popsána jejich implementace v jazyce R.
V druhé části práce jsou pak popsané metody aplikovány na různě velké datové soubory na dvou různých počítačích a následně jsou porovnávány výsledky jednotlivých metod a jejich časová náročnost.

Annotation

Title: Geometric principles of selected data mining methods

This thesis focuses on description of geometric principles of selected data mining methods. For this thesis were chosen following methods: Nearest neighbors, K-nearest neighbors, Support vector machine and Logistic regression.
The first part of this thesis concentrates on description of all of these methods and also on their implementation in language R.
In the second part are previously described methods applied on different datasets on two different computers. Subsequently, the results of these experiments are compared with each other, including their time consumption.

Obsah

Seznam obrázků

Seznam tabulek

1 Cíl práce

Tato práce se zaměřuje na popis vybraných klasifikačních algoritmů, patřících do kategorie učení s učitelem a jejich využití při analýze dat. Jejím cílem je seznámit s principem fungování daného algoritmu a následně ukázat jeho implementaci v jazyce R.

V první části této práce budou popsány metody nejbližší sousedi, K-nejbližších sousedů a support vector machines s využitím jádra. Po vysvětlení každé z metod následuje popis její implementace – soupis a vysvětlení příkazů použitých k zadání požadovaných metod do prostředí jazyka R.

Ve druhé části bude práce s těmito metodami prakticky předvedena na reálných datech z vybraných datasetů. Na každý dataset budou aplikovány všechny metody a následně bude porovnáván výsledek jejich aplikace a jejich časová náročnost při aplikaci na dvou různých počítačích.

2 Úvod

2.1 Učení s učitelem

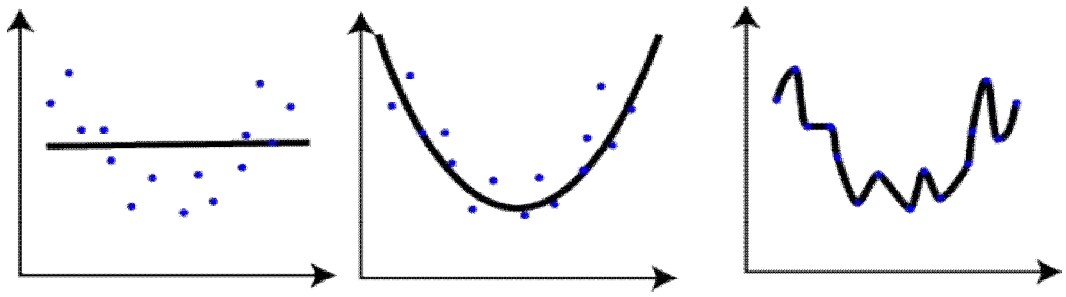
Kapitola volně zpracována dle 1.

Vybrané algoritmy patří do kategorie Učení s učitelem. Pro tuto kategorii je charakteristická práce s dvěma množinami dat. První množina dat se skládá ze dvou podskupin - trénovacích dat a testovacích dat. Obě tyto podskupiny obsahují nezávislé proměnné a jednu či více označených závislých proměnných, tedy konkrétní zařazení dat do tříd.

Oproti této množině dat, kde známe jejich kategorizaci, obsahuje druhá část data pouze s nezávislými proměnnými, neznáme u nich tedy jejich zařazení do tříd.

Princip agoritmů učení s učitelem je následující. Data, u kterých známe jejich klasifikaci, se rozdělí na trénovací a testovací část. Na základě trénovacích dat se vytvoří model, který je následně schopný podle naučených vztahů mezi závislými a nezávislými proměnnými navrhnout hodnotu závislé proměnné pro data bez zařazení. Tento model je testován na testovací množině dat, a následně použit pro predikci tříd u dat bez zařazení.

Při tvorbě modelu je důležité určit správnou míru přesnoti vyjádření trénovacích dat. Při nedostatečné přesnoti modelu nastane takzvaný underfitting, kdy model nebude odpovídat trénovacím datům. Díky tomu nedojde ani ke správné kategorizaci u dat bez zařazení. Naopak při overfittingu, neboli přeučení, dojde až k přílišnému sledování trénovacích dat, díky čemuž je tento model pro predikci také nevhodný. Grafické znázornění lze vidět na následujícím obrázku.



Obr. 1 Porovnání underfitingu, správného modelu, overfittingu

Zdroj: https://shapeofdata.wordpress.com/2013/03/26/general-regression-and-over-fitting/

2.2 Normalizace dat

Kapitola volně zpracována dle 1, 3.

Základem pro správnou klasifikaci je příprava dat. Před aplikováním algoritmu je potřeba převést data do správné podoby – normalizovat je. Jinak může dojít k nesprávnému vlivu různých proměnných na výsledek.
Například pokud výsledek závisí na dvou proměnných, z nichž jedna je v rozmezí několika tisíc a druhá v rámci desítek či stovek, stejná změna druhého parametru ovlivní výsledek více než stejná změna prvního. V takovém případě může i malá odchylka v datech způsobit závažnou nepřesnost výsledného zpracování.

Praktická demonstrace lze předvést na následujícím příkladu:
Při rozhodování, kam se osoba přestěhuje, vycházíme z preferencí ostatních osob. Výběr se provádí na základě tří kritérií. Výše platu, která ovlivní luxus oblasti, do které se přestěhuje, časová vzdálenost od práce a počet vysokých škol ve městě. Výše platu i vzdálenost (pokud bychom ji počítali v sekundách) se může pohybovat ve velkých rozmezích, zatímco počet škol s největší pravděpodobností nepřesáhne hranici 10.
Zatímco změna platu o 10 vybranou lokalitu neovlivní, změna počtu škol o stejnou hodnotu lokalitu změní.

Čím větší je původní rozsah hodnot, tím těžší je správně určit základní jednotku, která bude určovat důležitost vlivu dané hodnoty na výsledek.
Jednou z možností, jak se k dané jednotce dostat, je vydělit kategorii s větším rozsahem hodnot takovým číslem, aby dopad všech kategorií na výsledek byl stejný. To ale záleží spíš na intuici, než na matematickém vyjádření.
Další možností je upravit rozsah hodnot tak, aby odpovídal číslům na stupnici mezi 0 a 1. U této varianty ale hrozí, že pokud jeden záznam výrazně převyšuje všechny ostatní, po převedení do intervalu mezi 0 a 1 dostaneme zbytečně malé hodnoty (0.001, 0.002), které budou moc blízko u sebe.
Třetí způsob upravení vstupních dat je dělení rozsahem hodnot dané kategorie.

Metoda normalizace zvaná z-score upravuje data tak, že od každé hodnoty odečte střední hodnotu všech dat a poté výsledek vydělí velikostí směrodatné odchylky.
Každá z popsaných variant má své výhody a nevýhody a vždy záleží na aktuálních datech a jejich využití, pro zvolení konkrétní metody.

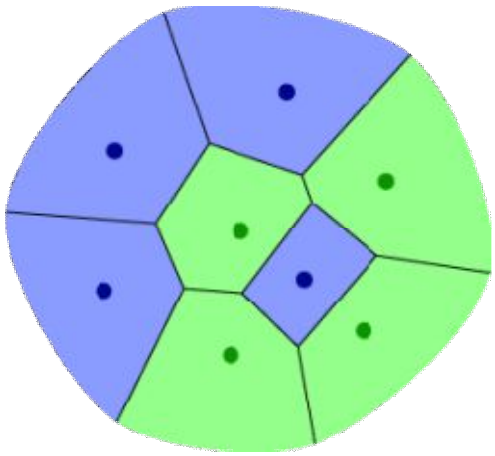
3 Algoritmus Nejbližší sousedi

Kapitola volně zpracována dle 4.

Algoritmus nejbližších sousedů patří mezi nejjednodušší klasifikační algoritmy. Předpokladem pro práci s ním je množina bodů, reprezentujících naše trénovací data. Body jsou umístěny v n-dimenzionálním prostoru, kde n je počet parametrů dat, a hodnota souřadnic je určena na základě hodnot těchto parametrů. U každého bodu navíc známe jeho přiřazení do určité kategorie (třídy).

Parametry dat zde představují nezávislé proměnné a konkrétní kategorie, do které je bod po proběhnutí algoritmu zařazen, je závislá proměnná. Při grafickém vyjádření algoritmu budou parametry zastupovány souřadnicemi bodu v prostoru a kategorie bude vyjádřena například popiskem konkrétního bodu či jeho obarvením.

Cílem algoritmu je přiřazení nových dat do jedné z tříd, na základě podobnosti s ostatními body. Novému bodu určíme souřadnice do prostoru definovaného trénovacími daty. Následně vypočítáme jeho vzdálenost od všech bodů trénovací množiny a určíme bod, který má tuto vzdálenost nejmenší. Tím vyhledáme jeho nejbližšího souseda a zkoumaný bod klasifikujeme stejným způsobem, jakým je klasifikován tento nejbližší bod.



Obr. 2 Voronoi buňky

Zdroj://shapeofdata.wordpress.com/2013/04/23/nearest-neighbors-classification

Při pohledu na nejbližší sousedy z grafického hlediska můžeme určit oblasti, ve kterých je vzdálenost od daného bodu menší než od všech ostatních. Takto vytyčené prostory

ve dvourozměrném prostoru se nazývají Voronoi buňky – viz obr.1. Platí, že každý bod uvnitř jedné Voronoi buňky bude klasifikován stejným způsobem.
Trojrozměrné body kolem sebe místo Voronoi buněk tvoří mnohostěny, u více dimezionálních bodů jsou tyto prostory nazývány pouze jako polytopy.

3.1 Náročnost

Podkapitola volně zpracována dle 6.

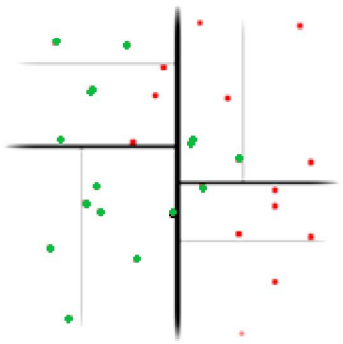
Jelikož algoritmus závisí na porovnávání vzdáleností prvku z testovací množiny od všech prvků z trénovací množiny, paměťová náročnost není nijak výrazná – závisí pouze na celkovém počtu prvků, se kterými pracujeme.
Ačkoli je algoritmus výhodný vzhledem k nízkým nárokům na paměť, porovnání právě klasifikovaných prvků se všemi prvky trénovací množiny je poměrně časově náročné. Náročnost je zde úměrná počtu prvků trénovací množiny. Lze ji částečně snížit vhodným setříděním prvků nebo použitím kd-stromů.

3.2 Kd- stromy

Podkapitola volně zpracována dle 2, 8.

Kd-strom je binární strom, pomocí kterého budeme reprezentovat trénovací množinu dat. Název vznikl ze samotné implementace této struktury – ukládá body, reprezentující jednotlivá data do k-dimenzionálního prostoru, kde K je počet parametrů dat.
Kd-strom postupně rekurzivně dělí prostor pomocí nadrovin. Nadroviny jsou zde rovnoběžné s osami prostoru, v dvou-dimenzionálním případě buď kolmé na osy či rovnoběžné s nimi. Dělení prostoru je ve stromu reprezentováno uzly. A v každém listu, který tímto dělením vzniká, se nachází jeden či více bodů z datového souboru.
Nejlepší způsob hledání směru rozdělení je výpočet rozptylu dat podle každé osy a následné zvolení směru kolmého na osu s největším rozptylem dat. Rozdělení prostoru nadrovinou je vhodné v místě hodnoty mediánu parametru dat dané osy. Tím vznikne rovina kolmá k dané ose, rozdělující data na dvě vyrovnané části. Tímto

způsobem vznikne vyvážený strom. Stále se v něm však mohou objevovat hubené obdélníkové regiony, ačkoli preferovaný výsledek jsou spíše čtvercové oblasti. Tomu zamezíme tím, že místo mediánu budeme hledat hodnotu aritmetického průměru a následně použijeme nejbližší bod k ní.



Obr. 3 Kd- strom
Zdroj: http://www.alglib.net/other/nearestneighbors.php

Na grafu je ukázán příklad kd-stromu ve dvou-dimenzionálním prostoru. Červené body značí data, přímky nadroviny. Čím užší je linka značící přímku, tím větší rekurzi dělení prostoru reprezentuje.

Hledáním nejbližších sousedů v kd-stromu dochází k hledání všech bodů, které mají určitou vzdálenost od daného bodu. Nejdříve najdeme větev stromu obsahující daný bod, a následně prohledáváme další body stejné větve, dokud nedojdeme do okamžiku, kdy jsme od daného bodu vzdáleni více, než potřebujeme.

Použitím kd-stromu při hledání nejbližších sousedů se náročnost algoritmu značně sníží. Při vhodném uspořádání stromu je hodnota log2(n), kde n je počet uzlů stromu.

3.3 Implementace v R

Podkapitola volně zpracována dle 7,9

V jazyce R budeme algoritmus implementovat jako algoritmus k-nejbližších sousedů, přičemž budeme brát v potaz pouze jednoho nejbližšího souseda. Tento algoritmus se nachází v balíčku class a jeho znění je následující.

knn(train, test, cl, k)

kde:

- **train** reprezentuje matici nebo datový rámec trénovací množiny
- **test** reprezentuje matici nebo datový rámec testovaného případu
- **cl** je vektor klasifikačního označení, vyjadřuje kolik prvků dané kategorie je v souboru trénovacích dat
- **k** reprezentuje počet sousedů, se kterými budeme nový bod porovnávat, v tomto případě bude k=1

Před samotným zadáním příkazu tedy načteme do prostředí balíček class zadáním

library(class).

Po zavedení knihovny načteme data, se kterými budeme pracovat. Do proměnné test načteme data, u kterých chceme zjistit jejich klasifikaci. K tomu se využijí příkazy:

test <-read.table("cesta_k_souboru", sep=",")

Obdobně do proměnné train uložíme trénovací data, vynecháme zde ale sloupec, ve kterém je definované zařazení jednotlivých prvků do tříd.

tabulka2<-read.table("cesta_k_souboru", sep=", ")

train<-tabulka2[,-sloupec_se_zarazenim]

Vynechaný sloupec převedeme na faktor a uložíme pod názvem cl.

cl<-factor(tabulka2[,cislo_sloupce])

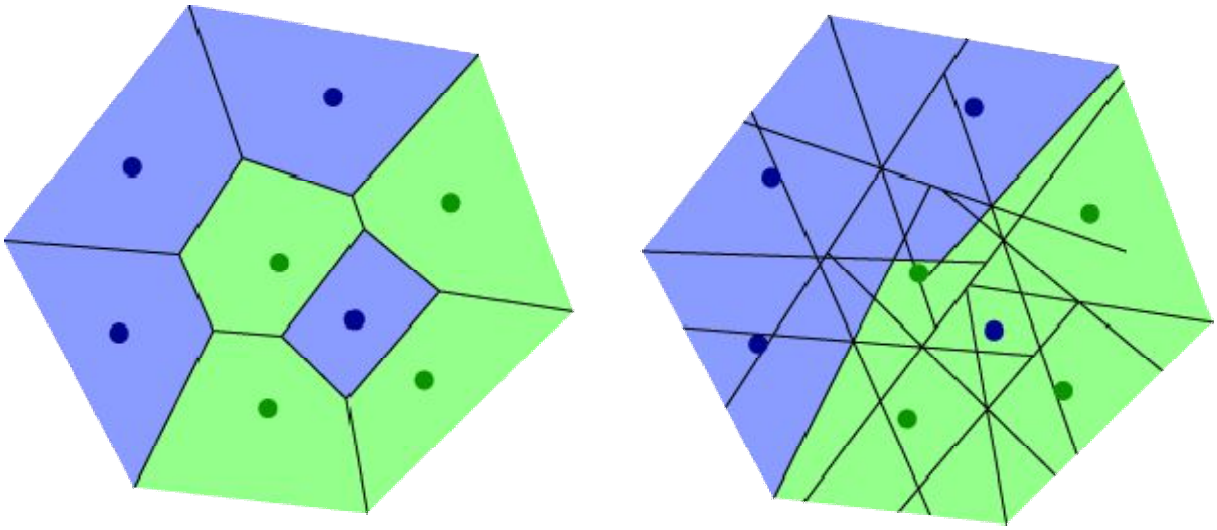
Nyní již zbývá pouze provést příkaz **knn(train, test, cl, 1)** a pro daná data zjistíme jejich zařazení do kategorie.

4 Algoritmus K-nejbližší sousedi

Kapitola volně zpracována dle 5, 10, 11

KNN (Algoritmus K-nejbližších sousedů) je jeden z nejznámějších statistických metod v oblasti rozpoznávání vzorů. Je znám už přes 40 let, původní algoritmus byl navrhnut T.M. Coverem a P.E. Hartem v článku Nearest neighbor pattern z roku 1967.

KNN je složitější variantou algoritmu nejbližších sousedů. Na rozdíl od již popsaného algoritmu nejbližších sousedů se klasifikovaný bod neporovnává se svým nejbližším sousedem, ale bere se v úvahu K nejbližších sousedů. Nový bod je následně zařazen do kategorie, která je mezi K sousedy reprezentována nejvícekrát.
Algoritmus nejbližších sousedů lze vyjádřit jako K-nejbližších sousedů, kde K=1.



Obr. 4 Porovnání nejbližších sousedů s KNN
Zdroj http://shapeofdata.files.wordpress.com/2013/05/07/k-nearest-neighbors

Při porovnání grafického znázornění nejbližších sousedů (obr. 4 vlevo) s grafickým znázorněním K-nejbližších sousedů (obr. 4 vpravo) je zřejmé, že ačkoli při použití K sousedů se v grafu nachází více čar, celkový výsledek je jednodušší.
Porovnáváním klasifikovaných dat s více sousedy dojde k vyhlazení výsledného grafu. Špatně zadaná, nebo šumem ovlivněná trénovací data neovlivní správné zařazení nových dat, viz modrý bod obklopený zelenými body na obr. 4.

Náročnost algoritmu závisí kromě množství dat, se kterými pracuje, především na parametru K. Zvolení konkrétního K ovlivní výsledek klasifikace - při zvolení příliš malé hodnoty hrozí overfitting, naopak při zvolení příliš velké hodnoty můžeme ztratit podstatné detaily. Zároveň se zvětšujícím se K se zvedají i výpočetní nároky.
Neexistuje jediná správná metoda, jak zvolit nejvhodnější K. Vždy závisí na datech, se kterými pracujeme. Jedním z přístupů, jak zjistit K, je heuristika pro výpočet K podle počtu datových bodů v souboru dat. Další možností je vyzkoušení klasifikace části dat, u kterých již známe správné zařazení do kategorií na trénovací množině s více K, a následně vybrání takového K, u kterého došlo k největší shodě s očekávaným výsledkem.

4.1 Implementace v R

Podkapitola volně zpracována dle 7, 9

Jak již bylo zmíněno u implementace nejbližších sousedů, v jazyce R je příkaz pro K-nejbližších sousedů shodný s příkazem pro nejbližší sousedy. Jedinou změnou je nastavení parametru K na číslo větší než 1.

k= ..vybraná hodnota k

knn(train, test, cl, k)

5 Algoritmus Support Vector Machines

Kapitola volně zpracována dle 12, 13, 14, 15

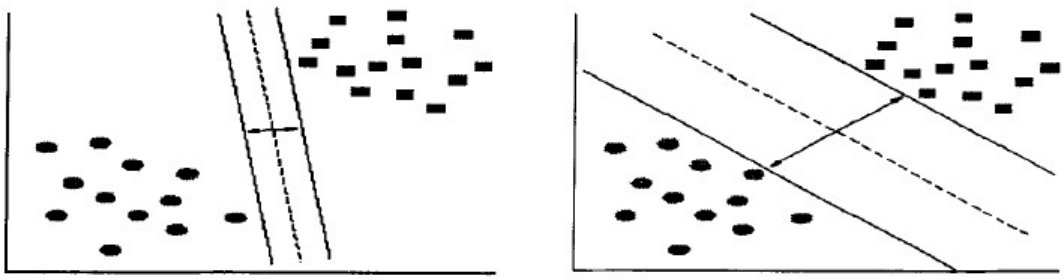
SVM (Support vector machines) algoritmus je založený na Vapnikově teorii statistického učení. Primárně je SVM binární klasifikační algoritmus, což znamená, že řeší klasifkaci dat mezi dvěma kategoriemi.

Při práci s více kategoriemi lze tento problém vyřešit vícenásobným rozkladem dat na dvojice kategorií, provedením jednotlivých zařazení, a opětovným seskupením. Při seskupení dojde ke kombinaci více binárních klasifikací a na základě volícího systému je zvolena žádná či jedna z kategorií.

I zde jsou jednotlivá data reprezentována jako body, respektive vektory, v n-dimenzionálním prostoru, přičemž parametry dat určují souřadnice jednotlivých bodů.

Při použití SVM určíme z trénovacích dat hranici mezi jednotlivými kategoriemi těchto dat, čímž vzniknou dvě oddělené skupiny a následně přiřazujeme nová data k jedné či druhé skupině. V grafickém znázornění v dvourozměrném prostoru si lze jako hranici mezi skupinami představit přímku procházející mezi body patřící do jedné kategorie a body patřící do druhé.

Dojde-li k situaci, kdy lze mezi vybranými kategoriemi vložit více přímek, (přímky jsou vůči sobě jen nepatrně pootočené), SVM se snaží o výběr takové přímky, která zajistí co největší separaci kategorií. Vybírá tedy takovou přímku, která má co největší vzdálenost od nejbližších bodů obou kategorií. Těmto bodům se pak říká podpůrné vektory (support vectors), z čehož vznikl i název algoritmu.

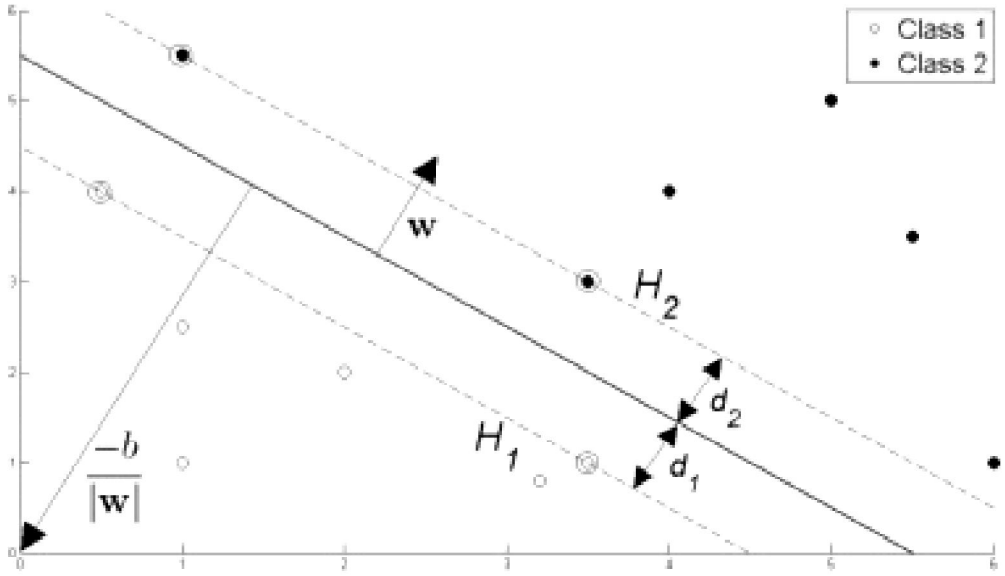


Obr. 5 Výběr správné přímky
Zdroj: SVM classification: Its contents and challenge

V grafickém znázornění si lze místo přímky představit obdélník. Snahou SVM je tento obdélník umístit takovým způsobem, aby velikost jeho strany mezi oběma kategoriemi byla co největší. Ve vícerozměrném prostoru je pak přímka nahrazena nadrovinou a obdélník nahrazen vícerozměrným regionem.

Při setu n dat, které obsahují data patřící do dvou lineárně oddělitelných tříd, lze dělící přímku vyjádřit jako $w.x + b = 0$, kde w je normálový vektor této přímky a $b/|w|$ je kolmá vzdálenost přímky od počátku souřadnic. Body patřící do jedné třídy se pak dají vyjádřit jako $x_i w + b > 1$, zatímco body patřící do druhé třídy mají rovnici $x_i w + b < -1$.

To lze shrnout do jedné rovnice: $y_i(x_i w + b) - 1 > 0$ pro $i = 1..n$ a y_i značící třídu. Podpůrné vektory pak tvoří pro každou třídu jednu přímku, rovnoběžnou s dělící přímkou. Tyto přímky se dají vyjádřit jako $w.x_i + b = 1$ a $w.x_i + b = -1$. Cílem svm modelu je pak určit co největší vzdálenosti d_1 a d_2 těchto přímek od dělící přímky, přičemž $d_1 = d_2$.



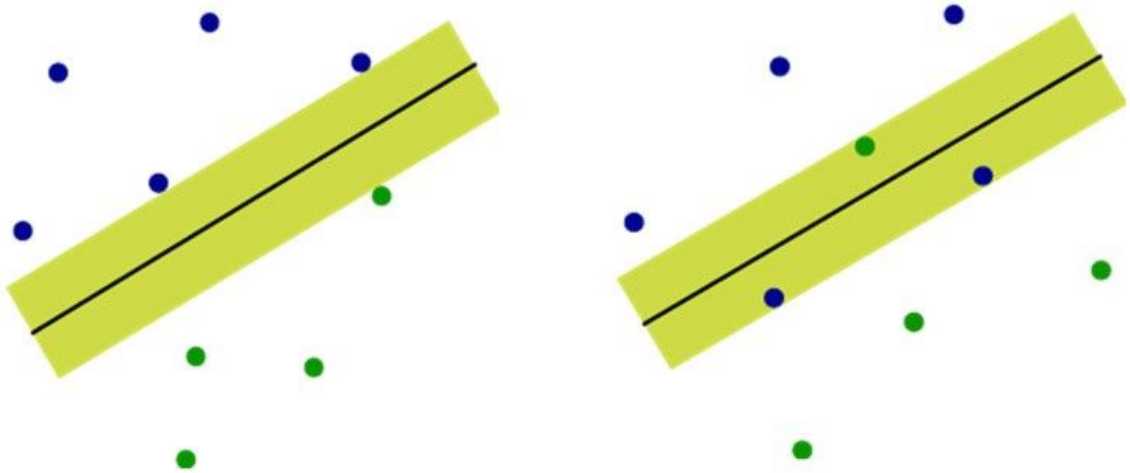
Obr. 6 Dělící přímka
Zdroj: převzato z [27]

Ne vždy je však rozdělení přímkou na dvě zcela odlišné kategorie možné. Pokud se kategorie překrývají, respektive jejich data jsou vzájemně prolnutá, nelze mezi nimi vést žádnou přímku, která by je jednoznačně oddělila. Vždy budou na jedné či druhé

straně přímky osamocené body, patřící ke kategorii s většinou bodů na druhé straně přímky. V tomto případě se hledá taková přímka, která rozděluje obě skupiny s minimálním množstvím bodů na špatné straně přímky.

Podpůrnými vektory jsou nyní nazývaný body nejvzdálenější od dělící přímky, a zároveň umístěné na špatné straně této přímky. Oproti separovatelným kategoriím zde nedochází k hledání přímky, která by měla od podpůrných vektorů co největší vzdálenost, ale naopak se hledá taková přímka, která je k podpůrným vektorům nejblíže.

Analogicky v grafickém znázornění umísťujeme mezi kategorie co nejužší obdélník, oproti co nejširšímu, jak tomu bylo u separovatelných kategorií.



Obr. 7 Porovnání separovatelných skupin(vlevo) a neseparovatelných(vpravo)
Zdroj: http://shapeofdata.files.wordpress.com/

5.1 Jádro

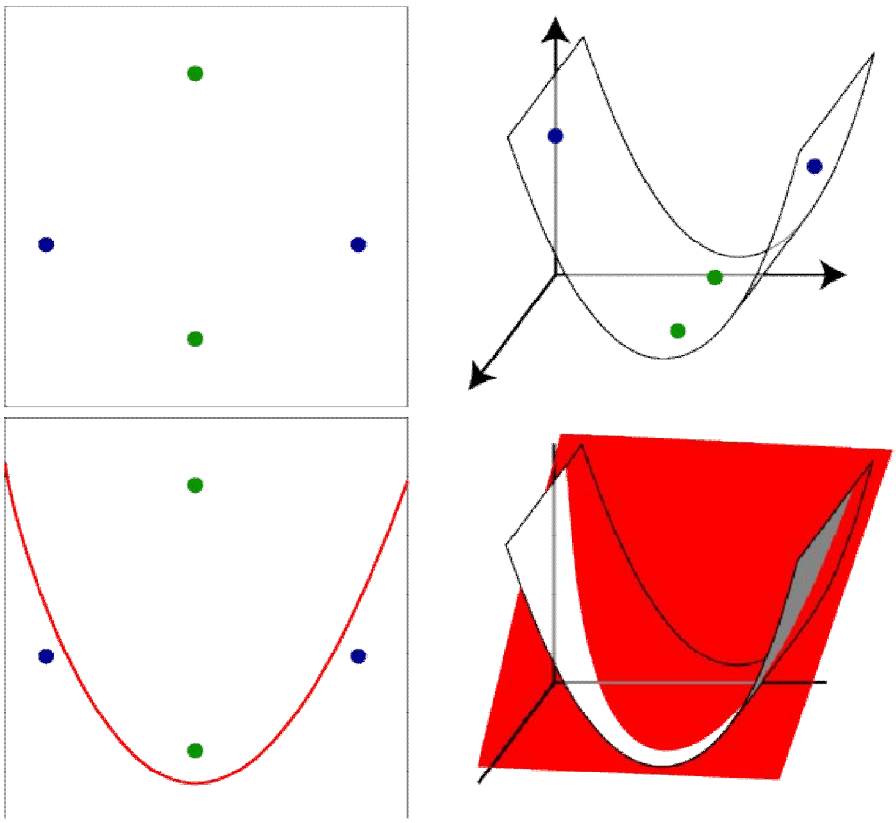
Podkapitola volně zpracována dle 19

Jak již bylo řečeno, dokonalé separování pomocí přímky není vždy možné. Lineární separace je vhodná pouze v případech, kdy data při znázornění netvoří zakřivené tvary. V takovém případě je vhodné nahradit dělící přímku křivkou, případně vícerozměrným zakřiveným tvarem.

Místo přímé změny lineárního algoritmu a hledání křivky lze použít metodu zvanou jádro. Jádro posouvá data do vícerozměrného prostoru, než v jakém se právě nachází. Díky přidání více dimenzí se zvětší i flexibilita, se kterou můžeme definovat dělící

hranici tříd, jelikož přibývají parametry dělící křivky (roviny či nadroviny ve více rozměrném prostoru), které můžeme upravovat a tím přizpůsobit křivku požadovanému tvaru.

Ve vícerozměrném prostoru třídy odděluje jednoduchá přímka, rovina, či nadrovina, která po promítnutí zpět do prostoru v původní dimenzi vytvoří hledanou složitější křivku. Konkrétní příklad lze ukázat na oddělení tříd ve dvourozměrném prostoru. Pokud jsou tyto třídy ve dvourozměrném prostoru separovatelné pouze pomocí křivky, je možné, že ve trojdimenzionálním prostoru budou separovatelné pomocí roviny, která se do původního dvourozměrného prostoru promítne jako hledaná křivka. Viz obr. 8.



Obr. 8 Příklad použití kernelu
Zdroj: http://shapeofdata.wordpress.com/2013/05/27/kernels/

SVM algoritmus se při použití jádra aplikuje na data posunutá do vyšší dimenze. Zvolení vhodné dimenze závisí na konkrétních datech, se kterými se pracuje. S posunutím sice vzniká více parametrů dat a jednodušší hledání oddělení tříd,

ale zároveň narůstá výpočetní složitost a hrozí přílišné zpřesnění modelu a nastání overfittingu.

Nevýhodou SVM je lehké znepřesnění několika málo špatně zadanými body či šumem v trénovacích datech. Například pokud se jen několik bodů vlivem chyby vzdálí od ostatních bodů patřících do stejné kategorie, dojde k rozdílnému umístění dělící přímky, či nadroviny ve vícerozměrném prostoru, oproti umístění, které by správně mělo nastat a následná klasifikace dat nebude správná.

5.2 Implementace SVM

Kapitola volně zpracována dle 16, 17, 18

Jedním ze způsobů jak pracovat se SVM v R, je pomocí příkazu svm() z balíčku e1071. Ten ale není součást základních knihoven R, a je potřeba ho doinstalovat. K tomu lze použít příkaz

install.packages("e1071")

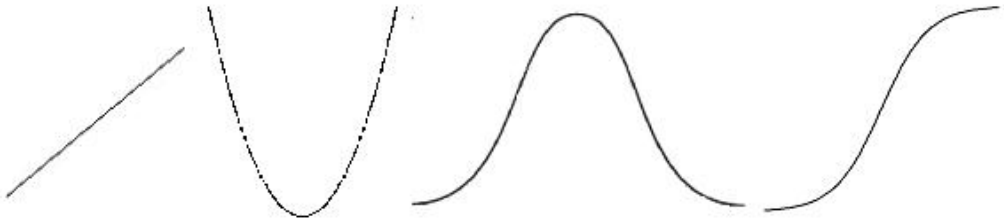
a následně zvolit z nabídky mirror, odkud se balíček stáhne. Po stažení stačí už jen zvolit práci s knihovnou pomocí příkazu

library(e1071).

Samotný příkaz má mnoho parametrů, mezi nevýznamější patří zejména

- **formula** – symbolický popis budoucího modelu
- **data** – volitelný datový rámec obsahující proměnné modelu. Defaultně jsou proměnné brány z prostředí, ze kterého je příkaz svm() volán
- **x** – matice či vektor
- **y** – odpovídající vektor s označením kategorie pro každý řádek či komponentu x. Pro účely klasifikace nabývá podobu faktoru, pro potřeby regrese je zastoupen numerickým vektorem
- **type** – parametr určující, jestli bude SVM použit ke klasifikaci, regresi nebo detekci aktualit. Možné hodnoty jsou následující:
  - **C-classification** – defaultní nastavení při zadání y jako faktoru
  - **nu-classification**
  - **one-classification** - použito při práci pouze s jednou třídou
  - **eps-regression** - defaultní nastavení při zadání y jako numerického vektoru
  - **nu-regression**

- **kernel** – druh jádra použitý při trénovací a předpovídací části algoritmu. Možné hodnoty:
  - **linear –** vyjádřeno vzorcem $u´v$
  - **polynomial –** vyjádřeno vzorcem $(\gamma u´v + coef\ 0)^{degree}$
  - **radial basis –** vyjádřeno vzorcem $e^{(-\gamma|u-v|^2)}$
  - **sigmoid –** vyjádřeno vzorcem $\tanh(\gamma u´v + coef\ 0)$



Obr. 9 Přímka, Parabola, Gaussova křivka, Sigmoid
Zdroj: vlastní práce

V případě zvolení jiného než lineárního jádra lze zvolit i parametr **gamma.** V případě zvolení polynomického nebo sigmoidního jádra se nastavuje parametr **coef**. Jádra polynomiálního typu navíc využívají i parametr **degree.** Defaultní nastavení těchto parametrů je gamma =1/(dimenze dat), coef=0 a degree=3.

Praktické použití příkazu SVM bude ukázáno na datasetu Iris, obsaženém v R. Prvním krokem je určení datasetu, na kterém bude klasifikace probíhat - **data(iris).**

Nyní je potřeba zadat nezávislé proměnné do parametru x a závislé proměnné do parametru y. Parametr x je zadán pomocí příkazu subset, kterým jsou z datasetu Iris vybrána všechny data, kromě hodnot Species, což je zde závislá proměnná. Ta je přiřazena do parametru y.

x=subset(iris, select=-Species)

y= Species

Pokud nezvolíme klasifikaci na základě jiného jádra než přednastaveného radiálního, jsou již známé všechny potřebné parametry a lze zadat příkaz k výpočtu modelu pro support vector machines.

model <- svm(x, y)

Dalším příkazem

summary (model)

dojde k vypsání informací o zjištěném modelu. Zjistíme tak například účel SVM, typ jádra, počet podpůrných vektorů a počet tříd, které jsou od sebe oddělovány.

Po zjištění modelu už nyní můžeme řadit nová data do jednotlivých kategorií. Nejdříve je nutné provést výběr dat, která chceme klasifikovat. Následujícím příkazem vybereme všechna data, kromě parametru Species z datového setu Iris a přiřadíme je do vektoru x.

x <- iris [,-5]

Následně vytvoříme nový vektor, v tomto případě pojmenovaný *zarazeni*, který bude obsahovat hodnoty kategorií, přiřazené vstupním datům podle SVM modelu.

zarazeni <-predict(model, x)

Pokud chceme ověřit přesnost zařazování podle konkrétního modelu, porovnáme počet dobře zařazených dat s celkovým počtem zařazovaných.

y<- iris[,5], sum (zarazeni==y)/lenght(y)

Případně příkazem **table(zarazeni, y)** vytvoříme matici záměn. Charakteristickým rysem matice záměn je přehledné zobrazení zařazení dat modelem vůči skutečnému zařazení, do kterého patří. Jednotlivé řádky reprezentují kategorie dat a počet skutečného výskytu dat v těchto kategoriích, zatímco sloupce zobrazují počty dat v kategoriích, které byly určeny pomocí modelu. Data, ve kterých se model shodl se skutečností, jsou zobrazena na hlavní diagonále tabulky.

6 Logistická regrese

Kapitola volně zpracována dle 2, 20

Lineární regrese je binomická funkce určující pravděpodobnost hodnoty závislé proměnné v závislosti na hodnotách nezávislých proměnných. Jelikož je to binární funkce, rozhoduje se u závislé proměnné pouze mezi dvěma hodnotami. Při kategorizaci dat to znamená, že lze testovaná data zařazovat pouze mezi dvěma kategoriemi.

Logistická regrese vychází z lineární regrese. Oproti ní ale vytváří lineární model založený na transformované cílové proměnné. Pravděpodobnost jedné z kategorií se tedy dá vyjádřit jako:

$$log(Pr[1|a_1, a_2, a_3, \dots, a_{k1}]) / (Pr[1|a_1, a_2, a_3, \dots, a_{1k}])$$

Transformovaná proměnná je dále aproximována jako u lineární regrese, čímž vznikne následující model:

$$Pr[1|a_1, a_2, a_k] = 1/(1 + exp\ (-w_0 - w_1 a_1 - w_2 a_2 - \cdots - w_k a_k)),$$

kde w_0 až w_k značí váhy jednotlivých nezávislých proměnných. Pro určení správných hodnot vah se využívá metody *log-likelihood*, kde je cílem maximalizovat následující součet hodnot:

$$\sum_{i=1}^{n} (1 - x^{(i)})\, log(1 - Pr[1|a_1{}^{(i)}, a_2{}^{(i)}, \dots, a_k{}^{(i)}]) + x^{(i)} log\ (Pr[1|a_1{}^{(i)}, a_2{}^{(i)}, \dots a_k{}^{(i)}])$$

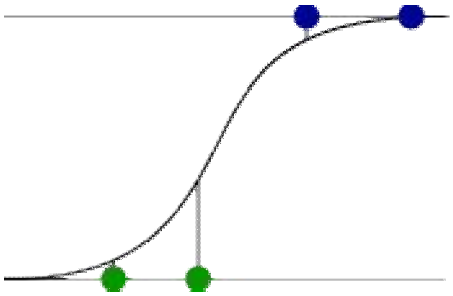
kde $x^{(i)}$ je buď jedna nebo nula.

Při klasifikaci každý bod z testovaných dat obdrží vektor vah vypočítaný z modelu. K zařazení do první ze dvou tříd dojde, pokud

$$(w_0{}^{(1)} - w_0{}^{(2)}) + (w_1{}^{(1)} - w_1{}^{(2)}) a_1 + \dots + (w_k{}^{(1)} - w_k{}^{(2)}) > 0.$$

Indexy jedna zde značí váhy vypočítané pro první kategorii, indexy dva druhou.

Z toho vyplývá, že při grafickém znázornění klasifikace pomocí logistické regrese je důležitým prvkem křivka, oddělující od sebe jednotlivé kategorie.



Obr. 9 Logistická regrese
Zdroj: https://shapeofdata.wordpress.com/2013/05/17/logistic-regression/

Na rozdíl od algoritmu SVM zde ale není pozice dělící křivky ovlivňována jen několika nejbližšími body z každé kategorie, nýbrž do jejího umístění zasahují všechny body datasetu. Každý z bodů má na pozici křivky jinak velký vliv - body, které leží nejblíže, jsou v tomto ohledu nejdůležitější.

Díky vlivu všech bodů na křivku, je logistická regrese méně náchylná na šum v datech než SVM – několik málo chybně zadaných bodů nemá na umístění logistické křivky takový vliv, jako tomu bylo u hraniční přímky u Support vector machine, jelikož větší množství správně zadaných bodů s největší pravděpodobností převáží vliv těch odchýlených ze své kategorie.

6.1 Implementace v R

Kapitola volně zpracována dle 25, 26

K určení modelu logistické regrese slouží v jazyce R příkaz **glm(),** obsahující více parametrů, ze kterých využijeme především tyto:

- **formula –** určuje jakou proměnnou se snažíme odvodit, a na základě kterých proměnných ji budeme odvozovat

  Nejčastější formulace formule je pomocí znaku ~ (tildy), který odděluje závislou proměnnou vlevo a nezávislé proměnné vpravo.

- **data –** definují datovou oblast, ze které čerpáme proměnné. Pokud tento parametr není definován, použijí se automaticky data z aktuálního pracovního prostředí.

- **family –** při použití logistické regrese musí být tento parametr vždy nastaven na hodnotu **binomial**

Získaný model následně můžeme použít pro předpověď pravděpodobnosti příslušnosti dalších dat k jedné či druhé kategorii. K tomu nám poslouží příkaz

pravdepodobnosti <- predict(model, data, type="response"),

kde **model** značí model získaný předchozím příkazem glm, **data** jsou data, u kterých zjišťujeme příslušnost ke kategoriím a parametr **type="response"** určuje, že hledáme pravděpodobnosti.

Jelikož výsledkem je pouze seznam pravděpodobností jednotlivých dat, nikoli kategorie, je nutné nyní převést výsledky do jiného formátu. To lze udělat například následujícím způsobem:

Funkcí **Kategorie <- rep(nazev,pocet)** vytvoříme vektor Kategorie obsahující název první kategorie tolikrát, kolik je instancí dat. Dále pomocí

kategorie [pravdepodobnosti > 0,5] = nazev2

určíme ty datové instance, u kterých je pravděpodobnost >0,5 a příslušný název kategorie přepíšeme na druhou hodnotu. Tím získáme vektor obsahující název odhadnuté kategorie pro každý bod podle vytvořeného modelu logistické regrese.

7 Aplikace

V této části budou dříve popsané metody prakticky aplikovány na různých datasetech na dvou různých počítačích a výsledky jednotlivých aplikací vzájemně porovnány.

Charakteristiky použitých počítačů jsou popsané v následující tabulce.

Charakteristiky	**PC 1**	**PC2**
Procesor	Intel® Core™2 Duo T7500 @2.20GHz 789 MHz	Intel® Pentium® Dual E2180 @2.00 GHz 2.00 GHz
Přidělená paměť	2.99 GB RAM	512 MB RAM
Operační systém	Microsoft Windows XP, Ultra Edition	Windows 7, Ultimate

Tabulka 1 Charakteristiky Pc
Zdroj: vlastní zpracování

Na obou počítačích bylo s daty pracováno pomocí jazyka R verze 3.1.0 (2014-04-10), s přezdívkou Spring Dance. Jako uživatelské rozhraní jazyka R bylo použito R Studio. R studio je integrované vývojové prostředí pro R, které je k dispozici jako open source, nebo komerčí verze, přičemž pro účely této práce byla využita open source verze 0.98.1091.0, která patří pod licenci AGPL v3. [21]

8 Poker Hand dataset

Kapitola zpracována za použití datasetu získaného z 23

Poker Hand dataset je soubor 1 025 010 záznamů, vytvořený v lednu 2007 Franzem Oppacherem a Robertem Cattralem z Carletonské univerzity v Ottawě.

Dataset je rozdělen na trénovací část (25 010 záznamů) a testovací část (1 000 000 záznamů). Každý záznam obsahuje hodnoty 11 proměnných, z nichž prvních deset popisuje barvu a hodnotu vytažené karty a poslední, jedenáctý atribut, určuje druh možné výherní pokerové kombinace karet.

Proměnná	Hodnota	Význam
S1, S2, S3, S4, S5	1- 4	reprezentace barvy: Srdce, Listy, Žaludy, Kule
C1, C2, C3, C4, C5	1- 13	reprezentace hodnosti: Eso, 2, 3, …, Král
CLASS	0 - 9	reprezentace výherní kombinace 0: vysoká karta 1: jeden pár 2: dva páry 3: tři stejné 4: postupka 5: barva 6: plný dům 7: čtyři stejné 8: čistá postupka 9: královská postupka

Tabulka 2 Proměnné datasetu PokerHand
Zdroj: vlastní tvorba

8.1 Nejbližší sousedi

Při aplikaci nejbližších sousedů došlo k následnému zařazení dat do tříd. Jelikož dataset obsahoval i informaci o správném zařazení instancí, lze porovnat počet instancí v každé kategorii získaný algoritmem nejbližších sousedů s počtem, který by měl nastat při dokonalé klasifikaci. Jak je vidět v tabulce 3, počet správně zařazených instancí je

přibližně poloviční oproti celkovému počtu instancí v jednotlivých kategoriích. Pravděpodobnost správného zařazení je při aplikaci nejbližších sousedů 65,6%.

Třída	0	1	2	3	4	5	6	7	8	9
Originální zařazení	501209	422498	47622	21121	3885	1996	1424	230	12	3
Správně zařazeno	372774	263649	11099	6266	1402	688	494	30	12	1

Tabulka 3 Zařazení NN, Poker hand
Zdroj: vlastní práce

Časová náročnost je vzhledem k množství testovaných dat v řádu minut. Navíc je zde znát i velký rozdíl mezi počítači. Konkrétní výsledky lze vidět v tabulce 4.

PC B	user	system	elapsed	PC B	user	system	elapsed
	728.04	0.11	748.45		1309.16	2.31	1461.46

Tabulka 4 Časová náročnost NN, Pokerhand
Zdroj: vlastní práce

8.2 Knn

Na datasetu byla několikrát aplikována klasifikace pomocí K nejbližších sousedů, pokaždé s jinak velkým parametrem k. Výsledky klasifikace se od sebe liší podle toho, jak velké K bylo použito. Ve srovnání s algoritmem nejbližších sousedů, kde K=1, je však téměř ve všech provedených pokusech znát upřesnění klasifikace. Přesnost všech pokusů KNN se pohybuje mezi 64,9% a 69,9%. Nejpřesnějšího výsledku dosáhla funkce s hodnotou K=10, kde pravděpodobnost správného zařazení je 69,9%. Porovnání výsledků klasifikací s různými K je shrnuto v tabulce 6.

	0	1	2	3	4	5	6	7	8	9
Originál	501209	422498	47622	21121	3885	1996	1424	230	12	3
K= 2	373002	260875	8935	4907	1185	471	361	18	0	1
K= 3	393196	272717	6282	3822	984	335	278	15	0	0
K= 5	405236	276243	4064	2633	761	161	173	6	0	0
K=10	419991	276628	1644	1061	367	20	36	1	0	0

Tabulka 5 Zařazení KNN pro různá K, Pokerhand
Zdroj: vlastní práce

V závislosti na použitém K se liší i časová náročnost algoritmu, viz tabulka 6.

	PC A	user	system	elapsed	PC B	user	system	elapsed
K= 2		594.64	0.03	596.06		1283.39	2.58	1910.08
K= 3		926.48	0.58	1059.78		1263.99	2.37	1770.36
K= 5		709.33	0.05	715.97		1440.64	2.54	1817.78
K=10		607.52	0.04	608.50		1305.95	1.98	1790.88

Tabulka 6 Časová náročnost KNN pro různá K, Pokerhand
Zdroj: vlastní práce

8.3 SVM

Algoritmus Support vector machine vytvořil na základě trénovacího datasetu model pro 10 tříd s 22 706 podpůrnými vektory. Konkrétní rozložení vektorů v jednotlivých třídách je rozepsáno v tabulce 7.

Třída	0	1	2	3	4	5	6	7	8	9
Počet vektorů	5	5	10335	10453	93	513	1206	54	36	6

Tabulka 7 Rozložení podpůrných vektorů SVM, Pokerhand
Zdroj: vlastní práce

Při zařazování testovacích dat tento model nebyl moc úspěšný. Na rozdíl od nejbližších sousedů a k-nejbližších sousedů zařadil data jen do dvou z deseti nalezených tříd. Přesto je úspěšnost správného zařazení 55,6%. Zároveň i časová náročnost byla mnohem vyšší než u předchozích algoritmů, viz tabulka 10. Díky tomu se zdá být nejméně vhodným algoritmem pro dataset Poker hand. Výsledek klasifikace pomocí SVM je zaznamenán v tabulce 8.

	0	1	2	3	4	5	6	7	8	9
Originál	501209	422498	47622	21121	3885	1996	1424	230	12	3
SVM	413034	143746	0	0	0	0	0	0	0	0

Tabulka 8 Zařazení SVM, Pokerhand
Zdroj: vlastní práce

	PC A	user	system	elapsed	PC B	user	system	elapsed
Model		337.94	0.03	338.73		514.31	1.33	729.76
Predikce		3966.11	0.86	3999.80		4354.27	6.96	4886.55
Celkem		4304.05	0.89	4338.53		4868.58	8.29	5616.31

Tabulka 9 Časová náročnost SVM, Pokerhand
Zdroj: vlastní práce

9 Dataset Gisette

Kapitola zpracována za použití datasetu získaného z 28

Dataset Gisette byl vytvořen Isebellou Guyon z původního datasetu Yanna LeCuna a Corriny Cortes MNIST. Dataset obsahuje 13 500 instancí o 5 001 atributech a je rozdělen na tři části. V rámci této práce bylo pracováno s trénovacím oddílem, obsahujícím 6 000 záznamů, a s testovacím oddílem, obsahujícím 1 000 záznamů.

Účelem klasifikace tohoto datasetu je rozpoznání ručně napsané číslice. Cílem je rozhodnout, jestli ručně napsaná číslice, zaznamenaná na obraze o rozměrech 28x28 je číslice 4 nebo číslice 9.

Proměnná	Hodnota	Význam
V1, V2, ...,V5000	integer	Nezávislé proměnné k určení výsledné číslice
V 5001	-1, 1	Proměnná značící rozpoznanou číslici -1 – rozpoznaná číslice je 4 1 – rozpoznaná číslice je 9

Tabulka 10 Proměnné datasetu Gisette
Zdroj: vlastní práce

9.1 Nejbližší sousedi

Klasifikace pomocí nejbližších sousedů byla u Gisette velmi úspěšná. Jak je vidět v tabulce 11, úspěšnost zařazení dat do příslušné třídy je 93,57%.

Kategorie	-1	1
Originál	3000	3000
Knn	2809	2805

Tabulka 11 Zařazení NN, Gisette
Zdroj: vlastní práce

U časové náročnosti je znát velký rozdíl mezi počítači A a B. Ačkoli je časový údaj naměřený na počítači A podobný časovému údaji naměřenému u datasetu Pokerhand, na počítači B byl zjištěn údaj více jak devětkrát větší.

	PC A	user	system	elapsed	PC B	user	system	elapsed
NN		475,88	0.19	478,15		4432.61	2.98	4401.04

Tabulka 12 Časová náročnost NN, Gisette
Zdroj: vlastní práce

9.2 K-nejbližší sousedi

Funkce KNN pro všechna použitá K byla velmi úspěšná při kategorizaci Gisette dat. Ve všech testovaných případech přesáhla pravděpodobnost správného zařazení 92%. Nejúspěšnější hodnotou parametru K byla hodnota K=5, kde úspěšnost dosáhla 94,93%. V tabulce 13 je zaznamenáno konkrétní zařazení testovaných dat do tříd.

Kategorie	-1	1
Originál	3000	3000
K= 2	2782	2792
K= 3	2837	2848
K= 5	2838	2858
K=10	2813	2856

Tabulka 13 Zařazení KNN pro různá K, Gisette
Zdroj: vlastní práce

Tabulka 14 zobrazuje časovou náročnost při jednotlivých aplikacích KNN. Stejně jako u nejbližších sousedů je zde vidět značný rozdíl mezi údaji získanými z obou počítačů. Počítač B se zde projevil jako mnohem pomalejší z obou strojů. Rozdíl mezi nimi je téměř desetinásobný.

	PC A	user	system	elapsed	PC B	user	system	elapsed
K= 2		479.81	0.28	482.17		4725.69	2.90	4786.68
K= 3		477.23	0.22	478.61		4830.57	2.64	4867.05
K= 5		477.36	0.39	478.75		4931.33	0.41	4986.44
K=10		479.52	0.44	483.10		4876.28	0.42	5285.18

Tabulka 14 Časová náročnost KNN s různým K, Gisette
Zdroj: vlastní práce

9.3 SVM

Aplikací SVM algoritmu na Gisette dataset vznikl model s rovnoměrným rozdělením podpůrných vektorů mezi obě kategorie, jak lze vyčíst z tabulky 15.

Třída	-1	1
Počet vektorů	500	500

Tabulka 15 Rozložení podpůrných vektorů SVM, Gisette
Zdroj: vlastní práce

Při následovné kategorizaci podle tohoto modelu byla zjištěna mnohem menší přesnost zařazení, než tomu bylo u algorimů KNN. Veškerá data byla označena jako rozpoznání číslice 4, což v tomto případě značí 50% úspěšnost správného zařazení.

Třída	-1	1
Originál	3000	3000
SVM	3000	0

Tabulka 16 Zařazení SVM, Gisette
Zdroj:vlastní práce

Časová náročnost SVM metody byla mnohem menší než použití NN či KNN. Přesné naměřené hodnoty jsou sepsané v následující tabulce.

	PC A	system	user	elapsed	PC B	system	user	eapsed
Model		26.05	0.09	26.19		31.17	0.30	34.96
Predikce		49.85	0.38	50.29		58.94	0.67	59.74
Celkem		75.90	0.47	76.48		90.11	0.97	94.71

Tabulka 17 Časová náročnost SVM, Gisette
Zdroj: vlastní práce

9.4 Logistická regrese

Algoritmus logistické regrese byl při aplikaci na dataset Gisette na počítači A úspěšný z 46,23%, což je zatím nejmenší úspěšnost kategorizace na tomto datasetu. Oproti NN či KNN je úspěšnost téměř poloviční.

Počítač B nebyl schopen klasifikovat celý soubor dat najednou. Největší možné množství dat, se kterými byl schopen pracovat, bylo 5 700 instancí. Úspěšnost

klasifikace těchto záznamů byla 53,82% %. Porovnání úspěšnosti klasifikace na obou počítačích je shrnuto v tabulce 18.

	PC A			PC B		
Třída		-1	1		-1	1
Originál		3000	3000		2856	2844
Logistická regrese		1330	1444		1598	1470

Tabulka 18 Zařazení Logistická regrese, Gisette
Zdroj: vlastní práce

Co se týče časové náročnosti, logistická regrese byl sice pomalejší než algorimus SVM, zato ve srovnání s KNN metodou dosahovaly naměřené údaje přibližně stejných hodnot. V porovnání obou počítačů byl počítač B opět pomalejší než počítač A, i přes to, že pracoval s menším množstvím dat. Přesné údaje jsou v tabulce 19.

	PC A	system	user	elapsed	PC B	system	user	elapsed
Model		379.58	1.02	384.88		901.08	14,42	1790.47
Predikce		3.28	0.14	3.39		110.72	16.79	2590.66
Celkem		382.86	1.16	388.37		1011.80	31.21	4381.13

Tabulka 19 Časová náročnost Logistické regrese. Gisette
Zdroj: vlastní práce

10 Dataset Chess (King-Rook vs. King)

Kapitola zpracována za použití datasetu získaného z 24

Dataset Chess, s přeným názvem Chess endgame database for white king and rook against black king, byl vytvořen v roce 1994 Michaelem Bainem a Arthurem van Hoffem v Turinském Institutu v Glassgow, UK.

Dataset je tvořen 28 056 záznamy se sedmi proměnnými, ze kterých prvních šest proměnných určuje postavení tří figurek na šachovnici (bílý král, bílá věž, černý král), a poslední proměnná určuje počet tahů do výhry bílého nad černým, případně nastání remízy. Počet kategorií je celkem 18.

Dataset nebyl explicitně rozdělen na trénovací a testovací část. K rozdělení dat došlo manuálně - do trénovací části bylo přiřazeno prvních 10 záznamů z každé kategorie, celkem 180 záznamů. Zbytek dat tvoří testovací část.

Proměnná	Hodnota	Význam
V1	a, b, ..., h	Pozice bílého krále
V2	1, 2, ..., 8	Pozice bílého krále
V3	a, b, ..., h	Pozice bílé věže
V4	1, 2, ..., 8	Pozice bílé věže
V5	a, b, ..., h	Pozice černého krále
V6	1, 2, ..., 8	Pozice černého krále
V7	„draw“, „zero“, „one“, „two“, „three“, „four“, „five“, „six“, „seven“, „eight“, „nine“, „ten“, „eleven“, „twelve“, „thirteen“, „fourteen“, „fiften“, „sixteen“	Počet tahů do výhry bílého nad černým

Tabulka 20 Proměnné datasetu Chess
Zdroj: vlastní práce

10.1 Nejbližší sousedi

Při aplikaci nejbližších sousedů na dataset Chess bylo klasifikováno 27 876 instancí z čehož jen 8 889 zařazení odpovídalo třídě, do které dané instance opravdu patřily.

Porovnání jednotlivých zařazení se skutečnou příslušností ke třídě lze vidět v následující tabulce.

Třídy	draw	zero	one	two	three	four	five	six	seven	eight
Originál	2786	17	68	236	71	188	461	582	673	1423
K=1	1420	9	65	187	60	110	306	304	9	674

Třídy	nine	ten	eleven	twelve	thirtten	fourteen	fifteen	sixteen
Originál	1703	1975	2844	3587	4184	4543	2156	380
K=1	445	245	580	755	1423	1722	463	112

Tabulka 21 Klasifikace NN, Chess
Zdroj: vlastní práce

Časová náročnost u tohoto algoritmu nebyla nijak veliká, a mezi oběma počítači se mnoho neliší, viz tabulka 22.

	PC A	user	system	elapsed	PC B	user	system	elapsed
Nejbližší sousedi		0.13	0.02	0.14		0.14	0.00	0.88

Tabulka 22 Časová náročnost NN, Chess
Zdroj: vlastní práce

10.2 K-nejbližší sousedi

Na dataset Chess byl několikrát aplikován algoritmus nejbližších sousedů, pokaždé s odlišnou hodnotou parametru K. Dokud byla hodnota K menší než 10, neklesla úspěšnost klasifikace pod 30%. Při výrazném zvednutí hodnoty K se úspěšnost výrazně snížila. Klasifikace s hodnotou K=100 dosáhla dokonce pouhých 12,66% úspěšnosti. Nejúspěšnější klasifikace bylo dosaženo při použití K=2, kde úspěšnost dosáhla 31,5%. Přesné výsledky jednotlivých pokusů jsou v tabulce 23.

Třídy	draw	zero	one	two	three	four	five	six	seven	eight
Originál	2786	17	68	236	71	188	461	582	673	1423
K= 2	1400	9	66	181	52	96	225	302	9	639
K= 5	1556	8	68	194	48	62	157	283	35	520
K=10	1956	10	68	183	49	16	144	287	39	328

K=50	1261	2	66	81	39	7	94	316	40	227
K=70	527	1	32	58	30	20	126	136	75	230
K=100	356	1	13	25	10	33	61	92	90	166

Třídy	nine	ten	eleven	tweve	thirteen	fourteen	fifteen	sixteen
Originál	1702	1975	2844	3587	4184	4543	2156	380
K= 2	460	241	579	845	1389	1666	491	131
K= 5	515	267	596	675	1306	1603	556	157
K= 10	547	182	642	600	1273	1564	643	205
K= 50	768	196	488	692	1098	1505	835	250
K= 70	529	160	586	679	852	1258	540	152
K=100	283	149	299	425	625	616	243	43

Tabulka 23 Zařazení KNN pro různá K, Chess
Zdroj: vlastní práce

Algoritmus K-nejbližších sousedů nebyl při aplikaci na dataset Chess náročný. Ani jeden pokus nepřesáhl časový interval 2 sekund.

	PC A	user	system	elapsed	PC B	user	system	elapsed
k=2		0.17	0.00	0.18		0.14	0.00	0.66
k=5		0.28	0.00	0.28		0.14	0.00	0.15
k=10		0.29	0.00	0.29		0.17	0.00	0.17
k=50		0.71	0.00	0.71		0.62	0.00	0.63
k=70		0.85	0.00	0.85		0.88	0.00	0.88
k=100		1.15	0.00	1.16		1.21	0.00	1.24

Tabulka 24 Časová náročnost KNN pro různá K, Chess
Zdroj:vlastní práce

10.3 SVM

Při tvorbě modelu klasifikace pomocí metody Support vector machine bylo určeno 176 podpůrných vektorů s následujícím rozdělením do jednotlivých tříd.

Třída	draw	zero	one	two	three	four	five	six	seven	eight	nine
Vektory	10	10	10	9	9	10	10	10	10	9	10

Třída	ten	eleven	twelve	thirteen	fourteen	fifteen
vektory	10	10	10	10	10	9

Tabulka 25 Rozdělení podpůrných vektorů, Chess
Zdroj: vlastní práce

Z výsledků aplikace získaného modelu pro klasifikaci testovacích dat byla vytvořena tabulka 26. Z porovnání zařazení testovaných dat do tříd pomocí SVM modelu a správných hodnot zjistíme, že úspěšnost klasifikace touto metodou je pouze 3,8%.

Třída	draw	zero	one	two	three	four	five	six	seven	eight
Originál	2786	17	68	236	71	188	461	582	673	1423
SVM	43	7	44	14	38	9	5	10	0	18

Třída	nine	ten	eleven	twelve	thirteen	fourteen	fifteen	sixteen
Originál	1702	1975	2844	3587	4184	4543	2156	380
SVM	10	32	9	10	48	701	41	34

Tabulka 26 Zařazení SVM, Chess
Zdroj: vlastní práce

Ani algoritmus SVM nebyl při aplikaci na dataset Chess nijak výrazně náročný. Byl zde však znát rozdíl mezi počítači A a B. Na počítači A časový interval nikdy nepřesáhl 1 sekundu, zatímco na počítači B celkový výpočet modelu a následná klasifikace trvala necelých 9 sekund. Časové údaje jsou zaznamenány v tabulce 27.

	PC A	user	system	elapsed	PC B	user	system	elapsed
Model		0.08	0.00	0.08		0.09	0.00	0.36
Predikce		0.83	0.01	0.84		0.85	0.22	8.05
Celkem		0.89	0.01	0.90		0.94	0.22	8.41

Tabulka 27Časová náročnost SVM, Chess
Zdroj: vlastní práce

11 Dataset Teaching Assistant Evaluation

Kapitola zpracována za použití datasetu získaného z 22

Dataset Teaching Assistant Evaluation (Teaching) byl sestaven Wei-Yin Lohem z oddělení statistiky z univerzity Wisconsin-Madison z roku 1997. Dataset obsahuje 151 záznamů hodnocení asistentů profesorů z univerzity Wisconsin-Madison. Záznamy byly posbírány během tří zimních a dvou letních semestrů a obsahují záznamy o 25 instruktorech ve 26 kurzech, vyučovaných v různě velkých třídách. Výsledné hodnocení může nabýt tří různých hodnot (nízké, střední, vysoké), přičemž všechna hodnocení jsou v datasetu zastoupena přibližně ve stejném množství.
Dataset nebyl explicitně rozdělen na trénovací a testovací data. Do trénovací části bylo zařazeno prvních 15 záznamů z každého druhu hodnocení. Zbytek záznamů datasetu tvoří testovací část.

Proměnná	Hodnota	Význam
V1	1, 2	Značí, jestli je učitel rodilý mluvčí
V2	1,2, …, 25	Kategorická proměnná označující učitele
V3	1,2, …, 26	Kategorická proměnná označující předmět
V4	1,2	Značí zimní nebo letní semestr
V5	3 .. 66	Velikost třídy
V6	1, 2, 3	Značí hodnocení třídy 1 – Nízké hodnocení 2 – Střední hodnocení 3 – Vysoké hodnocení

Tabulka 28 Proměnné datasetu Teaching
Zdroj: vlastní práce

11.1 Nejbližší sousedi

Metoda nejbližších sousedů zařadila data s přesností 50%. Konkrétní zařazení je zaznamenáno v tabulce 29.

Třída	1	2	3
Originál	34	35	37
Nejbližší sousedi	12	17	24

Tabulka 29 Zařazení NN, Teaching
Zdroj: vlastní práce

Dataset Teaching je poměrně malý, jak z hlediska počtu záznamů, tak podle počtu atributů. Díky tomu proběhla klasifikace okamžitě a v rámci desetin sekundy nezaznamenal ani jeden z počítačů žádné měřitelné výsledky.

	user	system	elapsed
PC A, B	0.00	0.00	0.00

Tabulka 30 Časová náročnost NN, Teaching
Zdroj: vlastní práce

11.2 K-nejbližší sousedi

Metoda K-nejbližších sousedů byla při kategorizaci dat poměrně úspěšná – úspěšnost správného zařazení dat se v testovaných případech pohybovala mezi 33% a 38%. Nejúspěšnějším pokusem kategorizace byla aplikace funkce, ve které byla pro parametr K zvolena hodnota 2 a 4. V těchto případech dosáhla pravděpodobnost správného zařazení shodného výsledku 37,73%. Tabulka 31 zobrazuje přesnost klasifikace při zvolených K.

Třída	1	2	3
Originál	34	35	37
K= 2	9	17	13
K= 3	7	16	17
K= 5	10	14	12
K=10	12	12	16

Tabulka 31 Zařazení KNN pro různá K, Teaching
Zdroj: vlastní práce

Algoritmus K-nejbližších sousedů se při aplikaci na dataset Teaching nijak neliší v časové náročnosti od algoritmu nejbližších sousedů, nehledě na velikost

parametru K. Ve všech testovaných případech, kdy K nabývalo hodnot 2, 3, 5 a 10, nebyl zaznamenán jiný časový údaj než 0.00.

	user	system	elapsed
K=2,3,5,10	0.00	0.00	0.00

Tabulka 32 Časová náročnost KNN pro různá K, Teaching
Zdroj: vlastní práce

11.3 SVM

Při aplikaci algoritmu Support vector machine byl vytvořen model pro klasifikaci obsahující 41 podpůrných vektorů s následujícím rozdělením do jednotlivých tříd, viz tabulka 33.

Třídy	1	2	3
Vektory	14	13	14

Tabulka 33 Rozložení vektorů SVM, Teaching
Zdroj: vlastní práce

Podle vytvořeného modelu byla kategorizována data z testovací části datasetu. Jak lze vyčíst z tabulky 34, metoda SVM dosáhla nejlepších výsledků klasifikace ze všech testovaných metod na tomto datasetu. Pravděpodobnost správného zařazení do kategorie byla 53,77 %.

Třída	1	2	3
Originál	34	35	37
Zařazení	19	16	22

Tabulka 34 Zařazení SVM, Teaching
Zdroj: vlastnípráce

Stejně jako předcházející algoritmy nejbližších sousedů a K-nejbližších sousedů, ani support vector machine nebyl při aplikaci na Teaching dataset časově náročný. Rozdíly mezi počítači A a B jsou téměř nepodstatné, pouze v setině vteřiny.

	PC A	user	system	elapsed	PC B	user	system	elapse
Model		0.03	0.00	0.00		0.00	0.02	1.61
Predikce		0.00	0.00	0.00		0.00	0.00	0.00
Celkem		0.03	0.00	0.03		0.00	0.02	1.61

Tabulka 35 Časová náročnost SVM,Teaching
Zdroj:vlastní práce

12 Shrnutí výsledků

Na čtyři rozdílné datasety byly aplikovány metody Nejbližších sousedů, K-nejbližších sousedů, Support vector machine a Logistická regrese. Datasety se od sebe navzájem lišily jak počtem záznamů, tak počtem atributů v záznamu. Zároveň data z každého datasetu patřila do různého počtu tříd.

Největší dataset z hlediska počtu atributů byl dataset Gisette, který obsahoval záznamy o 6 000 nezávislých proměnných. Na druhou stranu tento dataset obsahoval data patřící jen do dvou tříd. To umožnilo na tento dataset, jako na jediný, aplikovat algoritmus logistické regrese.
U Gisette souboru byla znát velká časová a výpočetní náročnost. Projevil se zde i největší rozdíl mezi počítači co se týče časové náročnosti. Zatímco počítač A zvládl všechny algoritmy NN a KNN aplikované na tento dataset v rámci časového intervalu do 10 minut, doba zpracovávání algoritmů na počítači B přesáhla jednu hodinu.
Tento dataset byl nejúspěšnějším příkladem klasifikace pomocí NN a KNN algoritmů, kde bylo dosaženo přes 93% shody výsledků se správným zařazením. Zato při práci s logisticou regresí počítač B zde nezvládl pracovat se všemi testovacími daty najednou, a výkonější počítač A zde dosáhl výsledku pouze 46,23% úspěšnosti.

Podle počtu záznamů byl největší dataset Pokerhand. Zde se pracovalo s 1 000 000 testovaných dat, které byly kategorizovány do 10 tříd.
Časová náročnost všech algoritmů aplikovaných na Pokerhand dataset se pohybovala do hodiny a půl, a mezi počítači se lišila v rámci desítek minut. Díky tomu lze tento dataset označit jako nevíce časově náročný, ze všech testovaných datasetů. U tohoto datasetu byla pozorována velká vyrovnanost pravděpodobnosti úspěšné kategorizace.

Dataset Teaching byl nejmenším datasetem použitým v rámci této práce – obsahuje pouze 151 záznamů, a dělí data do 3 kategorií. Algoritmus SVM zde dosáhl nejmenší úspěšnosti ve srovnání s jeho aplikací na ostatní datasety - pouhých 3,8%.

Zároveň zde byla minimální časová náročnost. Naměřené hodnoty u Teaching datasetu se lišila od nuly pouze při tvorbě klasifikačního modelu pomocí SVM, a to pouze na jednom z počítačů a jen v rámci setin vteřiny.

Čtvrtým zkoumaným datasetem je dataset Chess, skládající se z 28 056 záznamů o 6 atributech. Ze všech datasetů jsou zde data dělena do nejvíce tříd - 17. Stejně jako u datasetu Pokerhand zde byla znát poměrná vyrovnanost v pravděpodobnosti správného zařazení dat různými algoritmy. Časová náročnost se zde pohybovala okolo 1 sekundy, a mezi počítači A a B nebyl znát téměř žádný rozdíl.

Shrnutí úspěšnosti jednotlivých metod v rámci datasetů je shrnut v následující tabulce:

Dataset / Metoda	Pokerhand	Gisette	Chess	Teaching
NN	65,6%	93,6%	31,9%	50,0%
KNN (nejlepší výsledek)	69,9%	94,9%	31,5%	37,7%
SVM	55,6%	50,0%	3,8%	53,8%
Logistická regrese	-	53,8%	-	-

Tabulka 36 Shrnutí úspěšnosti metod v rámci jednotlivých datasetů
Zdroj: vlastní práce

13 Závěry a doporučení

V rámci této práce byly popsány principy vybraných metod učení s učitelem. Každá z metod byla aplikovaná na čtyři různě veliké datasety, o různém počtu atributů a tříd.
Z pozorování nelze určit, že by jedna metoda vynikala v úspěšnosti správné klasifikace dat nad ostatními metodami. Vždy záleželo na konkrétním datasetu, která z metod byla nejúspěšnější. Stejně tak se lišila i nejméně časově náročná metoda v rámci datasetu.

14 Seznam použité literatury

[1] SOUKUP Tom, DAVIDSON Ian. Visual Data Mining:Techniques and Tool for Data Visualisation and Mining. Canada: John Wiley & Sons,Inc., 2002. 382str. ISBN 0-471-14999-3.

[2] WITTEN Ian, FRANK Eibe. Data Mining:Practical Machine Learning Tools and Techniques. San Francisco: Elsevier Inc. 2005, 2nd ed., 524. ISBN 0-12-088407-0.

[3] JOHNSON, Jesse. Data Normalization[online]. April 30, 2013. [cit. 2014-05-08] < http://shapeofdata.wordpress.com/2013/04/30/data-normalization/>

[4] JOHNSON, Jesse. Nearest Neighbors Classification [online]. April 23, 2013. [cit. 2014-05-10] <http://shapeofdata.wordpress.com/2013/04/23/nearest-neighbors-classification/>

[5] JOHNSON, Jesse. K-Nearest Neighbors [online]. May 17, 2013. [cit. 2014-05-11]< http://shapeofdata.wordpress.com/2013/05/07/k-nearest-neighbors//>

[6] PIVOŇKOVÁ Alena, ŠMERÁK Petr, Rozpoznávání: Klasifikace podle nejbližších sousedů, zápis z přednášky [online], Praha:ČVUT, 2002. [cit. 2014-04-22]
<cmp.felk.cvut.cz/cmp/courses/recognition/zapis.prednasky/zapis_03/nearest-neigh03.doc >

[7] VLADIMIROV Mikhail, Using the k-Nearest Neighbors Algorithm in R [online]. November 24, 2013. [cit. 2014-05-13]
<http://blog.webagesolutions.com/archives/1164>

[8] Nearest neighbor search with kd-trees [online]. 1999. [cit. 2014-05-13] <http://www.alglib.net/other/nearestneighbors.php>

[9] Factors in R [online]. 3 Feb 2006. [cit. 2014-05-13] <https://www.stat.berkeley.edu/classes/s133/factors.html>

[10] SHANG Wenqian, HUANG Houkuan, ZHU Haibin, LIN Yongmin, WANG Zhihai, QU Youli, An Improved kNN Algorithm – Fuzzy kNN [online]. Springer Berlin Heidelberg, December 2005, [cit. 2014-11-29], online ISBN 978-3-540-31599-5

[11] OUGIAROGLOU Stefanos, NANOPOULOS Alexandros,PAPADOPOULOS Apostolos N., MANOLOPOULOS Yannis, WELZER-DRUZOVEC Tatjana, Adaptive k-Nearest-Neighbor Classification Using a Dynamic Number of Nearest Neighbors. Springer Berlin Heidelberg , October 2007, [cit. 2014-11-29], online ISBN 978-3-540-75185-4

[12] JIANG Jingqing, WU Chunguo, LIANG Yanchun, Multi-category Classification by Least Squares Suport Vector Regression, Springer Berlin Heidelberg, June 2005, [cit. 2014-12-2], online ISBN 978-3-540-32065-4

[13] YU Hwanjo, KIM Sungchul , SVM Tutorial—Classification,Regression and Ranking, Springer Berlin Heidelberg , 2012, [cit. 2014-12-3], online ISBN 978-3-540-92910-9

[14] YUE Shihong, LI Ping, HAO Peiyi, SVM classification: Its contents and challenges, Editorial Committee of Applied Mathematics , September 2003, [cit. 2014-12-3], online ISSN 1993-0445

[15] JOHNSON, Jesse. Linear separation and Support Vector Machines [online]. May 14, 2013. [cit. 2014-12-2] <http://shapeofdata.wordpress.com/2013/05/14/linear-separation-and-support-vector-machines/>

[16] MEYER David,DIMITRIADOU Evgenia, HORNIK Kurt, WEINGESSEL Andreas, LEISCH Friedrich , CHANG Chih-Chung, LIN ChihChen, Package 'e1071', September 1, 2014, [cit. 2014-12-4] <http://cran.r-project.org/web/packages/e1071/e1071.pdf>

[17] MEYER David, Support Vector Machines,[online]. [cit. 2014-12-4] <http://www.inside-r.org/node/57517>

[18] TING Kai Ming, Encyclopedia of Machine Learning – Confusion Matrix (2010), [cit. 2014-12-9] Springer US, 209 str., Online ISBN 978-0-387-30164-8

[19] JOHNSON, Jesse. Kernels [online]. April 27,2013. [cit. 2014-12-10] <http://shapeofdata.wordpress.com/2013/05/27/kernels/>

[20] JOHNSON, Jesse. Logistic regression[online]. May 17,2013 [cit. 2015-01-17] <https://shapeofdata.wordpress.com/2013/05/17/logistic-regression/>

[21] R Studio[online].[cit. 2015-01-14]<http://www.rstudio.com/products/RStudio>

[22] BACHE, K., LICHMAN, M. (2013). UCI Machine Learning Repository [http://archive.ics.uci.edu/ml/datasets/Teaching+Assistant+Evaluation]. Irvine, CA: University of California, School of Information and Computer Science.

[23] BACHE, K., LICHMAN, M. (2013). UCI Machine Learning Repository [Sciencehttps://archive.ics.uci.edu/ml/datasets/Poker+Hand]. Irvine, CA: University of California, School of Information and Computer

[24] BACHE, K., LICHMAN, M. (2013). UCI Machine Learning Repository [https://archive.ics.uci.edu/ml/datasets/Chess+(King-Rook+vs.+King-Pawn)]. Irvine, CA: University of California, School of Information and Computer Science

[25] GERMÁN, Rodríguez. Introducing R – Generalized Linear Models[online]. (1994-2014) [cit. 2015-01-20] <http://data.princeton.edu/R/glms.html>

[26] Fitting Generalized Linear Models [online]. [cit. 2015-01-20] <https://stat.ethz.ch/R-manual/R-devel/library/stats/html/glm.html>

[27] FLETCHER , Tristan. Support vector Machines Explained. [online] March 01, 2009. [cit. 2015-01-26] <http://www.tristanfletcher.co.uk/SVM%20Explained.pdf>

[28] BACHE, K., LICHMAN, M. (2013). UCI Machine Learning Repository [http://archive.ics.uci.edu/ml/datasets/Gisette]. Irvine, CA: University of California, School of Information and Computer Science

FIM UHK

UNIVERZITA HRADEC KRÁLOVÉ
Fakulta informatiky a managementu
Rokitanského 62, 500 03 Hradec Králové, tel: 493 331 111, fax: 493 332 235

Zadání k závěrečné práci

Jméno a příjmení studenta: **Eliška Rejmanová**

Obor studia: Aplikovaná informatika
Jméno a příjmení vedoucího práce: **Jiří Haviger**

Název práce:
Geometrické základy vybraných dataminingových metod

Název práce v AJ:
Geometric principles of selected datamining methods

Podtitul práce:

Podtitul práce v AJ:

Cíl práce: Popis geometrických základů vybraných dataminingových metod a jejich implementace v R, testování těchto metod na vybraných datových souborech a porovnání výsledků získaných jednotlivými metodami.

Osnova práce:
1. Úvod
2. Popis metod
3. Testování vybraných metod
4. Analýza výsledků
5. Závěr

Projednáno dne:

Podpis studenta

Podpis vedoucího práce