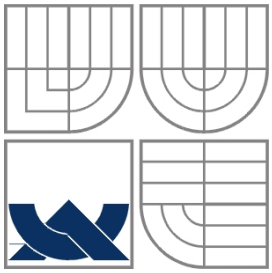
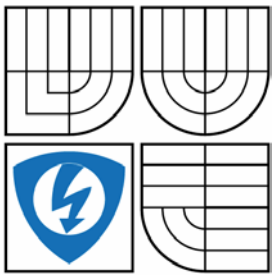


VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA ELEKTROTECHNIKY A KOMUNIKAČNÍCH TECHNOLOGIÍ
ÚSTAV AUTOMATIZACE A MĚŘICÍ TECHNIKY

FACULTY OF ELECTRICAL ENGINEERING AND COMMUNICATION
DEPARTMENT OF CONTROL AND INSTRUMENTATION

SROVNÁNÍ SPRÁVNOSTI KLASIFIKACE POMOCÍ TRADIČNÍCH MODELŮ A META-MODELŮ

COMPARISON OF ACCURACY ACHIEVED BY TRADITIONAL MODELS AND ENSEMBLE METHODS

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE
AUTHOR

ONDŘEJ ZAPLETAL

VEDOUCÍ PRÁCE
SUPERVISOR

Ing. PETR HONZÍK, Ph.D.

BRNO 2014

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

Fakulta elektrotechniky a komunikačních technologií

Ústav automatizace a měřicí techniky

Bakalářská práce

bakalářský studijní obor
Automatizační a měřicí technika

Student: Ondřej Zapletal ***ID:*** 119681
Ročník: 3 ***Akademický rok:*** 2013/14

NÁZEV TÉMATU:

Srovnání správnosti klasifikace pomocí tradičních modelů a meta-modelů

POKYNY PRO VYPRACOVÁNÍ:

Cílem bakalářské práce je na minimálně 20 datových souborech srovnat správnost klasifikace dosažené pomocí 3 různých meta-modelů a 5 tradičních modelů za srovnatelných podmínek a na základě vlastních experimentů pak rozhodnout, zda a případně za jakých okolností má smysl meta-model použít.

1) zpracujte rešerši zabývající se srovnáním obou přístupů k modelování
2) popište stručně princip fungování (meta) modelů a databáze, které budou použity při testování
3) navrhněte postup experimentu včetně metodiky vyhodnocení výsledků
4) realizujte experimentální část v programu RapidMiner
5) vyhodnoťte výsledky, pokuste se o jejich interpretaci a případné doporučení
6) naprogramujte vybraný model a srovnejte dosažené výsledky s modelem v RapidMineru

DOPORUČENÁ LITERATURA:

Hastie T., Tibshirani R., Friedman J.:The Elements of Statistical Learning: Data Mining, Inference, and Prediction, Second Edition. Springer Series in Statistics, 2013.
(http://www-stat.stanford.edu/~tibs/ElemStatLearn/)

Termín zadání: 10.2.2014 ***Termín odevzdání:*** 26.5.2014

Vedoucí práce: Ing. Petr Honzík, Ph.D.
Konzultanti bakalářské práce:

doc. Ing. Václav Jirsík, CSc.
předseda oborové rady

UPOZORNĚNÍ:

Abstrakt

Tato práce se zabývá empirickým porovnáváním tradičních modelů a meta-modelů v úlohách klasifikace. Na 20 datových souborech je statisticky porovnána přesnost 12 modelů programu *RapidMiner*.
V druhé části je popsána vlastnoručně naprogramovaná aplikace v programovacím jazyce C#, která implementuje 6 modelů. Čtyři z nich jsou porovnány s modely ekvivalentními modely programu *RapidMiner*.

Klíčová slova

Strojové učení, meta-learning, k-nejbližších sousedů, rozhodovací strom, neuronové sítě, metoda podpůrných vektorů, naivní Bayesův klasifikátor, Bagging, Stacking, Random Forest, Decision Stump

Abstract

This thesis deals with empirical comparison of traditional and meta-learning models in classification tasks. Accuracy of 12 *RapidMiner* models was statistically compared on 20 data sets.
Second part of this thesis consists of description of self-programed application in programing language C#, which implements 6 different models. Four of those are compared with equivalent models of program *RapidMiner*.

Keywords

Machine learning, meta-learning, k-nearest neighbor, decision tree, neural net, support vector machines, naive Bayes classifier, Bagging, Stacking, Random Forest, Decision Stump

Bibliografická citace:

ZAPLETAL, O. *Srovnání správnosti klasifikace pomocí tradičních modelů a meta-modelů.* Brno: Vysoké učení technické v Brně, Fakulta elektrotechniky a komunikačních technologií, 2014. 63 s. Vedoucí bakalářské práce Ing. Petr Honzík, Ph.D..

Prohlášení

„Prohlašuji, že svou bakalářskou práci na téma Srovnání správnosti klasifikace pomocí tradičních modelů a meta-modelů jsem vypracoval samostatně pod vedením vedoucího bakalářské práce a s použitím odborné literatury a dalších informačních zdrojů, které jsou všechny citovány v práci a uvedeny v seznamu literatury na konci práce.
Jako autor uvedené bakalářské práce dále prohlašuji, že v souvislosti s vytvořením této bakalářské práce jsem neporušil autorská práva třetích osob, zejména jsem nezasáhl nedovoleným způsobem do cizích autorských práv osobnostních a jsem si plně vědom následků porušení ustanovení § 11 a následujících autorského zákona č. 121/2000 Sb., včetně možných trestněprávních důsledků vyplývajících z ustanovení části druhé, hlavy VI. díl 4 Trestního zákoníku č. 40/2009 Sb.

V Brně dne: **26. května 2014** ……
podpis autora

Poděkování

Děkuji vedoucímu bakalářské práce Ing. Petru Honzíkovy, Ph.D. za účinnou metodickou, pedagogickou a odbornou pomoc a další cenné rady při zpracování mé bakalářské práce.

V Brně dne: **26. května 2014** ……

podpis autora

Obsah

Seznam Obrázků

Seznam Tabulek

1 ÚVOD

Obor strojového učení získává v posledních letech velkou popularitu. Ačkoliv je tento obor teoreticky rozpracovaný již několik desetiletí, k opravdovému průlomu došlo až s rozmachem výpočetní techniky. Protože neustále nabývá množství dat, které nás v našem životě obklopuje, je výhodné a někdy dokonce nezbytné automatizovat stále více činností, u kterých jsme donedávna byly odkázání na lidskou sílu.

Velké nadnárodní společnosti investují obrovské finanční prostředky do vývoje technologií, které jsou schopné provádět úkony donedávna prováděné pouze lidmi. Mezi činnosti, které stroje dnes již běžně zvládají, patří rozeznávání řeči, obrazu a psaného textu. Tohoto ohromného pokroku bylo možné dosáhnut hlavně díky strojovému učení.

Mezi jednu z nejužitečnějších úloh disciplíny strojového učení patří klasifikace. Cílem této práce je empiricky porovnat tradiční a meta-learningový přístup k modelování klasifikačních úloh. Pro tento účel byli vybráni populární zástupci z obou kategorií a otestováni na 20 datových souborech.

Hlavní část porovnání byla provedena pomocí programu RapidMiner. Zároveň však byla v rámci práce vytvořena aplikace v programovacím jazyce C#, která implementuje několik modelů obou kategorií.

Cílem práce je porovnat modely a pokusit se stanovit, zda a za jakých okolností se vyplatí meta-learningové modely použít.

1.1 Definice klasifikační úlohy

Strojové učení (*Machine Learning*) je vědní disciplína zabývající se algoritmy a postupy, které umožňují strojům proces učení. Jde o jednu z disciplín, která spadá do oblasti *umělé inteligence* (*Artificial Intelligence*). Pro svoje účely do velké míry využívá oblastí pravděpodobnosti, statistiky a optimalizace. Strojové učení můžeme dále rozdělit na *učení s učitelem* (*Supervised learning*) a *učení bez učitele* (*Unsupervised learning*). Učení s učitelem zahrnuje nejčastěji úlohy *klasifikační* (*Classification*) a *regresní* (*Regression*). Regrese spočívá v hledání funkčních závislostí v datech, které umožnují predikovat výstupní kvantitativní veličinu. Princip klasifikace je do jisté míry obdobný, ale je hledána funkční závislost dat, která od sebe efektivně separuje instance různých tříd, tedy kvalitativních veličin. Další text je zaměřen pouze na představení různých klasifikačních modelů úlohy, ale podstatná část z nich má ekvivalentní regresní modely. [14]

Pro zjednodušení jsou v teoretické části práce definovány postupy pro tzv. *binární klasifikaci*, tedy klasifikaci do dvou tříd, nejčastěji reprezentovány dvojicí hodnot $\{0,1\}$, případně $\{-1,1\}$. Všechny rozebírané modely buďto zvládají klasifikaci do více tříd, nebo lze aplikovat postup klasifikace *one-versus-the-rest*, kdy je vyhodnocována jedna třída proti zbytku, a nebo *one-versus-one*, kdy je vyhodnocována kombinace všech možných dvojic. [26]

Je nutné zmínit, že toto jednoduché řešení má své neduhy. Při opakovaném dělení vstupního prostoru se mohou v souhrnném řešení vyskytovat oblasti, které jsou součástí několika různých podprostorů dílčích řešení, a tak vzniká nejistota správného řešení. [19]

1.1.1 Problematika dat

Nejobecnějším způsobem dělení dat je dělení na data kvantitativní a kvalitativní. Kvantitativní data jsou číselné hodnoty, u kterých lze definovat velikost a rozdíl mezi dvěma hodnotami. Kvalitativní data mohou být nominální a ordinální. Ordinální data lze seřadit podle velikosti, ale nejde určit, jaký je mezi dvěma hodnotami rozdíl, v případě nominálních dat nejde určit ani pořadí. [14]

Všechny modely pracují s datovými soubory, tedy s naměřenými daty, reprezentujícími reálnou úlohu. Jednotlivé prvky datového souboru budou v následujícím textu nazývány instance. Pod tímto pojmem je myšlena položka dat, daná vstupními parametry a výstupní veličinou, v případě klasifikace kvalitativní. Vstupní parametry budou dále též nazývány atributy a mohou být jak kvantitativní, tak kvalitativního charakteru.

Modely strojového učení reprezentují nalezené závislosti v datech. Tyto závislosti jsou využívány k predikci výstupní veličiny pro nová data.

Data mohou být výsledky měření, vyhodnocená data ankety, data nasbíraná interakcí se zákazníky atp. Při odvozování skutečností z těchto dat je třeba brát ohled na možné nesprávné informace, které data obsahují. Data obsahují chyby měření, výsledky ankety obsahují lživé výpovědi, nebo dokonce v datech úplně chybí důležitá informace ovlivňující vyhodnocovanou veličinu. V mnohých úlohách bývají data velmi zašuměná, tedy obsahující velký podíl chybových dat (v angličtině *outliers*), nebo některé instance nemají hodnoty u některých atributů.

U problémů skutečného světa je velice častý problém v nedostatku dat, protože získávání kvalitních dat je finančně i časově náročné. Tímto faktem je také ovlivněn výběr modelu, protože některé modely snášejí omezené množství dat lépe, než jiné. Zároveň je výběr modelu ovlivňován typem vstupních dat, tedy hodnotami atributů, protože ne všechny modely umí pracovat jak s kvalitativními, tak kvantitativními veličinami.

1.1.2 Učení modelu

Učení modelu spočívá v hledání funkční závislosti mezi vstupními veličinami a výstupní veličinou, která co nejlépe reprezentuje danou úlohu. Při učení s učitelem je potřeba stanovit chybovou funkci, což je vlastně zpětná vazba, která určuje, jak dobře se daný model blíží tréninkovým datům.

V případě regresní úlohy může být příkladem *chybové funkce* $L\big(y, f(Xx)\big) = (y - f(x))^2$, kde y představuje skutečnou výstupní hodnotu udanou daty a $f(x)$ je výstupní hodnota modelu získaná na základě vstupních dat. Protože u diskrétních veličin nelze stanovit vzdálenost mezi jejími hodnotami, je v případě klasifikační úlohy nahrazena jinou penalizační funkcí. [13]

Veličina G může nabývat K různých diskrétních hodnot. Je zvykem chybovou funkci reprezentovat maticí $\boldsymbol{L}$ (*nákladová matice*) o rozměrech $K \times K$. $\boldsymbol{L}$ má na diagonále hodnotu 0 a všude jinde pozitivní hodnoty. $\boldsymbol{L}(k, l)$ je cena, kterou je placena nesprávná klasifikace, tedy například klasifikace skutečné třídy G_k jako nesprávnou třídu G_l. Obvykle je cena špatné klasifikace stanovena stejná pro všechny třídy, ale v některých reálných problémech je tato rozdílná penalizace žádoucí. Například v případě klasifikování, zda pacient trpí zákeřnou chorobou, je bezpochyby vhodnější, aby model dával větší důraz na chybu falešné negativní klasifikace, než je tomu u falešné pozitivní klasifikace. [13]

Ne ve všech datových souborech jsou všechny třídy zastoupeny stejně. Pokud je v datech četnost některé třídy v porovnání s ostatními třídami nízká, je pravděpodobné, že bude správnost její predikce omezená. Pokud je tomu tak, je možné pomocí nákladové matice zvýšit významnost této třídy.

1.2 Rešerše tradičního a ensemble přístupu k modelování

V rámci rešerše nebyla nalezena žádná studie, která by plošně porovnávala tradiční a meta-lerningové modely na větším počtu datových souborů. Nejčastěji byly nalezeny články porovnávající tradiční modely s jeho ensemble verzí.

Mezi všemi tradičními modely je rozhodovací strom konzistentně mnoha zdroji ([13], [19], [21] a [26]) označován jako ten, který má největší přínos z meta-learningového přístupu.

Pro model *k-nejbližších sousedů* byl v článku [12] navrhnut způsob jak zlepšit jeho robustnost pomocí *Boostingu*.

V [25] je na 13 datových souborech dosaženo lepších výsledku pomocí *Bagging SVM ensemble* oproti jeho prosté formě, na druhou stranu v [24] nebylo významné zlepšení *SVM* pomocí ensemble metod prokázáno.

V článku [11] je porovnáváno 11 tradičních a 4 ensemble modely na problému odhadování podílu alkoholu v hroznovém víně během dozrávání. Nejlepších výsledků dosáhl *Bagging* s base modelem *REPTree*. Článek však porovnává modely pouze na jednom datovém souboru.

Další článek porovnávající tradiční metody s meta-learnigovými je [3] ve kterém je zkoumáno rozeznávání skládání proteinů. V tomto článku byla použita téměř identitcká sestava modelů jako v této bakalářské práci a to (*k-nejbližších sousedů*, *Naivní Bayesův klasifikátor*, *rozhodovací stromy*, *metoda podpůrných vektorů*, *Neuronové sítě*, *Bagging* a *Boosting*). Podle dosažených výsledků se jako nejlepší jevil *Boosting* s *rozhodovacím stromem*. Nebylo prokázáno žádné výrazné zlepšení pří vytváření *ensemble Neuronových sítí* a v případě *SVM* došlo dokonce ke zhoršení.

1.3 Použitá terminologie

(Kapitola 1.1)

Strojové učení (*Machine Learning***)** – vědní disciplína zabývající se algoritmy a postupy umožňující strojům proces učení;

Umělé inteligence (*Artificial Intelligence***)** – vědní obor zabývající se technologií, která projevuje jisté známky inteligence;

Učení s učitelem (*supervised learning*) – typ učení, při kterém dochází k zpětné vazbě na základě úspěchů a neúspěchů;

Učení bez učitele (*unsupervised learning*) – typ učení bez zpětné informační vazby úlohy;

Regresní úloha (Regression) – úloha učení s učitelem, na základě které dochází k určování kvantitativní výstupní veličiny;

Klasifikační úloha (Classification) – úloha učení s učitelem, na základě které dochází určování kvalitativní výstupní veličiny;

Binární klasifikace – klasifikační úloha s výstupní třídou nabývající dvojice hodnot;

One-versus-the-rest, one-versus-one – postupy pro převedení binární klasifikační metody na metodu klasifikující libovolný počet tříd;

Instance – jeden datový záznam;

Atributy – vstupní parametr datového záznamu;

Outliers – instance, která není reprezentativní vzorek zkoumané závislosti;

Chybové funkce – zpětná vazba procesu učení. Udává informaci, jak dobře model reprezentuje tréninková data;

Nákladová matice – kvalitativní ekvivalent chybové funkce;

(Kapitola 2.1)

Instance based learning – učení na bázi instancí;

Líné/neparametrické metody (*lazy/non-parametric methods*) – metody, u kterých v tréninkové fázi nedochází k učícímu procesu;

Normalizace – proces transformace vstupních hodnot do určeného rozsahu;

Kletba dimenzionality (*curse of dimensionality*) – neschopnost modelu dobře se vypořádat s problémy s velkou dimenzí vstupního prostoru;

Euklidovská metrika - předpis definující vzdálenost dvou prvků v n–rozměrném prostoru;

(Kapitola 2.2)

Decision Tree – rozhodovací strom;

Random Tree – náhodný strom;

Decision Stump – speciální rozhodovací strom s hloubkou 1;

Uzel (*node*) – základní stavební prvek stromu;
Větev (*branch*) – spojení mezi dvěma uzly;
List (*leaf*) – uzel na konci struktury;
Kořenový list (*Root Node*) – uzel na vrcholu struktury stromu;
Hloubka (*depth*) – počet uzlů mezi kořenovým uzlem a listem;
Informační zisk (*information gain*)**, Gini index** – jsou to kritéria pro hledání optimální rozhodovací hranice;
Prořezávání (*pruning*) – redukce přebytečných větví stromu za účelem zlepšování klasifikace;
Error-Complexity, Reduced-Error - Různé algoritmy pro prořezávání rozhodovacích stromů;
Bias – úroveň odchylky modelu od tréninkových dat;
Variance – úroveň rozptylu jednotlivých řešení modelu;
Ensemble, **meta-learning** - vzájemně zaměnitelné pojmy popisující přístup k modelování;
Přeučení (*overfitting*) – model je naučen na datovém souboru i závislosti, které v datovém souboru neexistují;

(Kapitola 2.3)

Neural Net – neuronová síť;
Neuron – základní výpočetní jednotka sítě inspirovaná funkcí buněk centrální mozkové soustavy;
Preceptron – model neuronové sítě s jednou skrytou vrstvou;
Vícevrstvý preceptron (*multilayer preceptron*) – model neuronové sítě obsahující více než jednu skrytou vrstvu;
Aktivační funkce - tato funkce obecně transformuje vstupní proměnné blíže nespecifikovaným způsobem na výstupní, běžně se používá *Sigmoid funkce*;
Softmax – speciální druh aktivační funkce používaný na výstupu neuronové sítě;
Back – propagation – algoritmus učení neuronové sítě;
Váhy (*weights*) – parametr určující vliv jednotlivých prvků sítě;
Suma čtverců odchylek (*sum-of-squared errors*), **vzájemná entropie** (*cross entropy*) – chybové funkce používané neuronovými sítěmi;
Lineární logistická regrese (*linear logistic regression*) – lineární klasifikační model;
Pravidlo předčasného zastavení (*early stopping rule*) – omezení počtu iterací algoritmu *back-propagation*;
Gradient descent metody – metody hledání lokálního minima sledováním poklesu gradientu, tohoto postupu využívá algoritmus *back-propagation*;
Learning rate – parametr, který určuje, jak moc jsou aktualizovány váhy v každém kole *back – propagation*;
Online – způsob učení modelu;
Line search – metoda pro optimalizování *learning rate* parametru;
Weight decay – penalizace chybové funkce neuronové sítě předcházející přeučení;
Ridge regression – metoda penalizace chybové funkce pro lineární modely;

(Kapitola 2.4)

Naive Bayes Classifier – naivní Bayesův klasifikátor;
Bayesův teorém – pravděpodobnostní teorém o ovlivňování podmíněné pravděpodobnosti jeho opačným jevem;
Apriorní (prior) pravděpodobnost – pravděpodobnost představující předpoklad o charakteristice problému;
Věrohodnost (*likelihood*) – pravděpodobnost jednotlivých hypotéz;
Posteriorní (posterior) **pravděpodobnost** – výsledná pravděpodobnost představující normalizovanou kombinaci apriorní pravděpodobnosti a věrohodnosti;

Maximum a posterior (MAP) – pravidlo pro určování klasifikační třídy na základě posteriorních pravděpodobností;

Bayesův optimalní klasifikátor (*Bayes optimal classifier*) – teoretický postup určení klasifikační třídy na základě bayesova teorému;

Log-Sum-Exp – metoda převedení čísel do logaritmické soustavy pro práci s velice malými čísly;

(Kapitola 2.5)

Support Vector Machines (SVM) – metoda podpůrných vektorů;

Off-the-shelf model – model, který je schopný fungovat bez komplikovaného nastavení;

Sparse řešení – řešení, které je závislé na malém počtu parametrů, v tomto případě instancí;

Linear Discriminat Analys (LDA) – lineární klasifikační model;

Marginální oblast – oblast, ve které se nenacházejí instance mezi rozhodovací hranicí a prvním podpůrným vektorem;

Slack proměnné – proměnné umožňující výskyt instancí na nesprávné straně rozhodovací hranice;

Quadratic programming with linear inequality constraints – speciální případ matematického optimalizačního problému;

Lagrangeovy multiplikátory – metoda řešení problému hledání extrému funkce;

Kernel metody – metody využívající kernel funkcí, transformují problému na problém ve vyšší dimenzi bez nutnosti transformace souřadnic původních dat;

Nadrovina (*hyperspace*) – součást prostoru s n-1 dimenzí;

Inner product- matematická operace dvou prvků;

(Kapitola 3)

Boosting, Bagging – homogenní ensemble metody;

Slabý (*weak*) **model** – model, který je jen o něco lepší než náhodný výběr;

Silný klasifikátor (*strong classifiers*) - model, který sám o sobě dosahuje dobrých klasifikačních schopností;

Kombinace klasifikátorů (*combining of classifier*) – metoda kombinující různé modely s cíle získání lepší predikce;

Soubor slabých modelů (*ensambles of week learners*) – model kombinující různé slabé modely s cíle získání modelu s velkou predikční schopností;

Rozpoznávání vzorů (*pattern recognition*) – odvětví umělé inteligence;

Base model – dílčí model ensemble modelu;

Committe – soubor modelů;

(Kapitola 3.1)

AdaBoost, MadaBoost, BrownBoost, **FilterBoost** – varianty *Boosting* modelů

Sekvenční ensemble (sequential ensemble) – ensemble model, ve kterém jsou base modely vytvářeny postupně;

Aditivní modelování – postup modelování, ve kterém jsou modely aditivně spojovány;

Exponenciální chybové funkce – druh chybové funkce;

Occamovo ostří (Occam's razor) – teoretické kritérium pro výběr správného modelu;

Bayesian learning – Bayesovské učení;

(Kapitola 3.2)

Paralelní ensemble (*parallel ensemble*) - ensemble model, ve kterém jsou base modely vytvářeny nezávisle na sobě;

Bootstrap – metoda určování chyby modelu;

Bootstrap rozloženích – rozložení datového souboru vygenerované pomocí *bootstrap* metody;
Hlasování (voting) – kombinování predikce hlasováním každého modelu;
Random Forest, **VR-Tree** – speciální případy *Bagging* modelu;

(Kapitola 3.3)

Crisp label – vektor obsahující počet prvků, rovnající se počtu klasifikačních tříd;
Majority Voting, **Plurality Voting**, **Weighted Voting**, **Soft Voting**, **Stacking** – kombinační metody;
LOOCV validace – vynech-jednu-instanci křížová validace;
Modely první úrovně (*first-level learners*) – název base modelu pro *Stacking* metody;
Model druhé úrovně (*second-level learner*) – model kombinující base modely pro *Stacking* metody;

(Kapitola 4.1)

Poměr rychlosti-přesnosti-komplexnosti (speed-accuracy-complexity tradeoff) – kritéria výběru modelu;
Misclassifiaction rate – poměr špatně klasifikovaných dat;
Křížové validace (*cross validation*) - metoda určení přesnosti modelu;

(Kapitola 4.2)

5-fold, **10-fold** –Typy křížové validace
Shuffeled sampeling, stratified sampling – metody dělení datového souboru;

1.4 Seznam použitých notací

(Kapitola 1.1)

$f(x_i)$ – funkční závislost reprezentující model (predikce modelu na instanci i)
y_i- skutečná třída instance i.
$L(\cdot,\cdot)$ – libovolnou chybová funkce.
$\mathbf{L}(\cdot,\cdot)$ – nákladová matice
$\boldsymbol{x_i}$ – vektor vstupních veličin
D_t – množina tréninkových dat

(Kapitola 2.1)

p, d – počet vstupních atributů
N – počet instancí
N_k – instance *k*-tého nejbližšího souseda
$a_j(\boldsymbol{x}_i)$ – hodnot *j*-tého atributu vstupního vektoru $\boldsymbol{x}_i$.
$d(\cdot,\cdot)$ – metrika definující vzdálenost bodů.

(Kapitola 2.2)

$H(S)$ – entropie
$I(A,S)$ – informační zisk
S_i – podmnožina datového souboru při tvorbě rozhodovacího stromu.
c – klasifikační třída

(Kapitola 2.3)

Z_m – skrytá jednotka (*hidden unit*) neuronové sítě
T_k – výstup třídy k neuronové sítě
$\sigma(v)$ – Aktivační funkce *sigmoid* neuronové sítě
$g_k(T)$ – aktivační funkce *softmax*
α_m – váhy vstupů neuronové sítě
β_m – váhy výstupu skryté vrstvy neuronové sítě
$R(\theta)$ – suma čtverců odchylek (*sum-of-squared error*) chybová funkce
γ_r – *learning rate*
$J(\theta)$ – penalizace chybové funkce
λ – nastavitelný parametr penalizace

(Kapitola 2.4)

$P(y|\boldsymbol{x})$ – podmíněná pravděpodobnost
$\propto$-proporcinální

(Kapitola 2.5)

β – parametr roviny v p rozměrném prostoru
ξ_i – *slack* proměnná
$\text{sign}(x)$ – funkce vrací 1 v případě že x nabývá pozitivních hodnot -1 v případě negativních hodnot
M – je šířka marginálního pásu

(Kapitola 3.1)

$G(x)$ – *ensemble* model (*Boosting* a *Bagging*) nebo obecně klasifikační model.
$G_m(x)$ – base model
β_m váha přírůstku modelu platí $\alpha_m = 2\beta_m$.
α_m- váha base modelu

(Kapitola 3.2)

$G_b(x)$ – base model (Bagging)

(Kapitola 3.3)

$H(\boldsymbol{x})$ – ensemble model
h_m – base model
ω_m – váha base modelu
$\hat{f}_m^{-i}$ – *base* model trénovaný na všech tréninkových datech kromě instance i

(Kapitola 4.1)

err – chyba modelu
$\boldsymbol{I}(\cdot)$ – vrací hodnotu 1, pokud je $\cdot$ pravdivé a 0 v opačném případě

(Kapitola 4.2)

$\hat{f}^{-\kappa(i)}$ – obecný model trénovaný na všech kromě i-té instance

(Kapitola 4.3)

C^{-i} – soubor modelů trénovaných na Bootstrap rozložení neobsahujícím instanci i
$\hat{f}^{*b}$ – model trénovaný na datech vygenerovaných *bootstrap* rozložením

(Kapitola 5.4)

p – obecně pravděpodobnost
μ – střední hodnota normálního rozložení
σ^2 – rozptyl

2 TRADIČNÍ MODELY

Tradiční modely spočívají ve vytvoření jednoho modelu pro účel klasifikace. Ačkoliv jak bude v dalších kapitolách ještě zmíněno, není jednoduché pomocí této definice striktně rozhodnou u všech modelů, zda patří mezi tradiční modely, nebo ne.

Tradiční modely se třídí podle velmi bohaté škály různých kritérii. V této kapitole jsou rozebírány zástupci nejznámějších a nejpopulárnějších z nich.

2.1 Učení založené na instancích (Instance based learning)

Metody tohoto typu jsou založeny na předpokladu, že podobnost instancí ve vstupních datech, značí podobnost v datech výstupních. Vstupními daty jsou myšleny hodnoty jednotlivých atributů a daty výstupními jsou klasifikační třídy. Podobnost je vyhodnocována ve smyslu vzdálenosti neboli lépe blízkosti. Tato vzdálenost je definována metrikou, běžně euklidovskou. Na základě tohoto předpokladu lze velice jednoduše a efektivně klasifikovat.

Patří mezi takzvané líné (lazy) metody, někdy také nazývané neparametrické. Pro líné metody je specifické, že nevytváří žádný model. Výpočet je vždy prováděn až při určování nového prvku podle uložených instancí. Z toho zároveň plyne jeho slabá schopnost extrapolace, protože model je schopný efektivně predikovat pouze v prostoru, ve kterém je dostatečné pokrytí uloženými záznamy. [14]

S ohledem na výpočetní náročnost mají tyto metody svá úskalí. Se stoupajícím počtem instancí se hledání blízkých sousedů stává stále náročnějším problémem, který je v případě velkých datových souborů potřené řešit. Obecně jsou dva přístupy řešení tohoto problému, které je možné i kombinovat. První přístup, je zmenšení oblasti, ve které se hledají blízké instance, takže je vstupní prostor rozdělen do ne nutně disjunktivních podmnožin, ve kterých se hledají blízké instance, a tím lze efektivně zmenšit náročnost úlohy. Druhá možnost je, uchovávat pouze podmnožinu instancí v paměti modelu, a ty potom používat pro určování třídy. Výběr podmnožiny instancí je možné provádět na základě příspěvku k určování třídy, nebo hustoty pokrytí vstupního prostoru. [14]

Za předpokladu, že není žádoucí některý atribut zvýhodnit nebo naopak penalizovat, je pro správnou funkci instančních modelů důležité, normalizovat vstupní data. Protože je hledána podobnost ve vstupním prostoru na základě vzdálenosti, je žádoucí, aby měly všechny atributy stejný podíl na určování výstupu. Toho lze dosáhnout transformováním rozsahu hodnot každého atributu na nové rozložení v definovaném rozsahu (například 0-10). V praxi to může představovat problém, protože není zaručeno, že klasifikovaná data budou v rozsahu původních vstupních hodnot. V tomto případě je nutné stanovit, jak bude nakládáno s daty, která jsou mimo rozsah [14].

Hlavním problémem metod, hledajících blízké instance ve stavovém prostoru, jsou úlohy s velkým množstvím atributů. Počet atributů určuje počet dimenzí stavového prostoru. Se zvyšujícím se počtem dimenzí přestává být koncept blízkého okolí bodu jednoduše identifikovatelný. Tento fenomén se nazývá *kletba dimenzionality* (*curse of dimensionality*) poprvé představen v [23].

2.1.1 K-nejbližších sousedů (k-nearest-neighbor)

K-nejbližších sousedů je jeden z nejjednodušších modelů patřících do skupiny učení na bázi instancí. Pro učení jsou použita veškerá dostupná data. Třídu $G(x)$ určíme podle toho, do jaké třídy patří K-nejbližších sousedů. Hledání nejbližšího souseda N_k v p-dimenzionálním prostoru podle euklidovské metriky, lze matematicky popsat jako

$$N_k = \arg \min_{x_i \epsilon D_t} \sqrt{\sum_{j=1}^{p} \left(a_j(\boldsymbol{x}_i) - a_j(\boldsymbol{x}_n)\right)^2}, \tag{2.1.1}$$

kde D_t je množina tréninkových dat obsahující prvky x_i. K prvku x_n se hledá nejbližší soused. Veličina $a_j(x)$ představuje hodnotu j-tého atributu vstupního vektoru $\boldsymbol{x}$. Počet dimenzí vstupního prostoru je značen p. [14]

Takto je nalezeno K - nejbližších sousedů. Predikovaný label je potom určen jako nejčastěji zastoupená třída mezi sousedy. V případě, kdy některé třídy mají stejný počet zástupců, je rozhodující nejbližší prvek z nerozhodných tříd. Podle [21] je možné se nerozhodných situaci vyvarovat použitím lichých hodnot K, což spolehlivě funguje pouze pro binární klasifikaci.

Alternativou k nastavení, ve kterém mají všechny prvky v okolí určovaného bodu stejný podíl na klasifikaci prvku, je zohledňování vlivu jednotlivých sousedů podle jejich vzdálenosti. Platí tedy, že vzdálenější prvky budou mít menší vliv na klasifikace třídy. Jedna z možností, jak vyjádřit působení prvku podle vzdálenosti, je definice váhy

$$w_i = \frac{1}{d(x_i, x_n)^2}. \tag{2.1.2}$$

Váha je tedy převrácená hodnota umocněné euklidovské vzdálenosti $d(x_i, x_n)$ mezi prvky x_i a x_n. Výsledná třída tedy odpovídá třídě s největší souhrnnou váhou mezi nejbližšími sousedy.

Volbou parametru K lze ovlivňovat generalizační schopnost modelu. Při $K = 1$ je chyba modelu na tréninkových datech rovna 0, ale výsledná dělící hranice je velice členitá a je velice nepravděpodobné, že by příliš dobře odhadovala skutečné závislosti dat, protože model je v tomto případě citlivý na šum v datech. Tento fakt lze řešit zvyšováním K. Vhodné K lze nalézt pomocí *křížové validace*.

Pro úlohy s malým počtem atributů je však KNN model velice efektivní a je dnes stále používaný hlavně díky jednoduché implementaci [13]. Jedna z jeho nevýhod je, že výsledný model lze jen těžko interpretovat, především v datových souborech s větším počtem dimenzí, kdy selhává lidská intuice představy vícerozměrného prostoru.

2.2 Rozhodovací Stromy (Decision Trees)

Rozhodovací Stromy jsou nelineární hierarchické systém, umožňující nalezení a uložení znalostí a jejích následné využití k analýze nových dat [14]. Tento model je inspirován vzhledem stromu otočeného vzhůru nohama a na rozdíl od vývojového diagramu, nejsou v jeho struktuře žádné smyčky. Struktura se typicky skládá z uzlů

(*node*) a větví (*branch*). Jsou tři základní druhy uzlů: kořenový uzel (*Root Node*), uzel a list (*leaf*). Kořenový uzel je vždy jenom jeden a je na vrcholu struktury. Pomocí větví jsou k němu navázány další uzly, kterých může být libovolný počet. Na konci struktury je vždy tak zvaný list. Tedy uzel, ze kterého nevede žádná další větev směrem dolů, a který obsahuje informaci o klasifikované třídě. Počet uzlů mezi kořenovým uzlem a listem se nazývá hloubka (*depth*). Nejdelší rameno určuje hloubku stromu.

Každý uzel, kromě listu, obsahuje informaci, podle které se třídí data na podmnožiny. Funkce stromu je následující. Je dán naučený (tímto je myšleno, že struktura je hotová) strom a nová instance. Postupně se prochází uzly. Začíná se na vrcholu stromu v kořenovém uzlu a postupuje se do nižší úrovně po větvi, která je určená podmínkou uzlu. Tento proces je opakován tak dlouho, dokud není dosažen list. Podle listu je pak klasifikovaná instance. Vytváření struktury stromu, neboli jeho učení je možné dělat ručně z tréninkových dat. To je ovšem velice pracné, a proto se v drtivé většině případů využívá algoritmu pro jeho automatickou stavbu. Postupů pro učení stromů je velké množství. *Kritériem jejich odlišnosti jsou zejména topologie, typy vstupních a výstupních proměnných. Jeden typ stromu však muže být generován různými postupy – algoritmy* [14].

2.2.1 Algoritmus pro stavbu stromu

Úkol algoritmu, který automaticky generuje strom, je výběr vhodného atributu pro rozdělení tréninkových dat. U stavby stromu se nejčastěji využívá redukce entropie, nebo-li maximalizace *informačního zisku* (*information gain*). Alternativou informačního zisku je například Gini Index. [13]

Entropie $H(S)$ v uzlu je definována jako

$$H(S) = -\sum_{c \in C} p(c) \log_2 p(c), \tag{2.2.1}$$

kde $p(c)$ je podíl třídy c v tréninkovém datovém souboru. Snížení entropie, nebo-li informační zisk po rozdělení dat v uzlu podle atributu A na $S_1, \ldots, S_k$ podmnožin je

$$I(A,S) = H(S) - \sum_{i=1}^{k} \frac{|S_i|}{|S|} H(S_i). \tag{2.2.2}$$

Pro kvantitativní veličiny vstupů se nejčastěji volí $k = 2$, což znamená, že uzel je vždy rozdělen na dva. Algoritmus tedy prověřuje všechny atributy a hledá takovou hodnotu atributu, která nejlépe dělí tréninkový datový soubor (přináší největší informační zisk). Na každou z podmnožin S_i, je tento postup aplikován opakovaně tak dlouho, dokud není dosaženo ukončovací podmínky. [7]

Podmínka může představovat různé druhy kritérií. Nejjednodušší je stanovení podmínky tak, aby se strom rozvětvoval tak dlouho, dokud v uzlu nezůstanou data pouze jedné třídy. Pokud bude v platnosti tato ukončovací podmínka, bude strom dokonale klasifikovat tréninková data, tedy jeho tréninková chyba bude 0, ale pravděpodobně si nepovede tak dobře na datech testovacích. To je způsobeno tím, že strom byl naučen závislosti v datech, která se v nich ve skutečnosti nevyskytují. Pokud je strom naučen závislostem vycházejícím z těchto dat, dochází k jeho přeučení. Problematika zabývající se řešením přeučení rozhodovacích stromů je rozsáhlá a

přesahuje rozsah této práce, proto zde budou rozebrány pouze některé základní způsoby jak přeučení předcházet.

Do jisté míry může být řešením tohoto problému méně přísná ukončovací podmínka, například, že k vytvoření listu je potřeba nejméně 90% podíl jedné třídy. Zároveň sem patří například i minimální velikost uzlu pro provedení rozdělení nebo minimální velikost listu, myšleno ve smyslu množství dat, které do nich spadá. Tím je zvolněna přísnost a výsledný strom není tolik rozvětvený a i když stoupne chyba na tréninkových datech, je pravděpodobné, že strom bude lépe generalizovat na testovacích datech. Tyto parametry lze například nastavit pomocí křížové validace.

Poněkud sofistikovanější a účinnější způsob, jak zvýšit generalizační schopnosti, je takzvané *prořezávání* (*pruning*). Jsou v zásadě dva typy prořezávání *před prořezávání* (*pre-pruning)* a *prořezávání* (*post pruningp*). [26]

Pre-pruning se provádí při stavbě stromu a lze jím nahradit ukončovací podmínku. *Pre-pruning* spočívá v testování každého uzlu a porovnávání, jestli vytvoření dceřiných uzlů přinese nějaké zlepšení v predikci. Pokud tomu tak není, je z uzlu vytvořen list. Příkladem tohoto kritéria může být stanovení minimálního informačního zisku. Lze předpokládat, že přeučený strom používá, ke konci stromu irelevantní atributy pro provedení rozvětvení. Pokud tomu tak skutečně je, informační zisk takového rozdělení bude malý. Tím pádem lze stanovit kritérium minimálního informačního zisku. [21]

Post pruning se provádí až po vytvoření struktury stromu. Většina těchto technik potřebuje rozdělit tréninková data a na dvě části. Ačkoliv je možné aplikovat nejprve *pre-pruning* a výsledný strom prořezat na základě nezávislého datového souboru. Některé postupy jako například *Error-Complexity* prořezávání se provádí na stromu, který má v každém koncovém uzlu pouze instance jedné třídy. Postup u *Error-Complexity* prořezávání spočívá ve vytvoření několika nových stromů s rozdílnou důkladností prořezávání a pomocí nezávislých dat vybrat ten, který má nejmenší klasifikační chybu. U *Reduced-Error* prořezávání jsou nezávislá data použita rovnou na prořezávání. Pro každý uzel je vyhodnoceno, jaká je jeho chyba, pokud je uzel nahrazen listem a porovnána s neprořezaným stromem. Pokud je jeho klasifikační chyba menší, když je prořezán, je tento uzel nahrazen listem. [18]

Modely rozhodovacích stromů jsou omezeny na vzájemně ortogonální rozhodovací hranice, přesto však mají malý *bias*. Pokud je ukončovací podmínka nastavena tak, aby docházelo k větvení tak dlouho, dokud nejsou všechny instance správně klasifikovány, dostaneme se do stavu kdy *bias* modelu je 0. Model potom bude mít velkou tzv. *varianci*. Právě tato variance se bude podílet na snížené generalizační schopnosti modelu. Snížením této variance je dosaženo úspěchu u *ensemble* metod, jako je například *Bagging,* jak bude rozepsáno podrobněji v kapitole zabývající se *ensemble* metodami [13].

2.2.2 Náhodné stromy (Random Trees)

Náhodné stromy jsou rozhodovací stromy, které mají odlišný postup hledání nejlepších atributů pro rozdělení datových souborů. Náhodné stromy do procesu zavádějí náhodu. Z množiny atributů je vybrána náhodná podmnožina. Její velikost je parametr, který lze měnit. V každém uzlu jsou nejdříve náhodně vybrány parametry a poté se postupuje stejně jako v klasickém rozhodovacím stromu a vybírá se hodnota,

například podle největšího informačního zisku. Ukončovací podmínka je klasifikace všech instancí do správné třídy, neprovádí se žádná forma prořezávání. [7]

Výsledný strom většinou není příliš dobrý klasifikátor a běžně se nepoužívá jako jednoduchý model. Jeho hlavní využití je vhodné v ensemble modelech. Například v *Bagging ensemblech* a hlavně v modelu *Random Forest*, který bude vysvětlen později. [7]

2.2.3 Decision Stumps

Decision Stump je další model založený na principu rozhodovacích stromů. Je to vlastně rozhodovací strom s hloubkou 1. Protože dochází k rozdělení dat vždy na dvě části pomocí jedné lineární hranice, lze na *decision stumps* pohlížet jako na lineární model. Model není příliš samostatně použitelný, stejně jako je tomu v případě náhodného stromu. *Decision stump* je s oblibou využíván u *Boosting* metod a to hlavně v případě modelu *AdaBoost.* [26]

2.2.4 Shrnutí modelů rozhodovacích stromů

Modely rozhodovacích stromů mají jednu velice silnou stránku a to je interpretovatelnost jejich výsledků. Grafické znázornění stromů je jednoduše čitelné a lze z něj odpozorovat závislosti jednotlivých veličin. To je velice významné v mnoha oblastech vědních i běžném životě. Patří mezi ně například medicína, finance, bankovnictví anebo marketing. Jejich interpretovatelnost je hlavní důvod, proč jsou do dnes ještě běžně používány.

2.3 Neuronové sítě (Neural nets)

Teorie neuronových sítí má svoje počátky v roce 1943, kdy Warren McCulloch a Walter Pitts představili tzv. *McCulloch-Pitts* (*M-P*) model [26]. Během mnoha let zkoumání této problematiky se termín *neuronových sítí* rozvinul natolik, že zahrnuje širokou škálu modelů a učících metod. [13]

Neexistuje žádná formální definice *umělé neuronové sítě*. V následujícím textu jsou neuronové sítě popisovány jako nelineární statistický model, tvořený sítí výpočetních buněk, nazývaných neurony. Síť je tvořena váženými spoji a jako celek je schopna aproximovat nelineární funkční závislosti. [2]

2.3.1 Preceptron

V této kapitole bude podrobněji rozebrán jednovrstvý *preceptron* podle [13]. V tomto podání je neuronová sít tvořena dvoj-vrstvým modelem. Model je možné aplikovat jak pro klasifikační, tak regresní úlohy. Pro klasifikační úlohu s K třídami je na výstupu sítě K jednotek Y_k. Kdy Y_k vyjadřuje příslušnost k třídě k pomocí {0,1} hodnot.

$$Z_m = \sigma(\alpha_{0m} + \alpha_m^T X), \qquad m = 1, \dots, M, \tag{2.3.1}$$

$$T_k = \beta_{0m} + \beta_m^T Z, \qquad k = 1, \dots, K, \tag{2.3.2}$$

$$f_k(X) = g_k(T), \qquad k = 1, \dots, K, \tag{2.3.3}$$

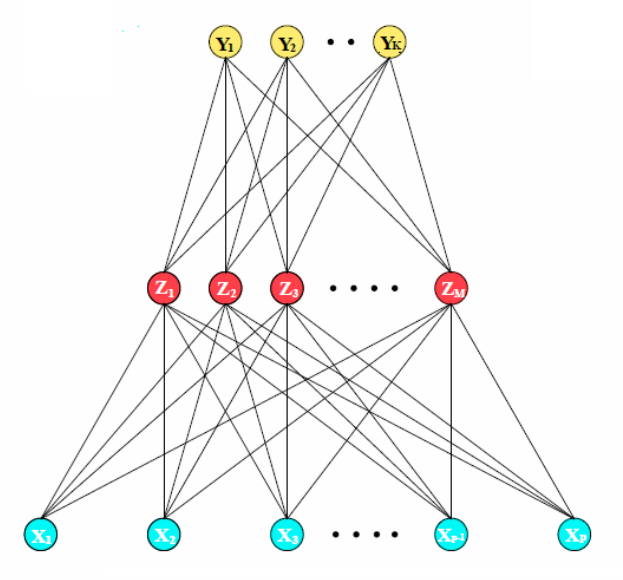
kde $Z = (Z_1, Z_2, \ldots, Z_M)$, a $T = (T_1, T_2, \ldots, T_K)$. Aktivační funkce $\sigma(v)$ je většinou *sigmoid* definována jako

$$\sigma(v) = \frac{1}{1 + e^{-v}}. \tag{2.3.4}$$

Výstupní funkce $g_k(T)$ transformuje vektor výstupů T na pravděpodobnost jednotlivých výstupů a nazývá se *softmax* aktivační funkce

$$g_k(T) = \frac{e^{T_k}}{\sum_{l=1}^{K} e^{T_l}}. \tag{2.3.5}$$

Jednotky Z_m jsou nazývány skryté a celá vrstva se potom nazývá *skrytou* (*hidden layer*). Obecně může mít model libovolný počet skrytých vrstev. V takovém případě se jedná o *vícevrstvý preceptron* (*multi layer preceptron*). Struktura sítě je znázorněna na (Obrázek 1: Schéma jednovrstvého preceptronu. [13]**Chyba! Nenalezen zdroj odkazů.**).



Obrázek 1: Schéma jednovrstvého preceptronu. [13]

2.3.2 Back-Propagation

Popis algoritmu *back-propagation* je opět adoptován z [13]. Parametry $\{\alpha_{0m}, \alpha_m;\ m = 1, 2, \ldots, M\}$ a $\{\beta_{0k}, \beta_k;\ k = 1, 2, \ldots, K\}$ modelu se nazývají váhy (*weights*). Cílem nastavení těchto parametrů je dobrá aproximace tréninkových dat. V případě regrese hledáme optimální hodnoty těchto parametrů pomocí minimalizace:

$$R(\theta) = \sum_{k=1}^{K} \sum_{i=1}^{N} (y_{ik} - f_k(x_i))^2. \tag{2.3.6}$$

$R(\theta)$ se v tomto případě nazývá *suma čtverců odchylek* (*sum-of-squared errors*) chybová funkce. Pro klasifikační úlohu je také možné použít *suma čtverců odchylek* chybovou funkci, anebo *vzájemnou entropii* (*cross entropy*):

$$R(\theta) = -\sum_{i=1}^{N}\sum_{k=1}^{K} y_{ik} \log f_k(x_i). \tag{2.3.7}$$

Ač by se tomu tak mohlo na první pohled zdát, obvykle není žádané nalezení globálního minima funkce $R(\theta)$, protože je pravděpodobné, že by model byl v tomto případě přeučen. Proto je vhodné zavést nějakou formu ochrany proti přeučení. Buď přímo konkrétním penalizačním prvkem, nebo nepřímo, pomocí *pravidla předčasného zastavení* (*early stopping rule*).

Obecný postup minimalizace $R(\theta)$ je aplikování tzv. *gradient descent* metody, v tomto případě *back-propagation*. Následující postup *back-propagation* pro *sum-of-squared errors* chybovou funkci $R(\theta) = \sum_{i=1}^{N} R_i$ je odvozen pomocí pravidla *řetězového diferencování*. Derivace chybových funkcí jsou definovány jako:

$$\frac{\partial R_i}{\partial \beta_{km}} = -2\big(y_{ik} - f_k(x_i)\big) g_k'(\beta_k^T z_i) z_{mi} \tag{2.3.8}$$

a

$$\frac{\partial R_i}{\partial \alpha_{ml}} = -\sum_{k=1}^{K} 2\big(y_{ik} - f_k(x_i)\big) g_k'(\beta_k^T z_i) z_{mi}, \beta_{km} \sigma'(\alpha_m^T x_i) x_{il.} \tag{2.3.9}$$

Aktualizace parametrů pomocí *poklesu gradientu* v $(r+1)$ iteraci je potom rovna:

$$\beta_{km}^{(r+1)} = \beta_{km}^{(r)} - \gamma_r \sum_{i=1}^{N} \frac{\partial R_i}{\partial \beta_{km}^{(r)}} \tag{2.3.10}$$

a

$$\alpha_{ml}^{(r+1)} = \alpha_{ml}^{(r)} - \gamma_r \sum_{i=1}^{N} \frac{\partial R_i}{\partial \alpha_{ml}^{(r)}}, \tag{2.3.11}$$

kde γ_r je tzv. *learning rate*.

Proces probíhá ve dvou směrech. Nejprve jsou směrem dopředu spočítány hodnoty $\hat{f}_k(x_i)$ podle stanovených vah. Následně je spočítána odchylka žádaných hodnot a zpětně jsou podle gradientu chyby aktualizovány parametry.

Výhoda *back-propagation* algoritmu je, že každá jednotka přebírá a přeposílá informaci pouze jednotkám se kterými sdílí vazbu. Této vlastnosti lze dobře využít při implementaci úlohy pro paralelní výpočty. Další předností je možnost takzvaného *online* trénování, kdy pro trénování sítě není použit celý datový set, ale každý tréninkový cyklus je proveden pouze pro jednu instanci vstupních dat. V tomto případě jsou sumy v rovnicích (2.3.10) a (2.3.11) nahrazeny jednou hodnotou. Díky této úpravě je možné postupně zpracovat i opravdu velké datové soubory, anebo nasadit neuronovou síť v aplikace s postupně přicházejícími daty.

Parametr *learning rate* γ_r může být brán jako konstanta nebo například optimalizován metodou *line search*. Je třeba podotknout, že výběr parametru γ_r je klíčová část metody, protože ovlivňuje její konvergenci.

Nakonec je ještě nutné zmínit jakým způsobem vhodně zvolit počáteční hodnoty vah. Pokud jsou hodnoty vah blízké 0, blíží se funkce, aproximovaná neuronovou sítí, téměř lineární. Následnou aktualizací vah se potom funkce přizpůsobuje podle dat. V případě,

že by hodnoty vah byly rovny 0, vycházely by nulové derivace a nedocházelo by k žádné změně parametru. Při opačné situaci, kdy by hodnoty parametrů začínaly blízko 1, je pravděpodobné nalezení nepřesných řešení. [13]

2.4 Bayesovské učení (Bayesian learning)

Oblast zabývající se pravděpodobnostními modely založenými na *Bayesovu* teorému, je velice rozsáhlá. V následujícím textu je nastíněna základní problematika sestavování pravděpodobnostního modelu a podrobněji je popsán *Naivní Bayesův* (*Naive Bayes*) klasifikátor.

Pravděpodobnostní modely jsou velice důležité v problémech, kdy není možné nalézt přesné řešení a je nutné stanovit, jak pravděpodobné jsou jednotlivé hypotézy. Protože v reálném světě téměř nikdy není známé přesné řešení, je tento postup přinejmenším žádoucí.

Proces kognitivního učení spočívá v hledání *posteriorní* (*posteriori*) pravděpodobnosti na základě *věrohodnosti* (*likelihood*) hypotézy ovlivněné *apriorním* (*prior*) přesvědčením. [19]

Ke klasifikování testové instance $\boldsymbol{x}$ lze formulovat pravděpodobnostní model, který sestavuje posteriorní pravděpodobnost $P(y|\boldsymbol{x})$ rozdílných y. Predikovaná třída je vybrána podle y s nejvyšší posteriorní pravděpodobností $P(y|\boldsymbol{x})$. Takto je definováno *maximum a posterior* (MAP) pravidlo. Podle *Bayesova* teorému je podmíněná pravděpodobnost třídy y na vstupu $\boldsymbol{x}$ určena jako

$$P(y|\boldsymbol{x}) = \frac{P(\boldsymbol{x}|y)P(y)}{P(\boldsymbol{x})}, \tag{2.4.1}$$

kde $P(y)$ je určeno početním podílem třídy y v tréninkovém setu. Pravděpodobnost $P(\boldsymbol{x})$ není pro klasifikaci relevantní, protože je porovnávána u různých y na stejném $\boldsymbol{x}$. Jediná neznámá v rovnici (2.8) je potom $P(\boldsymbol{x}|y)$. Pokud je odhad $P(\boldsymbol{x}|y)$ přesný, výsledkem je nejlepší možný klasifikátor z daných tréninkových dat a to *Bayesův optimalní klasifikátor* (*Bayes optimal classifier*) [26].

2.4.1 NaiveBayes klasifikátor

NaiveBayes klasifikátor je pravděpodobně jeden z nejběžněji používaných modelů založený na Bayesovu teorému ve strojovém učení. [21]

Odhadování podmíněné pravděpodobnosti $P(\boldsymbol{x}|y)$ není triviální, protože spočívá v nalezení exponenciálního množství sdružených pravděpodobností jednotlivých atributů. Za určitých předpokladů lze tento problém zjednodušit. [26]

U *Naivního Bayesova* klasifikátoru je předpokládáno, že p atributů je v každé třídě navzájem nezávislých. Potom je podmíněná pravděpodobnost dána jako:

$$P(\boldsymbol{x}|y) = \prod_{i=1}^{p} P(x_i|y), \tag{2.4.2}$$

což naznačuje, že je pouze potřeba vypočítat každý atribut v každé třídě, aby bylo možné určit podmíněnou pravděpodobnost, a tak se vyhnout výpočtu sdružených pravděpodobností. [26]

Ačkoliv se tento předpoklad zdá být poměrně ukvapený, funguje překvapivě dobře a metoda často překonává sofistikovanější alternativy [13].

V tréninkové fázi určí *Naivní Bayesův klasifikátor* pravděpodobnost $P(y)$ pro všechny $y \in Y$ a $P(x_i|y)$ pro všechny atributy $i = 1, \dots, p$ a všechny hodnoty x_i z tréninkových dat. V testovací fázi bude testovací instance x predikovaná jako ze třídy y, pokud tato třída má největší hodnotu

$$P(\boldsymbol{x}|y) \propto P(y) \prod_{i=1}^{p} P(x_i|y) \tag{2.4.3}$$

mezi všemi třídami. [26]

V případě velkého množství atributů vzniká problém s vymizením hodnot, protože je počítán součin dílčích pravděpodobností, které se mohou blížit 0, protože dnešní počítače nejsou schopny pracovat s čísly s nekonečnou přesností. Tento problém je možné řešit převedením problému do logaritmického tvaru, kde se součin pravděpodobností převádí na součet logaritmů pravděpodobností. Velice efektivní je metoda *Log-Sum-Exp*. [17]

Předností *Naivního Bayesova* klasifikátoru je jednoduchost výpočtu. Toto dovoluje nasazení v aplikacích s obrovským datovými soubory s velkým počtem vstupních atributů. Například filtrování spamu v emailech na základě výskytu klíčových slov, kdy atributy, které jsou v tomto případě jednotlivá slova, mohou nabývat několika tisíců. [17]

2.5 Metoda podpůrných vektorů (Support Vector Machnies)

Metoda podpůrných vektorů, dále nazývána metoda *SVM*, byla původně navržena pro binární klasifikaci, později však byla rozšířena, jak pro úlohy regresní, tak úlohy klasifikační s více třídami. [19]

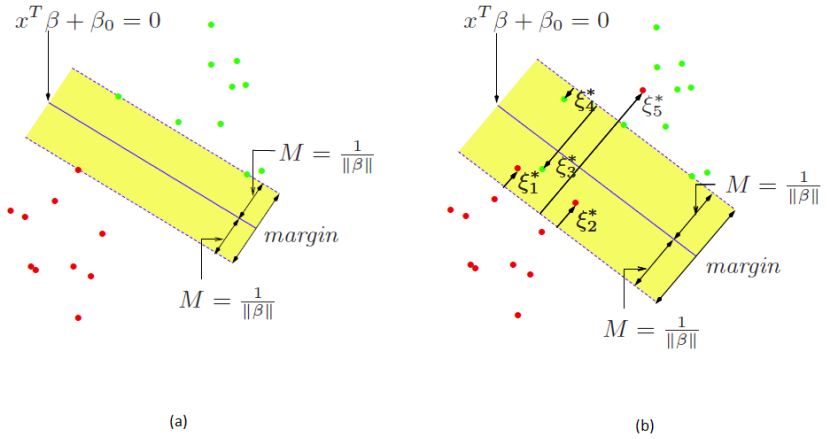
Jak je zmíněno ve [21], je v dnešní době *SVM* jeden z nejoblíbenějších *off-the-shelf* modelů. Velice atraktivní vlastností *SVM* modelů jsou poměrně dobré generalizační schopnosti modelu i v případě malého datového souboru.

SVM model spojuje 3 základní myšlenky: mapovaní do speciálního nad-prostoru pomocí Kernel metod, tzv. *sparse* řešení a maximalizování marginální oblasti. [19] Všechny tři zde budou podrobněji vysvětleny.

Při analýze problému separování dvou tříd v prostoru R^2, mohou obecně nastat čtyři rozdílné situace: třídy jsou lineárně oddělitelné bez vzájemného překrytí, třídy jsou lineárně oddělitelné se vzájemným překrytím, třídy jsou nelineárně oddělitelné bez vzájemného překrytí, třídy jsou nelineárně oddělitelné se vzájemným překrytím.

2.5.1 Lineární situace

První dvě situace nejsou příliš zajímavé, protože jejich řešení není příliš obtížné pomocí lineárních modelů jako je *LDA* nebo *Logistic regression*. Bude na nich ovšem vysvětlen princip řídkého řešení a maximální marginální vzdálenosti.



Obrázek 2: (a) – lineárně oddělitelný problém bez vzájemného překrytí, (b) - lineárně oddělitelný problém se vzájemným překrytím [13]

Je dán soubor tréninkových dat tvořený N páry $(\boldsymbol{x}_1, y_1), (\boldsymbol{x}_2, y_2), \dots, (\boldsymbol{x}_N, y_N)$, kde $\boldsymbol{x}_i \in \boldsymbol{R}^2$ a $y_i \in \{-1,1\}$. Podmnožina těchto dat je vyobrazena na (Obrázek 22). Vlevo (a) je případ první situace a vpravo (b) druhé situace. V obou případech je uprostřed oblasti vyznačena rozhodovací hranice obklopená žlutou oblastí nazývanou *marginální*. Rozhodovací hranice je definována jako

$$\{x : f(x) = \boldsymbol{x}^T\beta + \beta_0 = 0\}, \tag{2.5.1}$$

kde β je jednotkový vektor $\|\beta\| = 1$. Pokud definujeme klasifikační pravidlo podle $f(x)$, je dáno jako

$$G(x) = sign(\boldsymbol{x}^T\beta + \beta_0). \tag{2.5.2}$$

V první situaci je možné najít takovou funkci $f(x) = \boldsymbol{x}^T\beta + \beta_0$, která zaručí $y_i f_i(x) > 0$ pro $\forall i$. Tato funkce je rovnice přímky, která vytvoří největší *marginální* oblast mezi prvky obou tříd. Problém se dá přeformulovat jako:

$$\max_{\beta,\beta_0,\|\beta_0\|=1} M \text{ pro } y_i(\boldsymbol{x}^T\beta + \beta_0,) \geq M, i = 1, \dots, N, \tag{2.5.3}$$

kde M je šířka marginálního pásu. [13]

V případě druhé situace, již není možné najít funkci přímky, která oddělí od sebe prvky obou tříd. Je ovšem možné maximalizovat *marginální* oblast, která oddělí dvě třídy, pokud jsou některé instance z uvažování vyjmuta. Vyjmutím je myšleno, že mohou zůstat uvnitř marginální oblasti nebo i na její špatné straně. [13]

K tomu je potřeba definovat tzv. *slack* proměnné $\xi = (\xi_1, \xi_2, \dots, \xi_N)$. Jejich vlastností je, že $\xi_i = 0$ v případě, že leží vně marginální oblasti na správné straně. Pokud leží uvnitř a na správné straně je $0 < \xi_i \leq 1$. Třetí možnost je, že se nachází na špatné straně v tom případě, je $\xi_i > 1$. Modifikovaná rovnice (2.5.3) vypadá takto:

$$y_i(\,\boldsymbol{x}^T\beta + \beta_0,) \geq M(1-\xi_i), \tag{2.5.4}$$

pro $\forall i,\ \xi_i \geq 0, \sum_{i=1}^{N} \xi_i \leq konst.$ Hodnota ξ_i v podmínce $y_i(\omega^t x + b) \geq M(1-\xi_i)$ je proporcionální hodnota, o kolik je predikce $f(x_i) = \boldsymbol{x}^T\beta + \beta_0$, na špatné straně hranice. Takže omezením sumy $\sum \xi_i$ je omezena celková proporcionální vzdálenost o kolik predikce spadají na špatnou stranu hranice. Pokud platí, že $M = 1/\|\beta\|$ lze (2.5.3) a (2.5.4) přepsat jako

$$\min\|\beta\|\ pro \begin{cases} y_i(\boldsymbol{x}^T\beta + \beta_0) \geq 1 - \xi_i\ \ \forall i, \\ \xi_i \geq 0, \sum \xi_i \leq konst. \end{cases} \tag{2.5.5}$$

Nalezení minima v rovnici (2.5.5) se provádí pomocí tzv. *quadratic programming with linear inequality constraints*. U modelu *SVM* se k tomu nejčastěji používá *Lagrangeových multiplikátorů* (*Lagrange multipliers*). [13]

Nebude zde uveden postup, ale pouze výsledek:

$$\hat{\beta} = \sum_{i=1}^{N} \hat{\alpha}_i y_i x_i, \tag{2.5.6}$$

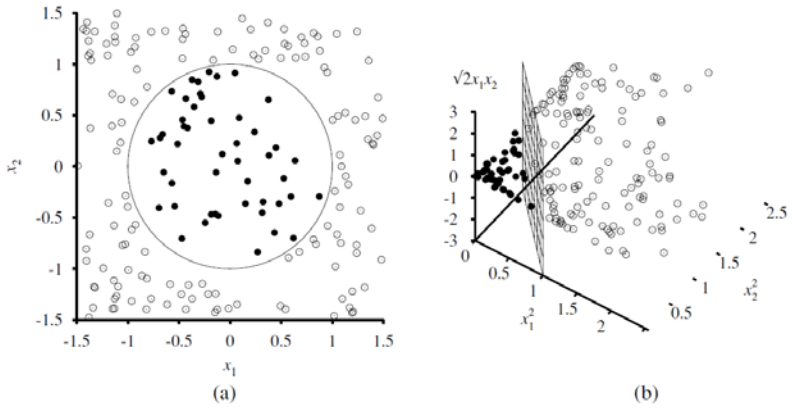
kde $\hat{\alpha}_i$ jsou nenulové koeficienty pouze pro ty body, které jsou na správné hranici *marginální* oblasti. Tyto body se nazývají *podpůrné vektory* (*support vectors*). [13]

Marginální hranice je tedy stanovena pouze na základě těchto vektorů a není nijak ovlivněna ani body vně marginální oblasti ani body uvnitř. Tato vlastnost je velice lákavá, protože řešení není ovlivňováno body uvnitř třídy a není ovlivňována chybovými daty. Výsledné řešení se poté označuje jako *sparse*, protože záleží na malém počtu instancí. Klíčem úspěchu je správné stanovení konst. v rovnici (2.5.5). Tento parametr nastavuje kolik instancí je ponecháno na špatné straně rozhodovací hranice. [13]

2.5.2 Nelineární situace

V případě nelineárně oddělitelných instancí je již situace složitější. Při pohledu na (**Chyba! Nenalezen zdroj odkazů.**-a) je evidentní, že ve vstupním prostoru neexistuje lineární rozhodovací hranice, která by mohla data spolehlivě rozdělit. V tomto případě je nezbytné využít *Kernel* metod. [21]

Problematika *Kernel* metod opět značně přesahuje záběr této práce, takže zde bude pouze popsán jeden konkrétní případ bez přesného vysvětlení. Podrobné informace lze nalézt zejména v [13] a [21.

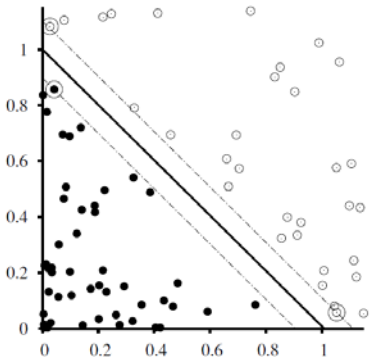


Obrázek 3: (a) - nelineárně separovatelný problém, (b) – nelineárně sepraovatlný problém převedný na separovatlný pomcí kernel funkcí. [3]

Instance jsou odděleny skutečnou rozhodovací hranicí, která lze popsat jako $x_1^2 + x_2^2 \leq 1$, neboli rovnice kružnice. Pokud jsou vstupní data transformována pomocí funkcí

$$f_1 = x_1^2, f_1 = x_2^2 \; a \; f_3 = \sqrt{2}x_1x_2, \tag{2.5.7}$$

lze znázornit podle (Obr. 3-b). Ve výsledném prostoru jsou třídy velice jednoduše oddělitelné pomocí lineární rozhodovací hranice. Na (Obr. 4) je patrné, že tato hranice je opět stanovena pomocí maximalizování *marginální* oblasti. [21]



Obrázek 4: Zobrazení rozhodvací hranice s marginální oblasti v narovině převedené pomocí kenrel funkcí [3]

Funkce f_1, f_2 a f_3 jsou ve skutečnosti mapování *nadrovin* (*hyperspace*) pomocí *lineární kernel* (*linear kernel*) funkce. [21]

Další populární *kernel* funkce jsou *polynomiální* a *RBF kernel*. Tento postup někdy nazývaný jako *kernel trik* (*kernel trick*) je možné použit u všech metod, které využívají pouze tzv. *inner product* mezi vstupními vektory k výpočtu třídy. Tyto metody se pak nazývají Kernel metody (*Kernel methods*) [26].

Nyní zbývá doplnit, že v případě nelineárně oddělitelné situace s vzájemným překrytím, je postupováno analogicky jako v lineární situaci, jen v prostoru s vyšší dimenzí.

3 META-LEARNING

V zahraniční literatuře jsou meta-learningové metody často nazývány ensemble metody. Různé zdroje se poněkud rozchází ohledně přesné definice ensemble metod. Například v [13], [21] a [26]. Dokonce existují různé pohledy na to, které modely patří mezi *ensemble*, a které patří mezi tradiční [13] a [26]. Příkladem jsou například neuronové sítě. Ačkoliv jsou běžně řazeny mezi tradiční modely, lze uvažovat každý neuron jako jeden model a tím pádem, jejich kombinace do struktury sítě je kvalifikuje jako jistá kombinace jednoduchých base modelů. Nicméně v rámci této práce jsou mezi ensemble modely uvažovány *Boosting*, *Bagging* a *kombinační metody*.

V této práci budou podrobněji rozebírány dva směry, které nyní utváří část *Ensemble* metod. Těmito směry jsou: kombinace klasifikátorů (*combining of classifiers)* a soubor slabých modelů (*ensambles of week learners*). [26]

Kombinace klasifikátorů byla převážně studována v komunitě *rozpoznávání vzorů* (*pattern recognition*). Je zde brán důraz na silné klasifikátory (*strong classifiers*). Tedy vhodná kombinace těchto klasifikátorů, která zajistí lepší predikční schopnosti celku oproti individuálním modelům. Soubor slabých modelů byl převážně studován v komunitě *strojového učení*. Cílem snažení bylo vytvořit silné klasifikátory pomocí vhodné kombinace jednoduchých base modelů. [26]

Jedním kritériem dělení *ensemble* metod je způsob vytváření modelů, kde mohou být vytvářeny paralelně nebo sekvenčně. Jediným zde rozebíraným zástupcem sekvenčního vytváření modelů je *Boosting*, zatímco paralelně jsou vytvářeny *Bagging* a kombinační metody. Další dělení je podle druhů modelů, ze kterých se *ensemble* skládá. Rozlišovány jsou *homogenní* a *heterogenní ensemble* modely. Příkladem homogenních jsou *Bagging* a *Boosting*, ale i ojediněle některé *kombinační metody*. Převážná většina modelů kombinačních metod však je heterogenních, tedy jednotlivé base modely jsou různého typu. [26]

3.1 Boosting

Některé zdroje [13] mluví o *Boosting* metodě jako jedné z nejvýznamnějších myšlenek představených během posledních dvou desetiletí. Původně byl navržen jako binární klasifikační model, ale později byl rozšířen i pro regresní úlohy. [13]

Hlavní myšlenka je, tak jako třeba u *Bagging* modelů, správnou kombinací slabých (*weak*) modelů dosáhnout tzv. *committe*, která dosahuje daleko lepší predikční schopnosti než je tomu u kteréhokoliv z dílčích modelů. Podobnost s *Bagging* modelem zde ovšem končí, protože postup kombinování jednotlivých modelů je značně odlišný. Jak již bylo řečeno *Boosting* patří mezi tzv. *sekvenční ensemble* (*sequential ensemble*) metody. Vytváření dílčích modelu probíhá postupně, protože učení nového modelu je ovlivňováno modely předešlými.

3.1.1 AdaBoost

Název je tvořen spojením slov *Adaptive Boosting*. V této sekci bude představen model *AdaBoost.M1* podle [13]. Odvození se dívá na *AdaBoost* jako na problém

aditivního modelování pomocí *exponenciální* chybové funkce. Dále je předpokládáno, že úloha spočívá v binární klasifikaci a třída *y* nabývá hodnot {-1,1}.

Procedura se snaží aproximovat skutečnou závislost aditivním generováním klasifikátorů $G_m(x)$, které jsou vytvářeny na upravených datových vzorcích. Rozložení dat je ovlivňováno úspěšností klasifikace předešlých modelů. Při vytváření nového modelu je vždy brán větší ohled na instance, které jsou špatně klasifikovány v předešlé iteraci. Predikce výsledného ensemble modelu je sestavena pomocí váženého hlasování dílčích base modelů. Výsledná predikovaná třída je určena podle

$$G(x) = \text{sign}\left(\sum_{m=1}^{M} \alpha_m G_m(x)\right), \tag{3.1.1}$$

kde $\alpha_1, \alpha_2, \ldots, \alpha_M$ jsou váhy pro každý $G_m(x)$ model. Váha každého modelu je nastavena pomocí *exponenciální chybové funkce*

$$L\big(y, f(x)\big) = \exp\big(-yf(x)\big). \tag{3.1.2}$$

Je-li hledán *m*-tý komponent z rovnice (3.1.1), jsou hledány parametry

$$(\beta_m, G_m) = \arg\min_{\beta,G} \sum_{i=1}^{N} \exp\big[-y_i\big(f_{m-1}(x_i) + \beta G(x_i)\big)\big], \tag{3.1.3}$$

kde $f_{m-1}(x_i)$ představuje model v předešlém kroku a $\beta G(x_i)$ představuje nový přírůstek modelu. Vzorec (3.1.3) lze upravit jako

$$(\beta_m G_m) = \arg\min_{\beta,G} \sum_{i=1}^{N} \omega_i^{(m)} \exp\big(-\beta y_i G(x_i)\big), \tag{3.1.4}$$

kde $\omega_i^{(m)} = \exp\big(-y_i f_{m-1}(x_i)\big)$. Jelikož $f_{m-1}(x_i)$ není závislá na β_m ani G_m plní $\omega_i^{(m)}$ funkci váhy jednotlivých instancí. Lze odvodit, že

$$\beta_m = \frac{1}{2}\log\frac{1 - err_m}{err_m}, \tag{3.1.5}$$

kde chyba err_m je definována jako

$$err_m = \frac{\sum_{i=1}^{N} \omega_i^{(m)} I(y_i \neq G_m(x_i))}{\sum_{i=1}^{N} \omega_i^{(m)}}. \tag{3.1.6}$$

Tímto je určena váha modelu $G_m(x)$. Nové váhy datových instancí jsou vypočteny jako

$$\omega_i^{(m+1)} = \omega_i^{(m)} \cdot e^{\alpha_m I(y_i \neq G_m(x_i))} \cdot e^{-\beta_m}, \tag{3.1.7}$$

kde $\alpha_m = 2\beta_m$.

Jsou dvě možnosti, jakým způsobem se projevuje váha jednotlivých instancí při vytváření base modelu. V případě, že je model schopný s nimi pracovat, je vliv evidentní. Pokud tomu tak není, je proveden náhodný výběr instancí s opakováním, kdy větší šanci na výběr mají instance s vyšší váhou. [13]

AdaBoost je náchylný vůči šumu dat. Problém tkví v *exponenciální* chybové funkci. V případě instance, která se nachází daleko za rozhodovací hranicí, je její váha po špatné klasifikaci neustále zvyšována, což model nutí snažit se je správně klasifikovat. Bylo vyvinuto několik modelů, které se snaží tento problém řešit. Jedním z nich je *MadaBoost*, který je *AdaBoostu* velice podobný. Jediný podstatný rozdíl je, že *MadaBoost* omezuje shora váhu špatně klasifikované instance. Dalšími příklady robustnějších modelů jsou *BrownBoost* a *FilterBoost* [26].

3.2 Bagging

BootstrAp aGGregatING patří mezi tak zvané *paralelní ensemble* (*parallel ensemble*) metody. Slovem paralelní je míněna nezávislost mezi jednotlivými modely při trénování. Modely jsou trénovány nezávisle na sobě na rozdíl od *Boosting* metod. Tento fakt skýtá velký potenciál v možnostech paralelních výpočtů. Paralelní přístup lze využít jak ve fázi učení, tak pří klasifikaci dat. [26]

Kombinací predikcí nezávislých dílčích modelů klesá u výsledného modelu chyba. Teorie naznačuje [26], že čím více nezávislých modelů se na klasifikaci podílí, tím menší je celková chyba. V praxi není možné získat opravdu nezávislé modely, protože jsou v zásadě vytvářeny pomocí jednoho datového souboru s konečným a mnohdy velmi malým počtem instancí.

Částečným řešením tohoto problém je trénovat modely na vždy odlišném vzorku z datového souboru. K dosažení skutečné nezávislosti se na první pohled zdá být dobrý nápad trénovat model na disjunktivních podmnožinách dat. Mnohdy však kvůli nedostatku dat toto řešení konstruuje modely, které nedostatečně reprezentují skutečnou závislost. *Bagging* tento problém, jak již název napovídá, řeší pomocí generování datových vzorků s *Bootstrap* rozložením. [26]

Bootstrap je podrobněji popsán v kapitole 4.3. *Bootstrap* je aplikován pro výběr vzorků s opakováním pro vytvoření *Bootstrap* rozložení. Vytvářením base modelů na různých *Bootstrap* rozloženích, je dosažena dostatečná nezávislost modelů při zachování rozložení původního datového souboru.

Predikce výsledných modelů je na závěr spojena. Toto spojení bývá tzv. *hlasování* (*voting*) pro klasifikaci a *průměr* (*average*) pro regresní modely.

V případě binární klasifikace tříd $K \in \{0,1\}$ je výsledná třída určena jako

$$G(x) = \text{sign}\left(\sum_{b=1}^{B} G_b(x)\right), \tag{3.2.1}$$

kde $G_b(x)$ jsou jednotlivé modely vytvořené pomocí bootstrap rozložení z původního datového souboru. [13]

Podle [26] je *Bagging* nejvíce vhodný pro modely, které nalézají nestabilní řešení, jako například *rozhodovací stromy*. Čím méně je model stabilní, tím většího zlepšení lze zpravidla s *Bagging* modelem dosáhnout.

3.2.1 Random Forest

Při porovnávání výkonosti *Bagging* a *Boosting* modelů je v literatuře (například [13] a [26]), často uváděno, že *Boosting* dosahuje lepších výsledků. V [6] byl navrhnut *Random Forest* model, vycházející z podstat *Baggingu*, který klade větší důraz na získávání co nejméně závislých modelů.

Jako base model používá speciální druh rozhodovacího stromu někdy nazývaný náhodný strom. Při vytváření modelu dochází ke zvýšení rozmanitosti jednotlivých stromů a tím zmenšení jejich závislosti. Během stavby jednotlivých stromů je v každém uzlu náhodně vybrána podmnožina atributů, ze kterých se vybírá nejlepší kandidát pro rozdělení dat. [26]

Ještě dále tímto směrem se vydává [16] s modelem *VR-Tree*. Kde se kromě atributů vybírají náhodně i hodnoty, ze kterých se vybere nejlepší dělící hodnota. Není zpravidla dobré nechat vytváření stromu čistě náhodě, proto je v algoritmu představen parametr α. V každém uzlu je s pravděpodobností α rozhodnuto zda bude uzel vytvořen jako u klasického rozhodovacího stromu nebo bude jeho vytváření podrobeno náhodě. V případě $\alpha = 1$ jde o klasický rozhodovací strom, v případě $\alpha = 0$ je strom vytvořen zcela náhodně. [26]

Velká přednost *Random Forest* modelů je jejich přirozená schopnost klasifikovat do více než dvou tříd. Ta je samozřejmě způsobena použitím stromů jako base klasifikátorů. Právě tato vlastnost může za jejich preferování v některých úlohách na úkor modelů jako *SVM* nebo *AdaBoost*. Protože postup učení a klasifikace je analogický jako u *Baggingu* je hojně využíváno paralelního programování. Oproti *Bagging* modelu s rozhodovacím stromem je dokonce i menší výpočetní náročnost, protože při každém vytváření nových uzlů je vybíráno mezi menším počtem atributů a *VR-Tree* dokonce menším množství hodnot [17].

Problém modelu *Random Forest* jsou úlohy s velkým množstvím vstupních atributů, mezi kterými je malý podíl relevantních. Je to způsobeno tím, že při výběru dělících atributů je malá šance, že budou vybrány ty správné [13].

3.3 Kombinační Metody (Combination Methods)

Jak již bylo diskutováno v případě *Boosting* a *Bagging* modelů, bylo dokázáno, že kombinací více jednoduchých modelů dochází ke zlepšení klasifikačních vlastností oproti jednotlivým modelům. V případě *Boosting* modelu je aplikován jeden druh modelu opakovaně se snahou lépe klasifikovat data. Na druhou stranu u modelu *Bagging* je pomocí kombinace nezávislých modelů jednoho druhu na rozdílných datech redukována variance predikce a tím zvýšena přesnost modelu jako celku.

Kombinační metody jsou zobecněním těchto poznatků. Pokud je řeč o kombinačních metodách, je většinou myšleno trénování různých druhů modelů nezávisle na sobě a poté vhodnou kombinací dílčích predikcí odhadovat predikci *ensemble* modelu jako celku. [26]

3.3.1 Hlasování (Voting)

Nejpřímočařejším způsobem kombinace se jeví hlasovaní dílčích base modelů. V následující sekci jsou stručně popsány rozdílné postupy kombinování metod hlasováním podle [26]. Za předpokladu, že je k dispozici sada M base klasifikátorů $\{h_1, \dots, h_M\}$, jejichž úkolem je predikovat třídu z l možných tříd $\{c_1, \dots, c_l\}$. Při vstupu $\boldsymbol{x}$ je výstup h_m definován jako vektor

$$\left(h_m^1(\boldsymbol{x}), \dots, h_m^l(\boldsymbol{x})\right)^T, \tag{3.3.1}$$

kde $h_m^j(\boldsymbol{x})$ je výstup klasikátoru pro třídu j.

V zásadě mohou mít jednotlivé modely dva různé výstupy pro $h_m^j(\boldsymbol{x})$. Prvním z nich je tzv. *crisp label*, kdy $h_m^j(\boldsymbol{x}) \in \{0,1\}$, který se rovná 1 v případě, že klasifikátor určuje tuto třídu a rovná se 0, pokud tomu tak není. Druhá varianta je možná pokud base klasifikátor zvládá vyjádřit výstup predikce jako pravděpodobnost. Pravděpodobnost třídy klasifikátoru $h_m^j(\boldsymbol{x}) \in [0,1]$, což je vlastně posterirní pravděpodobnost $P(c_j|\boldsymbol{x})$.

Hlas Většiny (Majority Voting)

Každý klasifikátor dává hlas jedné třídě c_j. Výslednou třídou je stanovena ta, která dostane nadpoloviční většinu hlasů. Pokud nadpoloviční většiny žádná třída nedosáhne, není instance klasifikována. [26]

Výsledná třída je určená podle

$$H(\boldsymbol{x}) = \begin{cases} c_j \text{ pokud } \sum_{m=1}^{M} h_m^j(\boldsymbol{x}) > \frac{1}{2}\sum_{k=1}^{L}\sum_{m=1}^{M} h_m^k(\boldsymbol{x}), \\ \text{v opačném případě není provedena klasifikace.} \end{cases} \tag{3.3.2}$$

Hlasování s nejlepším výsledkem (Plurality Voting)

U této metody je třída stanovena podle nejpočetněji zastoupené třídy mezi predikcemi base modelů. V případě nerozhodného výsledku je zvolena libovolná třída. Na rozdíl od většinového hlasu zde nedochází k situaci, kdy třída není určena. [26]

Třída je tedy určena jako

$$H(\boldsymbol{x}) = c_{\arg\max_j \sum_{m=1}^{M} h_m^j(\boldsymbol{x})}. \tag{3.3.3}$$

Vážené hlasování (Weighted Voting)

V případě, že jsou použity base modely s různou přesností, zdá se být dobrý nápad dát větší váhu přesnějším base modelům. Při této úpravě je výsledná třída klasifikována podle

$$H(\boldsymbol{x}) = c_{\arg\max_j \sum_{m=1}^{M} \omega_m h_m^j(\boldsymbol{x})}, \tag{3.3.4}$$

kde ω_m představuje váhu base modelu. Váha je většinou normalizována, aby $\sum_{m=1}^{M} \omega_m = 1$. Za předpokladu nezávislých klasifikátoru je $\omega_m \propto \log \frac{p_m}{1-p_m}$, kde p_m je přesnost modelu. Protože tento předpoklad je hodně nepřesný, není zaručeno, že váženým hlasováním lze dosáhnout lepšího výsledku než u většinového hlasování. [26]

Jemné hlasování (Soft Voting)

Zatímco předešlé metody hlasování mohou být použity pro oba druhy výstupů, kde pravděpodobnost třídy je nahrazena *MAP* (*maximum a posteriori*) odhadem, je jemné hlasování vhodné pouze pro pravděpodobnostní výstup. Kromě toho je většinou použito jemné hlasování aplikované na úlohy s homogenními ensemble modely. Pokud jsou výstupy všech modelů brány se stejnou váhou je

$$H^j(\boldsymbol{x}) = \frac{1}{M} \sum_{m=1}^{M} h_m^j(\boldsymbol{x}). \tag{3.3.5}$$

V případě aplikování vah je

$$H^j(\boldsymbol{x}) = \sum_{m=1}^{M} \omega_m h_m^j(\boldsymbol{x}), \tag{3.3.6}$$

nebo

$$H^j(\boldsymbol{x}) = \sum_{m=1}^{M} \omega_m^j h_m^j(\boldsymbol{x}). \tag{3.3.7}$$

Rozdíl mezi (3.3.6) a (3.3.7) je v uvažování vah zvlášť pro každý klasifikátor (ω_m) nebo pro každou třídu každého klasifikátoru (ω_m^j). Populárnější je druhá možnost, kdy řešením optimalizačního problému je

$$\omega^j = \arg\min_{\boldsymbol{\omega}^j} \sum_{i=1}^{N} \left(\sum_{m=1}^{M} \omega_m^j h_m^j(\boldsymbol{x}_i) - \mathrm{I}\big(f(\boldsymbol{x}_i) = c_j\big) \right)^2, \tag{3.3.8}$$

kde $i \in \{1, \dots, N\}$ představuje tréninkové instance a $j \in \{1, \dots, l\}$ možné třídy. [26]

3.3.2 Kombinace učením (Stacking)

Kombinační metody lze formulovat jako obecný postup kombinace výstupů několika modelů pro zpřesnění predikční schopnosti celku podle

$$f(y|\boldsymbol{x}, \boldsymbol{\omega}) = \sum_{m=1}^{M} \omega_m f_m(y|\boldsymbol{x}), \tag{3.3.9}$$

kde ω_m je modifikovatelný parametr a $f_m(y|\boldsymbol{x})$ funkce posteriorní pravděpodobnosti třídy y jednotlivých base modelů. Zatímco u kombinace pomocí hlasování je váha ω_m odvozena na základě vlastností dílčích modelů, je v případě *Stacking* modelu odvozena pomocí učícího procesu. V tomto případě je hledán vektor vah

$$\boldsymbol{\omega} = \arg\min_{\boldsymbol{\omega}} \sum_{i=1}^{N} L\left(y_i, \sum_{m=1}^{M} \omega_m f_m(\boldsymbol{x}) \right), \tag{3.3.10}$$

kde $L(\cdot,\cdot)$ je libovolná chybová funkce. [19]

Tato definice není žádoucí, protože dochází k přeučení celku způsobené upřednostněním nejkomplexnějšího modelu $f_m(\boldsymbol{x})$. Řešením je použití křížové validace. V případě *Stacking*u se běžně používá *LOOCV validace*, která je podrobněji popsána v kapitole 4.1. Upravená verze (3.3.10) je potom

$$\boldsymbol{\omega} = \arg\min_{\boldsymbol{\omega}} \sum_{i=1}^{N} L\left(y_i, \sum_{m=1}^{M} \omega_m \hat{f}_m^{-i}(\boldsymbol{x})\right), \tag{3.3.11}$$

kde $\hat{f}_m^{-i}$je base model trénovaný na všech tréninkových datech kromě instance i. [19]

Pro Stacking neboli *stacked genaralization* model existuje speciální terminologie označující jeho jednotlivé součásti. Dílčí base modely jsou nazývány modely *první úrovně* (*first-level learners*) a model reprezentující minimalizaci chybové funkce se nazývá model *druhé úrovně* (*second-level learner*). [26]

Modely první úrovně mohou být stejného typu, vytvářející homogenní ensemble, nebo to mohou být různé druhy modelů, které tak vytvářejí heterogenní ensemble. Častější je druhá varianta. Model druhé úrovně může být teoreticky libovolný model minimalizující chybovou funkci na tréninkových datech. [26]

4 URČOVÁNÍ PŘESNOSTI MODELU

Typická jsou data, která jsou k dispozici rozdělena na data tréninková a testovací. Pomocí tréninkových dat je vybrán a naučen model. K odhadu přesnosti modelu jsou použita data testovací. [19]

Poměr špatně klasifikovaných dat (*misclassifiaction rate*) je definován jako

$$err = \frac{1}{N}\sum_{i=1}^{N} \boldsymbol{I}(f(x_i) \neq y_i), \tag{4.1.1}$$

kde $f(x)$ představuje klasifikátor a y skutečnou třídu. Poměr špatně klasifikovaných dat na tréninkovém setu není dobrým kritériem pro výběr modelu. Daleko vhodnější je generalizační chyba, což je předpokládaná hodnota průměrné chyby na nových datech. Generalizační chybu lze aproximovat pomocí určení poměru špatně klasifikovaných dat na dostatečně velkém nezávislém datovém souboru. K tomuto účelu jsou využívána testovací data. [19]

Mohlo by se zdát lákavé využít testovacích dat jako měřítko pro výběr vhodného modelu, ale je důležité se tomuto nutkání vyhnout, protože odhad generalizační chyby modelu vybraného na základě testovacích dat, bude příliš optimistický. Vhodnějším řešením je rozdělit tréninková data a menší část těchto dat použít jako validační datový soubor, který není použit při trénování modelu. Na základě chyby naučeného modelu na validačních datech se vybírá vhodný model. Vybraný model je možné následně naučit na celém tréninkovém souboru a generalizační chybu odhadovat podle testovacích dat. [19]

Tato doporučení platí pouze v případě, kdy je k dispozici dostatečné množství dat. V případě, kdy tomu tak není, je možné místo dělení tréninkových dat použít metodu *křížové validace* (*cross validation*). V extrémních případech je dokonce možné použít *LOOCV* podrobněji popsáno v následující kapitole.

S ohledem na komplexnost modelu se často bere v potaz tzv. *Occamovo ostří* (*Occam's razor*), tedy při výběru se upřednostňuje jednodušší hypotéza konzistentní s daty. [19]

4.1 Křížová validace (Cross Validation)

Jde o velice jednoduchou metoda určení přesnosti modelu. Využívá se jí hlavně pokud není dostatek dat. Jak již bylo uvedeno v předešlé kapitole, v ideální situaci jsou data rozdělena na tréninková a testovací, pokud by při tomto rozdělení došlo ke zmenšení tréninkového datového souboru natolik, že výsledný model nesprávně reprezentuje skutečnost, může být částečným řešením křížová validace (dále CV).

CV má parametr *K*, který určuje, na kolik částí budou data rozdělena a zároveň kolik iterací bude provedeno. K-1 částí je v každém kole použito jako tréninková množina a zbylá část je použita jako nezávislý testovací soubor. Proces je opakován *k*-krát a vždy vynechá jinou část dat, aby se vystřídala všechna. Výsledkem tohoto procesu je průměrná hodnota individuálních podílů špatně klasifikovaných instancí u každého k opakování. [13]

Teoreticky může být K libovolné, ale doporučeným kompromisem je podle [4] a [15] volit $k = 5$ nebo $k = 10$. Metoda *CV* se potom nazývá *5-fold cross validation* respektive *10-fold cross validation* (někde uváděny jako *5-fold CV* a *10-fold CV*). V extrémním případě lze použít i *vynech-jednu-instanci křížová validace (leave-one-out cross-validation)* neboli *LOOCV*. V tomto případě je odhad přesnosti modelu pomocí *CV* definován jako

$$CV(\hat{f}) = \frac{1}{N}\sum_{i=1}^{N} L\left(y_i, \hat{f}^{-\kappa(i)}(x_i)\right), \tag{4.2.1}$$

kde $\hat{f}^{-\kappa(i)}$ představuje model trénovaný na všech kromě *i*-té instance.

Kromě počtu částí, na které jsou původní data rozdělena, je velice důležitý způsob rozdělení. V případě rozdělení podle pořadí jednotlivých instancí hrozí velké riziko nevyváženého podílu tříd v jednotlivých částech. Proto je žádoucí rozdělovat data tzv. *náhodným rozdělením* (*shuffeled sampeling*), protože tato podmínka sama o sobě vyvážené rozdělení všech tříd nezaručí, používá se tzv. *rovnoměrné náhodné rozdělení* (*stratified sampling*).

4.2 Bootstrap metody (Bootstrap Methods)

Je dán tréninkový datový soubor obsahující N instancí. Každá instance se skládá ze vstupního vektoru $\boldsymbol{x_i}$ a výstupu y_i. *Bootstrap* spočívá v náhodném výběru N prvků s opakováním z původního datového souboru. Tím je vytvořeno *Bootstrap* rozložení datového souboru.[13]

Na B takových rozloženích je možné odhadovat přesnost modelu. Podle [5] je pravděpodobnost vybrání i-té položky z datového souboru alespoň jednou zhruba $1 - (1/e) \approx 0{,}632$, z čehož vyplývá, že v každém *Bootstrap* rozložení není přibližně 36,85% instancí z původního souboru.

Přesnost se potom určuje obdobně jako v případě křížové validace. A tedy vynech-jeden (leave-one-out) Bootstrap odhad predikční chyby je

$$\widehat{Err} = \frac{1}{N}\sum_{i=1}^{N}\frac{1}{|C^{-i}|}\sum_{b\in C^{-i}} L\left(y_i, \hat{f}^{*b}(x_i)\right), \tag{4.3.1}$$

kde C^{-i} je soubor modelů trénovaných na *Bootstrap* rozložení neobsahujícím instanci i a $\hat{f}^{*b}$ je model vytvořený na konkrétním *Booststrap* rozložení. [13]

Bootstrap rozložení se využívá méně často pro určení přesnosti modelu než křížová validace. Jak je ale uvedeno v kapitole (3.2) dá se s úspěchem využívat pro generování datových souborů se sníženou vzájemnou závislostí, pří zachování reprezentace charakteru problému. [26]

5 NÁVRH A REALIZACE EXPERIMENTŮ

V praktické části práce, je popsán návrh experimentů porovnávajících přesnost tradičních a meta-learningových modelů na 20 různých datových souborech. Návrh experimentů byl rozdělen do těchto částí:

- výběr modelů (kapitola 5.1)
- výběr datových souborů (kapitola 5.2)
- předzpracování datových souborů (kapitola 5.3)
- hledání vhodných parametrů individuálně pro každý model (kapitola 5.3)
- odhad přesnosti modelu metodou *10-fold Cross Validation* (kapitola 5.3)
- porovnání dosažených výsledků přesností pomocí *Wilcoxon Signed Rank Testu* (kapitola 5.4)
- stanovení statisticky významného rozdílu mezi dosaženými výsledky jednotlivých modelů. (kapitola 5.4)

Podle zadání byly naprogramovány některé modely v programovacím jazyce *C#*. V kapitole 5.5 je popsána vlastnoručně naprogramovaná aplikace.

5.1 Výběr modelů.

Cílem této bakalářské práce bylo srovnání 5 tradičních a 3 meta-learningových modelů. V (Tab. 1) je výčet vybraných tradičních modelů, které byly vybrány s ohledem na co největší rozsah běžně používaných metod ve strojovém učení. V prvním sloupci tabulky je obecně užívaný název vybraného modelu, ve druhém sloupci je název konkrétního modelu, který byl vybrán z nabídky operátorů v programu RapidMiner.

Tabulka 1: Tradiční modely

Název modelu	Operátor programu RapidMiner
k-nebližších sousedů	k-NN
Rozhodovací stromy	Decision Tree
Neuronové sítě	W-MultilayerPerceptron
Naivní Bayesův klasifikátor	W-NaiveBayesMultinomial
SVM	W-SMO

V (Tab. 2) je výčet použitých ensemble modelů. Protože bylo záměrem důkladné porovnání obou přístupů k modelování, byly pro každý z modelů *AdaBoost* a *Bagging* vybrány 3 různé varianty base modelů. U *Baggingu* to je *rozhodovací strom*, *neuronová síť* a *Random Forest*, na který je v této práci nahlíženo jako na speciální případ *Baggingu*. U *AdaBoostu* je to také *rozhodovací strom* a *neuronová síť*, ale třetí byl zvolen *Decision stump*. V prvním sloupci (Tab. 2) je název ensemble modelu s jeho base modelem. Ve druhém a třetím sloupci jsou potom vybrané operátory programu *RapidMiner*.

Tabulka 2: MetaLearningové modely

Název modelu	Operátor programu RapidMiner	
	Ensemble	Base
Bagging (Rozhodovací strom)	W-Bagging	W-J48
Bagging (Neuronová síť)	Bagging	Neural Net
Random Forest	W-RandomForest	-
AdaBoost (Decision Stump)	W-AdaBoostM1	W-DecisionStump
AdaBoost (Rozhodovací strom)	W-AdaBoostM1	W-J48
AdaBoost (Neuronová síť)	W-AdaBoostM1	W-MultilayerPerceptron
Stacking (k-nebližších sousedů, Neuronová síť, Naivní Bayesův Klasifikátor, SVM)	W-Stacking	W-IBk
		W-MultilayerPerceptron
		W-NaiveBayesMultinomial
		W-SMO

Legenda ke všem následujícím tabulkám v této práci: KNN- *k-nejbližších sousedů*, DT – *Rozhodovací strom*, NN – *Neuronová síť*, SVM – *metoda podpůrných vektorů*, NB – *Naivní Bayesův klasifikátor*, DS – *Decision Stump*, B – *Bagging*, A – *AdaBoost*, ST – *Stacking*, [W] - model implementovaný vlastnoručně naprogramovanou aplikací *WML*.

5.2 Výběr datových souborů

Pro potřeby porovnání bylo vybráno 20 datových souborů standardně používaných jako referenční data v komunitě, zabývající se strojovým učením. Byl brán ohled na různorodost v oblasti původu dat, počtu instancí, počtu atributů a počtu klasifikačních tříd. Zároveň byly použity datové soubory s různým poměrem kvantitativních a kvalitativních veličin mezi vstupními atributy. Všechny datové soubory jsou ke stažení z databáze *UCI Machine Learning Repository* [22] podle referenčních názvů. V této kapitole budou stručně popsány. Údaje o počtu instancí a atributů v (Tab. 3) odpovídají upraveným verzím datových souborů. Tyto úpravy jsou blíže popsány v kapitole 5.3.

Balance Scale Data Set

Datový soubor simuluje výsledky psychologického experimentu. Instance jsou klasifikovány do 3 tříd. Jsou to váha nakloněná *doprava*, *doleva* nebo *rovná se*. Skutečná třída se řídí jednoduchým pravidlem páky podle toho, která hodnota je větší (*vzdálenost od středu vlevo * levá váha*) a (*vzdálenost od středu vpravo * pravá váha*) pokud se sobě zásuvky rovnají, jde o třídu *rovná se*. *Vzdálenost od středu vlevo*, l*evá váha*, *vzdálenost od středu vpravo*, *pravá váha* jsou čtyři vstupní atributy.

Bank Marketing Data Set

Data portugalské bankovní instituce. Marketingová telefonní kampaň, nabízející služby zákazníkům. Klasifikační třída značí, jestli zákazník nabídku přijal nebo nepřijal. U tohoto datového souboru je zajímavé, že je zde velký nepoměr mezi zastoupením jednotlivých tříd, téměř 9:1.

Banknote authentication Data Set

Identifikace pravých a falešných bankovek. Hodnoty jednotlivých atributů jsou vyextrahované z obrazové informace.

Blood Transfusion Service Center Data Set

Na základě daných údajů o četnosti návštěvy dárců krve je za úkol predikovat, jestli byli dárci darovat krev v březnu roku 2007.

Breast Cancer Wisconsin (Prognostic) Data Set

Určování jestli jde o benigní nebo zhoubný nádor prsu podle zadaných atributů.

Breast Tissue Data Set

Klasifikace prsní tkáně podle údajů impedančních měření na různých frekvencích. Mezi klasifikačními třídami jsou například tumoroidní, tukové nebo žlázové tkáně.

Cardiotocography Data Set

Diagnostika plodu pomocí CTG. Klasifikace do tříd *normální*, *podezřelý* a *patologický*.

Japanese Credit Screening Data Set

Kreditní prověrka žadatelů japonské finanční instituce. Klasifikace do dvou tříd *prošel* a *neprošel*.

Fertility Data Set

Posuzování plodnosti na základě životosprávy žen. Klasifikační třídy jsou *normální* a *ovlivněné*.

Heart Disease Data Set

Predikce srdečního onemocnění pacientů. Třída 0 značí zdravého pacienta. Třídy 1-4 představují různé druhy srdečního onemocnění.

Ionosphere Data Set

Posuzovaní kvality ionosféry podle průchodnosti pulsů generovaných 16 vysokofrekvenčními anténami o výkonu 6,4 kW. Klasifikované třídy jsou *Dobrá* a *Špatná*. *Dobrá* reprezentuje instance, s kvalitní strukturou ionosféry. *Špatná* reprezentují ty instance, u kterých struktura ionosféry není kvalitní.

Letter Recognition Data Set

Úloha spočívá v identifikaci všech 26 velkých písmen abecedy. Datový soubor je uměle vytvořen kombinací 20 druhů fontů. Písmena jsou převedena na 16 numerických atributů představujících počet hran, a hustotu rozložení jednotlivých pixelů obrazu.

Musk (Version 2) Data Set

Klasifikace molekul, které buď patří mezi vzorky pižma, nebo ne.

Nursery Data Set

Datový soubor sestavený při analýze vhodnosti uchazečů o předškolní vzdělávání (*nursery schools*). Třídy představují, jakou dítě dostane úroveň doporučení.

Ozone Level Detection Data Set

Předvídaní dní s nízkou ozonovou vrstvou podle vstupních atributů.

Seeds Data Set

Klasifikace tří druhů semen pšenice podle charakteristických znaků extrahovaných s rentgenových snímků.

Tic-Tac-Toe Endgame Data Set

Analýza rozehraných her piškvorek. Třída značí, jestli je možné hru vyhrát z perspektivy hráče křížků nebo ne.

Wine Data Set

Chemická analýza složení 3 odrůd vína vypěstovaných v jedné oblasti. Hladiny každé ze 13 složek charakterizují výslednou odrůdu.

Yeast Data Set

Analýza kvasinek. Predikce buněčných oblasti obsahujících protein.

Weight Lifting Exercises DataSet

Hodnocení efektivity posilovacích technik.

Tabulka 3: Soupis parametrů jednotlivých datových souborů

	Název datového souboru	Instance	Atributy	Třídy
1	Balance Scale Weight & Distance Database	624	4	3
2	Bank Marketing Data Set	18 084	32	2
3	Banknote authentication Data Set	1 372	4	2
4	Blood Transfusion Service Center Data Set	748	4	2
5	Breast Cancer Wisconsin (Prognostic) Data Set	682	9	2
6	Breast Tissue Data Set	106	9	6
7	Cardiotocography Data Set	2 126	40	10
8	Japanese Credit Screening Data Set	650	45	2
9	Fertility Data Set	100	9	2
10	Heart Disease Data Set	294	13	5
11	Ionosphere Data Set	350	34	2
12	Letter Recognition Data Set	20 000	16	26
13	Musk (Version 2) Data Set	476	166	2
14	Nursery Data Set	12 960	24	5
15	Ozone Level Detection Data Set	350	34	2
16	Seeds Data Set	210	7	3
17	Tic-Tac-Toe Endgame Data Set	958	27	2
18	Wine Data Set	178	13	3
19	Yeast Data Set	1 484	8	10
20	Weight Lifting Exercises DataSet	2 674	55	2

5.3 Realizace v prostředí RapidMiner

Hlavní část práce byla realizována v programu *RapidMiner 5*. *RapidMiner* je softwarová platforma firmy stejného názvu, která poskytuje integrované prostředí pro *strojové učení*, *data mining*, *text mining*, predikční analýzu a business analýzu atd. [20] Aktuální verze použitá pro celou práci byla 5.3.015. *Rapid?iner* je program napsány v programovacím jazyce Java.

5.3.1 Předzpracování datových souborů

Originální datové soubory stažené ze serveru UCI byly ve většině případu ve formátu csv. V programu RapidMiner jsou nástroje pro vkládání těchto souborů a vytváření tzv.

repositories. Jde o datovou strukturu, která obsahuje veškerá data z originálního souboru ve specifickém binárním formátu. Každý sloupec představuje jeden atribut a každý řádek představuje jednu instanci. Kromě standardních atributů je možné některým sloupcům přiřadit speciální vlastnosti, což je hlavně atribut, který představuje třídu. Specifikovat atribut třídy je nezbytné pro funkci veškerých modelů. Specifikace třídy je možné provést jak již při vkládání souboru do *repository*, tak až v samotném experimentu v programu RapidMiner pomocí speciálního operátoru

Ne všechny vybrané modely dokáží pracovat se všemi typy dat, a proto bylo nutné nastavit podmínky, které umožnují porovnat všechny modely na stejném datovém souboru.

Podmínky předzpracování

Data nesmí obsahovat instance s chybějícími hodnotami atributů. Aplikován operátor *Filter Example*s s podmínkou *no_missing_attributes.*

Žádný ze vstupních atributů nesmí být vyjádřen jako kvalitativní veličina. Aplikován operátor *Nominal to Numerical.* Pro převedení kvalitativních hodnot bylo použito tzv. *dummy coding*. To spočívá v nahrazení kvalitativní veličiny, N fiktivními atributy, kde N odpovídá počtu možných hodnot kvalitativní veličiny. Pro každou instanci je u n-tého atributu nastavena hodnota 1, pokud původní atribut nabýval n-té hodnoty původní veličiny, kde $n \in \{1, \dots, N\}$ a 0 pro všechny ostatní fiktivní atributy.

Třída musí být vyjádřena v kvalitativní veličině. Kvůli jisté nedokonalosti procesu importování *csv* souborů jsou někdy kvalitativní veličiny nesprávně identifikovány jako kvantitativní. Pro tyto datové soubory byl použit operátor *Numerical to Polynominal*, který provede převedení numerických hodnot na kvalitativní.

Datový soubor nesmí obsahovat žádný atribut, který nemá vliv na klasifikovanou třídu. Příkladem je například implicitně zadaný index instance nebo datum pořízení vzorku, které nijak neovlivňuje výstupní hodnotu. Pro tyto případy byl použit operátor *Select Attributes*, kde jako parametr byly nastaveny všechny vhodné atributy.

Všechna data musí být normalizována ve stejném rozsahu. Použit operátor *Normalize* s metodou *range transformation*. Rozsah výsledných hodnot byl nastaven v rozsahu 0 – 10.

Je vhodné zmínit, že některé modely (např. *Random Forest*) jsou citlivé na poměr relevantních vstupních atributů. Protože však problematika tzv. *feature selection* je sama o sobě poměrně rozsáhlá, byla tato forma předzpracování vynechána.

Výsledek předzpracovaní datových souborů byl uložen jako soubor typu aml pomocí operátoru *Write AML*. Výsledek této operace je soubor aml, který uchovává informace o atributech jako je jméno typ atd. a soubor dat, který obsahuje data ve speciálním formátu. Soubor dat byl použit jako vstupní formát aplikace WML naprogramované v jazyce C#.

5.3.2 Hledání vhodných parametrů

Pro nalezení vhodných parametrů modelu bylo použito procesu v rozložení na (Obr. 5). V části *1* jsou použity operátory *Read AML, Optimize Parameters (Grid)* a *Write as*

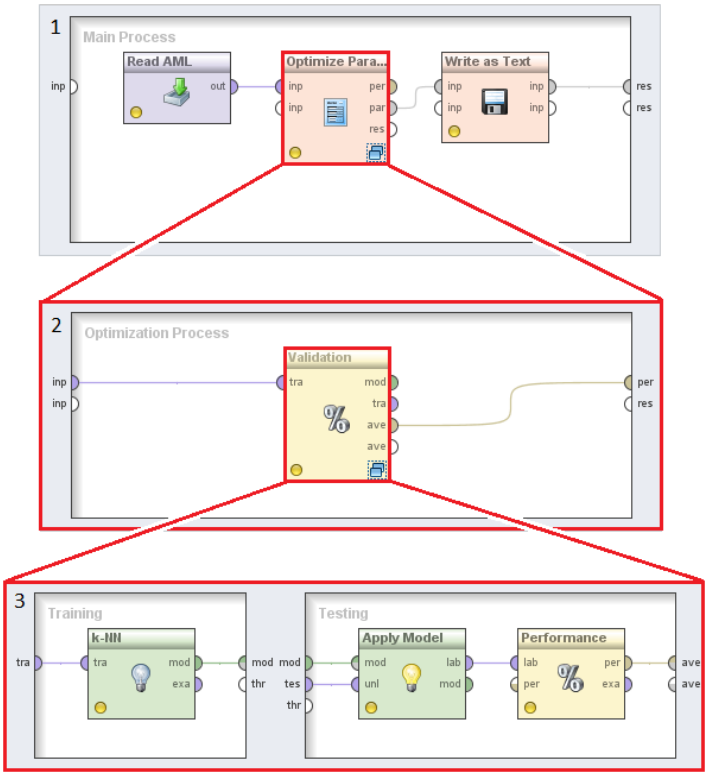
Text. V operátoru *Read AML* je specifikována cesta k *aml* souboru. Operátor *Write as Text* zapisuje výsledky dosažené operátorem *Optimize Parameters (Grid)* do textového souboru.

Operátor *Optimize Parameters (Grid)* prochází lineární kombinace vybraných parametrů v nastaveném rozsahu a vyhodnocuje nejlepší dosažený výsledek vnitřního procesu, podle zadaného kritéria. Jako výstup procesu jsou parametry a dosažený výsledek nejlepší kombinace.

Vnitřní proces je zobrazen na (Obr. 5) v části 2. Kritériem pro nalezení nejlepší kombinace byl použit operátor *X-Validation*, což je implementace metody *CV*, v tomto případě *5-fold CV*. Tato metoda byla zvolena především kvůli menší výpočetní náročnosti než přesnější 10-*fold CV*. Vnitřní struktura operátoru *X-Validation* je zobrazena na (Obr. 5) v části 3.

V tomto konkrétním případě jsou hledány parametry pro model k-nebližších sousedů reprezentovaný operátorem *k-NN*. Testování modelu je vždy provedeno operátorem *Apply Model*, který přebírá informace o modelu z operátoru *k-NN*. Výstupem operátoru *Apply Model* je datový soubor s predikovanými třídami. Operátor *Performance (Classification)* porovnává chybu testovacího modelu podle zadaného kritéria. V tomto procesu je kritériem *accuracy*, což je relativní počet správně klasifikovaných instancí.

Všechny textové soubory, které obsahují optimální hodnoty parametrů, se nacházejí na přiloženém CD.

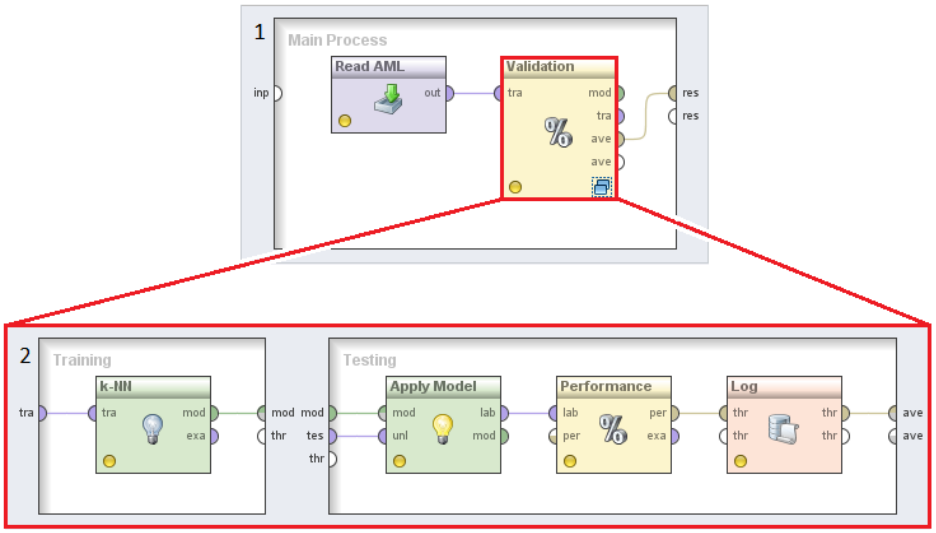


Obrázek 5: Náhled procesu hledání vhodných parametrů. [10]

5.3.3 Odhad přesnosti modelu

Při odhadu výsledné přesnosti bylo použito procesu v rozložení na (Obr. 6). Hlavní rozdíly od procesu v kapitole 5.3.2 jsou, že operátor *Optimize Parameters (Grid)* byl

nahrazen operátorem *X-Validation* implementujícím *10-fold CV* jak je vidět v části 1. V části 2, která představuje vnitřní strukturu operátoru *X-Validation*, byl přidán operátor *Log*. *Log* je zde, aby průběžně zaznamenával do souboru hodnoty jednotlivých přesností během vykonávání operátoru *X-Validation.* Tyto výsledky jsou potřebné pro vyhodnocení *Wilcoxon signed rank testu*. Parametry jednotlivých modelů, byly nastaveny podle parametrů nalezených pomocí procesu popsaného v kapitole 5.3.2.



Obrázek 6: Náhled procesu určování přesnosti metodou 10-fold Cross Validation. [10]

5.4 Porovnání modelů programu RapidMiner

5.4.1 Porovnání modelů pomocí Wilcoxon signed rank testu

Pro porovnání modelů na jednotlivých souborech byl použit *Wilcoxon signed rank test*, což je neparametrický statistický test hypotéz. Hypotéza H_0 říká, že střední hodnoty rozložení obou vzorků jsou stejné. V tomto případě byl proveden jednostranný test a hypotéza H_1říká, že střední hodnoty rozložení obou vzorků jsou odlišné a v konkrétním případě, že přesnost referenčního modelu na daném datovém souboru je lepší.

Pomocí průměrné hodnoty přesnosti odvozené metodou *10-fold CV* byl stanoven referenční model pro každý datový soubor. Pro porovnání výsledků jednotlivých modelů byla naprogramována konsolová aplikace, která bude stručně popsána v následující kapitole. Její funkce spočívá v načtení textových souborů obsahujících výčet jednotlivé přesnosti, určených metodou *křížové validace*. Pro každý datový souboru byl pomocí jednostranného *Two sample Wilcoxon signed rank* testu [1] porovnán referenční model s ostatními. Pří vyvrácení hypotézy H_0 byla soupeři referenčního modelu zaznamenána prohra, v opačném případě je zaznamenána remíza. Remíza je značena 0, prohra je značena -1. (Tab. 4) zobrazuje výsledné hodnoty každého souboje.

5.4.2 Testování statistické hypotézy

Pro porovnání jednotlivých modelů byl použit parametrický statistický test hypotéz. Pro statistický test významnosti je stanoven předpoklad že skoré modelů má binomické rozložení. Pro výpočet hraničního skoré bylo binomické rozložení aproximováno normálním rozložením se střední hodnotou $\mu = 10$ a rozptylem $\sigma^2 = 5$. Podle $\mu = N * p$ a $\sigma^2 = N * p(1-p)$, kde $N = 20$ je počet datových souborů a $p = 0{,}5$, protože apriorně se předpokládá, že jsou si modely rovny.

Je stanovena hypotéza H_0, která říká, že oba modely dosahují stejné přesnosti. Alternativní hypotéza H_1 je jednostranná a říká, že referenční model dosahuje větší přesnosti. Hypotézu H_0 je možné zamítnout s hladinou významnosti $\alpha = 0{,}05$.

Hodnoty proher a remíz z (Tab. 4) byly vztaženy vůči modelu s nejvyšší průměrnou přesností na všech modelech. A zobrazeny v (Tab. 5). Vítězství značené jako jedna představuje pro konkurenční model jeden bod, remíza představuje půl bodu a prohra nula bodů. Podle definovaného rozložení je hodnota pravděpodobností pro skoré 6,5 rovna $P\,(X < 6.5) = 0{,}0582$ a pro skoré 6 se rovná $P\,(X < 6) = 0{,}0367$. Proto, aby bylo možné zamítnout hypotézu H_0 musí být skoré konkurenčního modelu maximálně 6, protože skoré může nabývat hodnot po půl bodech.

Tabulka 4: Zobrazené výsledky Wilcoxon Signed Rank testu.

	KNN	DT	NN	SVM	NB	B(DT)	B(NN)	RF	A(DT)	A(DS)	A(NN)	ST
1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1		-1
2	-1	-1	-1	-1	-1		-1	-1	0	0	-1	-1
3	0	-1	-1	0	-1	-1	-1	-1	0	0		0
4	0	0	0	0	0	0	0	-1	0		-1	0
5	0	-1	0	0	-1	0	0	0	0	0	0	
6	0	0	-1	0	-1	-1	-1	0	0	-1	0	
7	0	0	0	0	0	0	0	0	0	-1	0	
8	0	0	-1	-1	0	0	-1		-1	0	-1	0
9	0	0	0	0	-1	0		0	0	0	0	0
10	0	-1	0	0	-1	0	0	0	0	-1	-1	
11	-1		-1	0	-1	0	0	0	0	0	0	0
12	-1	-1	-1	-1	-1	-1	-1	-1		-1	-1	-1
13	-1	-1	0	-1	-1	-1	0	-1	0	-1	-1	
14	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	
15	-1	0	-1	-1	-1	0	0		0	0	0	0
16	0	-1	0	0	-1	0	0	0	0	0	0	
17	-1	-1	-1	0	-1	0	-1	-1		-1	-1	0
18	0	-1	0	0	-1	0	0	0	0	0		0
19	-1	-1	-1	-1	-1	0	-1	-1	-1	-1	-1	0
20	0	0	0	0	-1	0	0	0	0	0	0	

Prohra je značena -1 (červené pole) a remíza je značena 0 (zelené pole). Pro každý datový soubor je vybrán jeden referenční model (světle šedé pole) a ostatní jsou s ním srovnávány.

Tabulka 5: Zobrazené výsledky Wilcoxon Signed Rank testu vztažené vůči nejlepšímu modelu

	KNN	DT	NN	SVM	NB	B(DT)	B(NN)	RF	A(DT)	A(DS)	A(NN)	ST
1	0	0	0	0	0	0	0	0	0	0	1	
2	0	0	0	0	0	1	0	0	1	1	0	
3	0	-1	-1	0	-1	-1	-1	-1	0	0	0	
4	0	0	0	0	0	0	0	-1	0	0	-1	
5	0	-1	0	0	-1	0	0	0	0	0	0	
6	0	0	-1	0	-1	-1	-1	0	0	-1	0	
7	0	0	0	0	0	0	0	0	0	-1	0	
8	0	0	-1	-1	0	0	-1	0	-1	0	-1	
9	0	0	0	0	-1	0	0	0	0	0	0	
10	0	-1	0	0	-1	0	0	0	0	-1	-1	
11	-1	0	-1	0	-1	0	0	0	0	0	0	
12	0	0	0	0	0	0	0	0	1	0	0	
13	-1	-1	0	-1	-1	-1	0	-1	0	-1	-1	
14	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	
15	-1	0	-1	-1	-1	0	0	0	0	0	0	
16	0	-1	0	0	-1	0	0	0	0	0	0	
17	-1	-1	-1	0	-1	0	-1	-1	0	-1	-1	
18	0	-1	0	0	-1	0	0	0	0	0	0	
19	-1	-1	-1	-1	-1	0	-1	-1	-1	-1	-1	
20	0	0	0	0	-1	0	0	0	0	0	0	
Scóré:	7	**5,5**	**6**	7,5	**3**	8,5	7	7	9,5	7	7	-

Nejlepší model (světle šedé pole). Výhra je značena 1 (zelené pole). Prohra je značena -1 (červené pole) a remíza je značena 0 (žluté pole). V posledním řádku je zobrazeno skoré. Pokud je skoré statisticky významně horší než u referenčního modelu je zobrazeno červeně.

5.5 Realizace vlastním kódem v jazyce C#

Pro účely této práce byly vytvořeny dvě aplikace v programovacím jazyce C#. První aplikace podle zadání implementuje vybrané modely strojového učení. Aplikace byla nazvána *WFA Machine Learning*. V následujícím textu nazývána *WML*

Koncepce *WML* byla inspirována aplikací *RapidMiner*. Byl proto vytvořen framework, ve kterém je každý model brán jako operátor, který má definovány obecné operace pomocí rozhraní a je tím pádem jednoduché vytváření ensemble modelů. Ve zdrojovém kódu byla vytvořena poměrně rozsáhlá dokumentace v jazyce XML, která popisuje strukturu a jednotlivé metody. *WML* je takto nachystána na možné rozšíření.

V aktuální verzi *WML* bylo vypracováno 5 modelů. Mezi čtyři tradiční patří *k-NN*, *Rozhodovací strom C4.5*, *Náhodný strom* a *Decision Stummp*. Jako zástupce ensemble modelů byl vytvořen *Bagging*, který používá jako *base* model buď *rozhodovací strom* a nebo *decision stump* a dále pak jeho modifikace s modelem *náhodného stromu*, které se jinak říká *Random Forest*.

Program byl vyvíjen jako konzolová aplikace, takže nebyl kladen velký důraz na uživatelské rozhraní. Pro demonstrační účely byla vytvořena jednoduchá *Windows Form Aplikace*, která zvládá reprezentovat funkci jeho nejdůležitějších částí.

Druhá aplikace byla vytvořena pro analýzu dat reprezentujících přesnost jednotlivých modelů vyprodukovaných programem *RapidMiner*. Je to poměrně jednoduchá konzolová aplikace, která načítá textové soubory vytvořené pomocí programu

RapidMiner obsahující výsledky jednotlivých přesností *10-fold CV*. Pro každý z 20 datových souborů porovná dvojice složené z referenčního (na daném datovém souboru nejlepšího podle *10-fold CV*) a každého ze zbylých modelů pomocí *Wilcoxon signed rank testu*. Výstup aplikace je textový soubor obsahující statistiku každého modelu. Zdrojové soubory obou aplikací jsou přiloženy na CD.

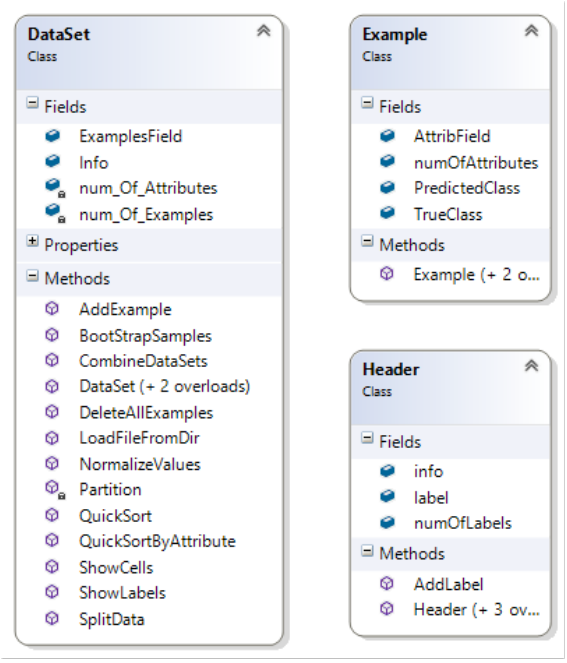
5.5.1 Popis zdrojového kódu WML

Zdrojový kód aplikace je poměrně rozsáhlý, a proto byl rozdělen do 6 souborů, které budou postupně podrobněji rozebrány.

DataSetClass.cs

Soubor obsahuje třídy *DataSet*, *Example* a *Header*, které uchovávají informace o datovém souboru. Prvek *ExampleField* reprezentuje datovou konstrukci List třídy *Example*. *Info* představuje instance třídy *Header*. Na (Obr. 7) jsou zobrazeny diagramy těchto tříd. Ve třídě *DataSet* jsou mezi jinými implementovány hlavně metody:

LoadFileFromDir - Pro načítání dat ze souboru.
BootStrapSamples - Pro vytvoření Bootstrap rozložení.
QuickSortByAttribute - K seřazení instancí podle zadaného atributu algoritmem quicksort.
SplitData- Pro rozdělí rovnovážně instance všech tříd do daného počtu nových datových souborů.

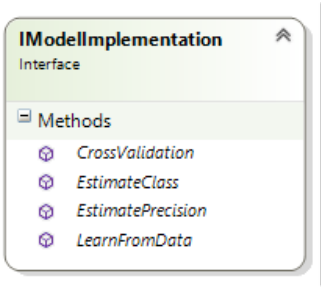


Obrázek 7: Náhled tříd obsažených v souboru DataSetClass.cs aplikace W*ML* [8]

IModelImplementation.cs

V tomto soboru je definováno rozhraní (*interface*). Rozhraní je implementováno třídami *KnnModel*, *BaggingModel*, *RandomForestModel*, *DecisionTreeModel*, *RandomTreeModel* a *DecisionStumpModel*. Všechny zmíněné modely tedy musely implementovat vlastní verzi těchto metod:

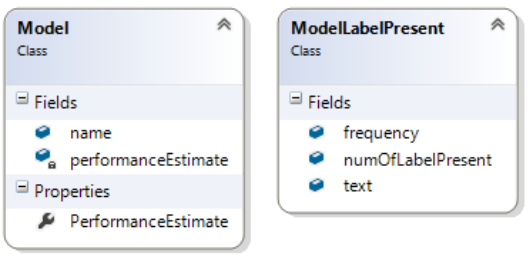
CrossValidation - Metoda plnící funkci křížové validace. Její vstupní parametry jsou Datový soubor a počet datových rozložení *K*. Výstupním parametrem je pole obsahující dosaženou přesnost na jednotlivých datových rozloženích.
EstimateClass - Metoda, která provádí proces klasifikování neznámé instance. Vstupní hodnota je objekt třídy *Example* a výstupní parametr je string hodnota klasifikované třídy.
EstimatePrecision - Metoda volající *EstimateClass* na všechny instance v tréninkovém datovém souboru, který je zadán jako vstupní parametr. Výstupní hodnota je průměrná přesnost modelu na daném datovém souboru.
LearnFromData - Metoda, která učí model na základě vstupního parametru tréninkového datového souboru.



Obrázek 8: Rozhraní v souboru IModelImplementation.cs aplikace *WML* [8]

ModelClass.cs

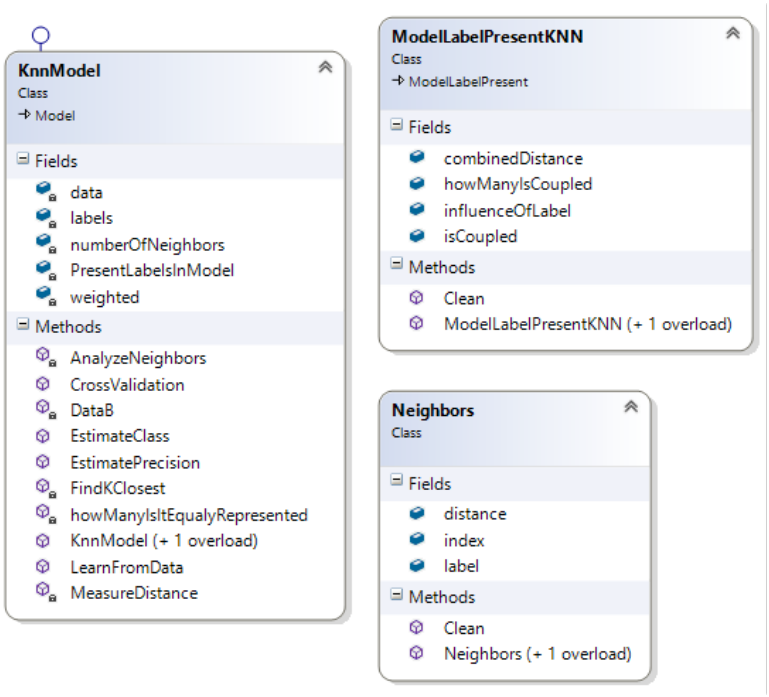
Model třída obsahuje třídu *Model* a *ModelLabelPresent*. Z každé z těchto tříd dědí ostatní modely. Třída *Model* neimplementuje žádné rozhraní a obsahuje pouze název modelu a jeho přesnost. Třída *ModelLabelPresent* obsahuje informace o třídách obsažených v aktuálním modelu. Parametry jsou četnost jednotlivých tříd, počet tříd a pole *string* hodnot obsahující názvy jednotlivých tříd.



Obrázek 9: Náhled tříd obsažených v souboru ModelClass.cs aplikace *WML* [8]

KnnClass.cs

Tento soubor obsahuje 3 třídy *KnnModel*, *Neighbors* a *ModelLablelPresentKNN*. *KnnModel* dědí ze třídy *Model* a implementuje rozhraní *IModelImplementation*. Model je schopný hledat teoreticky libovolný počet nejbližších sousedů a zvládá posuzovat vážený vliv nejbližších sousedů. Nejbližší sousedy hledá pomocí metody *FindKClosest*, která vrací pole objektů *Neighbors* představující *K* nejbližších sousedů. Metoda *AnalyzeNeighbors* analyzuje všechny nejbližší sousedy a podle zadaných parametrů vrací *string* hodnotu klasifikované třídy. *ModelLablelPresentKNN* je součastí *KnnModel* a nese informaci o třídách datového souboru a také přechodně uchovává, kolik instancí, jaké třídy se nachází mezi nejbližšími sousedy. Nerozhodné situace vznikající stejně zastoupenými třídami mezi nejbližšími sousedy, jsou rozhodnuty podle nejbližší instance mezi nerozhodnými třídami. Pokud ani toto kritérium neurčí jasně třídu a nastane znovu nerozhodná situace, tedy dvě instance jsou stejně vzdálené od určované, je vybrána ta třída, která má menší index.



Obrázek 10: Náhled tříd obsažených v souboru KnnClass.cs aplikace *WML* [8]

TreeClass.cs

TreeClass.cs je nejrozsáhlejší část programu, a to proto, že zahrnuje tři modely na bázi rozhodovacích stromů. Soubor obsahuje třídy *TreeModel*, *DecisionTreeModel*, *RanodomTreeModel*, *DecisionStump*, *Node*, *ModelLabelPresentTree* a strukturu *TreeRuleOfDivision*.

Třída TreeModel dědí ze třídy *Model* a obsahuje základní definici modelu rozhodovacího stromu. ModelLabelPresentTree stejně jako u KNN obsahuje informace o třídách, které jsou přítomné v modelu. Další položka třídy je instance třídy Node (*rootNode*) reprezentující kořenový uzel rozhodovacího stromu.

Třída *Node*, obsahuje tyto prvky:

DataInNode - Uchovává instance, které při učení spadají do konkrétního uzlu.
Label - V případě, že se jedná o list, je to informace o klasifikační třídě, do které je instance zařazena nová instance.
LeafNode - Značí, zda se jedná o list.
LevelOfNode - Udává hloubku uzlu.
LowerNodes - Reference na dva uzly, které jsou ve struktuře pod tímto uzlem.
NodeEntropy - Udává hodnotu entropie obsaženou v datech daného uzlu.
RootNode - Značí, zda se jedná o list kořenový uzel.
RuleOfDivision - Obsahuje strukturu, která určuje pravidlo, pomocí kterého se klasifikují nové instance.
UpperNode - Reference uzlu nacházejícího se o úroveň výše ve struktuře stromu.

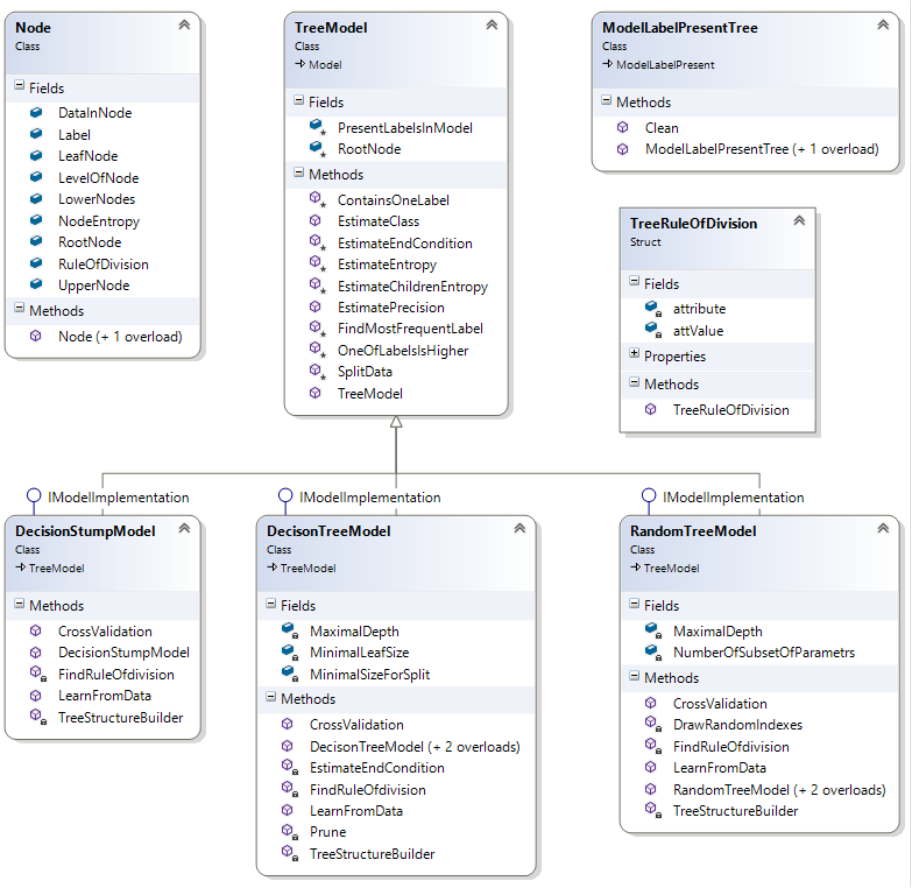
Třída *TreeModel* dále obsahuje metody společné pro všechny třídy na bázi stromů. Patří mezi ně zejména:

ContainsOneLabel - Metoda testuje, zda se v uzlu nachází pouze instance jedné třídy.
EstimateEndCondition - Metoda, která testuje souhrnně zadané ukončovací podmínky
EstimateEntropy - Vypočítá entropii dat obsažených v uzlu.
EstimateChildreEntropy - Vypočítá potenciální změnu entropie při rozdělení dat.
FindMostFrequentLabel - Metoda vrací nejčastěji zastoupenou třídu v uzlu.
OneOfLabelsIsHigher - Ukončovací podmínka založená na většinovém podílu jedné třídy.
SplitData - Rozdělí data podle definovaného pravidla.

Třídy *DecisonTreeModel*, *RandomModel* a *DecisionStumpModel* dědí z třídy *TreeModel* a každá z nich implemetnuje rozhraní *IModelImplementation* a zároveň každý z nich implementuje svoje verze metody *TreeStructureBuilder* a *FindRuleOfdivision*. *TreeStructureBuilder* je rekurzivní metoda, která vytváří strukturu stromu a *FindRuleOfdivision* je metoda, která hledá optimální atribut a jeho hodnotu pro rozdělení dat podle informačního zisku

Poslední částí souboru *TreeModel.cs* je struktura *TreeRuleOfDivision*, která pouze zapouzdřuje dvojici atribut a jeho hodnotu. Tato struktura je součástí každého uzlu a je podle ní nejprve vytvářen strom a následně jsou pomocí ní klasifikovány nové instance.

Třída *DecisionTreeModel* má nastavitelné 3 parametry. *MaximalDepth*, který ovlivňuje maximální hloubku stromu, *MinimalLeafSize*, který stanovuje jaká je spodní hranice počtu instancí pro vytvoření listu *a MinimalSizeForSplit*, který určuje, kolik musí mít uzel minimálně prvků, aby se mohl rozdělit. *RandomTreeModel* má též parametr *MaximalDepth* a dále *NumberOfSubsetOfParameters*, který určuje počet náhodně vybraných atributů při vytváření nového uzlu stromu.



Obrázek 11: Náhled tříd obsažených v souboru TreeClass.cs aplikace *WML*[8]

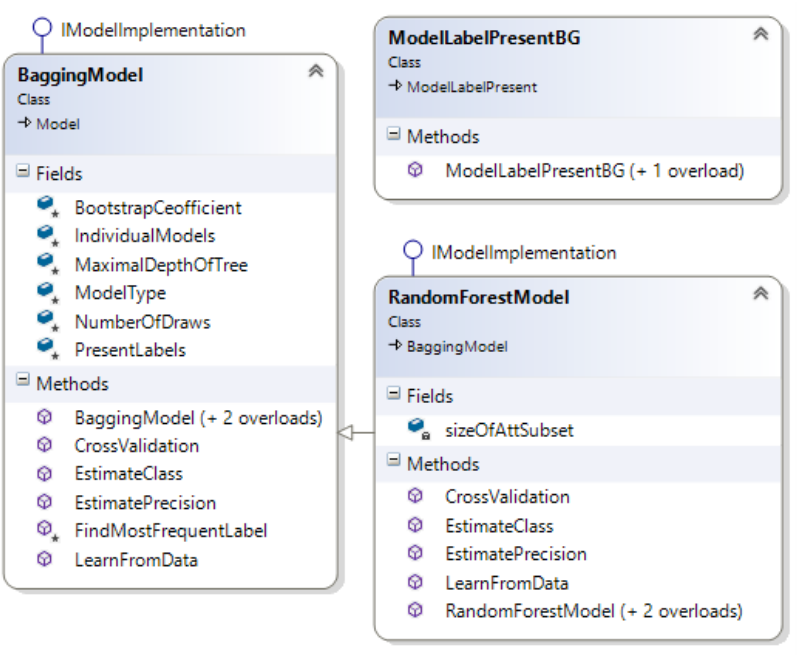
BaggingClass.cs

Poslední částí aplikace je třída implementující ensemble modely *Bagging* a *RandomForest*. Protože robustnost modelu k-NN není obvykle příliš zlepšitelná pomocí ensemble metod, byly oba modely zaměřeny na *base* model typu rozhodovacího stromu. Třída *BaggingModel* dědí ze třídy *Model*. Třída *RandomForestModel* dědí ze třídy *BaggingModel*.

Hlavní prvky definující modely jsou:

BootstrapCoeficient - Udává procentuální část datového souboru generovaného *Bootstrap* rozložení.
IndividualModels - Pole prvků *IModelImplementation*. Protože jde o pole rozhraní je teoreticky možné použít jako base model libovolný model implementující toto rozhraní.
MaximalDepthOfTree - Parametr pro base model typu rozhodovacího stromu.
ModelType - Definuje typ base modelu (pouze *BaggingModel*).
NumberOfDraws - Počet iterací ensemble modelu. Ovlivňuje velikost pole *IndividualModels*.
PresentLabels - Objekt třídy ModelLabelPresentBG.
SizeOfSubset - Udává počet atributů, které jsou náhodně vybrány při vytváření každého nového uzlu (pouze *RandomForestModel*).

Dále oba modely používají funkci *FindMostFrequentLabel*, která hledá nejčastěji predikovanou třídu mezi base modely ensemblu. Zbytek metod funguje obdobně jako u ostatních modelů.

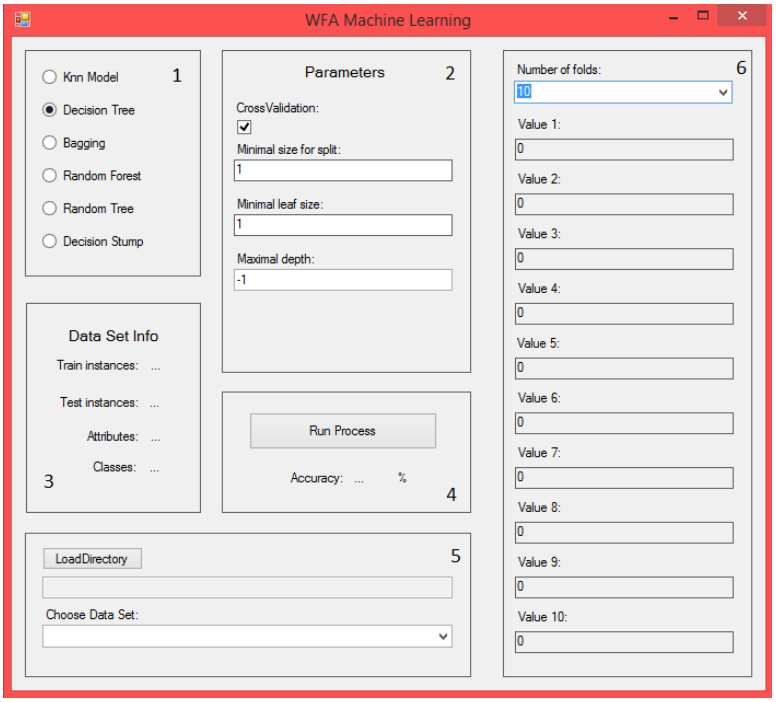


Obrázek 12: Náhled tříd obsažených v souboru BaggingClass.cs aplikace *WML*[8]

Uživatelské rozhraní

Na (Obr. 13) je náhled uživatelského rozhraní aplikace *WML*. Základní obrazovka je rozdělena do 5 panelů. V prvním panelu je na výběr ze 6 typů modelů. Ve druhém jsou nastavitelné parametry různé pro každý model. Panel číslo 3 zobrazuje informace o aktuálně načteném datovém souboru. Panel 4 obsahuje tlačítko *Run Process*, které pouští proces a hodnotu vypočtené přesnosti provedeného procesu *Accuracy*. Panel 5 obsahuje tlačítko *LoadDirectory*, pomocí kterého je spuštěn dialog pro nastavení adresáře v počítači obsahujícího datové soubory. Dalším prvkem je textové pole zobrazující cestu k aktuálně nastavenému adresáři. Třetí prvek je rolovací nabídka s výběrem ze 20 datových souborů.

V případě, že je zatržena možnost *CrossValidation*, v panelu 2 je rozšířen formulář doprava. Na rozšířeném prostoru se zobrazí panel 6. Na tomto panelu je na výběr mezi *5-fold* a *10-fold CV*. Pokud je zatržena možnost *CrossValidation*, je po stisknutí tlačítka *Run Process* provedena zvolená metoda *CrossValidation* na datovém souboru vytvořeném spojením tréninkových a testovacích dat. Po dokončení procesu jsou výsledky jednotlivých přesností vypsány do objektů *textbox* na panelu 6. Zároveň je průměrná hodnota na všech vzorcích vložena do položky *Accuracy* v panelu 4.



Obrázek 13 : Náhled uživatelského rozhraní aplikace *WML* [9]

5.5.2 Porovnání vlastnoručně naprogramovaných modelů

Vlastnoručně naprogramované modely *K-nejbližších sousedů*, *Bagging* (*rozhodovací stromy*) a *Random Forest* byly porovnány se svými ekvivalenty programu RapidMiner. Byla aplikována stejná metoda jako při porovnávání modelů RapidMiner, ale byly porovnány vždy jen dva ekvivalentní modely proti sobě. Výsledky porovnání jsou v kapitole 6.

6 DOSAŽENÉ VÝSLEDKY A VYHODNOCENÍ

Výsledky přesností modelů implementovaných programem *RapidMiner* získané metodou *10-fold CV* jsou v (Tab. 6). Průměrná přesnost jednotlivých modelů seřazena od nejlepší po nejhorší je v (Tab. 7). Jak je vidět nejlepší průměrné přesnosti dosáhl model *Stacking*. Tři modely jsou statisticky významně horší.

Tabulka 6: Výsledky přesností metody 10-fold CV modelů implementovaných v programu RapidMiner.

	KNN	DT	NN	SVM	NB	B(DT)	B(NN)	RF	A(DT)	A(DS)	A(NN)	ST
1	0,90	0,81	0,91	0,92	0,87	0,88	0,95	0,83	0,76	0,75	0,96	0,92
2	0,88	0,88	0,89	0,89	0,69	0,89	0,89	0,89	0,89	0,89	0,87	0,89
3	1,00	0,98	0,98	0,99	0,71	0,99	0,99	0,99	1,00	1,00	1,00	1,00
4	0,79	0,77	0,78	0,76	0,77	0,78	0,77	0,72	0,77	0,79	0,76	0,78
5	0,97	0,95	0,96	0,97	0,91	0,96	0,96	0,97	0,97	0,96	0,96	0,97
6	0,67	0,64	0,61	0,70	0,60	0,69	0,41	0,70	0,70	0,41	0,71	0,76
7	1,00	1,00	1,00	1,00	1,00	1,00	1,00	1,00	1,00	0,45	1,00	1,00
8	0,88	0,86	0,85	0,86	0,87	0,87	0,86	0,88	0,86	0,88	0,84	0,87
9	0,89	0,82	0,83	0,85	0,71	0,88	0,89	0,85	0,84	0,87	0,88	0,88
10	0,68	0,63	0,67	0,65	0,60	0,67	0,66	0,67	0,64	0,64	0,64	0,69
11	0,88	0,94	0,62	0,88	0,76	0,92	0,92	0,94	0,93	0,92	0,91	0,89
12	0,96	0,09	0,83	0,85	0,55	0,91	0,87	0,95	0,97	0,07	0,94	0,96
13	0,86	0,75	0,94	0,86	0,64	0,90	0,95	0,89	0,94	0,89	0,90	0,95
14	0,55	0,49	0,56	0,53	0,49	0,52	0,56	0,27	0,34	0,43	0,55	0,59
15	0,88	0,92	0,62	0,88	0,76	0,92	0,92	0,94	0,94	0,92	0,91	0,89
16	0,95	0,89	0,95	0,96	0,87	0,93	0,95	0,93	0,94	0,90	0,96	0,96
17	0,93	0,94	0,98	0,98	0,67	0,99	0,97	0,98	0,99	0,98	0,98	0,98
18	0,96	0,87	0,98	0,98	0,91	0,96	0,98	0,98	0,97	0,93	0,99	0,98
19	0,59	0,52	0,58	0,59	0,49	0,62	0,60	0,60	0,60	0,41	0,60	0,62
20	1,00	1,00	1,00	1,00	0,88	1,00	1,00	1,00	1,00	1,00	1,00	1,00
průměr	0,86	0,79	0,83	0,86	0,74	0,86	0,85	0,85	0,85	0,75	0,87	0,88

Ve sloupcích jsou modely a v řádcích datové soubory. Barvy buněk obsahující přesnost, znázorňují relativní přesnost v rámci jednoho datového souboru (zelená - nejlepší, červená - nejhorší).

Tabulka 7: Seřazení modelů podle průměrné přesnosti získané metodou *10-fold CV*

ST	A(NN)	B(DT)	KNN	SVM	RF	A(DT)	B(NN)	NN	DT	A(DS)	NB
0,88	0,87	0,86	0,86	0,86	0,85	0,85	0,85	**0,83**	**0,79**	0,75	**0,74**

Červeně jsou zobrazeny ty modely, které jsou podle statistického testu významně horší.

Podle postupu popsaného v kapitole 5.5.2 byly porovnány modely programů *RapidMiner* a *WML*. V (Tab. 8) jsou zobrazeny přesnosti získané metodou *10-fold CV* jednotlivých modelů. V (Tab. 9) jsou ekvivalentní údaje jako v (Tab. 4) s tím rozdílem, že jsou vždy porovnávány pouze dva modely stejného typu proti sobě. (Tab. 10) je ekvivalent (Tab. 5) se stejným rozdílem jako v předešlém případě. Průměrná přesnost modelů programu *RapidMiner* je vždy lepší než aplikace *WML*. Podle skoré v (Tab. 10) je však statisticky významný rozdíl pouze u modelu *Baggign* (rozhodovací stromy) a u zbylých dvou modelů jsou výsledky srovnatelné.

Porovnání programu *RapidMiner* a *WML* je problematické hlavně s toho důvodu, že program *RapidMiner* má více parametrů, které je možné u jednotlivých modelů nastavit.

Jedinou výjimkou je v tomto případě model *k-nejbližších sousedů*, protože u něj jsou nastavitelné parametry v zásadě dva, které byly oba implementovány. Další dva modely používají jako base model varianci *rozhodovacího stromu*, který není u obou aplikací ekvivalentní. Druhým problém je, že každá aplikace používá vlastní implementaci metody křížové validace.

Tabulka 8: Porovnání přesnosti modelů programů *RapidMiner* a *WML*

	KNN	KNN [W]	B(DT)	B(DT) [W]	RF	RF [W]
1	0,90	0,92	0,88	0,73	0,83	0,77
2	0,88	0,89	0,89	0,88	0,89	0,89
3	1,00	1,00	0,99	0,84	0,99	0,99
4	0,79	0,64	0,78	0,77	0,72	0,77
5	0,97	0,97	0,96	0,93	0,97	0,95
6	0,67	0,70	0,69	0,50	0,70	0,70
7	1,00	1,00	1,00	0,47	1,00	1,00
8	0,88	0,86	0,87	0,87	0,88	0,87
9	0,89	0,84	0,88	0,89	0,85	0,83
10	0,68	0,71	0,67	0,69	0,67	0,68
11	0,88	0,87	0,92	0,84	0,94	0,90
12	0,96	0,96	0,91	0,07	0,95	0,92
13	0,86	0,82	0,90	0,72	0,89	0,88
14	0,55	0,52	0,52	0,43	0,27	0,45
15	0,88	0,86	0,92	0,84	0,94	0,91
16	0,95	0,94	0,93	0,65	0,93	0,86
17	0,93	0,97	0,99	0,70	0,98	0,87
18	0,96	0,96	0,96	0,64	0,98	0,99
19	0,59	0,61	0,62	0,41	0,60	0,59
20	1,00	1,00	1,00	0,91	1,00	1,00
průměr	0,86	0,85	0,86	0,69	0,85	0,84

Barvy buněk obsahující přesnost, znázorňují relativní přesnost v rámci jednoho modelu (zelená - nejlepší, červená - nejhorší).

Pokud je přesnost jediné hledisko pro výběr modelu, vychází lépe ensemble modely, protože obsazují první 3 místa. Na druhou stranu tradiční modely *k-nejbližších sousedů* a *SVM* obsazují 4. a 5. místo. Jako rozhodně nejslabší je možné označit modely *rozhodovací strom* a *naivní Bayesův klasifikátor*. U modelu *Adaboost* s base modelem *Decision stump* sice statistický test neprokázal, že je horší než *Stacking*, ale na základě průměrné přesnosti skončil předposlední.

Na základě přesnosti lze meta-modely doporučit, protože celkově dosáhly dobrých výsledků. Výsledky dále ukazují, že ensemble modely zvyšují přesnost svých base modelů. Například pro *Adaboost* s *neuronovými sítěmi* byla přesnost zvýšena o 4%. Pro *Bagging* s *rozhodovacími stromy* to bylo 7%. Při rozhodování mezi tradičními a meta-modely tedy hraje velkou roli, jestli je v konkrétním případě zvýšené nároky na výpočetní a časovou náročnost. Pokud je brána v potaz rychlost učení a jednoduchost implantace tak se rozumnou volbou jeví i model *k-nejbližších sousedů* a *Random Forest.*

Tabulka 9: : Zobrazené výsledky *Wilcoxon Signed Rank testu* pro porovnání modelů programu *RapidMiner* a *WML*

	KNN	KNN [W]	B(DT)	B(DT) [W]	RF	RF [W]
1	0			-1		-1
2	0			-1	0	
3		0		-1	0	
4		-1		0	-1	
5		0		-1		-1
6	0			-1		0
7		0		-1		0
8		0		0		0
9		-1	0			0
10	0		0		0	
11		0		-1		-1
12	0			-1		-1
13		-1		-1		0
14		-1		-1	-1	
15		0		-1		-1
16		0		-1		0
17	-1			-1		-1
18		0		-1	0	
19	0			-1		0
20		0		-1		0

Prohra je značena -1 (červené pole) a remíza je značena 0 (zelené pole). Pro každý datový soubor je vybrán jeden referenční model (světle šedé pole) a ostatní jsou s ním srovnávány.

Tabulka 10: Zobrazené výsledky *Wilcoxon Signed Rank testu* vztažené vůči nejlepšímu modelu pro porovnání modelů programu *RapidMiner* a *WML*

	KNN	KNN [W]	B(DT)	B(DT) [W]	RF	RF [W]
1		0		-1		-1
2		0		-1		0
3		0		-1		0
4		-1		0		1
5		0		-1		-1
6		0		-1		0
7		0		-1		0
8		0		0		0
9		-1		0		0
10		0		0		0
11		0		-1		-1
12		0		-1		-1
13		-1		-1		0
14		-1		-1		1
15		0		-1		-1
16		0		-1		0
17		1		-1		-1
18		0		-1		0
19		0		-1		0
20		0		-1		0
Scóré:		8,5		**2**		8

Nejlepší model (světle šedé pole). Výhra je značena 1 (zelené pole). Prohra je značena -1 (červené pole) a remíza je značena 0 (žluté pole). V posledním řádku je zobrazeno skoré. Pokud je skoré statisticky významně horší než u referenčního modelu je zobrazeno červeně.

7 ZÁVĚR

Hlavním úkolem této bakalářské práce bylo empiricky porovnat tradiční a meta-learningový přístup k modelování. Na 20 datových souborech bylo otestováno 5 tradičních a 7 meta-learningových modelů. Zároveň byla vytvořena aplikace v programovacím jazyce C# s 6 různými modely. Modely *Neuronová síť*, *Rozhodovací strom* a *Naivní Bayesův klasifikátor* byly rovněž porovnány na stejných datových souborech s ekvivalentními modely programu *RapidMiner*.

Na základě výsledků přesnosti získaných metodou *10-fold cross validation* byl jako model s nejvyšší průměrnou přesností určen *Stacking*. Pomocí statistického testu bylo stanoveno, že 3 ze zbylých 11 modelů dosahují statisticky významně horších výsledků. U zbytku není možné na základě testu říci, že dosažené výsledky jsou horší.

Při porovnávání modelů programu *RapidMiner* s naprogramovanými modely nebyly v případě modelů *k-nejbližších sousedů* a *Random Forest* nalezeny statistické významné rozdíly. U modelu *Bagging* byl nalezen statisticky významný rozdíly. Rozdíl přesností obou modelů *Bagging* byl pravděpodobně způsoben tím, že ve vlastnoručně naprogramované aplikaci chybí implementace prořezávání u modelu *Rozhodovacího stromu.*

Dosažené výsledky ukazují, že meta-learningové modely dosahují převážné lepších výsledků než modely tradiční. Vyjímkou je model *AdaBoost* s base modelem *Decision Stump*, který dosáhl poměrně nízké průměrné přesnosti oproti ostatním ensemble modelům. Na druhou stranu tradiční modely *SVM* a *k-nejbližších sousedů* dosáhly poměrně vysoké přesnosti. Obzvlášť výsledek modelu *k-nejbližších sousedů*, který skončil na čtvrtém místě. Jeho výsledků je velmi zajímavý s ohledem na jeho jednoduchost.

Hlavním problém ensemble modelů je časová a výpočetní náročnost. Proces učení je v zásadě mnohem náročnější, než tomu je u klasických modelů. Proto je velice zajímavé že nejlepších výsledků dosáhl právě *Stacking*, který kombinuje 4 tradiční modely. Jeho proces učení je mnohdy méně náročný než u metod *Bagging* a *Boosting*.

Je nutné zmínit i model *Random Forest*, který v porovnání průměrné přesnosti vyšel až na 6 místě, ale rozdíl průměrné přesnosti není výrazný a jeho výpočetní náročnost oproti modelům *Bagging* a *Boosting* je výrazně menší.

8 LITERATURA

[1] - *Algorithmic Learning Theory* [online]. Berlin, Heidelberg: Springer Berlin Heidelberg, 1999-5-19 [cit. 2014-05-21]. Dostupné z: http://link.springer.com/10.1007/3-540-46769-6_2

[2] - Artificial_neural_network. In:*Wikipedia: the free encyclopedia* [online]. San Francisco (CA): Wikimedia Foundation, 2001- [cit. 2014-05-20]. Dostupné z: http://en.wikipedia.org/wiki/Artificial_neural_network

[3] - BITTENCOURT, V.G., M.C.C. ABREUT, M.C.P. DE SOUTO a A.M. DE P. CANUTO. An empirical comparison of individual machine learning techniques and ensemble approaches in protein structural class prediction. *Proceedings. 2005 IEEE International Joint Conference on Neural Networks, 2005.* [online]. IEEE, 2005, s. 527-531 [cit. 2014-05-20]. DOI: 10.1109/IJCNN.2005.1555886. Dostupné z: http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=1555886

[4] - Breiman, L. and Spector, P. (1992). Submodel selection and evaluation in regression: the X-random case, International Statistical Review 60: 291–319.

[5] - BREIMAN, Leo. Bagging Predictors. *Machine Learning* [online]. 1996, vol. 24, issue 2, s. 123-140 [cit. 2014-05-25]. DOI: 10.1023/A:1018054314350. Dostupné z: http://link.springer.com/10.1023/A:1018054314350

[6] - BREIMAN, Leo. Random Forests. *Machine Learning* [online]. 2001, vol. 45, issue 1, s. 5-32 [cit. 2014-05-25]. DOI: 10.1023/A:1010933404324. Dostupné z: http://link.springer.com/10.1023/A:1010933404324

[7] - Criminisi, A. Shotton, J. a Konukoglu E. *Decision Forests for Classification, Regression, Density Estimation, Manifold Learning and Semi-Supervised Learning.* Microsoft Research. 2011

[8] - Export funkce *ClassDiagram* programu Microsoft Visual Studio 2013.

[9] - Export pracovního prostředí naprogramované aplikace WML v jazyce C#.

[10] - Export pracovního prostředí programu RapidMiner.

[11] - FERNANDEZ MARTINEZ, Roberto, Ruben LOSTADO LORZA, Julio FERNANDEZ CENICEROS a F. Javier MARTINEZ-DE-PISON ASCACIBAR. Comparative analysis of learning and meta-learning algorithms for creating models for predicting the probable alcohol level during the ripening of grape berries. *Computers and Electronics in Agriculture* [online]. 2012, vol. 80, s. 54-62 [cit. 2014-05-25]. DOI: 10.1016/j.compag.2011.10.009. Dostupné z: http://linkinghub.elsevier.com/retrieve/pii/S0168169911002377

[12] - GARCÍA-PEDRAJAS, Nicolás a Domingo ORTIZ-BOYER. Boosting k-nearest neighbor classifier by means of input space projection. *Expert Systems with Applications* [online]. 2009, vol. 36, issue 7, s. 10570-10582 [cit. 2014-05-25]. DOI: 10.1016/j.eswa.2009.02.065. Dostupné z: http://linkinghub.elsevier.com/retrieve/pii/S0957417409002140

[13] - 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. ISBN 978-0-387-84857-0.

[14] - Honzík P.: Strojové učení. Elektonická skripta VUT. (CS)

[15] - Kohavi, R. (1995). A study of cross-validation and bootstrap for accuracy estimation and model selection, International Joint Conference on Artificial Intelligence (IJCAI), Morgan Kaufmann, pp. 1137–1143.

[16] - LIU, F.T., K.M, TING, Y. YU a Z.H. ZHOU. Spectrum of Variable-Random Trees. [online]. 2008, vol. 32, s. 355-384 [cit. 2014-05-25]. DOI: 10.1613.jair.2470. Dostupné z: https://www.jair.org/papers/paper2470.html

[17] - Machine Learning and Data Mining: CPSC 340. *Http://www.cs.ubc.ca/* [online]. 2012 [cit. 2014-05-25]. Dostupné z:http://www.cs.ubc.ca/~nando/340-2012/lectures.php

[18] - MINGERS, John. An Empirical Comparison of Pruning Methods for Decision Tree Induction. *Machine Learning* [online]. 1989, vol. 4, issue 2, s. 227-243 [cit. 2014-05-25]. DOI: 10.1023/A:1022604100933. Dostupné z: http://link.springer.com/10.1023/A:1022604100933

[19] - MURPHY, Kevin P. *Machine learning: a probabilistic perspective*. Cambridge: MIT Press, c2012, xxix, 1067 s. Adaptive computation and machine learning series. ISBN 978-0-262-01802-9.

[20] - RapidMiner. In: *Wikipedia: the free encyclopedia* [online]. San Francisco (CA): Wikimedia Foundation, 2014, 11 March [cit. 2014-05-12]. Dostupné z: http://en.wikipedia.org/wiki/RapidMiner

[21] - RUSSELL, Stuart J a Peter NORVIG. *Artificial intelligence: a modern approach*. 3rd ed. Upper Saddle River: Prentice Hall, 2010, xviii, 1132 s. Prentice Hall series in artificial intelligence. ISBN 978-0-13-604259-4.

[22] - *UCI Machine Learning Repository* [online]. 1987 [cit. 2014-05-12]. Dostupné z: https://archive.ics.uci.edu/ml/datasets.html

[23] - WILCOX, R. H. Adaptive control processes—A guided tour, by Richard Bellman, Princeton University Press, Princeton, New Jersey, 1961, 255 pp., $6.50. *Naval Research Logistics Quarterly* [online]. 1961, vol. 8, issue 3, s. 315-316 [cit. 2014-05-25]. DOI: 10.1002/nav.3800080314. Dostupné z: http://doi.wiley.com/10.1002/nav.3800080314

[24] - YAN-SHI DONG a KE-SONG HAN. A comparison of several ensemble methods for text categorization. *IEEE International Conference onServices Computing, 2004. (SCC 2004). Proceedings. 2004* [online]. IEEE, 2004, s. 419-422 [cit. 2014-05-25]. DOI: 10.1109/SCC.2004.1358033.

[25] - YE, Ren a P.N. SUGANTHAN. *Information Fusion (FUSION), 2012 15th International Conference on: Empirical comparison of bagging-based ensemble classifiers*. S.l.: [s.n.], 2012. ISBN 978-146-7304-177.

[26] - ZHOU, Zhi-Hua. *Ensemble methods: foundations and algorithms*. Boca Raton, FL: Taylor, 2012. ISBN 978-1439830031.

9 SEZNAM VZORCŮ

10 DODATKY

Obsah přiloženého CD:

- Aplikace *WML*
- Aplikace *Wilcoxon_signed-rank_test*
- Výsledky Testů metody 10-fold Cross Validation
- Parametry modelů nalezené programem Rapidminer
- Bakalářská práce v pdf formátu