FAKULTA
INFORMAČNÍCH
TECHNOLOGIÍ
ČVUT V PRAZE

# ZADÁNÍ BAKALÁŘSKÉ PRÁCE

| | |
|---|---|
| **Název:** | Optimalizace metod pro předzpracování dat pro co nejlepší klasifikaci |
| **Student:** | Jan Pancíř |
| **Vedoucí:** | doc. RNDr. Ing. Marcel Jiřina, Ph.D. |
| **Studijní program:** | Informatika |
| **Studijní obor:** | Znalostní inženýrství |
| **Katedra:** | Katedra aplikované matematiky |
| **Platnost zadání:** | Do konce letního semestru 2018/19 |

## Pokyny pro vypracování

Cílem práce je najít optimální pořadí a hodnoty parametrů různých metod pro předzpracování dat tak, aby bylo dosaženo co nejlepších výsledků při následné klasifikaci.

1) Prozkoumejte stávající přístupy (metody, algoritmy), které se zabývají optimalizací pořadí metod pro předzpracování dat a jejich parametrů pro dosažení co nejlepší výsledné klasifikace.
2) Navrhněte algoritmus pro optimalizaci pořadí metod pro předzpracování dat a jejich parametrů. Pro optimalizaci využijte zejména evoluční techniky. Z metod pro předzpracování dat využijte metody pro diskretizaci, vyvažování tříd (SMOTE), standardizaci dat, redukci dimenzionality (PCA), doplnění chybějících hodnot a detekci a odstranění odlehlých hodnot.
3) Implementujte navržený algoritmus ve vhodném programovacím jazyku.
4) Navržený a implementovaný algoritmus ověřte na reálných datech.
5) Vyhodnoťte dosažené výsledky a diskutujte možná zlepšení navrženého algoritmu.

## Seznam odborné literatury

[1] Guyon, I., Gunn, S., Nikravesh, M., & Zadeh, L. A. (Eds.). (2008). Feature extraction: foundations and applications (svazek 207). Springer.
[2] Hastie, T., Tibshirani, R., Friedman, J., & Franklin, J. (2005). The elements of statistical learning: data mining, inference and prediction. The Mathematical Intelligencer, svazek 27, č. 2, 83-85.
[3] Pyle, D. (1999). Data preparation for data mining, Morgan Kaufmann, 1. vydání

Ing. Karel Klouda, Ph.D.
vedoucí katedry

doc. RNDr. Ing. Marcel Jiřina, Ph.D.
děkan

V Praze dne 31. ledna 2018

FAKULTA
INFORMAČNÍCH
TECHNOLOGIÍ
ČVUT V PRAZE

Bakalářská práce

# Optimalizace metod pro předzpracování dat pro co nejlepší klasifikaci

***Jan Pancíř***

Katedra teoretické informatiky
Vedoucí práce: doc. RNDr. Ing. Marcel Jiřina, Ph.D.

29. dubna 2018

# Poděkování

Chtěl bych poděkovat panu doc. RNDr. Ing. Marcelu Jiřinovi, Ph.D. za odborné vedení práce a cenné rady, které mi pomohly tuto práci zkompletovat.

# Prohlášení

Prohlašuji, že jsem předloženou práci vypracoval(a) samostatně a že jsem uvedl(a) veškeré použité informační zdroje v souladu s Metodickým pokynem o etické přípravě vysokoškolských závěrečných prací.

Beru na vědomí, že se na moji práci vztahují práva a povinnosti vyplývající ze zákona č. 121/2000 Sb., autorského zákona, ve znění pozdějších předpisů. V souladu s ust. § 46 odst. 6 tohoto zákona tímto uděluji nevýhradní oprávnění (licenci) k užití této mojí práce, a to včetně všech počítačových programů, jež jsou její součástí či přílohou, a veškeré jejich dokumentace (dále souhrnně jen „Dílo“), a to všem osobám, které si přejí Dílo užít. Tyto osoby jsou oprávněny Dílo užít jakýmkoli způsobem, který nesnižuje hodnotu Díla, a za jakýmkoli účelem (včetně užití k výdělečným účelům). Toto oprávnění je časově, teritoriálně i množstevně neomezené. Každá osoba, která využije výše uvedenou licenci, se však zavazuje udělit ke každému dílu, které vznikne (byť jen zčásti) na základě Díla, úpravou Díla, spojením Díla s jiným dílem, zařazením Díla do díla souborného či zpracováním Díla (včetně překladu), licenci alespoň ve výše uvedeném rozsahu a zároveň zpřístupnit zdrojový kód takového díla alespoň srovnatelným způsobem a ve srovnatelném rozsahu, jako je zpřístupněn zdrojový kód Díla.

V Praze dne 29. dubna 2018 .....

### Odkaz na tuto práci

Pancíř, Jan. *Optimalizace metod pro předzpracování dat pro co nejlepší klasifikaci*. Bakalářská práce. Praha: České vysoké učení technické v Praze, Fakulta informačních technologií, 2018.

# Abstrakt

Práce se zabývá zefektivněním procesu vytěžování znalostí z dat. Zaměřuje se na optimalizaci pořadí metod pro předzpracování dat a jejich parametrů pro daný klasifikátor. Byl navržen algoritmus, který vytvoří plán jak předzpracovat vstupní data, aby naučený klasifikátor dosahoval co nejvyšší přesnosti klasifikace. Pro optimalizaci a nalezení plánu byl využit genetický algoritmus. Byla vytvořena aplikace v jazyce Java implementující tento algoritmus, která nalezne plán předzpracování dat pro libovolný numerický dataset s použitím následujících metod předzpracování dat: diskretizace, normalizace, redukce dimenzionality, odstranění odlehlých hodnot, vyvažování tříd a doplnění chybějících hodnot. Výsledky byly testovány na několika reálných datasetech. Algoritmus zlepšuje přesnost klasifikace v průměru o 4-9 %. Jedná se o nástroj, který umožňuje plně zautomatizovat proces předzpracování dat. Případně lze využít jako pomocný nástroj pro znalostního experta při tvorbě plánu předzpracování dat.

**Klíčová slova** předzpracování dat, optimalizace metod, genetický algoritmus, vytěžování znalostí z dat

# Abstract

The thesis deals with the efficiency of the process of extracting knowledge from the data. It focuses on optimizing the order of pre-processing methods and their parameters for the specific classifier. An algorithm has been designed to create a plan to pre-process input data so that the learned classifier achieves the highest accuracy of classification. The genetic algorithm was used to optimize and find the plan. A Java application implementing this algorithm has been developed to find a pre-processing plan for any numerical dataset using the following data preprocessing methods: discretization, normalization, dimensionality reduction, remote values removal, class balancing and missing values imputation. The results were tested on several real datasets. The algorithm improves the classification accuracy by an average of 4-9 %. This is a tool that allows you to fully automate the pre-processing process. Eventually it can be used as a help tool for a knowledge expert to create a pre-processing plan.

**Keywords** data preprocessing, method optimization, genetic algorithm, data mining

# Obsah

# Seznam obrázků

# Seznam tabulek

# Úvod

Strojové učení je v současné době jednou z hlavních oblastí umělé inteligence s širokým využitím v průmyslu či lékařství. Neustále se zdokonalují algoritmy pro tvorbu klasifikátorů s čím dál větší přesností predikce. Úspěšnost těchto klasifikátorů však nezáleží pouze na nich, ale také na kvalitě dat, na kterých je učíme. Metod pro předzpracování dat je mnoho a pro konkrétní data se hodí jen některé metody. Cílem práce je navrhnout algoritmus, který pro vstupní data a k nim zvolený klasifikátor automatizovaně vytvoří vhodný plán k jejich předzpracování. Plánem rozumím vhodnou posloupnost metod s nastavením jejich parametrů takovou, aby klasifikátor naučený na předzpracovaných datech dosahoval co největší přesnosti klasifikace. Výsledek práce bude prospěšný pro úlohy využívající strojové učení s častým střídáním dat s různou strukturou. Algoritmus pro tato data efektivně a rychle vytvoří vhodný plán k jejich předzpracování.

Téma jsem si zvolil z důvodu že problém, jak automatizovaně nalézt postup předzpracování dat pro jakákoliv data, nebyl doposud uspokojivě vyřešen. Dále věřím ve velký potenciál evolučních technik, proto jsem jako automatizovaný nástroj k řešení zvolil genetický algoritmus.

V práci se zabývám analýzou, návrhem, implementací a vyhodnocením postupu pro nalezení plánu předzpracování dat. Práce dále pokračuje v následující struktuře:

Nejdříve se v teoretické části věnuji základním prvkům strojového učení a vlivu předzpracování dat na učení modelů. Podrobněji se zabývám mnoha známými metodami pro předzpracování dat a popisem genetického algoritmu. Dále se věnuji současným řešením, které se tímto problémem zabývají. Následuje návrh mého řešení pro nalezení plánu předzpracování dat pomocí genetického algoritmu. V praktické části se zabývám implementací algoritmu pro nalezení plánu předzpracování dat a experimenty, které ověřují funkčnost tohoto konceptu na několika reálných datových souborech.

Kapitola 1

# Cíl práce

Cílem teoretické části je:

- Analyzovat stávající řešení zabývající se optimalizací pořadí metod pro předzpracování dat a jejich parametrů.
- Seznámit se s metodami a algoritmy pro předzpracování dat.
- Analyzovat techniku genetického algoritmu.
- Vytvořit návrh vlastního řešení pro nalezení plánu předzpracování dat.

Cílem praktické části je:

- Implementovat algoritmus pro nalezení optimálního pořadí metod předzpracování dat a hodnot jejich parametrů. Z metod pro předzpracování dat využít metody pro diskretizaci, normalizaci, vyvažování tříd (SMOTE), redukci dimenzionality (PCA), doplnění chybějících hodnot a detekci a odstranění odlehlých hodnot.
- Provést experimenty a ověřit úspěšnost navrženého algoritmu na několika reálných datových souborech.
- Dosažené výsledky vyhodnotit a diskutovat možná zlepšení navrženého algoritmu.

Kapitola 2

# Teoretické pozadí

V této kapitole seznámím čtenáře s hlavními nástroji využitými v mé práci a sjednotím názvosloví jednotlivých pojmů. Nejprve obecně uvedu úlohy strojového učení a zdůrazním vliv předzpracování dat na úspěšnost řešení těchto úloh. Dále definuji vybrané druhy metod pro předzpracování dat a algoritmy, které tyto metody realizují a jsou využity v mém řešení. Následně popíši genetický algoritmus, který je základním výpočetním prvkem v mém navrženém řešení.

## 2.1 Strojové učení

Strojové učení je jednou z hlavních oblastí pro vývoj umělé inteligence. Zabývá se technikami, jak umožnit počítačovému systému se učit a zdokonalovat se v řešení dané úlohy. V souvislosti s vytěžováním znalostí z dat [1] strojovým učením rozumíme algoritmy, které jsou schopné v datech rozpoznat různé vzory a závislosti a následně úspěšně predikovat hodnoty či třídy dat nově příchozích. Tyto algoritmy tvoří modely, též klasifikátory, které poté mohou řešit klasifikační či regresní úlohy.

**Klasifikační úloha** Model má za cíl predikovat třídu, do které daná instance dat patří.

**Regresní úloha** Model má za cíl predikovat chybějící spojitou číselnou hodnotu nové instance dat.

Úspěšnost predikce je velice závislá na kvalitě naučení modelu. Mít vhodná data, na kterých model učíme, je jedním z nejdůležitějších předpokladů pro úspěšný model. Způsobů, jak data před učením připravit a transformovat, je mnoho a k různým datům je nutné přistupovat individálně. Nalezení vhodných metod je proto časově nejnáročnější fází při vytěžování znalostí z dat. Jermyn a další v [2] odhadují, že fáze předzpracování dat zabírá alespoň 80 % času

potřebného k dokončení znalostního projektu. Automatizace tohoto procesu by byla velice účinným zefektivněním při tvorbě znalostních systémů.

### 2.1.1 Data

Data, která k naučení potřebujeme, musí mít specifickou formu. Skládají se z atributů (sloupců) a poté instancí (řádků), které v sobě nesou konkrétní hodnoty daných atributů. Takováto data můžeme zapsat do tabulky a hovoříme o nich jako o datové matici, či datasetu. Při učení modelu volíme většinou jeden atribut jako tzv. atribut třídy. Jedná se o cílový atribut, který chceme predikovat pro nová příchozí data u nichž jeho hodnotu neznáme.

Data v této formě většinou snadno získáme z datových úložišť či jiných zdrojů a hovoříme o nich jako o surových datech, tedy ještě nezpracovaných. Pro naučení úspěšného modelu je však nutné tato data ještě transformovat (předzpracovat) do vhodné formy. Způsob, jakým to provedeme, je však silně závislý na struktuře dat samotných a modelu, který chceme pro predikci používat.

Často je nezbytné vyřešit známe problémy, které se surovými daty můžeme mít. Máme-li dat příliš málo, model neodhalý veškerou znalost a bude predikovat nepřesně. Máme-li dat příliš mnoho, je třeba je vhodně redukovat, aby bylo učení modelu časově zvládnutelné. Dále v datech může být nevyvážené zastoupení instancí od různých tříd a model se tak nedokáže naučit málo zastoupené třídy správně klasifikovat. V datech mohou být zanesené chybné hodnoty, odlehlé hodnoty způsobené chybou měření, či nějaké hodnoty mohou chybět úplně. Některé modely si s chybějícími hodnotami či nečíselnými hodnotami neumí poradit, jako například neuronové sítě [3]. Díky těmto a mnoha dalším problémům je nezbytné data před učením vhodně pro model připravit, předzpracovat [4].

### 2.1.2 Klasifikace

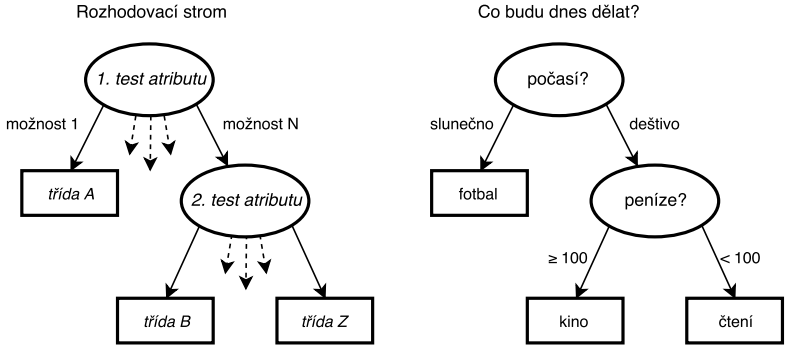
Klasifikace je proces nalézání množiny modelů (funkcí), které popisují a rozlišují datové třídy a koncepty, za účelem následné predikce tříd dat, u kterých jejich třídu neznáme [3]. Tyto modely jsou tvořeny na základě analýzy trénovací množiny dat, což jsou data se známou třídou. Modely lze reprezentovat pomocí mnoha algoritmů. Mezi nejznámější, které jsem zároveň použil pro testování svého řešení, patří rozhodovací strom, model nejbližších sousedů (K-NN) a Naivní Bayes [5].

**K-NN (Nearest Neighbour)** Algoritmus nejbližších sousedů je založen na principu nalezení $K$ nejbližších instancí (sousedů) k neklasifikované instanci a poté predikovaní cílové hodnoty na základě těchto sousedů. Každá instance tak vyjadřuje bod v n-dimenzionálním prostoru a $K$ nejbližších sousedů tak znamená $K$ nejbližších bodů v prostoru. Pro

výpočet vzdálenosti dvou bodů je možno použít mnoho metrik jako je Mahalanobisova, Cosinová či Euklidovská vzdálenost [3]. Často používanou Euklidovskou vzdálenost dvou bodů (instancí) v prostoru $X_1 = (x_{11}, x_{12}, ..., x_{1n})$ a $X_2 = (x_{21}, x_{22}, ..., x_{2n})$ spočteme

$$d(X_1, X_2) = \sqrt{\sum_{i=1}^{n}(x_{1i} - x_{2i})^2}$$

**Rozhodovací strom** Algoritmus generující stromovitou strukturu s jedním kořenem, kde každý vnitřní uzel představuje test na určitý atribut a každá vycházející větev z tohoto uzlu představuje výsledek daného testu. Instance tak od kořene prochází těmito testy až k libovolnému listu, který určuje do jaké třídy je instance klasifikována.




**Obrázek 2.1:** Obecné schema rozhodovacího stromu s konkrétním příkladem

**Naivní Bayes** Algoritmus reprezentující pravděpodobnostní klasifikátor založený na Bayesově větě, který předpokládá nezávislost jednotlivých atributů [6]. Výsledkem klasifikace je pravděpodobnost s jakou instance náleží daným třídám. Tedy pravděpodobnost, že instance $x$ náleží třídě $C_k$ se spočte

$$p(x|C_k) = \prod_{i=1}^{r} p(x_i|C_k)$$

kde $x_i$ jsou hodnoty atributů dané instance a $r$ je počet všech atributů.

## 2.2 Metody předzpracování dat

Možností jak data upravit či transformovat před učením modelu je mnoho. Existují metody sloužící především k opravě dat a jejich doplnění či rozšíření, mezi něž patří metody nahrazení chybějících hodnot, odstranění odlehlých hodnot a dále metody tvořící nové atributy, například výpočet věku podle

data narození apod. [7]. Dále existují metody, které datovými transformacemi přizpůsobují reprezentaci dat pro daný klasifikátor. Mezi ně patří metody normalizace, diskretizace, metody využívající lineární transformaci a další. Výběr metod a pořadí, v jakém je na data použijeme, může mít na učení různých klasifikátorů zcela odlišný dopad.

V následující sekci popíši základní druhy metod pro předzpracování dat a k nim algoritmy, pomocí nichž se dají na datech realizovat. Tyto vybrané algoritmy zároveň používám při testování svého konceptu.

### 2.2.1 Obohacení dat

Obecným cílem těchto metod je vygenerování nových umělých instancí, které podpoří učení modelu, avšak nezanesou do originálních dat novou informaci. Tyto metody jsou vhodné pro datasety s celkovým velmi malým počtem instancí či malým počtem instancí od určité třídní skupiny. Tento nedostatek řeší například algoritmus SMOTE (Synthetic Minority Over-sampling Technique) [8].

### 2.2.2 Odstranění odlehlých hodnot

Metody odstraňující instance, které obsahují hodnoty výrazně vzdálené od očekávaného rozsahu. Tedy hodnoty vzniklé chybou měření či jiným nechtěným zásahem. Statistickou metodou, která detekuje odlehlé hodnoty, je například mezikvartilové rozpětí IQR (Interquartile range) [9].

### 2.2.3 Nahrazení chybějících hodnot

V reálných datech jsou častým problémem chybějící hodnoty dat [10]. Tedy pro určité instance jsou hodnoty některých atributů nedostupné. Před použitím takovýchto neúplných dat ke strojovému učení je často nutné tato data upravit, protože některé modelovací nástroje (například neuronové sítě [3]) s chybějícími hodnotami neumí zacházet. Existuje několik technik, jak chybějící hodnoty v datech odstranit:

- ignorování instancí s neznámou hodnotou
- nahrazení hodnoty nejčastější hodnotou
- nahrazení hodnoty nejčastější hodnotou dané třídy
- nahrazení hodnoty průměrnou hodnotou
- nahrazení hodnoty pomocí K-NN

Je důležité dodat, že chybějící hodnota nemusí vždy znamenat chybu v datech. Chybějící hodnota může vyjadřovat to, že hodnota pro tuto instanci

nedává smysl či znamená tzv. možnost „nevím“. V takovýchto případech doplnění těchto hodnot může do dat vnést novou informaci a význam dat zkreslit [3]. Je tedy doporučeno data nejprve ručně analyzovat znalostním expertem a automatizovaný nástroj využít na data již bez chybějících hodnot.

### 2.2.4 Normalizace

Jedná se o techniky, které škálují numerické hodnoty dat do zvoleného rozsahu. Nejčastěji se data transformují tak, aby spadala do intervalu $[-1, 1]$, případně $[0, 1]$.

Normalizace dat si klade za cíl dát všem atributům v datech stejnou váhu. Je velice užitečná pro algoritmy jako jsou neuronové sítě, nejbližší sousedé (K-NN) nebo shluková analýza [3]. Normalizace zrychluje dobu učení a zabraňuje zvýhodnění atributů s vysokými hodnotami oproti atributům s nízkými hodnotami. Existují dvě nejčastější metody normalizace:

1. **min-max normalizace**

   Jedná se o lineární transformaci originálních dat. Předpokládejme, že $min_A$ a $max_A$ je minimum a maximum atributu $A$. Min-max normalizace přeškáluje hodnotu $v_i$ atributu $A$ na novou hodnotu $v'_i$ spadající do intervalu $[newmin_A, newmax_A]$, kde $newmin_A$ a $newmax_A$ je libovolně zvolené nové minimum a maximum. Pro výpočet nové hodnoty tedy platí

   $$v'_i = \frac{v_i - min_A}{max_A - min_A}(newmin_A - newmin_A) + newmin_A$$

2. **Z-score normalizace (standardizace)**

   Hodnoty atributu A jsou škálovány v závislosti na průměru a standardní odchylce. Nová hodnota $v'_i$ atributu A je tedy spočtena

   $$v'_i = \frac{v_i - \bar{A}}{\sigma_A}$$

   kde $\bar{A}$ je průměr hodnot atributu $A$ a $\sigma_A$ je standardní odchylka, tedy druhá odmocnina rozptylu hodnot atributu A. Tato metoda normalizace je vhodná pokud dopředu neznáme minimální a maximální hodnoty dat nebo se v datech vyskytují odlehlé hodnoty, které deformují výsledky min-max normalizace.

### 2.2.5 Diskretizace

Metoda transformující kvantitativní data na kvalitativní rozdělením numerických atributů na konečný počet nepřekrývajících se intervalů. Hodnoty se poté mapují do daných intervalů a tudíž se stávají diskrétními [11].

Reálná data jsou často spojité numerické hodnoty, mnoho algoritmů však vyžaduje data nominální, a proto je použití diskretizace pro určité modely nezbytné.

Výhodou diskretizace je zjednodušení dat, což zajišťuje rychlejší a přesnější učení modelů. Dále čitelnost, jelikož diskrétní atributy je mnohdy jednodušší pochopit a vysvětlit [12]. Naproti tomu diskretizace vždy způsobí ztrátu informace v datech a je tedy potřeba vhodně nastavit volitelné parametry této metody, aby byla ztráta přijatelná.

### 2.2.6 Redukce dimenzionality

U datasetů s velkým počtem atributů či obrovským množstvím instancí narážíme při učení modelu na tzv. problém dimezionality [13]. Tedy, že algoritmy nejsou schopné naučit model v přijatelném výpočetním čase. Pro tento problém existuje řada technik, které buďto odebírají či transformují atributy datasetu tak, aby došlo k co nejmenší ztrátě informace v datech.

První technikou je výběr rysů (Feature selection) [14]. Jedná se o proces identifikování a odebrání co nejvíce irelevantní a nadbytečné informace. Výsledkem je podmnožina atributů, která stále dobře popisuje původní problém k řešení klasifikačních či regresních úloh. Algoritmem pro výběr rysů z mnohodimenzionálních datasetů je například FCBF (Fast Correlation-Based Filter) [15].

Druhou technikou je prostorová transformace (Space transformation). Oproti výběru významných atributů tato metoda generuje zcela novou množinu atributů kombinující atributy původní. Algoritmus založený na lineárních transformacích je například analýza hlavních komponent (PCA) [16].

### 2.2.7 Redukce instancí

Cílem této techniky je co nejvíce zredukovat počet instancí v datasetu při co nejmenší ztrátě informace v datech. Tato metoda umožňuje zrychlit učení a vyhodnocování modelu. Nejjednodušším algoritmem pro realizaci je náhodný výběr instancí, ten však může generovat deformované datasety se ztrátou či zkreslením informací. Tento nedostatek řeší technika používající Kolmogorovův-Smirnovův test [17].

#### Kolmogorovův-Smirnovův test

Jedná se o neparametrický statistický test pro ověření, zda dvě jednorozměrné náhodné veličiny pocházejí ze stejného pravděpodobnostního rozdělení. V souvislosti s daty a strojovým učením dokáže tento test ověřit, zda dvě datové sady mají stejné pravděpodobnostní rozdělení všech hodnot. Jinými slovy pomocí něj dokáži zjistit reprezentativnost dané podmnožiny dat vůči původním surovým datům.

Mám dva numerické datasety, $A$ o velikosti $m$ a $B$ o velikosti $n$ a jejich empirické distribuční funkce $F_{A,m}$ a $F_{B,n}$. Hodnota Kolmogorovovy-Smirnovovy statistiky $D_{m,n}$ se spočte

$$D_{m,n} = max|F_{A,m}(x) - F_{B,n}(x)|$$

a vyjadřuje maximální vzdálenosti mezi empirickými distribučními funkcemi dvou datových vzorků. Uvažuji nulovou hypotézu $H_0$, že obě výběrové množiny mají stejné pravděpodobnostní rozdělení. Tuto hypotézu zamítnu, pokud hodnota $D_{m,n}$ přesáhne hodnotu kritické meze $D_\gamma$. Hodnotu kritické meze lze spočítat jako

$$D_\gamma = c(\gamma)\sqrt{\frac{m+n}{mn}}$$

kde $c(\gamma)$ je inverze Kolmogorovovy distribuce, kterou lze vyčíst například z následující tabulky:

**Tabulka 2.1:** Hodnota $c(\gamma)$ pro nejčastější hodnoty hladiny významnosti $\gamma$

| $\gamma$ | 0.1 | 0.05 | 0.025 | 0.01 |
|---|---|---|---|---|
| $c(\gamma)$ | 1.22 | 1.36 | 1.48 | 1.63 |

Díky tomuto testu mohu zajistit výběr podmnožiny dat z původního datasetu, která je se zvolenou pravděpodobností reprezentativním vzorkem původních dat.

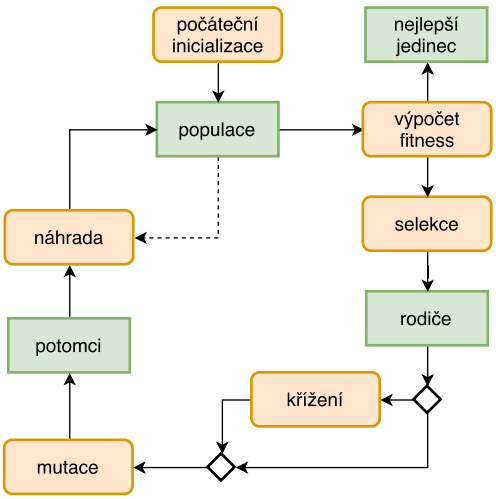
Alternativou ke K-S testu je Chí-kvadrát test. Z výsledků porovnání obou testů v [18] je zřejmé, že K-S test je vhodnější pro případ malého počtu dat a dosahuje menší míry chyb prvního druhu.

## 2.3 Genetický algoritmus

Jedná se o algoritmus ze skupiny evolučních algoritmů, který se pomocí principů evoluční biologie snaží nalézt řešení k úlohám, pro které je těžké vytvořit exaktní postup. Pomocí evolučních procesů jako je dědičnost, křížení, mutace a přirozený výběr tento algoritmus zdokonaluje skupinu řešení, dokud nejsme s nejlepším nalezeným řešením spokojeni.

Genetický algoritmus [19] (dále jen GA) se skládá z hlavního evolučního cyklu, do kterého zprvu vstupuje inicializovaná populace. Populací rozumíme skupinu jedinců (potencionálních řešeních) pro danou úlohu. Tito jedinci jsou ohodnoceni tzv. fitness funkcí a je jim přiřazena číselná hodnota fitness. Ta určuje, jak daný jedinec uspokojuje požadavky pro vyřešení úlohy. Zvolení správné funkce fitness je důležitý prvek pro úspěch celého algoritmu. Po výpočtu fitness si algoritmus zapamatuje nejlepší nalezené řešení (jedinec s nejvyšší

fitness) a poté provede selekci (výběr) nejlepších jedinců pro další generaci. Na vybrané jedince je použit operátor křížení a mutace (vysvětleno dále) a je vytvořena nová populace. Tato populace nahradí původní populaci jedinců a celý evoluční cyklus se opakuje. Každá iterace tohoto cyklu se nazývá generace. Ukončovací podmínka algoritmu je nejčastěji číslo generace, do které má evoluční cyklus běžet. Výsledkem je zapamatované nejlepší řešení.




**Obrázek 2.2:** Schema evolučního cyklu genetického algoritmu

### 2.3.1 Jedinec

Jedinec je v genetickém algoritmu tvořen jedním binárním vektorem nazývaným genotyp. Pomocí tohoto vektoru je třeba zakódovat výsledné řešení. Výpočet fitness zajišťuje funkce, která na vstupu obdrží tento genotyp, dekóduje ho a dané řešení ohodnotí číselnou hodnotou fitness.

### 2.3.2 operátory

V evolučním cyklu jsou na jedince či celou populaci použity různé operátory vzniklé inspirací z evoluční biologie. Jejich základní myšlenky a principy jsou následující:

- **Selekce** Operátor přirozeného výběru. Má za cíl vybrat z ohodnocené populace podmnožinu nejlepších jedinců při zachování dostatečné diverzity (rozmanitosti). Rozlišujeme dva základní druhy selekce:

  **ruletová selekce** RWS (Roulettewheel selection) používá princip přímé úměrnosti pravděpodobnosti výběru jedince na jeho hodnotě fitness. Pravděpodobnost výběru $i$-tého jedince s ohodnocením $f_i \geq 0$

z populace o velikosti $N$ je:

$$p_i = \frac{f_i}{\sum_{j=1}^{\infty} f_j} \quad kde\ i \in \{1, ..., N\}$$

Ruletová selekce nezaručuje dopředu konkrétní počet vybraných jedinců. Tomu se snaží zabránit varianta nazývaná Stochastické univerzální vzorkování SUS (Stochastic universal sampling) [20]. Díky proporcionálnímu výběru však operátory často nedosahují dostatečné diverzity a mají tendenci příliš upřednostňovat jedince s vysokou fitness.

**turnajová selekce** TS (Tournament selection) je založena na principu výběru nejsilnějšího jedince ze skupiny. Z populace je náhodně vybráno $T$ jedinců a ten, který má největší fitness, je vybrán do nové populace. Toto opakujeme dokud nemáme vybraný požadovaný počet jedinců. Metoda je silně závislá na velikosti parametru $T$ – nazývaném velikost turnaje – a proto často vyžaduje experimentální přístup ke konkrétnímu typu úlohy.

- **Křížení** Operátor, do kterého vstupují dva jedinci (rodiče) a výsledkem jsou dva noví jedinci (potomci), jejichž genotyp je složený z genotypů rodičů. Jedná se o důležitou schopnost GA pro přenos významné informace v populaci. Existuje několik technik, jak křížení docílit:

  **n-bodové křížení** Mějme rodiče $P_a, P_b$ a potomky $C_a, C_b$. Genotypy rodičů jsou shodně rozděleny $n$ řezy na $n+1$ částí. Pokud tyto části očíslujeme, pak platí že: Potomek $C_a$ má genotyp složený z lichých částí rodiče $P_a$ a sudých částí rodiče $P_b$. Druhý potomek $C_b$ přesně naopak.

  **jednotné křížení (uniformní)** Při tvoření genotypu prvního potomka je o každém jeho bitu náhodně rozhodnuto s pravděpodobností 50 %, zda bude pocházet od prvního, či druhého rodiče. Bity genotypu druhého potomka jsou poté vždy opačného původu než bity potomka prvního.

- **Mutace** Operátor zajišťující náhodné prohledávání prostoru blízkých řešeních. Jedná se o náhodné změny v genotypu jedince, které mají za cíl nalézt potenciální lepší řešení. O každém bitu v genotypu je náhodně s malou pravděpodobností rozhodnuto, zda se změní či nikoliv. Pravděpodobnost změny je nejčastěji od 0,001 do 0,05 [21].

### 2.3.3 Pokročilé techniky

Existuje řada technik, které zdokonalují genetický algoritmus pro zrychlení výpočtu či dosažení lepších výsledků. Zde jsou typické techniky jejichž podrobné použití v mém řešení je popsáno v sekci 4.5.

**Simulované žíhání** Do výpočtu je přidána tzv. teplota, která ovlivňuje šíři prohledávání prostoru řešení, typicky úpravou pravděpodobnosti operátoru mutace. Teplota poté zajišťuje možnost uniknutí z lokálního optima.

**Katastrofa** Technika spočívající v nahrazení několika jedinců v populaci zcela nově inicializovanými jedinci. Zabraňuje tak situaci, kdy populace degeneruje ve skupinu mnoha stejných jedinců.

**Ostrovní model** Evoluce probíhá nezávisle na několika populacích současně. Může docházet k přesunu vybraných jedinců mezi těmito populacemi, nazývanými též ostrovy. Technika se touto distribucí snaží zabránit uváznutí v lokálních optimech.

Kapitola 3

# Předešlá řešení

Obecné řešení, které by uspokojivě řešilo kompletní automatizaci vytěžování znalostí z dat, ještě vyvinuto nebylo. Existuje však řada technologií či vědeckých publikací, které se částí či obecnějším pojetím tohoto problému zabývají.

Komerční program IMB SPSS Modeller [22] pro vytěžování znalostí z dat obsahuje techniku automatického předzpracování dat. Jedná se však o pouhou transformaci dat pomocí zabudovaných pravidel s malou množinou metod předzpracování dat. Omezená je i možnost optimalizace tohoto předzpracování pro konkrétní klasifikátor. V experimentální části se věnuji porovnáním mých výsledků s výsledky tohoto nástroje.

Existuje několik technologií, které se zabývají automatizací celého procesu vytěžování znalostí z dat pro určitou datovou doménu. Například v technologii [23] je použito předdefinovaných schemat, pomocí nichž lze kompletní proces vytěžování znalostí z dat zautomatizovat pro určité známé případy. Tento přístup však lze aplikovat jen na omezenou skupinu problémů a vyžaduje náročnou přípravu daných schemat.

Dále existuje několik článků, které dokazují, že správné pořadí metod pro předzpracování dat je důležité pro přesnost klasifikace. Postup snažící se manuálně nalézt správnou sekvenci předzpracovacích metod pro mozkové počítačové rozhraní je představen v [24] a dokazuje, že pořadí metod hraje významnou roli v úspěšnosti klasifikace.

V minulé době se především kladl důraz na optimalizaci jednotlivých metod pro předzpracování dat. Mezi nedávné techniky, které se optimalizací zabývaly, patří například zdokonalení doplnění chybějících hodnot pomocí regresní imputace [25], dále optimalizace metody pro redukci dat a dimenzionality pomocí mravenčí kolonie, která vybere vhodnou podmnožinu atributů pro optimální klasifikaci [26].

Výběr optimálního pořadí metod předzpracování dat lze převést na kombinatorický problém [27], pro který existuje řada matematických metod k jeho vyřešení. Ve svém řešení určuji pořadí pomocí tzv. techniky vah jednotlivých

metod. Více v sekci 4.5.1 kódování jedince.

V mém řešení se zabývám obecným konceptem, který není limitován doménou dat ani specifickou skupinou metod pro předzpracování dat. Jedná se o nástroj, který může využít i výše zmíněné optimalizované metody. V implementační části detailněji popisuji způsob vložení libovolných metod do mého konceptu.

## 3.1 Technologie IPT

Podobným obecným řešením pro optimalizace procesu předzpracování dat se zabýval Miroslav Čepek. Vytvořil návrh konceptu pojmenovaný IPT (Inductive Preprocessing Technology) [7] a zabývá se v něm hledáním optimálního pořadí metod a nastavení těchto metod pro co nejlepší dosaženou klasifikaci.

Hlavní myšlenka technologie IPT je v jejím indukčním pojetí. Zpočátku model nemá žádné informace o datech. Během učení algoritmus zkoumá význam dat a postupně zdokonaluje strukturu modelu. Algoritmus tedy začíná s náhodnou sekvencí metod pro předzpracování dat a postupně pomocí přidávání, odebírání a modifikování těchto metod skládá optimální pořadí pro předzpracování. Algoritmus hledá nejjednodušší sekvenci metod, která dosáhne nejlepší přesnosti použitého klasifikátoru. IPT se skládá ze tří částí. Hledání sekvence metod, optimalizace parametrů a klasifikátoru. V následujících sekcích popíši základní prvky algoritmu IPT s dodanými odlišnostmi oproti mému řešení.

### 3.1.1 Sekvence metod pro předzpracování dat

Tato část algoritmu konstruuje sekvenci metod předzpracování dat, které se v daném pořadí použijí na vstupní dataset. Rozlišuje dva typy metod. Lokální metody jsou takové, které lze aplikovat pouze na jeden atribut v datasetu. Globalní metody lze naopak aplikovat pouze na celý dataset. Výsledkem výpočtu je poté jedna sekvence globálních metod a sekvence lokálních metod pro každý atribut v datasetu. Při použití těchto sekvencí na datasety jsou vždy nejprve aplikovány všechny lokální metody a poté globální metody.

Úspěšnost dané sekvence je vyjádřena hodnotou fitness. Ta je spočtena jako přesnost klasifikace modelu na testovacích datech a tento způsob používám i v mém řešení pro genetický algoritmus, viz. sekce 4.5. Algoritmus IPT ještě přidává modifikaci fitness, která hodnotu penalizuje za počet použitých metod v sekvenci. Jelikož je mým cílem nalézt plán pro co nejlepší klasifikaci a není kladen důraz na složitost tohoto plánu, tuto modifikaci nepoužívám. Předpokládám, že bude použit vždy relativně malý konstantní počet metod a není nutné vyžadovat co nejjednodušší výsledný plán na úkor jeho přesnosti.

Rozdělení dat na trénovací a testovací množinu se provádí náhodně pro každý výpočet fitness. Bylo ověřeno, že použití jednoho dělení na celý IPT algoritmus způsobí přeučení dané sekvence. Dělení těchto dat je však časově

velmi náročné, zejména pro velké datasety. Ve svém řešení jsem zvolil kompromis a využívám konstantní počet trénovacích a testovacích množin, více v sekci 4.5.4 ostrovní model.

Experimenty a výsledky techniky IPT ukázaly, že v průměru na různých typech datasetů nejlepší řešení hledá genetický algoritmus. Testování proběhlo s řadou různých technik jako je algoritmus simulovaného žíhání, náhodné prohledávání a další. Díky tomuto závěru jsem použil genetický algoritmus ve svém řešení a zabýval jsem se zejména jeho rozšířením o pokročilejší techniky využitelnými v rámci této úlohy.

### 3.1.2 Optimalizace parametrů

Nalezení optimálních parametrů metod spočívá v prohledávání prostoru reálných čísel, přirozených čísel a nominálních hodnot. Experimenty ověřily, že nastavení parametrů metod a pořadí těchto metod nejsou zcela nezávislé fáze a nelze je tedy plně separovat. Algoritmus IPT řeší tyto problémy dvěma různými optimalizačními technikami tak, že první technikou nalezne sekvenci metod, druhou technikou optimalizuje parametry těchto metod a poté vypočte fitness této celé sekvence. Experimenty a výsledky techniky IPT ukázaly, že v průměru na různých typech datasetů nejlepší řešení hledá algoritmus diferenciální evoluce [28]. Testování proběhlo s řadou různých technik jako je simulované žíhání a náhodné prohledávání. V mém řešení nepoužívám dvě odlišné optimalizační metody, ale na oba problémy využívám genetického algoritmu se dvěma binárními řetězci. Více v sekci 4.5.1 kódování jedince.

Kapitola 4

# Koncept vlastního řešení

V následující kapitole představím můj návrh algoritmu pro optimalizaci předzpracování dat DPO (Data Preprocessing Optimizer), který pro libovolný numerický dataset a zvolený klasifikátor nalezne optimální plán pro předzpracování dat. Koncept se zaměřuje na optimalizaci pouze pro klasifikační úlohy. Nejprve definuji obecnou strukturu algoritmu, jeho vstupy, výstupy a základní myšlenku výpočtu. Poté se detailněji zaměřím na jednotlivé části algoritmu. Popíši fázi přípravy dat a dělení dat. Následně detailně rozeberu části mého genetického algoritmu.

## 4.1 Vstupy

Vstupem do algoritmu je **dataset**, který chci předzpracovat, **klasifikátor**, pro který chci předzpracování dat optimalizovat, a **seznam metod** pro předzpracování dat, které jsou dostupné a mohu je pro transformaci dat použít.

**Dataset**

Vstupní datovou sadu nebo-li dataset definuji jako dvojici $(I, \vec{a})$, kde $I$ je množina instancí o velikosti $|I|$ a $\vec{a}$ je vektor atributů o délce $|\vec{a}|$. Každá instance $x$ je definována jako vektor hodnot všech atributů, tedy $x = (x_1, x_2, ..., x_{|\vec{a}|})$. Poslední atribut ($atribut_{|\vec{a}|}$) nazvu třída. Toto je cílový atribut, který chci predikovat.

**Tabulka 4.1:** Obecný formát vstupního datasetu zapsaného datovou maticí

| | $atribut_1$ | $atribut_2$ | ... | $atribut_{\|\vec{a}\|-1}$ | třída |
|---|---|---|---|---|---|
| $instance_1$ | $x_{1,1}$ | $x_{1,2}$ | ... | $x_{1,\|\vec{a}\|-1}$ | $x_{1,\|\vec{a}\|}$ |
| $instance_2$ | $x_{2,1}$ | $x_{2,2}$ | ... | $x_{2,\|\vec{a}\|-1}$ | $x_{2,\|\vec{a}\|}$ |
| ... | ... | ... | ... | ... | ... |
| $instance_{\|I\|}$ | $x_{\|I\|,1}$ | $x_{\|I\|,2}$ | ... | $x_{\|I\|,\|\vec{a}\|-1}$ | $x_{\|I\|,\|\vec{a}\|}$ |

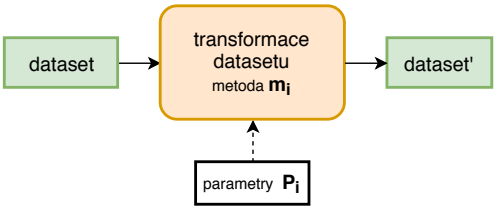
Pro navržený algoritmus je nutné, aby všechny hodnoty $x$ byly číselné, tedy musí se jednat o numerický dataset. Existuje však řada technik a metod předzpracování dat, které převádí nominální hodnoty na numerické. Koncept tak lze rozšířit na libovolný dataset, který lze transformovat do výše popsáného formátu.

**Klasifikátor**

Vstupním klasifikátorem je algoritmus strojového učení, pro který chci předzpracování dat optimalizovat. Jedná se o model, který je schopen se naučit na daných datech a poté klasifikovat data nově příchozí. Klasifikací myslím predikci hodnoty atributu třída pro libovolnou vstupní instanci.

**Seznam metod**

Seznamem metod rozumím množinu $M$ algoritmů $m_1, m_2, ..., m_{|M|}$, které nějakým způsobem transformují data. Chování těchto algoritmů je možné blíže specifikovat nastavením několika parametrů. Jedná se tedy o libovolné algoritmy vyhovující následujícímu schematu:




**Obrázek 4.1:** Obecné schema metody pro předzpracování dat

Jako příklad uvažujme metodu diskretizace popsanou v kapitole teoretické pozadí. Tato metoda provádí transformaci takovou, že atributy s numerickými hodnotami převádí na nominální hodnoty. Parametry této metody může být například mód, který určuje, zda se má provést diskretizace na základě frekvence či počtu binů, dále číslo určující přesný počet binů apod.

Metody nemusí transformovat celý dataset, ale i libovolnou podmnožinu atributů. Proto pro každou metodu, která umožňuje transformaci pouze části datasetu, uvažuji speciální parametr, který definuje, jakou podmnožinu metoda předzpracuje.

## 4.2 Výstupy

**Plán**

Hlavním výstupem algoritmu je plán pro předzpracování dat. Plánem rozumím posloupnost metod pro předzpracování dat a popis jejich parametrů. Formálně plán

$$Plan = ((m_1, P_1), (m_2, P_2), ..., (m_k, P_k))$$

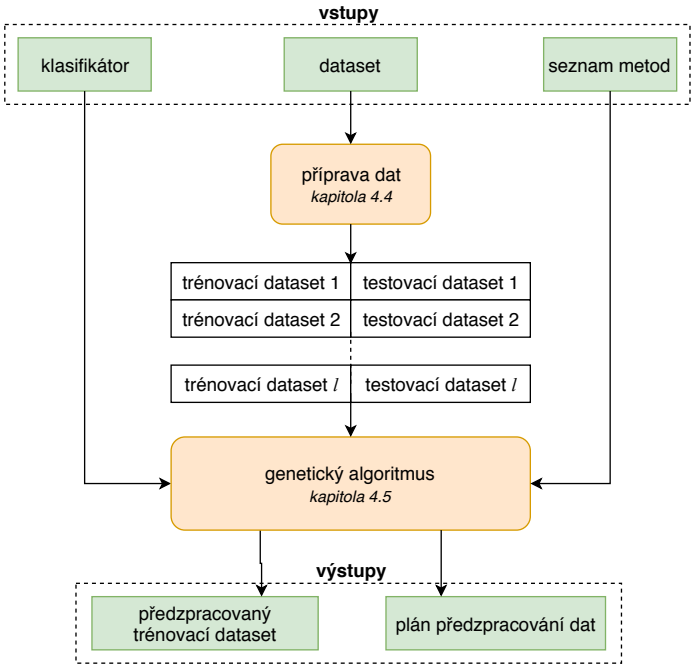
definuji jako uspořádanou k-tici dvojic $(m_i, P_i)$, kde $m_i$ je libovolná metoda pro předzpracování dat a $P_i$ je množina parametrů, určující chování metody $m_i$ a platí, že $i \leq k$, kde $k$ je počet metod použitých v daném plánu.

**Trénovací dataset**

Vedlejším produktem algoritmu je předzpracovaná datová sada vhodná k naučení vstupního klasifikátoru tak, aby dosahoval co největší úspěšnosti klasifikace.

## 4.3 Výpočet

Algoritmus se skládá ze dvou hlavních částí. První část je příprava dat, která se zabývá rozdělením vstupního datasetu na trénovací a testovací množiny potřebné pro učení a ověřování úspěšnosti modelu. Druhou částí je genetický algoritmus, což je hlavní výpočetní nástroj pro provedení optimalizace. Základní myšlenka je tedy taková, že probíhá evoluční cyklus nad množinou možných řešení, ve kterém se různými technikami snažím tato řešení vylepšit. Jednotlivá řešení jsou na konci cyklu vždy ohodnocena fitness funkcí. Hodnota fitness odpovídá procentuální úspěšnosti klasifikace testovacích dat pomocí modelu naučeném na datech trénovacích. Cyklus běží dokud není splněna ukončovací podmínka. Výsledkem je vrácení nejlepšího nalezeného řešení.




**Obrázek 4.2:** Obecné schema algoritmu DPO

## 4.4 Příprava dat

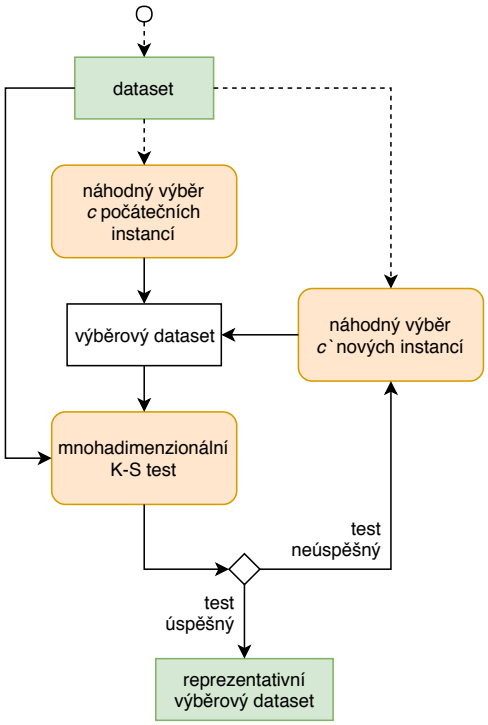
Před zahájením výpočtu je třeba surová data pro výpočet připravit. První technika je redukce dat, která je nezbytná pro velké datasety, aby je bylo možné strojově zpracovat v rozumném čase. Druhou technikou je rozdělení dat na trénovací a testovací množiny, které budou sloužit k naučení a ověření zvoleného klasifikátoru.

### 4.4.1 Reprezentativní redukce

Cílem redukce dat je co největší zmenšení datové množiny při zachování co největší reprezentativnosti k původním datům. Jinými slovy cílem je, aby prediktivní model naučený na této zmenšené množině dat byl stejně úspěšný jako hypotetický model naučený na původních surových datech.

Hlavním důvodem, proč data redukovat, je zejména časová náročnost výpočtu. Předpokládám, že surová data mohou mít obrovskou velikost (tisíce instancí) a každá operace pro přečtení těchto dat je pro evoluční techniky příliš časově náročná.

Pro ověření toho, zda je vybraná množina dat dostatečně reprezentativním vzorkem, používám statistický test Kolmogorovův-Smirnovův (K-S). Algoritmus pro nalezení této množiny je zobrazen v následujícím schematu.




**Obrázek 4.3:** Schema algoritmu pro vytvoření reprezentativního vzorku dat

Na vstupu je původní surový dataset o počtu instancí $|I|$. Z tohoto datasetu náhodně vyberu $c$ instancí a vytvořím tak výběrový dataset, pro který platí, že $c \leq |I|$. Následně spustím multidimenzionální Kolmogorovův-Smirnovův test mezi původním datasetem a nově vytvořeným výběrovým datasetem. Pokud je test úspěšný, výběrový dataset je reprezentativním vzorkem původních dat a je tedy možné ho použít pro učení a testování modelu. Pokud je test neúspěšný, vyberu náhodně $c'$ dalších (ještě nevybraných) instancí z původního datasetu a přidám je k výběrovému datasetu. Následně znovu ověřím reprezentativnost K-S testem. Tento cyklus opakuji, dokud není K-S test úspěšný. Cyklus se vždy zastaví, protože po konečném počtu iterací se původní dataset bude rovnat výběrovému datasetu a tedy K-S test musí z definice uspět, neboť dataset je sám k sobě reprezentativním vzorkem.

Multidimenzionální Kolmogorovův-Smirnovův test provádím formou jednoduchých K-S testů použitých na jednotlivé atributy obou datasetů. Pro každý atribut tedy vytvořím vektor hodnot z prvního datasetu a vektor hodnot z druhého datasetu. Na těchto dvou vektorech poté ověřuji, zda mají stejné pravděpodobnostní rozdělení viz. sekce 2.2.7 K-S test. Pro úspěch multidimenzionálního K-S testu je třeba, aby úspěšně skončily jednoduché K-S testy pro všechny atributy.

### 4.4.2 Rozdělení na trénovací a testovací množinu

Důležitou součástí přípravy dat je jejich rozdělení na trénovací a testovací množinu. Aby bylo učení modelu pro klasifikaci korektní a změřená úspěšnost klasifikace nebyla zkreslená, je nutné model učit a následně testovat na dvou různých množinách dat. Zároveň je nutné, aby obě množiny byly reprezentativním vzorkem původních dat.

K rozdělení dat používám techniku redukce z předešlé části. Pomocí této techniky vyberu reprezentativní vzorek původních dat a tuto množinu nazvu trénovacím datasetem. Poté ze zbylých původních dat vyberu tou samou technikou reprezentativní vzorek a nazvu ho testovací dataset. Zde může nastat problém nedostatku dat, tedy že pro testovací dataset nezbudou data, aby byl splněn multidimenzionální K-S test. V tom případně data doplňuji z trénovací množiny dokud není K-S test splněn. Pro tento případ sice mírně zkresluji měření výsledné úspěšnosti modelu, ale toto doplnění mi umožňuje použít algoritmus i na menších datasetech. Pro zpřesnění výpočtu úspěšnosti modelu používám více trénovacích a testovacích množin, více v sekci 4.5.4 ostrovní model.

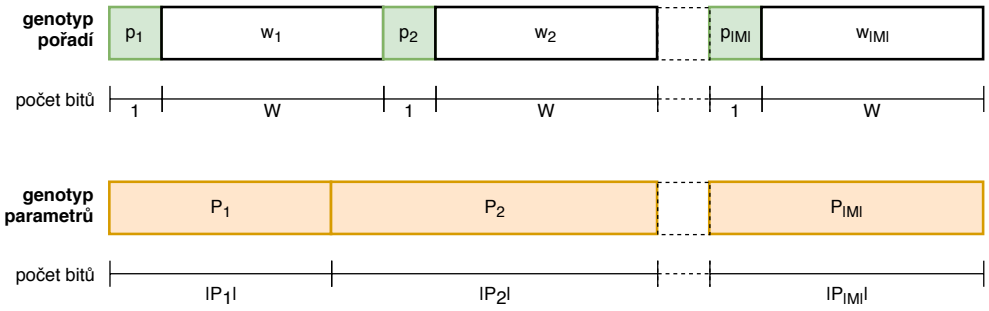
## 4.5 Genetický algoritmus

Hlavní technikou pro samotnou optimalizaci je v mém řešení genetický algoritmus. Pomocí této evoluční techniky mohu vyřešit dva základní problémy pro nalezení plánu předzpracování dat. Zaprvé to je nalezení sekvence metod,

které se na data použijí v daném pořadí, a zadruhé nalezení hodnot parametrů pro každou metodu. V následujících částech podrobně popíši koncept mého genetického algoritmu, kódování řešení, operátory a další pokročilé techniky použité pro zefektivnění celého výpočtu.

### 4.5.1 Jedinec

Jedincem v mém genetickém algoritmu popisuji právě jedno řešení dané úlohy, čili tento jedinec popisuje právě jeden plán předzpracování dat. Tento plán je v jedinci zakódován pomocí dvou binárních vektorů (genotypů). Tímto rozdělením na dva genotypy, oproti jednomu jako má klasický genetický algoritmus, tématicky rozlišuji dva výše zmíněné problémy, tedy hledání sekvence metod a hledání parametrů metod. Díky tomu mohu navrhnout různé operátory pro oba vektory. Kódování mých binárních vektorů popisuje následující schema:




**Obrázek 4.4:** Kódování genotypů jedince pro GA

Genotyp pořadí obsahuje bity $p_1, p_2, ...p_{|M|}$, kde $|M|$ je počet uvažovaných metod pro předzpracování dat, tedy velikost seznamu metod na vstupu algoritmu DPO. Tyto bity nazývám bity přítomnosti dané metody. Pokud je bit $p_i$ nastaven na 1, pak je metoda $m_i$ použita ve výsledném plánu. Naopak pokud je bit $p_i$ nastaven na 0, metoda $m_i$ není použita ve výsledném plánu.

Za každým bitem přítomnosti metody $p_i$ vždy následuje $W$ bitů určujících váhu metody $w_i$. Hodnota $w_i$ je přirozené číslo udávající prioritu dané metody a tedy definuje pořadí všech metod. Ve výsledném dekódovaném plánu poté platí, že pro každou dvojici metod $m_a$ a $m_b$ s váhami $w_a$ a $w_b$, kde $w_a > w_b$, se metoda $m_a$ provede před metodou $m_b$. Pokud nastane rovnost vah, je pořadí metod zvoleno libovolně. Tento případ lze eliminovat zvolením dostatečně velkého počtu bitů $W$. V mém řešení používám $W$ o velikosti $log_2|M|$ bitů.

Genotyp parametrů se skládá z $|P_1|+|P_2|+...+|P_{|M|}|$ bitů, kde $|P_i|$ je počet bitů, které si vyhrazuje metoda $m_i$ pro zakódování své množiny parametrů $P_i$. Metody, které nevyžadují žádné nastavitelné parametry, mají množinu $P$ rovnou prázdné množině a tedy $|P| = 0$.

Využívám binární genetický algoritmus, tedy řešení je kódováno binárním řetězcem a ne řetězcem reálných čísel. Zvolil jsem tento způsob zejména proto, že parametry metod často určují pouze určité módy chování dané metody a k jejich kódování tedy stačí malý počet bitů.

### 4.5.2 Oprava jedince

Přestože se jedná o univerzální koncept pro libovolný numerický dataset a libovolné metody předzpracování dat, je nezbytné pro určité případy omezit či upřesnit formu hledaného plánu. Například pro určité datasety je nějaký mód dané metody nepřípustný, či vyžadujeme, aby se určitá metoda vykonala před jinou metodou. Tyto nedostatky se v klasickém GA řeší opravnou funkcí. V mém případě využívám techniky genotypové masky s výchozím jedincem.

**výchozí jedinec** je jedinec, jehož některé bity v genotypech jsou inicializovány na předem zvolené hodnoty, které je nežádoucí měnit. Ostatní bity je možné zvolit náhodně.

**genotypová maska** je binární řetězec, který je před výpočtem inicializován a během výpočtu se již nemění. Obsahuje informaci o tom, které bity je možné měnit náhodnými operátory (počáteční inicializace a mutace) a pro které je změna zakázaná. Bity, které je zakázané měnit v průběhu GA jsou nastaveny na 1, ostatní na 0.

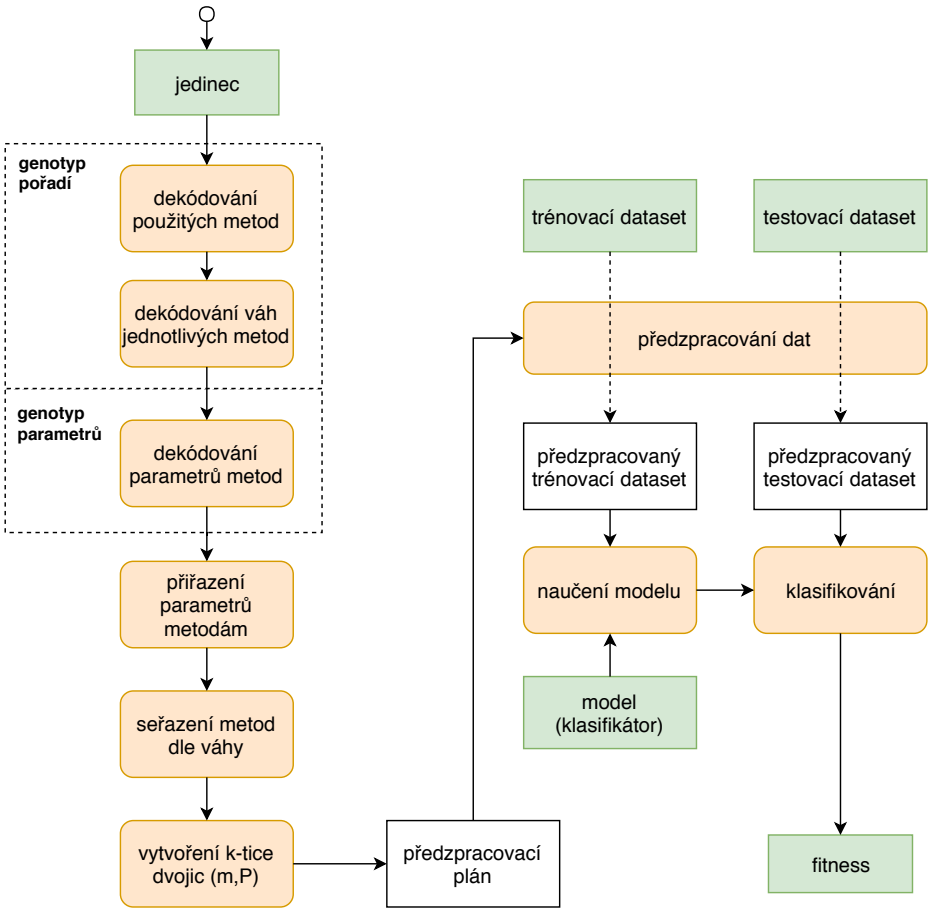
Při počáteční inicializaci jsou genotypy jedince nastaveny podle genotypů výchozího jedince. Poté je spuštěna náhodná inicializace, která pro každý bit v genotypech jedince určí jeho hodnotu. Toto se však neprovede pro bity, které jsou zamčeny genotypovou maskou.

Operátor mutace se při změně bitů dívá na genotypovou masku a změnu bitu provede pouze pokud je daný bit v masce odemčen, tedy nastaven na 0.

Výchozí jedinec a genotypová maska jsou sestaveny podle analýzy dat před zahájením výpočtu GA a není tedy třeba, aby oprava genotypu znovu analyzovala, zda nalezené řešení je validní. Tato technika zajišťuje efektivní zrychlení výpočtu oproti klasické opravné funkci, která by musela znovu analyzovat celý genotyp.

### 4.5.3 Výpočet fitness

Pro správný výpočet hodnoty fitness je nutné přesně definovat cílové řešení, kterého chci dosáhnout. V tomto případně chci nalézt takové řešení (plán), pomocí něhož model naučený na předzpracovaných trénovacích datech dosáhne co nejvyšší přesnosti klasifikace na předzpracovaných testovacích datech. Tedy hodnotu fitness mohu přímo vyjádřit jako úspěšnost klasifikace tohoto modelu




**Obrázek 4.5:** Schema výpočtu fitness pro jednoho jedince

na testovacích datech. Tímto způsobem výpočtu je předzpracování dat optimalizováno pro konkrétní zvolený klasifikátor. Průběh výpočtu hodnoty fitness včetně dekódování a sestavení plánu vyjadřuje schema 4.5.

Na vstupu je jedinec obsahující dva genotypy. Z genotypu pořadí dekóduji použité metody a k nim jejich váhy, z genotypu parametrů dekóduji parametry jednotlivých metod a ty následně k daným metodám přiřadím. Metody seřadím dle váhy a vytvořím k-tici dvojic $(m, P)$, která definuje plán pro předzpracování dat.

Následně tento plán použiji na transformaci trénovacího a testovacího datasetu, které jsou přiřazeny k jedinci, jehož fitness nyní počítám (více o přiřazení datasetů v následující sekci). Na předzpracovaném trénovacím datasetu naučím klasifikátor pro predikci cílového atributu třída. Učení klasifikátoru je závislé na konkrétním typu modelu (rozhodovací strom, Naivní Bayes, neuronová síť apod.). Pomocí tohoto klasifikátoru predikuji atribut třída pro

všechny instance z testovacího datasetu. Hodnota fitness je poté vypočtena jako

$$fitness = \frac{T_{correct}}{T_{all}} * 100$$

kde $T_{correct}$ je počet úspěšně klasifikovaných instancí z testovacího datasetu a $T_{all}$ je počet všech instancí v testovacím datasetu. Výsledná hodnota fitness je procentuální vyjádření klasifikační úspěšnosti z intervalu $\langle 0, 100 \rangle$.

### 4.5.4 Ostrovní model

Důležité opatření, které je u genetického algoritmu třeba zajistit, je to, aby jedinci v populaci neuvízli v lokálním optimu. Pro zajištění diverzity používám techniku ostrovního modelu.

Tato technika mi kromě diverzity umožňuje i velice dobře zamezit vysoké závislosti výpočtu na vybraných trénovacích a testovacích množinách dat. Bohužel jedním z problémů techniky redukování dat zmíněné výše je přítomnost náhody při výběru instancí. I přes Kolmogorovův-Smirnovův test se může stát, že vybrané množiny nebudou rozložením zcela odpovídat originálním datům, případně v nich bude poměrné zastoupení tříd mírně zdeformované. Pro snížení tohoto rizika problém distribuuji do několika nezávislých skupin.

Mám genetický algoritmus a v něm $l$ ostrovů. Každý ostrov má vlastní populaci jedinců a zároveň má přiřazenou vlastní trénovací a testovací množinu dat. Jedinci na daném ostrově přistupují při výpočtu fitness právě k těmto přiřazeným datasetům. Při výběru nejlepšího jedince z celé generace však jeho úspěšnost testuji na všech vytvořených datasetech (za každý ostrov jeden, tedy celkem $l$). Tímto snižuji riziko vzniklé z reprezentativní redukce dat. Více o výběru a testování nejlepších jedinců v sekci 4.5.8 selekce.

Jelikož jedinci na různých ostrovech se díky vybraným datasetům vyvíjejí lehce odlišně, je vhodné jejich informaci transportovat mezi jedince na jiných ostrovech. Pro přenos jedinců využívám techniku migrace. V každém generačním cyklu pomocí turnajové selekce vyberu $t$ jedinců z ostrova $i$ a přenesu je na ostrov $i + 1$. Z posledního ostrova jedinci migrují na první ostrov. Tímto migračním cyklem dávám důležitou znalost k dispozici všem jedincům v GA. Přesnou hodnotu migrace $t$ odhaduji v experimentální části.

### 4.5.5 Proměnlivé pravděpodobnosti operátorů

K zajištění dostatečné diverzity a zamezení uváznutí populace v lokálním optimu používám techniku inspirovanou simulovaným žíháním [29]. Pravděpodobnosti operátorů křížení a mutace jsou závislé na době nezlepšení nejlepšího nalezeného jedince. Pokud se po $g$ generacích neaktualizuje nejlepší nalezený jedinec, jsou pravděpodobnosti operátorů křížení a mutace upraveny tak, že

$$P_c = P_{c0} + \frac{g - g_{max}}{g}(\alpha - P_{c0})$$
$$P_m = P_{m0} + \frac{g - g_{max}}{g}(\beta - P_{m0})$$

kde $P_{c0}$ a $P_{m0}$ jsou počáteční pravděpodobnosti operátorů křížení a mutace, $g_{max}$ je konstanta určující počet generací než dojde k úpravě pravděpodobností a $\alpha$ a $\beta$ jsou zvolená reálná čísla z intervalu $(0, 1)$, pro která platí, že $P_{c0} < \alpha$ a $P_{m0} < \beta$. Tato čísla určují rychlost a přesnost při uniknutí z lokálního optima. Velké $\alpha$ a $\beta$ zajistí rychlé a daleké prohledávání i vzdálených řešení s rizikem ztráty již nalezených vhodných části genotypu. Nízké hodnoty naopak nasměrují hledání do blízkého okolí současných řešení bez rizika zdeformování populace.

### 4.5.6 Křížení

Operátor křížení je hlavním dopravcem již objevené informace v populaci a pro různé typy kódování je vhodný jiný způsob křížení. Díky tomu, že můj jedinec se skládá ze dvou různých genotypů, je možné provést na každém z nich odlišný způsob křížení. Porovnáním různých druhů křížení se věnuji dále v experimentální části.

### 4.5.7 Mutace se simulovaným žíháním

Operátor mutace v základu prohledává prostor blízkých řešení náhodně, ale je vhodné toto prohledávání usměrnit k potenciálně úspěšnějším řešením. Další technika, která je inspirována simulovaným žíháním, zavádí pravděpodobnost, se kterou jsem ochoten přijmout i horší řešení [29]. Tato pravděpodobnost se v závislosti na čase zmenšuje a tedy přijímám méně a méně horších řešení.

Mám řadu teplot $T$, pro kterou platí

$$T_{(t_g)} = \frac{T_0}{\log t_g}$$

kde $T_0$ je počáteční zvolená teplota a $t_g$ je čas reprezentovaný jako číslo aktuální generace. Dále mám dvě hodnoty fitness $f_o$ a $f_m$ pro originálního a zmutovaného jedince. Pokud $f_o < f_m$, pak původního jedince nahradím zmutovaným jedincem. Naopak pokud $f_o \geq f_m$, pak původního jedince nahradím zmutovaným jedincem s pravděpodobností $P_r$, která se spočte

$$P_r = e^{-\frac{F_m - F_o}{T_{t_g}}}$$

Pomocí této techniky tedy postupně v čase přecházím z dalekého prohledávání prostoru potenciálních řešeních k blízkému zdokonalování již úspěšných řešení pro nalezení skutečného optima.

### 4.5.8 Selekce

Pro výběr jedinců do další populace využívám turnajovou selekci. Tato technika umožňuje podpořit distribuci a zamezit uváznutí v lokálním optimu. Přesná hodnota velikosti turnaje je určena v experimentální části.

Při aktualizaci nejlepšího nalezeného jedince provádím dvě fáze výběru. Nejprve ze všech ostrovů získám nejlepšího jedince a vytvořím tak skupinu $l$ jedinců. Pro tyto jedince následně vypočtu průměrnou fitness na všech datasetech ze všech ostrovů. Pro průměrnou fitness jedince $I_i$ tedy platí, že

$$\bar{f} = \frac{\sum_{x=1}^{l} f_x(I_i)}{l}$$

kde $f_x$ je fitness funkce pro ostrov $x$ a $l$ je počet ostrovů. Tímto výrazně redukuji riziko zkreslení výsledku způsobené potencionálně zdeformovaným výběrem datasetů.

Kapitola 5

# Implementace algoritmu

Algoritmus implementuji formou jednoduché desktopové aplikace napsané v jazyce Java. Pro nástroje na vytěžování znalostí z dat využívám knihovnu Weka, která implementuje mnoho technik pro předzpracování dat, učení modelů i vyhodnocení samotné klasifikaci. V následujících sekcích uvedu hlavní důvody použití těchto technologií a dále čtenáře seznámím s uživatelským rozhraním aplikace, příkladem použití a detailnějším popisem důležitých tříd algoritmu, potřebných pro případné rozšíření aplikace.

Pro otestování funkčnosti a efektivity mého konceptu pro nalezení plánu předzpracování dat implementuji omezenou verzi celého obecného algoritmu. Aplikace na vstupu přijímá pouze **data s numerickými hodnotami** a využívá následující implementované metody: diskretizace, standardizace, normalizace, odstranění odlehlých hodnot, redukce dimenzionality (PCA), vyvažování tříd (SMOTE) a doplnění chybějících hodnot.

## 5.1 Knihovna Weka

Weka (Waikato Environment for Knowledge Analysis) [30] je svobodný software vyvinutý na univerzitě Waikato obsahující soubor nástrojů napsaných v jazyce Java pro vizualizaci, datovou analýzu a prediktivní modelování.

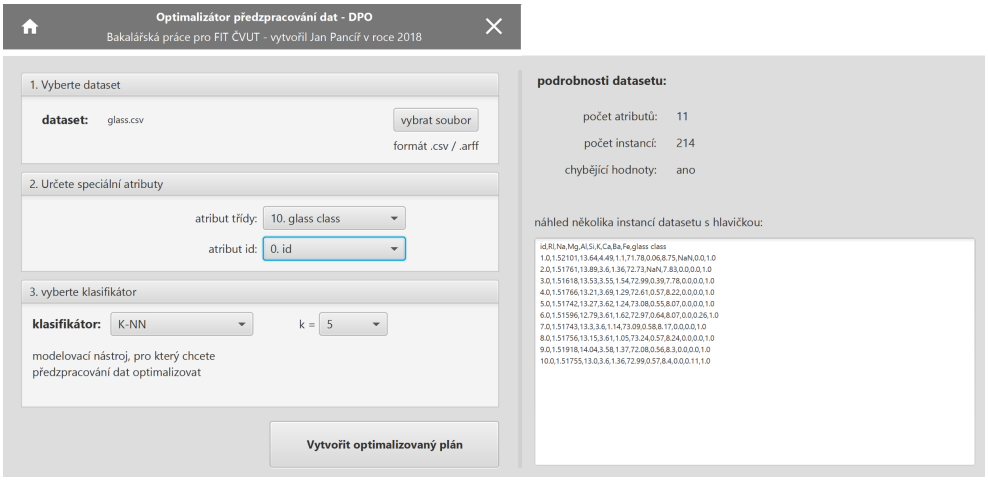
Tato knihovna obsahuje implementace základních metod předzpracování dat jako je diskretizace, normalizace, odstranění odlehlých hodnot, techniky redukce dimenzionality jako například PCA a další. Dále obsahuje algoritmy tvořící modely pro klasifikaci jako jsou K-NN, Naivní Bayes, rozhodovací strom, náhodný les a další. Tyto technologie mi umožňují snadno využít kvalitní otestované implementace algoritmů strojového učení, a proto jsem zvolil programovací jazyk Java, díky kterému mohu knihovnu Weka využít.

## 5.2 Uživatelské rozhraní

Aplikaci je možné ovládat jednoduchým uživatelským rozhraním, které uživatele provede všemi nezbytnými kroky pro vytvoření plánu předzpracování dat. Rozhraní se skládá ze dvou hlavních obrazovek. První obrazovka slouží k zadání a nastavení vstupů (obr. 5.1), druhá obrazovka obsahuje informace o průběhu výpočtu a možnost uložení nalezeného plánu (obr. 5.2).

Použití aplikace s popisem jednotlivých prvků je následující:

1. V hlavním menu vybrat možnost *Vytvořit plán*, zobrazí se obrazovka vstupů.



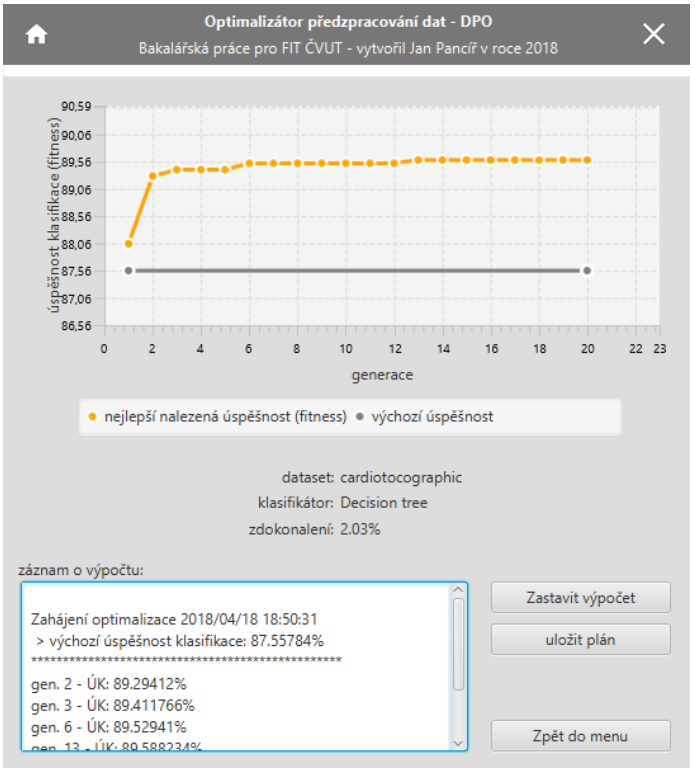

**Obrázek 5.1:** Náhled obrazovky zadání vstupů

2. Vybrat datový soubor (dataset), pro který chceme plán sestavit. Datový soubor může být ve formátu **CSV** nebo **ARFF**. V případě formátu csv musí jako první řádek obsahovat názvy atributů. Všechny hodnoty dat musí být numerické. Po načtení se v pravé obrazovce zobrazí základní informace o datasetu s náhledem několik instancí.

3. Z načtených názvů atributů je možné vybrat dva významné atributy. Atribut třída, tedy atribut, který chceme predikovat (výchozí nastavení je na poslední atribut v datasetu) a atribut ID, který identifikuje jednotlivé instance a je třeba ho při učení modelu ignorovat, je možné ho vynechat.

4. Vybrat klasifikátor, pro který chceme předzpracování dat optimalizovat. Je možné vybrat z nabídky: K-NN, rozhodovací strom, Naivní Bayes.

5. Zahájit výpočet stisknutím tlačítka *Vytvořit optimalizovaný plán*. Zobrazí se obrazovka výpočtu.




**Obrázek 5.2:** Náhled obrazovky výpočtu

6. Na obrazovce výpočtu je k dispozici graf průběhu výpočtu, který znázorňuje vývoj hodnoty fitness v závislosti na generacích GA, a záznam průběhu výpočtu s délkou trvání. Výpočet je možné předčasně ukončit tlačítkem *Zastavit výpočet*. Po dokončení výpočtu lze tlačítkem *uložit plán* nalezený plán předzpracování dat uložit ve formátu XML.

## 5.3 Formát plánu

Výstupem aplikace je plán pro předzpracování dat. Pro snadné strojové zpracování a zároveň pro dobrou čitelnost jsem jako výstupní formát zvolil textový soubor XML. Vygenerovaný plán se poté skládá ze dvou základních prvků:

**Informace o klasifikátoru** Jméno klasifikátoru, pro který byl plán optimalizován, s případnými parametry, které upřesňují jeho chování.

**Seznam metod** Uspořádaný seznam metod předzpracování dat v pořadí v jakém se mají na data použít. Ke každé metodě mohou být uvedeny parametry, které definují její chování.

Zde je ukázka plánu ve formátu XML:

```
<Classifier name="K-NN">
  <k>5</k>
</Classifier>
<PreprocessingMethods>
  <Method id="1" name="Discretize">
    <binningBy>equal_distance</binningBy>
    <bins>30</bins>
    <choosenColumns> 3 4 5</choosenColumns>
  </Method>
  <Method id="2" name="RemoveOutliers">
    <choosenColumns> 2 3 4</choosenColumns>
  </Method>
</PreprocessingMethods>
```

Tento plán předzpracování dat je optimalizovaný pro klasifikátor K-NN s k = 5. Popisuje předzpracování dat takové, že se nejprve na atributy 3, 4 a 5 použije metoda diskretizace (*Discretize*) podle vzdálenosti (*equal_distance*) a počtem binů 30. Poté se na atributy 2, 3 a 4 použije metoda odstranění odlehlých hodnot (*RemoveOutliers*). Podrobný popis všech implementovaných parametrů všech metod je dostupný ve zdrojových kódech aplikace nebo v přiloženém souboru popis_metod.txt.

## 5.4 Správa metod

V aplikaci je implementovaný kompletní obecný koncept algoritmu DPO, který byl cílem této práce, s několika metodami předzpracování dat. Pro zefektivnění a zvýšení využitelnosti aplikace je však nutné rozšířit seznam použitých metod o další druhy algoritmů předzpracování dat. V této sekci popíši důležité třídy mého implementovaného řešení pro případné snadné rozšíření aplikace.

### 5.4.1 třída PrepMethod

Jedná se o abstraktní třídu pro libovolnou metodu předzpracování dat. Veškeré metody pro předzpracování dat musí implementovat rozhraní této třídy.

Třída obsahuje dvě nutné proměnné, `bvLength` určuje délku binárního řetězce pro zakódování parametrů dané metody a `weight` je váha metody pro uspořádání metod v seznamu, není třeba měnit. Dále je nutné pro každou metodu implementovat následující funkce:

```
public String getName()
```

Funkce vrátí unikátní identifikátor dané metody předzpracování dat.

```
public void setUpFromBV(boolean[] binaryArray)
```

Funkce obdrží binární řetězec o délce `bvLength`, který dekóduje a nastaví tak hodnoty parametrů dané metody.

```
public void useOnDataset(Dataset inputDataset)
```

Funkce použije metodu předzpracování dat na předzpracování datasetu `inputDataset`.

```
public List<MyPair> getOptionsList()
```

Funkce vrátí všechny parametry této metody jako seznam dvojic (název parametru, hodnota).

```
public int updateBVLengthForDataset(Dataset dataset)
```

Pomocí této funkce lze upravit délku binárního řetězce `bvLength` podle formátu vstupních dat. Například některé metody vyžadují délku řetězce závislou na počtu atributů v datasetu apod.

```
public boolean [] createParameterMaskForDataset(Dataset dataset)
public boolean [] createParameterDefaultForDataset(Dataset dataset)
```

Funkce vytvoří a vrátí nastavení masky resp. výchozí hodnoty genotypu parametrů podle vstupního datasetu. Lze použít pokud chceme některé parametry zamknout na fixní hodnoty pro určitý typ dat.

### 5.4.2 třída MethodManager

Třída pro správu seznamu metod předzpracování dat. Seznam je reprezentován proměnnou `methods` a dále třída obsahuje následující funkce:

```
public void addAllMethods()
```

Funkce inicializuje seznam metod přidáním veškerých implementovaných metod pro předzpracování dat. Zde je třeba vytvořit instance všech nově implementovaných metod.

```
public void setupMethodsForInstances(Instances instances)
```

Funkce volaná před zahájením výpočtu pro zakázání určitých metod. Zde je možné vyřadit některé metody ze seznamu na základě analýzy vstupních dat.

## 5.5 Kódování parametrů metod

Chování metod lze ovlivnit nastavením jejich parametrů. Tyto parametry je pro GA nutné zakódovat do binárního řetězce. Jeho délka silně ovlivňuje čas výpočtu a je tedy třeba zvolit efektivní způsob kódování. Zde jsou použité druhy kódování pro čtyři různé typy parametrů, které se u metod nejčastěji používají.

**mody** Chování metody lze měnit nastavením tzv. modu, tedy parametr dosahuje malého počtu různých hodnot. Většinou se jedná o textový řetězec pro určení použitého algoritmu. *Například u metody diskretizace rozlišujeme mód diskretizace podle velikosti binů či podle frekvence.* Pro zakódování tohoto parametru používám $\log_2 h$ bitů, kde $h$ je počet různých hodnot, kterých parametr dosahuje. Každému modu je poté přiděleno unikátní číslo z intervalu $\langle 0, log_2 h)$ jako identifikátor modu.

**funkce** Chování metody lze měnit zapnutím či vypnutím určité funkce algoritmu. Tento parametr kóduji jedním bitem, kde hodnota 1 značí použití funkce, 0 nepoužití.

**čísla** Chování metody lze ovlivnit číselnou hodnotou ze známého intervalu $(h_{min}, h_{max})$. Tento číselný parametr kóduji $\log_2 (h_{max} - h_{min})$ bity jako číslo v přímém kódu. K dekódované hodnotě přičtu $h_{min}$ pro zisk výsledného čísla. Tento princip lze použít i pro kódování desetinných čísel vydělením výsledku zvolenou konstantou.

**atributy** Některé metody nevyžadují transformaci celého datasetu, ale lze je použít na libovolnou podmnožinu atributů. Kódování je formou $|\vec{a}|$ bitů, kde $|\vec{a}|$ je počet atributů datasetu a i-tý bit určuje použití metody na i-tý atribut.

***Příklad kódování:***
*metoda diskretizace (v knihovně Weka třída weka.filters.unsupervised.attribute.Discretize)*

*mod: podle velikosti binů / podle frekvence / optimalizovaný počet binů*

*3 různé hodnoty => 2 bity pro zakódování*

*číslo: maximální počet binů, zvolený interval* $\langle 1, 256)$

*256 hodnot =>* $log_2 256$ *= 8 bitů pro zakódování*

*atributy: indexy atributů pro použití metody*

$|\vec{a}|$ *bitů pro zakódování, kde* $|\vec{a}|$ *je počet atributů datasetu*

*Celková délka binárního řetězce pro zakódování parametrů metody diskretizace je* $2 + 8 + |\vec{a}|$ *bitů.*

Kapitola 6

# Experimenty a vyhodnocení

V následujících sekcích se věnuji experimentům a vyhodnocení dosažených výsledků na několika reálných datových souborech. Nejprve charakterizuji použité reálné datasety. Následně pomocí několika pokusů nastavuji hodnoty parametrů genetického algoritmu. Poté měřím zdokonalení úspěšnosti klasifikace mého řešení oproti nepředzpracovaným datům. Dále testuji úspěšnost plánu porovnáním různě naučených klasifikátorů pomocí programu Rapid Miner a následně porovnávám úspěšnost mého řešení s komerčním nástrojem IBM SPSS Modeler a jeho automatizovanou technikou předzpracování dat. V závěru této kapitoly diskutuji možná zlepšení mého řešení.

## 6.1 Použité datasety

Pro vyhodnocení dosažených výsledků bylo použito 7 reálných datasetů s různou datovou strukturou. Charakteristiku jednotlivých datasetů popisuje tabulka 6.1. Vybrané datasety se výrazně liší počtem atributů, počtem druhů hodnot klasifikační třídy i počtem instancí a zachycují tak různé druhy klasifikačních úloh. Všechny použité datasety jsou s podrobným popisem a zdrojem k dispozici v příloze.

**Tabulka 6.1:** charakteristika datasetů použitých pro experimenty a vyhodnocení

| dataset | instancí | atributů | druhů třídy |
|---|---|---|---|
| breast-cancer-wisconsin | 699 | 11 | 2 |
| cardiotocographic | 2126 | 26 | 10 |
| ecoli | 336 | 8 | 8 |
| glass | 214 | 11 | 7 |
| messidor-features | 1150 | 20 | 2 |
| sonar | 208 | 61 | 2 |
| transfusion | 748 | 5 | 2 |

Pro experimenty a následné ověření dosažených výsledků jsem použil tři z výše zmíněných datasetů, které nejlépe charakterizují různé druhy klasifikačních úloh. Následuje podrobnější popis těchto vybraných datasetů.

**dataset transfusion** (Blood Transfusion Service Center) [31]

Numerický dataset s menším počtem atributů (5) a počtem instancí jevících se jako střední počet (748). Popisuje úlohu nalezení klasifikátoru, který z dostupných hodnot o dárci krve bude predikovat, zda dárce daroval krev v květnu 2007. Jedná se o typický jednoduchý dataset pro klasifikační úlohy se dvěma třídami.

**Tabulka 6.2:** Ukázka instancí datasetu transfusion

| Naposledy | Počet | Krev (ml) | Poprvé | daroval krev? |
|---|---|---|---|---|
| 2 | 50 | 12500 | 98 | 1 |
| 0 | 13 | 3250 | 28 | 1 |
| 1 | 12 | 3000 | 35 | 0 |

**dataset cardiotocography** (Cardiotocography Data Set) [32]

Numerický dataset s větším počtem atributů (26) a počtem instancí jevících se jako velký počet (2126). Popisuje úlohu nalezení klasifikátoru, který z dostupných naměřených hodnot pacienta bude predikovat třídu jeho fetální srdeční frekvence. Jedná se o pokročilejší klasifikační úlohu o více jak dvou třídách.

**dataset sonar** (Connectionist Bench Sonar Data Set) [33]

Numerický dataset s větším počtem atributů (61) a počtem instancí jevících se jako malý počet (208). Dataset byl upraven tak, aby obsahoval chybějící hodnoty. Popisuje úlohu nalezení klasifikátoru, který z naměřených hodnot energií ve frekvenčních pásmech bude predikovat, zda byla nalezena mina či nikoliv. Hodnoty v datasetu jsou již normalizovány. Je nutné využít techniky nahrazení chybějících hodnot.

## 6.2 Nastavení parametrů GA

Konkrétní hodnoty klíčových parametrů v genetickém algoritmu mohou významně ovlivnit úspěšnost a rychlost výpočtu. V mém implementovaném řešení je možno tyto hodnoty měnit v souboru **GAConfig.txt**.

Nejprve jsem hodnoty odhadl ze zkušenosti a experimentů v [21]. Poté jsem provedl několik pokusů na datasetech popsaných výše s využitím různých klasifikátorů. Z pozorování jsem poté hodnoty parametrů upravil tak, aby řešení byla co nejúspěšnější a zároveň výpočet doběhl v přijatelném čase.

Zde je seznam parametrů GA s vysvětlením jejich funkce a odhadnutou optimální hodnotou pro implementovaný algoritmus DPO.

počet generací *generation_count* : Počet iterací evolučního cyklu, tedy délka a přesnost výpočtu. Nejlepší řešení byla nalezena vždy do 30. generace. Tedy dostačující hodnota je 30.

počet ostrovů *islands_count* : Počet ostrovů, tedy počet nezávislých populací mezi nimiž migrují jedinci. Optimální hodnoty jsem nalezl v rozsahu 4-6, větší hodnoty nemají v souvislosti s malým počtem generací smysl. Nízký počet ostrovů způsobuje malý počet testovacích a trénovacích množin a tedy nepřesnost výpočtu.

velikost populace *population_size* : Počet jedinců v jedné populaci. Vyšší hodnoty zpřesňují výpočet, ale výrazně ho časově zpomalují. Dostačující hodnotu jsem odhadl na 80 jedinců.

velikost katastrofy *catastrophe_size* : Počet nově inicializovaných jedinců doplněných v každé generaci. Postačují nízké hodnoty, jinak se jedná o zbytečné zavedení náhodného prohledávání prostoru. Optimální hodnotu jsem odhadl na přibližně jednu desetinu populace, tedy 8.

p. mutace *mutation_probability* : Počáteční pravděpodobnost změny bitu u operátoru mutace. Při použití různých technik simulovaného žíhání je vhodné zvolit vyšší hodnotu. Nalezená hodnota inspirovaná výsledky v [21] je 0.05 a k ní odhadnutá alfa konstanta 0.2.

p. křížení *crossover_probability* : Počáteční pravděpodobnost křížení dvou jedinců. Při použití různých technik simulovaného žíhání je vhodné zvolit vyšší hodnotu. Nalezená hodnota inspirovaná výsledky v [21] je 0.7 a k ní odhadnutá beta konstanta 0.9

teplota *initial_temperature* : Počáteční teplota pro úpravu pravděpodobností operátorů v závislosti na generaci. Pro optimalizaci v tomto malém počtu generací se nejlépe jevily hodnoty z rozsahu 4-10.

velikost turnaje *tournament_size* : Počet jedinců použitých pro výběr turnajovou selekcí. Vhodnou hodnotu jsem nalezl kolem jedné desetiny velikosti populace, tedy 8.

### 6.2.1 Pravděpodobnost K-S testu

Důležitým parametrem při přípravě dat je pravděpodobnost s jakou považuji libovolnou podmnožinu původních dat za reprezentativní vzorek těchto dat. V Kolmogorovovu-Smirnovovu testu jde o určení pravděpodobnosti s jakou dva náhodné výběry mají stejné pravděpodobnostní rozdělení. K-S test při reprezentativní redukci tedy zamítne hypotézu, že dataset A je reprezentativním vzorkem datasetu B, právě když je testem nalezená pravděpodobnost menší než zvolená pravděpodobnost. Z několika provedených pokusů vyplývá, že pro dostatečně kvalitní vzorky dat, které nezkreslují původní data, je nutné zvolit pravděpodobnost alespoň **85 %**. Nižší hodnoty způsobují velké výkyvy v úspěšnosti klasifikace a model naučený na takto vybraných množinách může

být přeučen. Vyšší pravděpodobnost zpřesňuje výpočet, avšak neumožňuje použití algoritmu na rozsáhlé datasety kvůli časové složitosti výpočtu.

### 6.2.2 Volba operátoru křížení

V mém řešení, kde používám jedince se dvěma genotypy, mohu pro každý z nich použít různý operátor křížení. V následujícím experimentu jsem na datasetu transfusion vyzkoušel několik typů křížení obou genotypů a sledoval jsem dvě hlavní kritéria. Zaprvé jaké procentuální zlepšení klasifikace nalezené řešení přinese a zadruhé v kolikáté generaci je toto řešení nalezeno. Zde je souhrn získaných průměrných hodnot z několika opakováních experimentu:

**Tabulka 6.3:** Porovnání operátorů křížení na datasetu transfusion

| **křížení G1** | **křížení G2** | **zlepšení** | **generace nalezení** |
|---|---|---|---|
| jednobodové | jednobodové | 1,5 % | 10 |
| uniformní | jednobodové | 1,5 % | 8 |
| jednobodové | uniformní | **2,5 %** | **7** |
| uniformní | uniformní | 2,2 % | 7 |

Pokusy jsem provedl i s vícebodovým křížením, které se však výsledky výrazně nelišilo od jednobodového křížení. Ze získaných hodnot tedy nejlépe vyhovuje použít **jednobodové křížení na genotyp pořadí** (G1) a **uniformní křížení na genotyp parametrů** (G2).

## 6.3 Dosažené výsledky

Implementovaný algoritmus DPO jsem použil pro nalezení plánu předzpracování dat pro všech sedm výše popsaných datasetů. Ke každému datasetu jsem nalezl tři plány, kde každý z nich byl optimalizovaný pro jiný klasifikátor (5-NN, Naivní Bayes, rozhodovací strom). Zdokonalení přesnosti klasifikace zvoleného klasifikátoru při použití nalezeného plánu předzpracování dat oproti klasifikátoru naučeném na původních nezpracovaných datech popisuje tabulka 6.4.

Výsledky ukázaly, že pro klasifikátory typu K-NN a rozhodovací strom jsou data předzpracovaná dle plánu algoritmu DPO klasifikována v průměru přesněji o 4-5 %. Pro pravděpodobnostní klasifikátor Naivní Bayes jsou data klasifikována přesněji v průměru až o 9 %. Existují však výkyvy v obou směrech. U dobře strukturovaných a již zpracovaných dat dosahuje algoritmus DPO zdokonalení kolem 1 %, jako například u datasetu *breast-cancer-wisconsin*. Naopak u hůře strukturovaných nezpracovaných datasetů, jako například *cardiotocographic* nebo *sonar*, dosahuje algoritmus DPO zdokonalení úspěšnosti klasifikace o více jak 10 %. Všechny vygenerované plány, rozšířená tabulka výsledků a grafy demonstrující průběhy výpočtů jsou k dispozici v příloze.

**Tabulka 6.4:** Dosažené zdokonalení úspěšnosti klasifikace pro vybrané datasety (v procentech).

| dataset | 5-NN | Naivní Bayes | Rozhodovací strom |
|---|---|---|---|
| breast-cancer-wisconsin | 1,20 | 2,58 | 3,67 |
| cardiotocographic | 4,95 | 12,13 | 2,03 |
| ecoli | 1,96 | 3,04 | 4,70 |
| glass | 8,41 | 18,69 | 4,83 |
| messidor-features | 1,52 | 10,57 | 5,11 |
| sonar | 7,69 | 13,94 | 9,42 |
| transfusion | 6,33 | 3,11 | 5,05 |
| **průměrné zdokonalení** | **4,58** | **9,15** | **4,97** |

## 6.4 Ověření plánu pomocí nástroje Rapid Miner

Pro ověření, zda je nalezený plán předzpracování dat skutečně optimální, používám nástroje Rapid Miner [34]. Umožňuje nasimulovat různé scénáře předzpracování dat a změřit úspěšnost klasifikátoru naučeném na těchto datech. V následujících testech jsem se zaměřil především na ověření vhodnosti vybraných metod a na pořadí, v jakém se na data použijí. Testy jsem provedl pouze s plány předzpracování dat, které bylo možné kompletně nasimulovat v programu Rapid Miner.

Využitím fáze přípravy dat algoritmu DPO jsem vygeneroval pro zvolené datasety jejich trénovací a testovací množiny (jedná se o reprezentativní vzorky ověřené K-S testem). V programu Rapid Miner jsem sestavil schema pro naučení zvoleného klasifikátoru na trénovací množině dat a ověření úspěšnosti klasifikace tohoto klasifikátoru na testovací množině dat.

Obě datové množiny jsem předzpracoval podle nalezeného plánu předzpracování dat, tedy jsem sestavil sekvenci metod předzpracování dat a nastavil jejich parametry dle výstupu algoritmu DPO. Následně jsem vytvořil několik alternativ plánu pro předzpracování dat záměnou pořadí jednotlivých metod či výměnou některé metody za jinou.

Výsledky naměřené úspěšnosti klasifikace s použitím plánu DPO a dvěma nejlepšími nalezenými alternativami popisují tabulky 6.5 až 6.9. Ověřil jsem, že naměřená úspěšnost klasifikace se shoduje s výstupem algoritmu DPO a všechny nalezené alternativy mají stejnou či horší úspěšnost klasifikace. Algoritmus DPO tedy generuje vhodný plán předzpracování dat, který skutečně zdokonaluje úspěšnost klasifikace zvoleného klasifikátoru.

**Tabulka 6.5:** Úspěšnost klasifikátoru **Naivní Bayes** pro dataset **sonar** dle postupů předzpracování dat (v procentech).

| | **postup předzpracování dat** | **úspěšnost klasifikace** |
|---|---|---|
| plán DPO | 1. normalizace<br>2. diskretizace<br>3. odstranění odlehlých hodnot | **85,57** |
| alternativa 1 | 1. standardizace<br>2. diskretizace | 83,65 |
| alternativa 2 | 1. odstranění odlehlých hodnot<br>2. diskretizace | 85,50 |

**Tabulka 6.6:** Úspěšnost klasifikátoru **Naivní Bayes** pro dataset **transfusion** dle postupů předzpracování dat (v procentech).

| | **postup předzpracování dat** | **úspěšnost klasifikace** |
|---|---|---|
| plán DPO | 1. standardizace<br>2. PCA | **79,50** |
| alternativa 1 | 1. PCA<br>2. standardizace | 78,40 |
| alternativa 2 | 1. normalizace<br>2. PCA | 79,33 |

**Tabulka 6.7:** Úspěšnost klasifikátoru **5-NN** pro dataset **transfusion** dle postupů předzpracování dat (v procentech).

| | **postup předzpracování dat** | **úspěšnost klasifikace** |
|---|---|---|
| plán DPO | 1. standardizace<br>2. odstranění odlehlých hodnot | **81,00** |
| alternativa 1 | 1. odstranění odlehlých hodnot<br>2. standardizace | 80,87 |
| alternativa 2 | 1. odstranění odlehlých hodnot<br>2. normalizace | 78,20 |

## 6.5 Porovnání s nástrojem IBM SPSS Modeler

Pro zjištění úspěšnosti mého řešení oproti jiným dostupným nástrojům jsem zvolil porovnání s komerčním nástrojem IBM SPSS Modeler [22], který obsahuje funkcionalitu automatického předzpracování dat. Tímto nástrojem jsem předzpracoval trénovací a testovací data a následně změřil úspěšnost klasifikace pro dané klasifikátory.

**Tabulka 6.8:** Úspěšnost klasifikátoru **rozhodovací strom** pro dataset **transfusion** dle postupů předzpracování dat (v procentech).

| | **postup předzpracování dat** | **úspěšnost klasifikace** |
|---|---|---|
| plán DPO | 1. odstranění odlehlých hodnot | **82,00** |
| alternativa 1 | 1. odstranění odlehlých hodnot<br>2. standardizace | 82,00 |
| alternativa 2 | 1. diskretizace | 79,60 |

**Tabulka 6.9:** Úspěšnost klasifikátoru **5-NN** pro dataset **cardiotocographic** dle postupů předzpracování dat (v procentech).

| | **postup předzpracování dat** | **úspěšnost klasifikace** |
|---|---|---|
| plán DPO | 1. normalizace<br>2. diskretizace | **79,20** |
| alternativa 1 | 1. standardizace | 79,04 |
| alternativa 2 | 1. normalizace<br>2. PCA | 66,00 |

Úspěšnost mého řešení oproti nástroji IBM SPSS Modeler je zobrazena v tabulce 6.10. Rozdíl ÚK udává, o kolik procent je úspěšnost klasifikace modelu naučeném na datech předzpracovaných podle algoritmu DPO větší, než úspěšnost klasifikace modelu naučeném na datech předzpracovaných nástrojem IBM SPSS Modeler. Ve všech testech dosáhl algoritmus DPO lepších výsledků.

**Tabulka 6.10:** Porovnání úspěšnosti klasifikace s použitím algoritmu DPO oproti použití nástroje IBM SPSS Modeler (v procentech).

| **dataset** | **klasifikátor** | **rozdíl ÚK** |
|---|---|---|
| transfusion | Naivní Bayes | 2,5 |
| transfusion | rozhodovací strom | 2,3 |
| transfusion | 5-NN | 2,2 |
| cardiotocographic | Naivní Bayes | 12,6 |
| cardiotocographic | rozhodovací strom | 2,2 |
| cardiotocographic | 5-NN | 0,1 |
| sonar | Naivní Bayes | 8,3 |
| sonar | rozhodovací strom | 2,9 |
| sonar | 5-NN | 0,8 |

## 6.6 Diskuze

V této části komentuji zkušenosti získané při tvorbě této práce. Nejprve uvádím možná využití konceptu v praxi, dále popisuji možnost rozšíření algoritmu pro řešení regresních úloh. Poté zdůrazňuji klíčové části celého algoritmu s možnými dalšími rozšířeními.

Z dosažených výsledků je zřejmé, že princip obecného automatizovaného předzpracování dat je možný a umožňuje efektivně zajistit zvýšení přesnosti klasifikace. Tento nástroj však stále neuvažuje další klíčové faktory, které rozhodují o výběru metod či klasifikátoru. Důležitým faktorem je například interpretovatelnost dat i výsledného modelu, která je pro některé úlohy velice důležitá i na úkor úspěšnosti klasifikace. Dále platí, že při zcela automatizovaném předzpracování dat se mnohdy nevyhneme drobné deformaci dat, tedy zanesení či drobného zkreslení původních dat. Pro výrazně citlivé klasifikační úlohy (například v lékařství) by automatizované předzpracování mělo být použito jen jako pomocný nástroj, vždy kontrolovaný znalostním expertem. Navržená technika je vhodná pro úlohy s častým střídáním dat s různou strukturou pro rychlé zlepšení úspěšnosti klasifikace daného modelu. Lze ji tedy aplikovat na projekty již používající automatizované systémy vytěžování znalostí z dat. Dále je vhodná jako pomocný nástroj při hledání optimálního předzpracování dat v komplexních a datově citlivých úlohách, například z oblasti lékařství či průmyslové výroby.

Ve své práci jsem navrhoval koncept pouze pro optimalizaci klasifikačních úloh. Algoritmus lze však jednoduše aplikovat i na řešení regresních úloh, například optimalizací předzpracování dat pro model neuronových sítí apod. Obecný koncept přípravy dat i genetického algoritmu lze ponechat, je třeba pouze upravit výpočet fitness jedince tak, aby hodnota odpovídala úspěšnosti predikce v regresní úloze.

Nejdůležitější částí celého konceptu pro úspěšné řešení se ukázala fáze přípravy dat. Vhodné rozdělení surových dat na trénovací a testovací množiny je nutnou podmínkou pro nalezení úspěšného plánu předzpracování dat. Zvolený statistický nástroj Kolmogorovův-Smirnovův test se ukázal jako dostačující technika pro nalezení reprezentativních trénovacích a testovacích datasetů. V mém řešení využívám zjednodušeného multidimenzionálního K-S testu, který předpokládá nezávislost atributů. Tento test by bylo vhodné rozšířit o zpracování i závislých atributů, aby nedocházelo k výběru případných zdeformovaných datasetových podmnožin.

Jako rozšíření mého konceptu se nabízí podpora datasetů s nečíselnými hodnotami. V mém řešení zmiňuji možnost převodu všech nominálních dat na numerická, jelikož pro tuto transformaci existuje již řada technik. Navržení optimálního algoritmu, který nalezne vhodné techniky pro transformaci těchto dat by bylo vhodným doplňkem celého konceptu.

Genetický algoritmus použitý jako optimalizační nástroj se ukázal jako vhodný avšak časově náročný výpočetní mechanismus. Pro můj obecný kon-

cept algoritmu DPO, který není omezen počtem metod předzpracování dat či počtem parametrů jednotlivých metod, je GA vhodným nástrojem, protože přidání nových metod či nových parametrů již výrazně neovlivňuje časovou náročnost výpočtu. Efektivita navrženého algoritmu se tedy projeví zejména při využití většího počtu metod předzpracování dat s podrobnými parametry pro jejich nastavení.

Genetický algoritmus navržený v mém řešení, lze využít i na jiné úlohy a problémy. Prvky jako jsou ostrovní model s migrací, genotypová maska, proměnlivá pravděpodobnost operátorů, mutace se simulovaným žíháním a další lze obecně aplikovat do libovolného genetického algoritmu. Princip rozdělení genotypu na dva binární řetězce, kde každý kóduje část z celé úlohy, se ukázal vhodný nejen jako odlišovací prvek, ale i jako efektivní technika pro zrychlení a zpřesnění výpočtu. Zobecněním této techniky lze tedy libovolnou úlohu řešenou genetickým algoritmem rozdělit na libovolný počet podúloh, kde každá podúloha má vlastní genotyp k zakódování a tedy vlastní operátory křížení, případně mutace atd. Nabízí se tedy snaha o prozkoumání této techniky a případné vytvoření nástroje, který pro každý genotyp nalezne operátory dosahujících nejlepších výsledků.

# Závěr

V práci jsem se zabýval návrhem a implementací algoritmu pro nalezení optimálního pořadí metod předzpracování dat a jejich parametrů pro co nejlepší možnou klasifikaci.

Nejprve jsem obecně uvedl základní pojmy strojového učení a zdůraznil jsem vliv předzpracování dat na úspěšnost klasifikace. Dále jsem se podrobněji věnoval vybraným metodám předzpracování dat a definoval jsem základní koncept genetického algoritmu. Prozkoumal jsem stávající přístupy a metody, které se zabývají automatizovaným předzpracováním dat – zejména technologii IPT.

Navrhl jsem algoritmus DPO využívající evoluční techniky, který pro libovolný numerický dataset a vybraný klasifikátor nalezne optimální plán předzpracování dat.

Implementoval jsem navržený algoritmus formou desktopové aplikace v jazyce Java s několika vybranými metodami předzpracování dat. Představil jsem hlavní implementované třídy pro snadnou možnost rozšíření aplikace o nové metody. Věnoval jsem se několika experimentům pro nalezení optimálních parametrů implementovaného genetického algoritmu. Úspěšnost mého konceptu jsem ověřil na sedmi reálných datasetech. Nalezený plán předzpracování dat zvyšuje úspěšnost klasifikace v průměru o 4-9 %. Dále jsem porovnal úspěšnost klasifikace s daty předzpracovanými nástrojem IBM SPSS Modeler a navržený algoritmus DPO dosahoval ve všech testech lepších výsledků. V závěrečné diskusi jsem se věnoval možným rozšířením konceptu a využitelnosti v praktických úlohách.

# Literatura


[1] Franklin, J.: The elements of statistical learning: data mining, inference and prediction [online]. *The Mathematical Intelligencer*, ročník 27, č. 2, 2005: s. 83–85. Dostupné z: `https://link.springer.com/content/pdf/10.1007/BF02985802.pdf`

[2] Jermyn, P.; Dixon, M.; Read, B. J.: Preparing clean views of data for data mining. *ERCIM Work. on Database Res*, 1999: s. 1–15.

[3] Han, J.; Kamber, M.; Pei, J.: *Data Mining: Concepts and Techniques*. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., třetí vydání, 2011, ISBN 0123814790, 9780123814791.

[4] Pyle, D.: *Data preparation for data mining*, ročník 1. Morgan Kaufmann, 1999.

[5] Bramer, M.: *Principles of Data Mining*, ročník 180. Springer, 01 2007, ISBN 978-1-84628-765-7.

[6] Murphy, K. P.: Naive bayes classifiers [online]. *University of British Columbia*, ročník 18, 2006. Dostupné z: `https://datajobsboard.com/wp-content/uploads/2017/01/Naive-Bayes-Kevin-Murphy.pdf`

[7] Čepek, M.: *Inductive Preprocessing Technology [online]*. teze disertační práce, České vysoké učení technické v Praze. Fakulta elektrotechnická. Katedra počítačů., 06 2013. Dostupné z: `http://hdl.handle.net/10467/15629`

[8] Bowyer, K. W.; Chawla, N. V.; Hall, L. O.; aj.: SMOTE: Synthetic Minority Over-sampling Technique [online]. , č. arXiv:1106.1813, Jun 2011. Dostupné z: `http://cds.cern.ch/record/1357766`

[9] Leys, C.; Ley, C.; Klein, O.; aj.: Detecting outliers: Do not use standard deviation around the mean, use absolute deviation

around the median [online]. *Journal of Experimental Social Psychology*, ročník 49, č. 4, 2013: s. 764 – 766, ISSN 0022-1031, doi:https://doi.org/10.1016/j.jesp.2013.03.013. Dostupné z: http://www.sciencedirect.com/science/article/pii/S0022103113000668

[10] Acuña, E.; Rodriguez, C.: *The Treatment of Missing Values and its Effect on Classifier Accuracy*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2004, ISBN 978-3-642-17103-1, 639–647 s.

[11] García, S.; Ramírez-Gallego, S.; Luengo, J.; aj.: Big data preprocessing: methods and prospects [online]. *Big Data Analytics*, ročník 1, č. 1, Nov 2016: str. 9, ISSN 2058-6345, doi:10.1186/s41044-016-0014-0. Dostupné z: https://doi.org/10.1186/s41044-016-0014-0

[12] Liu, H.; Hussain, F.; Tan, C. L.; aj.: Discretization: An Enabling Technique [online]. *Data Mining and Knowledge Discovery*, ročník 6, č. 4, Oct 2002: s. 393–423, ISSN 1573-756X, doi:10.1023/A:1016304305535. Dostupné z: https://doi.org/10.1023/A:1016304305535

[13] Bellman, R. E.: *Adaptive control processes: a guided tour*. Princeton university press, 2015.

[14] Guyon, I.; Gunn, S.; Nikravesh, M.; aj.: *Feature extraction: foundations and applications*, ročník 207. Springer, 2008.

[15] Lei, Y.; Huan, L.: Feature selection for high-dimensional data: A fast correlation-based filter solution [online]. 2003: s. 856–863. Dostupné z: http://www.aaai.org/Papers/ICML/2003/ICML03-111.pdf

[16] Dunteman, G. H.: *Principal Components Analysis [online]*. SAGE, 1989, ISBN 978-0-8039-3104-6. Dostupné z: https://books.google.com/books?id=Pzwt-CMMt4UC&pg=PA5

[17] contributors, W.: Kolmogorov–Smirnov test [online]. 2018. Dostupné z: https://en.wikipedia.org/wiki/Kolmogorov=Smirnov_test

[18] wang, H.-M.: Comparison of the Goodness-of-Fit Tests: the Pearson Chi-square and Kolmogorov-Smirnov Tests [online]. *Ling Tung University, Department of Information Management*. Dostupné z: https://pdfs.semanticscholar.org/0c3a/736708f50897690875b6e578e62657df437b.pdf

[19] J Russell, S.; Norvig, P.: *Artificial Intelligence: A Modern Approach*. 01 1995, ISBN 0137903952.

[20] Pohlheim, H.: Evolutionary Algorithms 3 Selection [online]. 2006. Dostupné z: http://www.geatbx.com/docu/algindex-02.html

[21] Schaffer, J.; Caruana, R.; Eshelman, L.; aj.: *A Study of Control Parameters Affecting Online Performance of Genetic Algorithms for Function Optimization.* 01 1989, 51-60 s.

[22] IBM: *IBM SPSS Modeler 14.2 Algorithms Guide.* 233 South Wacker Drive, 11th Floor, Chicago: IBM, il 60606-6412 vydání, 2011.

[23] Vishnubhotla, P. R.: Method and system for data mining automation in domain-specific analytic applications [online]. 09 2004. Dostupné z: https://patents.justia.com/patent/6799181

[24] Farquhar, J.; Hill, N. J.: Interactions Between Pre-Processing and Classification Methods for Event-Related-Potential Classification [online]. *Neuroinformatics*, ročník 11, č. 2, Apr 2013: s. 175–192, ISSN 1559-0089, doi:10.1007/s12021-012-9171-0. Dostupné z: https://doi.org/10.1007/s12021-012-9171-0

[25] Qin, Y.; Zhang, S.; Zhu, X.; aj.: Semi-parametric optimization for missing data imputation [online]. *Applied Intelligence*, ročník 27, č. 1, Aug 2007: s. 79–88, ISSN 1573-7497, doi:10.1007/s10489-006-0032-0. Dostupné z: https://doi.org/10.1007/s10489-006-0032-0

[26] Jensen, R.; Shen, Q.: Fuzzy-rough data reduction with ant colony optimization [online]. *Fuzzy Sets and Systems*, ročník 149, č. 1, 2005: s. 5 – 20, ISSN 0165-0114, doi:https://doi.org/10.1016/j.fss.2004.07.014, fuzzy Sets in Knowledge Discovery. Dostupné z: http://www.sciencedirect.com/science/article/pii/S0165011404003136

[27] Nocedal, J.; Wright, S. J.: *Numerical Optimization.* New York, NY, USA: Springer, druhé vydání, 2006.

[28] Storn, R.; Price, K.: Differential Evolution – A Simple and Efficient Heuristic for global Optimization over Continuous Spaces [online]. *Journal of Global Optimization*, ročník 11, č. 4, Dec 1997: s. 341–359, ISSN 1573-2916, doi:10.1023/A:1008202821328. Dostupné z: https://doi.org/10.1023/A:1008202821328

[29] Hwang, S.-F.; He, R.-S.: Improving real-parameter genetic algorithm with simulated annealing for engineering problems [online]. *Advances in Engineering Software*, ročník 37, č. 6, 2006: s. 406 – 418, ISSN 0965-9978, doi:https://doi.org/10.1016/j.advengsoft.2005.08.002. Dostupné z: http://www.sciencedirect.com/science/article/pii/S0965997805001377

[30] Hall, M.; Frank, E.; Holmes, G.; aj.: The WEKA Data Mining Software [online]. *SIGKDD Explor. Newsl.*, ročník 11, č. 1, Listopad 2009: s. 10–18, ISSN 1931-0145, doi:10.1145/1656274.1656278. Dostupné z: http://doi.acm.org/10.1145/1656274.1656278

[31] Yeh, I.-C.: Blood Transfusion Service Center Data Set [online]. 2008. Dostupné z: https://archive.ics.uci.edu/ml/datasets/Blood+Transfusion+Service+Center

[32] Marques de SÃ¡, J.: Cardiotocography Data Set [online]. 2010. Dostupné z: https://archive.ics.uci.edu/ml/datasets/Cardiotocography

[33] Sejnowski, T.: Connectionist Bench (Sonar, Mines vs. Rocks) Data Set [online]. 1988. Dostupné z: http://archive.ics.uci.edu/ml/datasets/Connectionist+Bench+(Sonar,+Mines+vs.+Rocks)

[34] Hofmann, M.; Klinkenberg, R.: *RapidMiner: Data Mining Use Cases and Business Analytics Applications [online]*. Chapman & Hall/CRC, 2013, ISBN 1482205491, 9781482205497.

Příloha **A**

# Seznam použitých zkratek

**DPO** Data Preprocessing Optimizer

**GA** Genetický algoritmus

**KS** Kolmogorov-Smirnov

**GUI** Graphical user interface

**XML** Extensible markup language

**IQR** Interquartile range

**SMOTE** Synthetic Minority Over-sampling Technique

**FCBF** Fast Correlation-Based Filter

**PCA** Principal component analysis

**RWS** Roulettewheel selection

**SUS** Stochastic universal sampling

**TS** Tournament selection

**IPT** Inductive Preprocessing Technology

**Weka** Waikato Environment for Knowledge Analysis

**CSV** Comma-separated values

**ARFF** Attribute-Relation File Format

**ÚK** úspěšnost klasifikace

Příloha **B**

# Obsah přiloženého CD

- `datasety` .. adresář s datasety využitými pro experimenty a vyhodnocení
- `DPO` ..... adresář se spustitelnou formou implementace
- `impl` ..... zdrojové kódy implementace
- `thesis` ..... zdrojová forma práce ve formátu LaTeX
- `výsledky`
  - `grafy` ..... grafy dokumentující průběhy výpočtů
  - `plány` ..... vytvořené plány předzpracování dat
  - `zdokonaleni.pdf` ..... souhrnná tabulka výsledků
- `BP_Pancir_Jan_2018.pdf` ..... text práce ve formátu PDF
- `aplikace DPO.txt` ..... požadavky ke spuštění aplikace
- `obsah CD.txt` ..... stručný popis obsahu CD