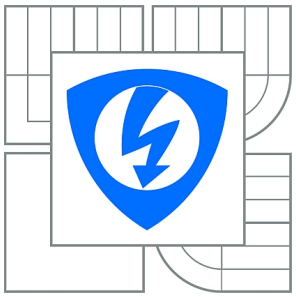
VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA ELEKTROTECHNIKY A KOMUNIKAČNÍCH TECHNOLOGIÍ

ÚSTAV TELEKOMUNIKACÍ

FACULTY OF ELECTRICAL ENGINEERING AND COMMUNICATION

DEPARTMENT OF TELECOMMUNICATIONS

PROCEDURÁLNÍ PROGRAMOVÁNÍ V DATABÁZI

PROCEDURAL PROGRAMMING IN DATABASE

BAKALÁŘSKÁ PRÁCE

BACHELOR'S THESIS

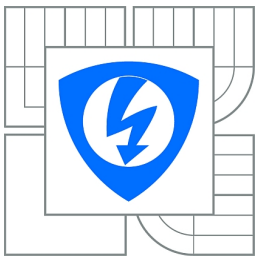
AUTOR PRÁCE — ADAM NIMRICHTER

AUTHOR

VEDOUCÍ PRÁCE — Ing. VÁCLAV UHER

SUPERVISOR

BRNO 2015



VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

Fakulta elektrotechniky a komunikačních technologií

Ústav telekomunikací

Bakalářská práce

bakalářský studijní obor
Teleinformatika

Student:	Adam Nimrichter	***ID:***	158203
Ročník:	3	***Akademický rok:***	2014/2015

NÁZEV TÉMATU:

Procedurální programování v databázi

POKYNY PRO VYPRACOVÁNÍ:

Nastudujte si problematiku strojového učení a programování v databázi pomocí procedurálního jazyka. Implementujte algoritmus dopředného výběru příznaků s použitím knihovny MADlib a ověřte jeho funkčnost.

DOPORUČENÁ LITERATURA:

[1] http://www.postgresql.org/docs/8.0/static/plpgsql.html
[2] ttp://www.postgresql.org/docs/8.4/static/external-pl.html
[3]http://www.joeconway.com/presentations/function_basics.pdf

Termín zadání:	9.2.2015	***Termín odevzdání:***	2.6.2015

Vedoucí práce: Ing. Václav Uher
Konzultanti bakalářské práce:

doc. Ing. Jiří Mišurec, CSc.
Předseda oborové rady

UPOZORNĚNÍ:

ABSTRAKT

Práce se zabývá ověřením konceptu provádět výpočty přímo v databázi. Popisuje databázi PostgreSQL, její vlastnosti a procedurální jazyk PL/pgSQL. Dále se věnuje metodám strojového učení, návrhu algoritmu pro dopředný výběr příznaků a ověřením jeho funkčnosti.

Hojně využívá rozšiřující knihovny analytických funkcí MADlib, která poskytuje implementace matematických, statistických a strojních učebních metod pro strukturovaná a nestrukturovaná data.

KLÍČOVÁ SLOVA

Databáze, PostgreSQL, Strojové učení, Dopředný výběr příznaků, MADlib.

ABSTRACT

Thesis deals with verification of concept of performing calculations inside database. Describes PostgreSQL database, its features and procedural language PL/pgSQL. Also focuses on machine learning methods, implementation of forward selection algorithm and verification of his functionality.

Frequently used tool is MADlib, which is an open-source library of scalable in-database algorithms for machine learning, statistics and other analytic tasks.

Nimrichter, A. *Procedurální programování v databázi*. Brno: Vysoké učení technické v Brně, Fakulta elektrotechniky a komunikačních technologií. Ústav telekomunikací, 2015. 54 s., 0 s. příloh. Bakalářská práce. Vedoucí práce: Ing. Václav Uher

SIX research centre
sensor, information and communication systems

Faculty of Electrical Engineering
and Communication

Brno University of Technology
Technicka 12, CZ-61600 Brno, Czechia

http://www.six.feec.vutbr.cz

Výzkum popsaný v této bakalářské práci byl realizovaný v laboratořích podpořených projektem Centrum senzorických, informačních a komunikačních systémů (SIX); registrační číslo CZ.1.05/2.1.00/03.0072, operačního programu Výzkum a vývoj pro inovace.

MINISTERSTVO ŠKOLSTVÍ, MLÁDEŽE A TĚLOVÝCHOVY

EVROPSKÁ UNIE
EVROPSKÝ FOND PRO REGIONÁLNÍ ROZVOJ
INVESTICE DO VAŠÍ BUDOUCNOSTI

2007–13
OP Výzkum a vývoj pro inovace

PROHLÁŠENÍ

Prohlašuji, že svou semestrální práci na téma Procedurální programování v databázi jsem vypracoval samostatně pod vedením vedoucího semestrální práce a s použitím odborné literatury a dalších informačních zdrojů, které jsou všechny citovány v práci a uvedeny v seznamu literatury na konci práce.

Jako autor uvedené semestrální práce dále prohlašuji, že v souvislosti s vytvořením této semestrální práce jsem neporušil autorská práva třetích osob, zejména jsem nezasáhl nedovoleným způsobem do cizích autorských práv osobnostních a/nebo majetkových a jsem si plně vědom následků porušení ustanovení § 11 a následujících zákona č. 121/2000 Sb., o právu autorském, o právech souvisejících s právem autorským a o změně některých zákonů (autorský zákon), ve znění pozdějších předpisů, včetně možných trestněprávních důsledků vyplývajících z ustanovení části druhé, hlavy VI. díl 4 Trestního zákoníku č. 40/2009 Sb.

V Brně dne ....

(podpis autora)

PODĚKOVÁNÍ

Děkuji vedoucímu bakalářské práce ing. Václavu Uhrovi za účinnou metodickou, pedagogickou a odbornou pomoc a další cenné rady při zpracování mé bakalářské práce.

V Brně dne ....

(podpis autora)

OBSAH

SEZNAM OBRÁZKŮ

SEZNAM TABULEK

SEZNAM GRAFŮ

ÚVOD

V dnešní moderní době, kdy je téměř vše kolem nás zaznamenáváno a řízeno pomocí počítačů, vzniká velké množství dat a záznamů, které je výhodné nějakým způsobem ukládat, třídit a dále zpracovávat. Místu, kde taková data shromažďujeme, se říká databáze. Databáze je vlastně novodobou formou kartotéky, která uchovává své záznamy v podobě počítačových dat.

Jedním z mnoha využití takto uložených dat je i možnost naučit systém předpovídat určité události, které jsou založeny na základě poznatků získaných z historických dat. Což je v praxi ohromně ekonomicky i časově výhodné. Technikám, které umožňují počítačovému systému učit se a předpovídat události, říkáme strojové učení.

Přínosem této práce je rozšíření knihovny o funkci dopředného výběru, která se využívá právě ke zpřesnění modelů pro strojové učení.

1 TEORETICKÝ ÚVOD

Bakalářská práce se zabývá databází PostgreSQL, jejím databázovým serverem a možnostmi provádění výpočtů přímo uvnitř databáze.

Práce je členěna na dvě části, část teoretickou a praktickou. Teoretická část popisuje princip relačních databází, PostgreSQL, procedurální programování v databázi a metody strojového učení.

Praktická část se naproti tomu zabývá instalací a nastavením databáze i jejího serveru, instalací knihovny MADlib, příklady algoritmů strojového učení, implementací dopředného výběru příznaků a ověřením jeho funkčnosti.

1.1 Databáze

„*Databázi si můžeme představit jako kartotéku, kde každý záznam je jedinečný a organizovaně nalezitelný lístek*.“ Tento pojem vznikl sloučením dvou anglických slov data a base (česky báze dat). Představuje prostor pro ukládání organizovaných a strukturovaných dat. Velmi často se v praxi používá pro označení celého databázového systému. [1]

1.1.1 Databázový systém

Vznikne spojením dvou termínů – databáze a Systému řízení báze dat (*SŘBD*), anglicky Database Management Systems (*DBMS*). Kde databáze představuje strukturovaně uložená vzájemně související data a systém řízení báze dat reprezentuje sadu nástrojů, aplikací a knihoven umožňujících jejich správu.

Tento pojem tak nepředstavuje pouze samotná data, ale také nástroje pro práci s nimi. [2] [3]

Každý databázový systém musí obsahovat nástroje pro:

- Vytvoření, vyhledání, aktualizaci a rušení uživatelských dat.
- Definici struktury dat.
- Zajištění integrity dat.
- Zajištění fyzické a logické nezávislosti dat.
- Mechanizmy pro zálohování dat.
- Bezpečnostní mechanismy pro zabránění neoprávněného přístupu k datům.
- Podpora dotazovacího jazyka.
- Správu současného přístupu více uživatelů k datům.

1.2 Jazyk SQL

Firma IBM v polovině 70. let minulého století provedla výzkum, který se zabýval možnostmi a využitím relačních databází. Cílem projektu bylo vytvořit sadu příkazů, jimiž by se relační databáze řídila. Hlavním úkolem pak bylo, aby se tyto příkazy co nejvíce podobaly běžnému jazyku (angličtině). Výsledkem projektu byl vznik nového jazyka SE-QUEL (Structured English Query Language).

Velkého potenciálu relačního přístupu si také začaly uvědomovat i jiné firmy. Například firma Oracle Corporation vypustila do světa svůj první relačně databázový systém pojmenovaný Oracle již koncem 70. let. Téměř ve stejnou dobu také vznikaly databázové systémy i jiných firem jako například Informix, SyBase, nebo Progress. Tyto systémy byly vybudovány na základech různých verzí jazyka SEQUEL, jenž byl později přejmenován na SQL (Structured Query Language).

Se stále rostoucím významem relačních databází přišla i nutnost jazyk SQL standardizovat. V roce 1986 tak Americký standardizační institut (ANSI) vytvořil standard jazyka SQL označovaný jako SQL86 podle roku, ve kterém byl přijat. Postupem času se však ukázalo, že původní standard nevyhovuje současným potřebám a má také spoustu zásadních nedostatků týkající se integrity dat. Roku 1992 byl proto schválen nový, vylepšený standard nazvaný SQL92 (SQL2). Pro databáze s objektovými prvky byl později vytvořen, doposud nejnovější, standard SQL99 (SQL3). [2]

Jazyk SQL dělíme do čtyř základních skupin podle toho, k čemu slouží. [3]

Jazyk pro dotazování (Data Query Language, DQL)

Jazyk pro dotazování je reprezentován příkazem `SELECT`. Tento příkaz umožňuje nalézt a zobrazit požadovanou množinu záznamů z databáze. Výsledek dotazu je vracen jako strukturovaná množina dat ve formě tabulky. Širší forma příkazu `SELECT` je složena z dalších příkazů označovaných jako klauzule (`FROM`, `WHERE`, `GROUP BY`, `HAVING`), což umožňuje vytvářet širokou škálu nejrůznějších dotazů.

Jazyk pro manipulaci s daty (Data Manipulation Language, DML)

Jazyk pro manipulaci s daty slouží pro úpravu dat v databázi. Nejčastěji s ním pracují koncoví uživatelé, či programátoři databázových aplikací. Umožňuje vkládat, upravovat a odstraňovat data z databáze. Zástupnými příkazy jsou `INSERT`, `MERGE`, `DELETE` a `UPDATE`.

Jazyk pro definici dat (Data Definition Language, DDL)

Jazyk pro definici dat zahrnuje příkazy vytvářející struktury databáze, jimiž jsou například tabulky, indexy, pohledy a další objekty. Vytvořené struktury lze také upravovat, doplňovat a mazat. Do této skupiny patří příkazy jako `CREATE`, `ALTER` a `DROP`.

Jazyk pro řízení dat (Data Control Language, DCL)

Jazyk pro řízení dat je využíván pro řízení provozu a údržbu databáze. Pomocí příkazů řadících se do této skupiny lze nastavovat přístupová práva a systémová oprávnění, což je značně výhodné pracuje-li například s jednou databází více lidí. Patří sem příkazy `GRANT`, `REVOKE`, `COMMIT` a `ROLLBACK`.

1.3 Databáze PostgreSQL

PostgreSQL (zkráceně Postgres) je relační databázový systém s otevřeným zdrojovým kódem (open source) a dobrou pověstí pro svou spolehlivost a bezpečnost. Lze ji nainstalovat a běžně používat na všech rozšířených operačních systémech včetně Linuxu, UNIXů (AIX, BSD, HP-UX, SGI-IRIX, Mac OS X, Solaris, Tru64) a Windows. [6]

1.3.1 Historie PostgreSQL

Na začátku všeho byl systém INGRES vyvíjený na kalifornské univerzitě v Berkley, jehož cílem bylo navrhnout databázi pro správu demografických dat a jejich grafickou prezentaci. Na úspěšný projekt navázal prof. Michael Stonebraker, který jej dále vyvíjel, ovšem už pod názvem Postgres. Jelikož byl Postgres založen na vlastním dotazovacím jazyce QUEL, byl upraven pro podporu SQL a pojmenován PostgreSQL95.

S rostoucí popularizací SQL se Postgres dostal mezi hlavní databázové systémy. Později se z Postgres stal projekt s otevřeným kódem a byl přejmenován na PostgreSQL. První vydaná verze PostgreSQL byla uvolněna roku 1996 jako program s otevřeným kódem a nesla označení 6.0. V současnosti (2015) se PostgreSQL nachází ve verzi 9.3.5. [4] [5]

1.3.2 Vlastnosti databáze PostgreSQL

Charakteristickou vlastností databáze PostgreSQL je, že svým uživatelům umožňuje její volné rozšiřování například o vlastní datové typy, operátory, funkce, agregační funkce a dokonce až o celé procedurální jazyky. I díky tomu vznikla celá řada různých rozšíření, jako třeba nadstavba PostGIS pro podporu geografických informačních systémů či Slony-I pro asynchronní replikaci databází.

PostgreSQL má plnou podporu pro:

- Foreign keys (cizí klíče)
- Joins (spojování tabulek)
- Triggers (spouště)
- Views (pohledy)
- Vlastní datové typy
- Agregační funkce
- Stored procedures (uložené procedury)
- Ukládaní binárních objektů (obrázky, zvuky, videa, …)

Obsahuje většinu datových typů standardů SQL92 a SQL99, například:
- INTEGER
- NUMERIC
- BOOLEAN
- CHAR
- VARCHAR
- DATE
- INTERVAL
- TIMESTAMP

Plně také splňuje podmínky *ACID* (Atomic, Consistent, Isolated, Durable), což je obecně uznávaný seznam požadavků na bezpečný transakční systém.

- Atomic V rámci transakce jsou provedeny všechny změny nebo žádná.
- Consistent Převedení dat z jednoho konzistentního stavu do druhého.
- Isolated Transakce není ovlivněna souběžnými transakcemi.
- Durable Pokud je transakce potvrzena, změny dat jsou trvalé.

PostgreSQL má plně implementovánu podporu pro národní znakové sady, vícebajtové znakové kódování, Unicode a zohledňuje i národní specifikace pro formátování, řazení a velikost znaků (case-sensitivity). Je vysoce rozšiřitelná jak z hlediska dat, které může spravovat, tak i v počtu uživatelů, kteří k datům přistupují. Některé obecné limity PostgreSQL jsou uvedeny v tabulce Tabulka 1. [6] [7] [8]

Limit	Hodnota
Maximální velikost Databáze	Neomezená
Maximální velikost tabulky	32 TB
Maximální velikost řádku	1,6 TB
Maximální velikost pole	1 GB
Maximální počet řádků na tabulku	Neomezen
Maximální počet sloupců na tabulku	250 až 1600 v závislosti na typu
Maximální počet indexů na tabulku	Neomezen

Tabulka 1: Limity PostgreSQL

1.3.3 Výhody a nevýhody PostgreSQL

Velkou a podstatnou výhodou PostgreSQL je jeho tolerantní licence s otevřeným zdrojovým kódem (*BSD*), umožňující jakkoliv upravovat a rozšiřovat databáze dokonce i pro komerční účely. Další značnou výhodou je kvalitní dokumentace celého projektu, umožňující dobrou orientaci v problematice.

Za nevýhodu lze označit snad jen malou rozšiřitelnost mezi běžnými uživateli. [6]

1.4 Procedurální jazyk PL/pgSQL

PostgreSQL umožňuje vytvářet a používat tzv. uložené procedury (Stored procedure). Uložená procedura je kód, který je uložen a spouštěn SQL serverem. Tento kód lze v PostgreSQL psát v některém z následujících programovacích jazyků: SQL, Perl, Python, TCL a PL/pgSQL. Nejpoužívanějším z nich je právě PL/pgSQL (Procedural Language/PostgreSQL), jenž byl speciálně navržen pouze pro psaní uložených procedur RDBMS PostgreSQL.

Jedná se o jednoduchý plnohodnotný programovací jazyk, který oproti obyčejnému SQL umožňuje například používání smyčky a dalších řídících struktur. Jeho podstatnou výhodou je automatické kešování prováděcích plánů. Nelze jej však použít pro návrh vlastního datového typu. [6]

1.5 Strojové učení

„*Strojové učení se zabývá přesně tím, co říká jeho název, tedy vývojem umělé inteligence, která se dokáže učit z dat a přizpůsobovat se změnám. Umělou inteligenci představují na počítači běžící algoritmy, které se jsou schopné rozhodovat nebo dokonce předpovídat budoucí vývoj na základě vstupních dat.*

Výsledky strojového učení využíváme nevědomky každý den. Je mozkem internetových vyhledávačů, antispamových filtrů chránících mailové schránky a stojí za většinou operací na světových finančních trzích. Uplatnění nacházejí všude tam, kde je potřeba přijímat rychlá, na datech založená rozhodnutí nebo předpovídat pravděpodobný budoucí vývoj.“ [9]

1.5.1 MacQueenova metoda (*k*-means)

Algoritmus *k*-means řadíme do skupiny nehierarchických metod shlukování. Jeho cílem je zařadit každý bod do shluku, k jehož středu je nejblíže a dosáhnout co nejmenších rozdílů uvnitř shluku. Středy jsou dokola přepočítávány jako aritmetický průměr všech bodů k nim přiřazených.

Jednotlivé kroky algoritmu:

1. Zadání požadovaného počtu *k* shluků.
2. Zařazení každého bodu do shluku, jehož střed je nejblíže.
3. Přepočet pozice středů, jako aritmetický průměr všech přiřazených bodů.
4. Pokud nedošlo ke změně žádného shluku skončit, jinak návrat ke kroku 2.

Podstatnou výhodou toho algoritmu je jeho jednoduchost a rychlost i při použití na velkém množství dat. Za nevýhodu lze označit to, že výběr umístění počátečních středů způsobí ovlivnění výsledku. Tento algoritmus tak není vhodné používat na datech, kde jsou očekávány překrývající se shluky. Existuje několik verzí tohoto algoritmu z nichž nejčastěji používanými jsou: Forgy, MacQueen, Kaufman a random. [13]

1.5.2 Křížová validace

Křížová validace je jedna z metod pro ověření výsledné kvality modelu. Využívá se pro objektivní odhad přesnosti klasifikátoru na neznámých datech. Její princip spočívá v rozdělení dat na dvě části. První část tvoří data pro trénink modelu a druhá část slouží pro následné testování přesnosti klasifikace. Jedná se tedy o metodu, s jejíž pomocí získáme nezaujatý odhad predikce chyby, který je důležitý jednak pro předpověď budoucí predikční přesnosti, tak i pro výběr vhodného klasifikátoru ze setu. [10]

1.5.3 Druhy křížové validace

Prvním krokem křížové validace je rozdělení data setu na několik stejně strukturovaných subsetů. Ve druhém kroku, je z těchto subsetů vybrán jeden pro natrénování algoritmu (tzv. training set). Na zbývajících datech potom proběhne ověření přesnosti (tzv. testing set). Aby se zvýšila objektivnost výsledku, provádí se hned několik opakování, přičemž je v každém cyklu zvolen vždy jiný subset pro natrénování modelu.

Existuje hned několik metod křížové validace: [10]

_K_-fold cross validation (Křížová validace o _k_-přeloženích)

Původní data set je u této metody náhodně rozdělen do k subsetů z nichž vždy pouze jeden je vybrán pro testování a zbytek k-1 je použit pro natrénování modelu. Proces je opakován k krát (počet přeložení), přičemž každý ze subsetů je vybrán pro testování právě jednou. Výhoda této metody spočívá v použití všech dat jak pro trénink, tak i pro testování. Často se používá křížová validace o 10 přeloženích (tzv. 10-fold cross validation).

Repeated random subsampling (Opakující se náhodné vzorkování)

Data jsou u této metody náhodně rozdělena do jednoho ze dvou subsetů, kdy jeden z nich slouží pro trénink a druhý pro testování. Velikost jednotlivých subsetů je stanovena uživatelem. Pro zvýšení objektivnosti metody se náhodné rozřazení dat do jednotlivých skupin může i několikrát opakovat. Výhoda metody je v nezávislosti poměru dat pro trénink/test na počtu iterací (foldů). Nevýhoda pak spočívá v možnosti, kdy jsou některá data vybrána vícekrát a některá ani jednou. Tato metoda bývá také někdy označována jako metoda Monte Carlo.

Leave-one-out cross validation (Křížová validace vyber jeden)

Z celkového množství dat je vybrán pouze jeden vzorek pro testování a zbytek je určen pro trénink. Metoda je časově a výpočetně náročná.

1.5.4 Rozhodovací stromy

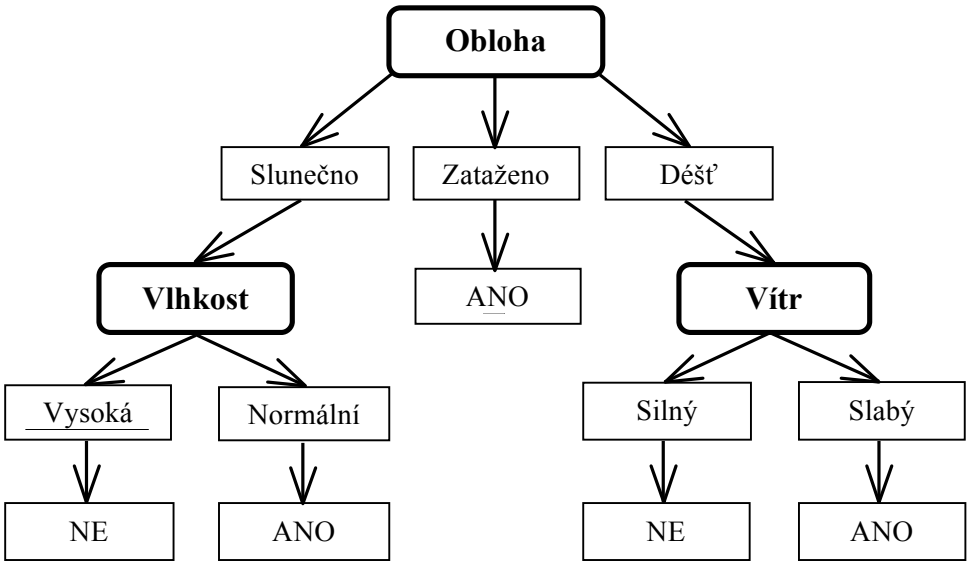
Jedná se o jeden z nejrozšířenějších a hojně využívaných nástrojů strojového učení. „*Rozhodovací stromy umožňují zobrazit logický vývoj časově na sebe navazujících alternativních rozhodnutí a náhodných situací. Jejich cílem je stanovit optimální strategii posloupnosti rozhodnutí, která vede k co nejpřesnější očekávané hodnotě zvoleného výběrového kritéria.*“ [11]

Primárně jsou tedy určeny pro úlohy klasifikace a predikce. Podstata klasifikace spočívá v přidání, ke všem již existujícím datovým záznamům (klientům, službám, poplatkům, ...), jednoho klíčového atributu který je zpravidla v binární (0, 1), nebo booleově podobě (ano, ne) - tedy například zda od nás zákazník odešel nebo zda je nadále naším klientem. Na základě ostatních zákaznických atributů (např. věk, stav konta, atd.) se poté strom naučí, které jejich kombinace vedou k tomu, zda od nás zákazník odešel nebo ne.

Použijeme-li pro naučení stromu historická data, můžeme poté s určitou přesností předpovídat, kteří z našich stávajících klientů se s nejvyšší pravděpodobností chystají odejít a pokusit se tak tomu včas zabránit (telefonátem, výhodnou nabídkou, …). [12]

Vlastnosti:

- Jednoduché, rychlé.
- Snadná vizualizace a interpretace výsledků.
- Nevyžadují žádnou speciální úpravu vstupních dat jako je např. převod na vektory.
- Zpracují jak kategorické, tak i číselné proměnné.
- Lze je kombinovat s ostatními rozhodovacími technikami.



Obrázek 1: Příklad vizualizace rozhodovacího stromu.

1.6 Dopředný výběr

Protože nadbytečné vstupy zpomalují výpočet a mohou také způsobovat nepřesnosti, je cílem dopředného výběru snížit jejich počet na minimum, ovšem za předpokladu zachování maximální přesnosti.

Jedná se o postup, kterým je vybrána podmnožina co nejvhodnějších příznaků z celé množiny vstupů. Je založen na principu výběru všech možných kombinací vstupů a jejich ohodnocení pro následné použití. To znamená, že každá ze vzniklých kombinací vstupů je předložena určitému algoritmu pro natrénování. Výsledek je pak vyhodnocen z hlediska dosažené kvality.

V prvním kole proběhne ohodnocení přesnosti klasifikace po jednom atributu a je vybrán ten, s nejvyšší přesností. Ve druhém kole se potom počítají přesnosti dvou atributů, přičemž jeden z nich je vybraným atributem z prvního kola v kombinaci se zbytkem atributů. Celý proces se pak opakuje dokud roste přesnost. [16]

1.7 MADlib

MADlib je klíčovou a hojně využívanou knihovnou celé bakalářské práce. Má kvalitně zpracovanou dokumentaci a na jejím vývoji se podílí početná skupina nezávislých vývojářů. Díky své otevřené implementaci udržuje aktivní vazby s probíhajícím akademickým výzkumem. Je také plně kompatibilní s vybranou databází Postgres, což byl jeden z mála důvodů proč byla vybrána pro použití v práci.

Jedná se o jedinou dostupnou knihovnu s otevřeným zdrojovým kódem pro rozšíření databázových analytických funkcí. Určena je pro databáze Postgres, Pivotal Greenplum a HAWQ. Její hlavní myšlenkou, je provádět operace s daty přímo v databázi a nepřesouvat je zbytečně přes různá výpočetní prostředí. Poskytuje paralelní implementace matematických, statistických a strojních učebních metod pro strukturovaná a nestrukturovaná data. [14]

Mezi hlavní funkce knihovny patří:

- Klasifikační metody
- Regresní metody
- Shlukové metody
- Titulkové modelování
- Asociační vytěžovací metody
- Deskriptivní statistika
- Ověření kvality

1.8 MacPorts

Při pokusu o přímou instalaci databáze se vyskytly problémy spojené s kompatibilitou jazyka PL/Python. Proto byl vybrán tento nástroj, pomocí kterého byly nainstalovány potřebné součásti pro řešení projektu.

MacPorts je snadno použitelný systém pro kompilaci, instalaci a správu programů s otevřeným kódem. Díky němu je možné stáhnout, kompilovat a instalovat aplikace pouze jediným příkazem. Je vyvinut pro OS X, nicméně je navrhnutý tak, aby mohl pracovat i na ostatních Unixových systémech. Aktuálně se nalézá ve verzi 2.3.3. [15]

1.8.1 Výhody použití MacPorts

- Čistá a snadná správa instalovaných programů.
- Přístup k rozsáhlé databázi aktuálního softwaru s otevřeným kódem skrze sjednocené rozhraní.
- Automatická instalace všech vyžadovaných podpůrných programů.
- Snadné, automatizované aktualizace zmíněného software.
- Umisťuje software do soukromého sandboxu, který jej udržuje od promíchání s ostatními programy a operačním systémem.
- Umožňuje vytvářet předkompilované binární instalátory přenesených aplikací a rychle nahrávat software na vzdálené počítače bez nutnosti kompilace zdrojového kódu.
- Lepší výkon aplikací

1.8.2 Možné alternativy MacPorts:

- Homebrew
- Fink
- Rudix
- Maclibre

2 PRAKTICKÁ ČÁST

V následujícím textu je probrána praktická část problematiky bakalářské práce.

Skládá se z popisu instalace databáze PostgreSQL a jejího databázového serveru, implementace knihovny MADlib, zprovoznění celého systému, příkladů algoritmů strojového učení a vývoje funkce pro dopředný výběr atributů a jeho testování.

2.1 Instalace MacPorts

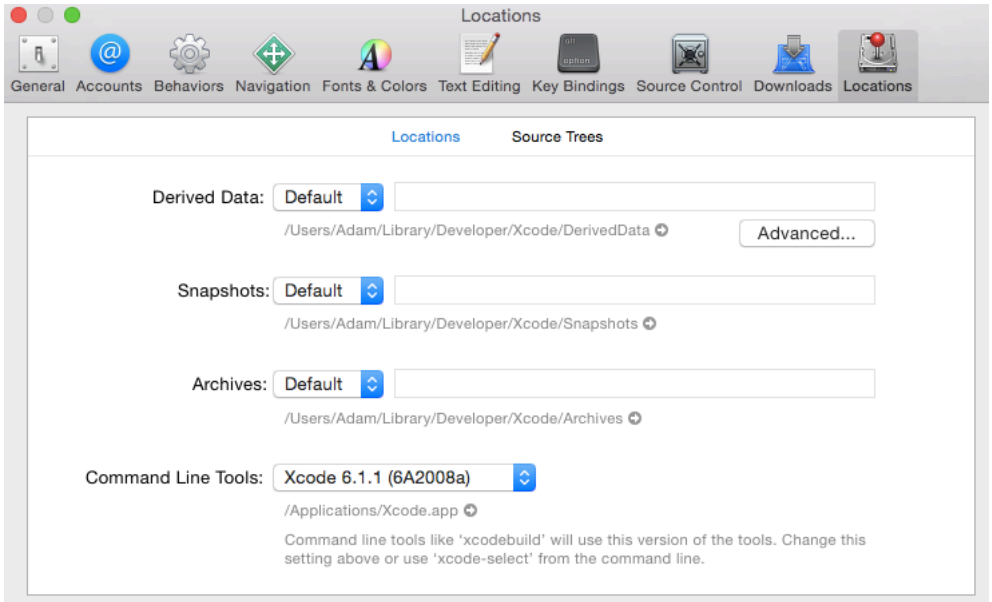
Aby mohl být MacPorts nainstalován, musela se nejdříve provést instalace programu Xcode a dalších vývojářských nástrojů potřebných pro jeho správnou funkci.

2.1.1 Instalace Xcode a vývojářských nástrojů

Xcode je balíček poskytovaný společností Apple. Obsahuje překladače, knihovny a další nástroje potřebné k vývoji aplikací pro OS X. Je volně přístupný ke stáhnutí z vývojářských stránek společnosti Apple.

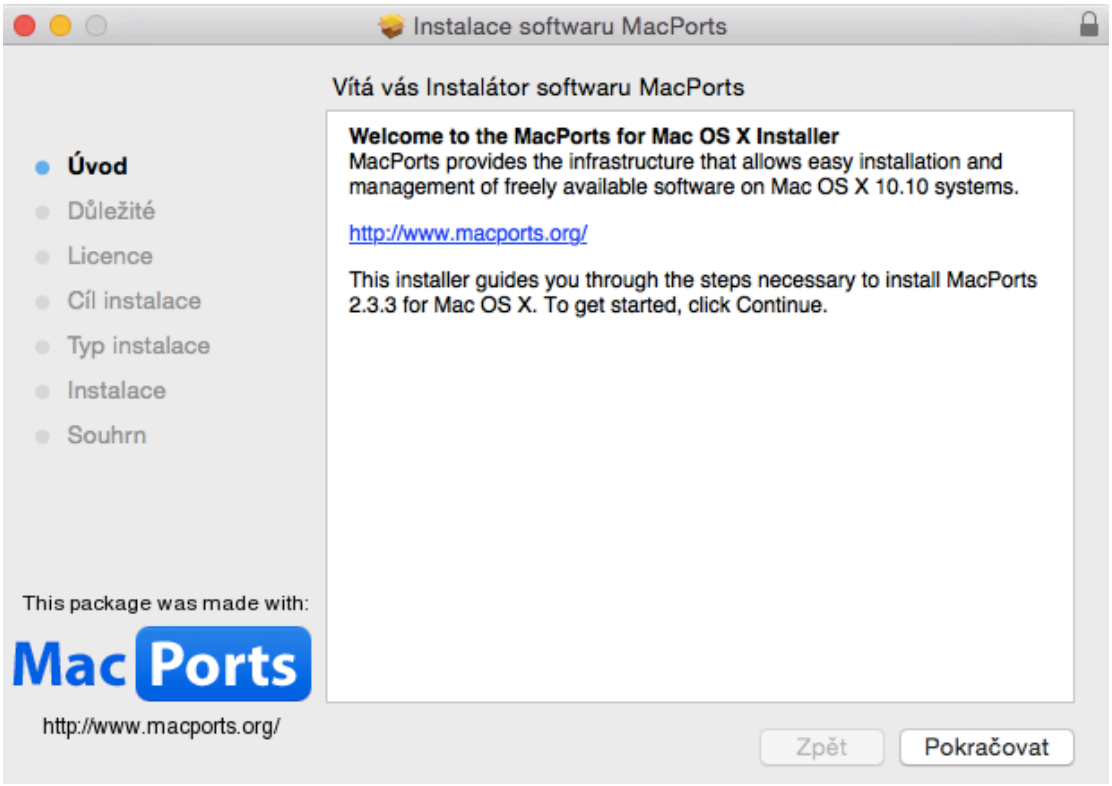
Xcode od verze 4.3 a výš nepodporuje automatickou instalaci vývojářských nástrojů příkazového řádku, které ale program MacPorts pro svou práci vyžaduje. Proto je nutné tyto nástroje stáhnout a nainstalovat manuálně.

Do terminálu zadáme příkaz: `xcode-select --install` dojde ke spuštění instalace vývojářských nástrojů. Poté je třeba, v již nahraném programu Xcode přejít do nastavení, v menu stažených souborů (downloads) tyto nástroje vybrat a tlačítkem instalovat potvrdit jejich implementaci. Nakonec v záložce locations změnit políčko nástrojů příkazového řádku na Xcode 6.1.



Obrázek 2: Nastavení vývojářských nástrojů v programu Xcode

Jakmile je Xcode včetně podpory vývojářských nástrojů úspěšně nainstalován může se přistoupit k instalaci samotného MacPorts. Instalační balíček je volně dostupný ke stažení na oficiálních stránkách aplikace.



Obrázek 3: Instalace MacPorts

Na závěr je doporučeno zadat do terminálu (příkazového řádku) příkaz `sudo port selfupdate` čímž se zajistí, že se právě nainstalovaný balíčkovací systém aktualizuje na nejnovější verzi (v současnosti 6.1). Tato verze umožňuje vyvíjet aplikace na nejnovější verze operačních systému Apple OS X Yosemite a iOS 8.

2.2 Instalace PostgreSQL

Díky předešlým krokům a již nahranému balíčkovému systému MacPorts je instalace databáze PostgreSQL velice snadná. Do terminálu stačí zadat příkaz `sudo port install postgresql93 +python postgresql-server +python` a tím dojde k nainstalování databáze a databázového serveru PostgreSQL do počítače.

Nakonec celého instalačního procesu ještě stáhneme klienta pgAdmin3, který slouží pro lepší orientaci v databázi a také i jako její správce.

2.2.1 Nastavení databáze:

Předtím než se databázový server poprvé spustí, je nejprve nutné vyhradit mu úložný prostor v paměti disku. V tomto prostoru se nachází soubor databází, které jsou řízeny

jedinou instancí běžícího databázového serveru. V souborových systémech je tento prostor reprezentován adresářem, v němž jsou uložena všechna data. Pro správné nastavení databáze je nutné provést řadu kroků popsaných níže. Příkazy provádíme pomocí příkazového řádku.

Prvním krokem je tedy vytvoření adresáře, do kterého později provedeme inicializaci.

```
sudo mkdir -p /opt/local/var/db/postgresql93/defaultdb
sudo mkdir -p /opt/local/var/log/postgresql
```

Jelikož datový adresář obsahuje všechna data uložená v databázi, je nezbytné zabezpečit jej proti neoprávněnému přístupu. Inicializační příkaz `initdb` lze proto také spustit pouze a jenom jako přihlášený PostgreSQL uživatel. Z tohoto důvodu je nutné změnit vlastníka i skupinu vlastníků již dříve vytvořeného adresáře. Změnu přístupových práv adresáře lze provést následujícími příkazy:

```
sudo chown postgres:Postgres/opt/local/var/db/postgresql93/defaultdb
sudo chown postgres:postgres /opt/local/var/log/postgresql
```

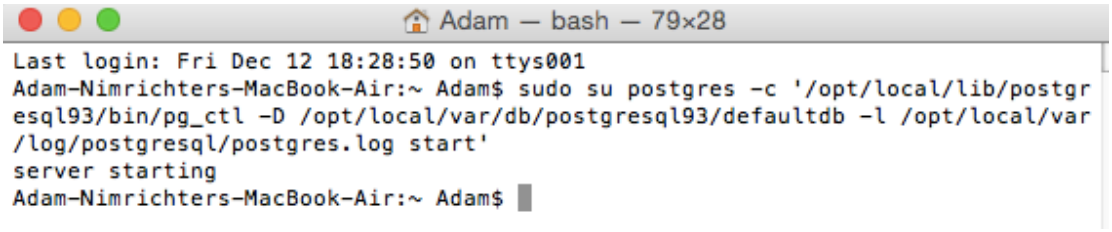
Jakmile jsou změněna přístupová práva, je nutné se příkazem `su` přihlásit jako uživatel `postgres` a spustit inicializační proces. Inicializační příkaz `initdb` musí být výhradně vykonán stejným uživatelem, jenž bude zároveň i vlastníkem procesu vykonávajícího chod serveru. Je to z toho důvodu, že server potřebuje mít přístup k souborům a adresářům, vytvořených při inicializaci.

```
sudo su postgres -c '/opt/local/lib/postgresql93/bin/initdb -D
/opt/local/var/db/postgresql93/defaultdb'
```

`Initdb` provede inicializaci a příznak -D představuje cestu pro požadované umístění v souborovém systému. Po vykonání příkazu se v prostoru souboru databází nachází výchozí databáze pojmenovaná `Postgres`.

Spuštění databázového serveru:

```
sudo su postgres -c '/opt/local/lib/postgresql93/bin/pg_ctl -D
/opt/local/var/db/postgresql93/defaultdb -l
/opt/local/var/log/postgresql/postgres.log start'
```



Obrázek 4: Spuštění databázového serveru

Nastavení cesty proměnného prostředí:

Proměnná PATH obsahuje seznam cest (adresářů), které se prohledávají v případě, že se má spustit nějaký nástroj, ke kterému nebyla zadána absolutní nebo relativní cesta.

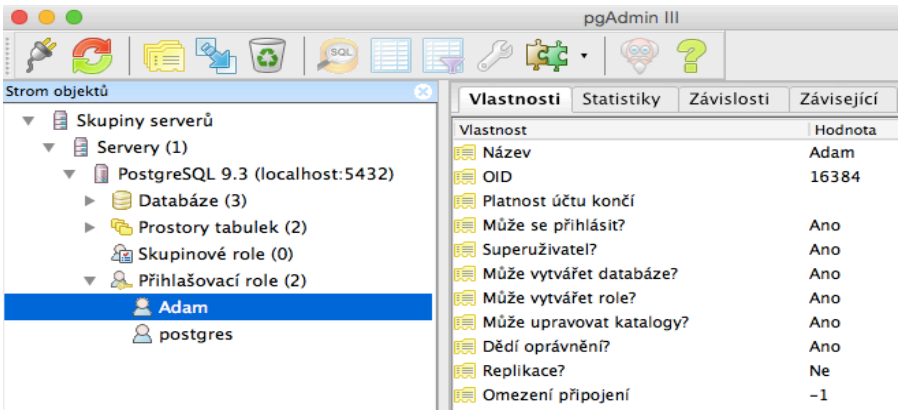
```
export PATH=/opt/local/lib/postgresql93/bin:$PATH
```

Uvedený příkaz nahradí obsah proměnné PATH hodnotou této proměnné doplněnou o adresář `/opt/local/lib/postgresql93/bin` Jednotlivé adresáře se oddělují dvojtečkou a proměnná PATH se prohledává zleva doprava.

Vytvoření uživatelského účtu:

vytvořit nového PostgreSQL uživatele mohou pouze superuživatelé, nebo také uživatelé s přidělenými příslušnými právy pro správu uživatelů. Příkaz `createuser` musí být tedy vykonán nějakým superuživatelem, nebo uživatelem s právem `createrole`. Vytvoření účtu superuživatele je možné pouze jiným superuživatelem. Superuživatelé mají neomezený garantovaný přístup do celé databáze, proto by se k přidělování těchto přístupových práv mělo přistupovat zřídka a s rozvahou.

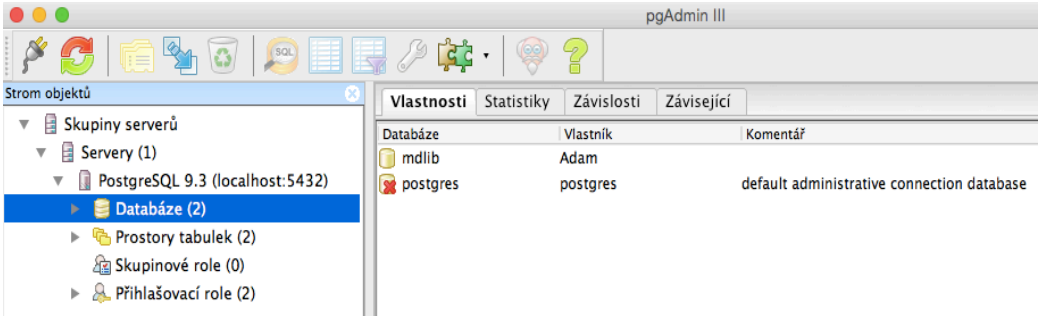
```
createuser --superuser Adam -U Postgres
```



Obrázek 5: Vytvoření uživatelského účtu

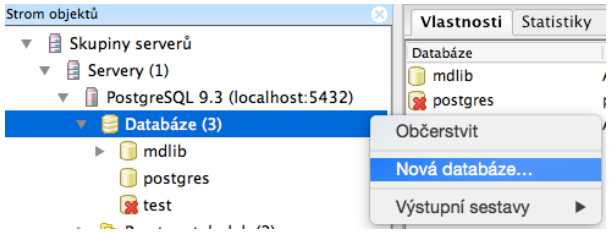
Vytvoření databáze:

```
createdb mdlib   -- vytvoří databázi pojmenovanou mdlib
```



Obrázek 6: Vytvoření databáze

Alternativní cestou jak založit novou databázi je vytvořit ji v programu pgAdmin3.



Obrázek 7: Vytvoření databáze v programu pgAdmin3

Přístup do databáze:

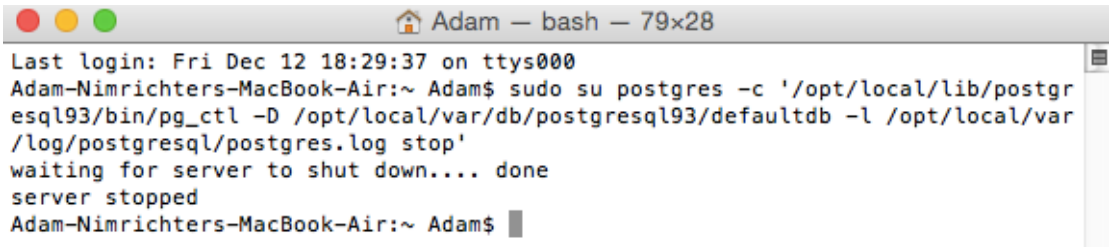
```
psql mdlib
```

Opuštění databáze:

```
mdlib=> \q
```

Vypnutí databázového serveru

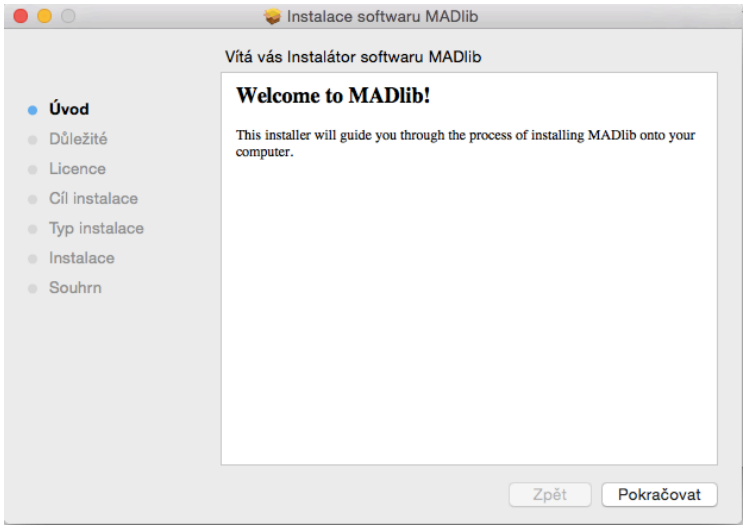
```
sudo su postgres -c '/opt/local/lib/postgresql93/bin/pg_ctl -D
/opt/local/var/db/postgresql93/defaultdb -l
/opt/local/var/log/postgresql/postgres.log stop'
```



Obrázek 8: Vypnutí databázového serveru

2.3 Instalace knihovny MADlib do databáze:

Prvním krokem je stažení instalačního balíčku pro daný operační systém (v našem případě Mac OS X) a jeho následná instalace.



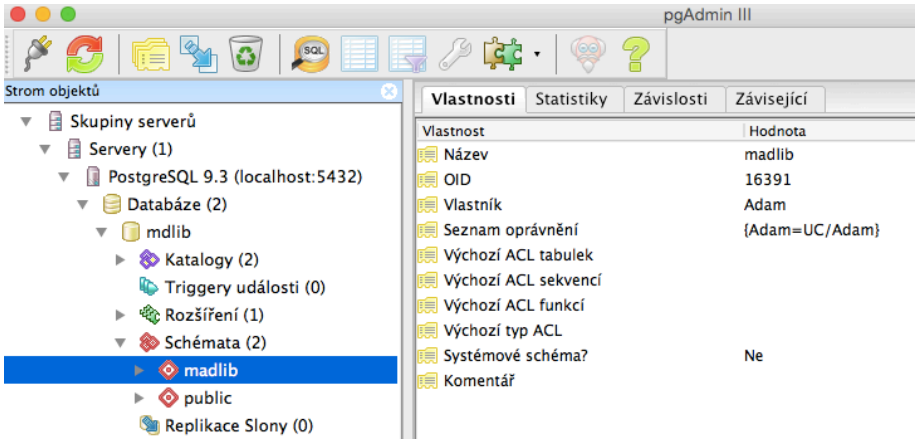
Obrázek 9: Instalace knihovny MADlib

Dalším krokem je nainstalovanou knihovnu nahrát do vytvořené databáze:

```
/usr/local/madlib/bin/madpack -p postgres -c
Adam@localhost:5432/mdlib install
```

Úspěšný import knihovny je doprovázen následujícím hlášením:

```
madpack.py : INFO : MADlib 1.6 installed successfully in MADlib
schema.
```



Obrázek 10: Implementace knihovny MADlib do databáze

2.4 Vzorový příklad *k*-means:

2.4.1 Zdrojová data

Nejprve ze všeho je třeba v databázi vytvořit novou tabulku se dvěma sloupci. Jeden pro jednoznačnou identifikaci každého řádku tabulky, a druhý pro souřadnice vstupních bodů. Jako datový typ byl pro první sloupec zvolen `INTEGER` a pro souřadnice vstupních bodů byl zvolen datový typ `DOUBLE PRECISION`.

```
CREATE TABLE public.km_sample(pid int, points double precision[]);
COPY km_sample (pid, points) FROM stdin DELIMITER '|';

1|{14.23,1.71,2.43,15.6,127,2.8,3.0600,0.2800,2.29,5.64,1.04,3.92,1065}
2|{13.2,1.78,2.14,11.2,1,2.65,2.76,0.26,1.28,4.38,1.05,3.49,1050}
3|{13.16,2.36,2.67,18.6,101,2.8,3.24,0.3,2.81,5.6799,1.03,3.17,1185}
4|{14.37,1.95,2.5,16.8,113,3.85,3.49,0.24,2.18,7.8,0.86,3.45,1480}
5|{13.24,2.59,2.87,21,118,2.8,2.69,0.39,1.82,4.32,1.04,2.93,735}
6|{14.2,1.76,2.45,15.2,112,3.27,3.39,0.34,1.97,6.75,1.05,2.85,1450}
7|{14.39,1.87,2.45,14.6,96,2.5,2.52,0.3,1.98,5.25,1.02,3.58,1290}
8|{14.06,2.15,2.61,17.6,121,2.6,2.51,0.31,1.25,5.05,1.06,3.58,1295}
9|{14.83,1.64,2.17,14,97,2.8,2.98,0.29,1.98,5.2,1.08,2.85,1045}
10|{13.86,1.35,2.27,16,98,2.98,3.15,0.22,1.85,7.2199,1.01,3.55,1045}
\.
```

2.4.2 Výpočet středů

Pro nalezení středu shluků je použita analytická funkce `kmeanspp` z knihovny MADlib.

```
kmeanspp(
      rel_source,  -- Název tabulky obsahující vstupní zdrojová data
      expr_point,  -- Název sloupce, ve kterém jsou uloženy souřadnice bodů
      k,           -- Požadovaný počet středů
      fn_dist,     -- Funkce požitá pro výpočet vzdálenosti od zdroje ke středu shluku
      agg_centroid,        -- Agregační funkce, použitá pro výpočet středu shluku
      max_num_iterations,  -- Maximální počet provedených iterací.
      min_frac_reassigned
      );
```

Požadovaný počet shluků byl nastaven na dva, středy shluků byly vyhledány agregační funkcí `avg`, která počítá průměrnou hodnotu ze všech bodů shluku. Výpočet vzdálenosti od bodu ke středu shluku je realizován funkcí `squared_dist_norm2`, tato funkce využívá pro výpočet čtvercovou Euklidovu vzdálenost, což má příznivý vliv na rychlost celého výpočtu.

```
SELECT * FROM madlib.kmeanspp( 'km_sample',
                               'points',
                               2,
                               'madlib.squared_dist_norm2',
                               'madlib.avg',
                               20,
                               0.001
                             );
```

2.4.3 Silhouette koeficient

Určuje, jak kvalitní je výsledný model. Nabývá hodnot <0,1>. Jedna znamená, že je model velmi kvalitní, prvky jednoho shluku jsou blízko u sebe a sousední shluky naopak daleko od sebe. Nula na druhou stranu znamená, že výsledný model je nekvalitní.

$$S(i) = \frac{b(i) - a(i)}{\max\{a(i), b(i)\}}$$

i - Jeden prvek v trénovací množině
a(i) - Průměrná vzdálenost prvku k ostatním prvkům v rámci stejného shluku
b(i) - Nejnižší průměrná vzdálenost k jinému shluku

```
simple_silhouette(
      rel_source,       -- Jméno relace obsahující vstupní body
      expr_point,       -- Výraz odkazující na sloupec, v němž jsou uloženy souřadnice
      centroids,        --Výraz obsahující souřadnice středů
      fn_dist     -- Funkce použitá pro výpočet vzdálenosti od bodu, ke středu shluku
                  )
```

```
SELECT *
FROM madlib.simple_silhouette
      ( 'km_sample','points',
      (SELECT centroids
      FROM madlib.kmeanspp(   'km_sample',
                              'points',
                              2,
                               'madlib.squared_dist_norm2',
                               'madlib.avg',
                              20,
                              0.001)),
                               'madlib.dist_norm2'
                             );
```

Výstupní panel

Datový výstup | Výklad | Zprávy | Historie

	simple_silhouette double precision
1	0.68978804882941

Obrázek 11: Výsledek výpočtu silhouette koeficientu

2.4.4 Přiřazení bodu do shluku

Zařazení všech vstupních bodů do určitého shluku je provedeno funkcí `closest_column(m,x)`, kde *m* představuje nalezené středy a *x* vstupní souřadnice bodů, které mají být zařazeny. Sloupec `cluster_id` označuje, do kterého shluku byl daný bod přiřazen. V tomto případě 0 pro první shluk a jedna pro druhý shluk.

```
SELECT data.*,
(madlib.closest_column(centroids, points)).column_id as cluster_id
FROM public.km_sample as data,
     (SELECT centroids
      FROM madlib.kmeanspp('km_sample', 'points', 2,
                           'madlib.squared_dist_norm2',
                           'madlib.avg', 20, 0.001)) as centroids
ORDER BY data.pid;
```

Výstupní panel

Datový výstup | Výklad | Zprávy | Historie

	pid integer	points double precision[]	cluster_id integer
1	1	{14.23,1.71,2.43,15.6,127,2.8,3.06,0.28,2.29,5.64,1.04,3.92,1065}	0
2	2	{13.2,1.78,2.14,11.2,1,2.65,2.76,0.26,1.28,4.38,1.05,3.49,1050}	0
3	3	{13.16,2.36,2.67,18.6,101,2.8,3.24,0.3,2.81,5.6799,1.03,3.17,1185}	1
4	4	{14.37,1.95,2.5,16.8,113,3.85,3.49,0.24,2.18,7.8,0.86,3.45,1480}	1
5	5	{13.24,2.59,2.87,21,118,2.8,2.69,0.39,1.82,4.32,1.04,2.93,735}	0
6	6	{14.2,1.76,2.45,15.2,112,3.27,3.39,0.34,1.97,6.75,1.05,2.85,1450}	1
7	7	{14.39,1.87,2.45,14.6,96,2.5,2.52,0.3,1.98,5.25,1.02,3.58,1290}	1
8	8	{14.06,2.15,2.61,17.6,121,2.6,2.51,0.31,1.25,5.05,1.06,3.58,1295}	1
9	9	{14.83,1.64,2.17,14,97,2.8,2.98,0.29,1.98,5.2,1.08,2.85,1045}	0
10	10	{13.86,1.35,2.27,16,98,2.98,3.15,0.22,1.85,7.2199,1.01,3.55,1045}	0

v pořádku

Obrázek 12: Výstup dotazu - přiřazení bodů do shluku

2.5 Příklad Iris flower data set:

Iris flower data set, je známý a hojně využívaný soubor dat na testování klasifikačních technik strojového učení. Obsahuje 3 různé druhy rostliny Kosatce (latinsky Iris).

Skládá se z padesáti příkladů měření čtyř charakteristických znaků každého ze tří druhů rostliny (Iris – Setosa, Iris – Virginica, Iris – Versicolor). Celkově tedy 150 rostlin a 600 datových bodů. Jednotlivé druhy se liší svými charakteristickými vlastnostmi, a to konkrétně délkou a šířkou kališních lístků (sepal width, sepal length) a také délkou a šířkou okvětních lístků (petal width, petal length) měřenou v centimetrech.

Sepal length	Sepal width	Petal length	Petal width	Species (druh)
5.1	3.5	1.4	0.2	Iris – Setosa
7.0	3.2	4.7	1.4	Iris - Versicolor
6.3	3.3	6.0	2.5	Iris - Virginica

Tabulka 2: Příklad záznamů Iris flower data setu

Celý soubor dat byl rozdělen na množinu dat pro trénování a testování v poměru 2:1, tak aby v obou množinách byly obsaženy všechny tři druhy Kosatce.
Na množině pro trénování se učí algoritmus, jak co zařadit, a testovací množina slouží k následnému ověření jeho přesnosti.

Soubor dat obsahuje slovní popis. Konkrétně se jedná o druhové jméno květiny, které je pro člověka na první pohled dobře čitelné a srozumitelné, pro počítač však nikoliv. Proto je vhodné se tohoto textu zbavit a nahradit jej například binárním číslem (místo textu Iris Setosa použít 100), nebo jako v našem případě každému řádku přiřadit unikátní identifikátor tzv. ID. Takto upravená data je již možno importovat do databáze.

2.5.1 Vytvoření zdrojových tabulek pro trénovací a testovací množinu

Následujícími příkazy provedeme vytvoření dvou nových tabulek (`train_data`, `test_data`), které následně naplníme rozdělenými a upravenými daty z předešlého kroku.

Vytvoření a naplnění tabulky `train_data` obsahující trénovací množinu dat:

```
CREATE TABLE public.train_data(pid int, points double precision[]);
COPY train_data (pid, points) FROM stdin DELIMITER '|';
1 | {5.1,3.5,1.4,0.2}
2 | {4.9,3.0,1.4,0.2}
3 | {4.7,3.2,1.3,0.2}
...
\.
```

Vytvoření a naplnění tabulky `test_data` obsahující testovací množinu dat:

```
CREATE TABLE public.test_data(id int, points double precision[]);

COPY test_data (id, points) FROM stdin DELIMITER '|';

1 | {5.5,4.2,1.4,0.2}
2 | {4.9,3.1,1.5,0.1}
3 | {5.0,3.2,1.2,0.2}
...
\.
```

2.5.2 Výpočet středu shluků

Dalším krokem byl výpočet středu shluků (anglicky centroids) a jejich následné uložení do nové tabulky databáze:

```
CREATE table iris_centroids as
      SELECT centroids
      FROM madlib.kmeanspp(
                              'train_data',
                              'points',
                              3,
                              'madlib.squared_dist_norm2',
                              'madlib.avg', 20, 0.001) as
                              centroids;
```

Pro vytvoření nové tabulky z výsledku dotazu byl použit příkaz `CREATE TABLE AS`. Vypočtené středy shluků jsou tak nyní uloženy v nově vzniklé tabulce s názvem `iris_centroids` a mohou být využity pro následující výpočty.

2.5.3 Výpočet Silhouette koeficientu

```
 SELECT * FROM madlib.simple_silhouette(
      'train_data',
      'points',
      (SELECT centroids
       FROM
      iris_centroids),
      'madlib.dist_norm2');
```

Výstupní panel

Datový výstup | Výklad

	simple_silhouette double precision
1	0.669326080980293

Obrázek 13: Výpočet silhouette koeficientu z trénovací množiny dat

2.5.4 Zařazení bodů do shluků

Model vypočítaných středů (tabulka `iris_centroids`), který je uložen v databázi byl aplikován na data z trénovací i testovací množiny. Protože byl tento model nastaven pro výpočet tří středů, došlo tak i k výslednému rozřazení bodů do tří shluků. Sloupec `cluser_id` na Obr. 13 představuje shluk, do kterého byla daná rostlina zařazena (Setosa 0, Versicolor 1, Virginica 2).

```
CREATE TABLE train_results AS SELECT data.*,
(madlib.closest_column(centroids, points)).column_id as cluster_id
FROM public.train_data as data,
      (SELECT centroids
      FROM iris_centroids) as centroids;


CREATE TABLE test_results AS SELECT data.*,
(madlib.closest_column(centroids, points)).column_id as cluster_id
FROM public.test_data as data,
      (SELECT centroids
      FROM iris_centroids) as centroids;
```

Vznikly dvě nové tabulky `train_results` a `test_results`. Které obsahují výstupní rozřazená data, přiřazená do jednotlivých shluků a jejich obsah je možno zobrazit dotazem:

```
SELECT * FROM train_results;
SELECT * FROM test_results;
```

Výstupní panel

Datový výstup | Výklad | Zprávy | Historie

	id integer	points double precision[]	cluster_id integer
16	16	{5.3,3.7,1.5,0.2}	0
17	17	{5,3.3,1.4,0.2}	0
18	18	{5.4,3,4.5,1.5}	1
19	19	{6,3.4,4.5,1.6}	1
20	20	{6.7,3.1,4.7,1.5}	1

v pořádku

Obrázek 14: Výstup zařazených dat do shluků

2.5.5 Výpočet přesnosti modelu

Z výstupu dotazu byl označen počet správně a nesprávně klasifikovaných dat jak trénovací, tak i testovací množiny a tyto hodnoty byly zaznamenány do Tab. 3. a 4.

TRAIN_RESULTS	Iris-Setosa	Iris-Versicolor	Iris-Virginica
Setosa 0	**33**	0	0
Versicolor 1	0	**34**	10
Virginica 2	0	0	**24**

Tabulka 3: Výsledek klasifikace bodů z trénovací množiny dat

$$Přesnost = \frac{Počet\ správně\ klasifikovaných\ dat}{Celkový\ počet\ dat} = \frac{33 + 34 + 24}{33 + 34 + 24 + 10} * 100 = 90{,}1\%$$

TEST_RESULTS	Iris-Setosa	Iris-Versicolor	Iris-Virginica
Setosa 0	**17**	0	0
Versicolor 1	0	**16**	5
Virginica 2	0	0	**11**

Tabulka 4: Výsledek klasifikace bodů z testové množiny dat

$$Přesnost = \frac{Počet\ správně\ klasifikovaných\ dat}{Celkový\ počet\ dat} = \frac{17 + 16 + 11}{17 + 16 + 11 + 5} * 100 = 89{,}8\%$$

Porovnáním výsledků z trénování a z testování lze zjistit, jak je model kvalitní.
Pokud by například z trénování bylo 98% a z testování 70%, znamenalo by to, že model byl přetrénován - naučil se pouze konkrétní příklady z trénovací množiny, ale nebyl by obecný, aby šel aplikovat na jakákoli data.

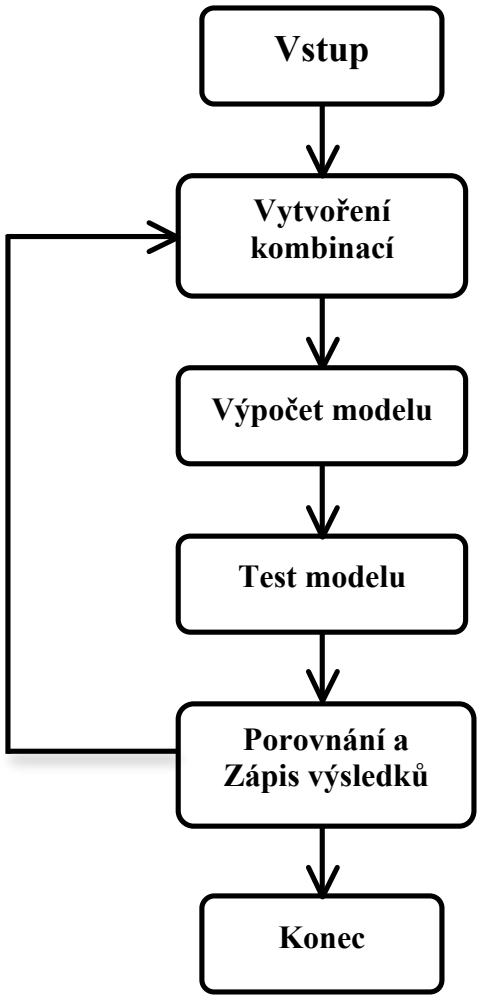
2.6 Dopředný výběr příznaků

Funkce byla napsána v jazyce PL/pgSQL. Pro výpočty modelu stromu a následné predikci hodnot byly použity funkce strojového učení z knihovny MADlib, konkrétně se jedná o `tree_train` a `tree_predict`.

2.6.1 Vytvoření funkce pro dopředný výběr

Funkce bude sloužit jako výběr nejvýhodnějšího atributu, či skupiny atributů, pro sestavení modelu rozhodovacího stromu ze zadaných dat. Jejím cílem tedy bude takovou skupinu najít a sestavit, vymodelovat model stromu, určit jeho výslednou predikční přesnost a výsledky uložit do databáze.

Popis funkce znázorňuje následující zjednodušený vývojový diagram.



Obrázek 15: Vývojový diagram funkce pro dopředný výběr

2.6.2 Popis jednotlivých bloků vývojového diagramu funkce

Vstup

Funkce pro dopředný výběr (dále `f_selection`) má čtyři vstupní parametry jejichž popis je uveden níže. Požadovaný datový typ každého z nich je uveden v hranatých závorkách.

`train_table_name [CHARACTER VARYING]`

Název tabulky obsahující data pro natrénování modelu rozhodovacího stromu.

`test_table_name [CHARACTER VARYING]`

Název tabulky obsahující data pro otestování přesnosti natrénovaného modelu.

`decision_parameter [CHARACTER VARYING]`

Jméno sloupce s klíčovým atributem obsahujícím výstupní odpověď.

`attributes [CHARACTER VARYING[]]`

Jednotlivé atributy tabulky, které mají být otestovány.

Protože se volání funkcí v jazyce PL/pgSQL provádí dotazem `SELECT` spojeným s názvem volané funkce, je spuštění inicializováno následovně:

```
select f_selection( train_table_name, test_table_name,
                    decision_parameter, attributes )
```

Vytvoření kombinací

Protože atribut, mající určitou hodnotu vypočtené přesnosti, může mít v kombinaci s jinými atributy daleko vyšší přesnost predikce, jsou v tomto kroku vždy vypočteny všechny možné kombinace atributů, kromě těch již vybraných. Jejich počet při každé iteraci odpovídá vzorci $celkový\ počet\ atributů\ -počet\ již\ použitých$.

Výpočet modelu

V tomto bodě se provádí výpočet modelu rozhodovacího stromu pro jednu z možných kombinací atributů, jenž byla sestavena v předchozím kroku. Je zde použita funkce `tree_train` z knihovny MADlib.

Test modelu

Zde probíhají výpočty, určující přesnost a tím i kvalitu výsledného natrénovaného modelu rozhodovacího stromu. Pro predikci je zde využito funkce `tree_predict` z knihovny MADlib. Ověřování přesnosti probíhá jak na datech pro testování, tak i na datech pro trénink za účelem zjištění, zda je model dostatečně univerzální. Mohlo by se totiž stát, že se model naučí jen konkrétní případy a nebude tak použitelný pro jakákoliv data. Vzorec pro výpočet procentuální přesnosti je následovný:

$$Přesnost = \frac{Počet\ správně\ klasifikovaných\ dat}{Celkový\ počet\ dat} * 100$$

Zápis výsledků

Výsledky algoritmu jsou zapisovány do tabulky `output`. Existence této tabulky v databázi je kontrolována při každém spuštění funkce. Pokud tabulka doposud neexistuje dojde k jejímu vytvoření a zápisu výsledků. Jestliže ale existuje, dojde pouze k zápisu.

Popis jednotlivých atributů tabulky `output` uveden v Tabulce 7.

Název sloupce	Datový typ	Popis sloupce
id	`SERIAL`	Primární klíč, jedinečný identifikátor každého řádku tabulky.
iteration	`INTEGER`	Stupeň iterace, udává kolik atributů bylo použito pro výpočet.
features	`VARCHAR`	Výčet atributů použitých pro sestavení modelu stromu.
tree_model	`VARCHAR`	Název tabulky, obsahující vypočítaný model stromu.
tree_result	`VARCHAR`	Tabulka obsahující informace o modelu vypočteného stromu.
train_accuracy	`NUMERIC`	Procentuální přesnost, s jakou natrénovaný model klasifikoval data na kterých se učil.
test_accuracy	`NUMERIC`	Procentuální přesnost, s jakou natrénovaný model klasifikoval testová data.

Tabulka 5: Popis výstupní tabulky `output`

Konec

K ukončení funkce dojde v případě, kdy se vypočtená přesnost již dále nenavyšuje nebo bylo dosaženo všech možných kombinací.

2.6.3 Funkce tree_train

`Tree_train` je funkce knihovny MADlib sloužící pro výpočet modelu rozhodovacího stromu. Syntaxe funkce je následovná:

```
tree_train(
          training_table_name, output_table_name,
          id_col_name, dependent_variable,
          list_of_features, list_of_features_to_exclude,
          split_criterion, grouping_cols,
          weights, max_depth,
          min_split, min_bucket,
          num_splits, pruning_params,
          surrogate_params, verbosity
          )
```

Popis jednotlivých parametrů funkce `tree_train` je uveden níže. V hranatých závorkách je uveden požadovaný datový typ na vstupu.

`training_table_name [TEXT]`

Tabulka, obsahující vstupní zdrojová data určená pro natrénování modelu stromu.

`output_table_name [TEXT]`

Název vygenerované tabulky obsahující model stromu. Pokud již existuje v databázi tabulka se stejným názvem, funkce je ukončena chybovým hlášením. Nově vytvořená tabulka obsahuje tyto sloupce:

Název sloupce	Datový typ	Popis sloupce
tree	`BYTE8`	Natrénovaný model rozhodovacího stromu uložený v binárním formátu.
cat_levels_in_text	`TEXT`	Úroveň kategorických proměnných (např. false, true).
cat_n_levels	`INTEGER`	Počet úrovní pro každou kategorickou proměnnou.
tree_depth	`INTEGER`	Maximální hloubka stromu po jeho natrénování (kořen má hloubku 0).
pruning_cp	`DOUBLE PRECISION`	Parametr, vyjadřující náklady pro vytvoření rozhodovacího stromu.

Tabulka 6: Popis výstupní tabulky `output_table_name`

Ve stejném kroku je také vytvořena tabulka, obsahující sumární informace o natrénovaném modelu. Sloupce, které tabulka obsahuje jsou uvedeny v následující tabulce.

Název sloupce	Datový typ	Popis sloupce
method	TEXT	Použitá metoda, výchozí funkce 'tree_train'.
is_classification	BOOLEAN	TRUE pro klasifikační rozhodovací stromy, FALSE pro regresní.
source_table	TEXT	Název zdrojové tabulky.
model_table	TEXT	Název tabulky, obsahující vypočítaný model stromu.
id_col_name	TEXT	Sloupec, obsahující jednoznačný identifikátor každého řádku.
dependent_varname	TEXT	Sloupec, obsahující výstupní odpověď pro trénovaná data.
independent_varname	TEXT	Nezávislé proměnné.
cat_features	TEXT	Výčet kategorických atributů.
con_features	TEXT	Výčet spojitých atributů.
grouping_col	TEXT	Názvy seskupených sloupců.
num_all_groups	INTEGER	Počet skupin v rozhodovacím stromu pro trénování.
num_failed_groups	INTEGER	Počet chybných skupin v rozhodovacím stromu pro trénování.
total_rows_processed	BIGINT	Celkový počet zpracovaných řádků.
total_rows_skipped	BIGINT	Celkový počet přeskočených řádků.
dependent_var_levels	TEXT	Proměnné pro klasifikaci.
dependent_var_type	TEXT	Datový typ proměnných pro klasifikaci.
input_cp	DOUBLE PRECISION	Náklady pro vytvoření rozhodovacího stromu.
independent_var_types	TEXT	Datový typ nezávislých proměnných.

Tabulka 7: Popis výstupní tabulky `output_table_name_summary`

`id_col_name [TEXT]`

Sloupec, obsahující jednoznačný identifikátor každého řádku. Jedná se o povinný argument používaný pro predikci a křížovou validaci.

dependent_variable [TEXT]

Sloupec, obsahující výstupní odpověď pro trénovaná data. `Boolean`, `integer` a `text` jsou datové typy považované za výstupy klasifikační, zatímco hodnoty datového typu `double precision` jsou považovány za výstupy regresní. Odpověď může být multinominální, ovšem se zvyšujícím se počtem různých odpovědí lineárně roste časová náročnost výpočtu trénovací funkce.

list_of_features [TEXT]

Názvy sloupců použitých jako prediktory. Může zde být také uveden zástupný znak '*' reprezentující všechny sloupce (mimo ty, které jsou zahrnuty jako vstupní argumenty funkce). Počet sloupců není omezen, a datové typy mohou být různé (`boolean, integer, text, double precision..`).

list_of_features_to_exclude [TEXT]

Sloupce, které mají být vyloučeny z listu pro predikci.

split_criterion [TEXT]

Rozdělovací kritérium. Výchozí nastavení je následující: 'gini' pro klasifikaci a 'mse' pro regresi.

grouping_cols [TEXT]

Čárkami oddělený seznam sloupců, pro seskupení dat podle nich. Toto nastavení vede k vytvoření několika rozhodovacích stromů pro každou skupinu. Výchozí hodnota je nastavena jako `NULL`.

weights [TEXT]

Sloupec obsahující váhu každého pozorování. Výchozí hodnota `NULL`.

max_depth [INTEGER]

Maximální hloubka každého uzlu u konečného stromu. Kořenový uzel má číslo 0. Výchozí hodnota je 10.

min_split [INTEGER]

Minimální počet pozorování, které musí existovat, aby mohlo dojít v uzlu k rozhodnutí o rozdělení. Optimální zvolená hodnota závisí na počtu řádků daného data setu. Výchozí hodnota je 20.

min_bucket [INTEGER]
Nejmenší možný počet pozorování v každém terminálním uzlu. Jestliže je zadán parametr `min_bucket`, `min_split` je automaticky nastaven na hodnotu `min_bucket*3`. Je-li naopak zadán parametr `min_split`, hodnota `min_bucket` je rovna `min_split/3`.

num_splits [INTEGER]
Tento globální parametr je využíván k výpočtu rozlišení rozdělení u kontinuálních prvků. Parametry kontinuálních hodnot jsou uskladňovány do diskrétních kvantilů, sloužících pro výpočet hranic rozdělení. S rostoucím počtem uskladněných hodnot roste přesnost výsledné predikce, ovšem za cenu delšího výpočetního času. Výchozí hodnota je 100.

pruning_params [TEXT]
Parametry pro prořezávání stromu.

surrogate_params [TEXT]
Páry kontrolující chování náhradního rozdělení každého uzlu stromu. Náhradní proměnná je dalším prediktorem, je spojena s hlavní proměnnou pro rozdělení a použije se pouze tehdy, je-li hlavní proměnná `NULL`.

verbosity [BOOLEAN]
Poskytuje detailní výpis výsledků trénování. Výchozí hodnota `FALSE`.

2.6.4 Funkce tree_predict

`Tree_predict` je rovněž funkcí knihovny MADlib a slouží pro predikci hodnot na základě již dříve vypočítaného modelu rozhodovacího stromu. Syntaxe funkce je následující:

```
tree_predict(  tree_model,
               new_data_table,
               output_table,
               type
            )
```

Popis jednotlivých parametrů funkce `tree_predict` je uveden níže. V hranatých závorkách je uveden požadovaný datový typ na vstupu.

`tree_model [TEXT]`
Název tabulky vytvořené funkcí `tree_train`, jenž obsahuje model natrénovaného rozhodovacího stromu.

`new_data_table [TEXT]`
Název tabulky obsahující predikovaná data. Očekává se, že tabulka bude obsahovat stejné parametry, které byly použity při trénování. Tabulka by taktéž měla obsahovat jednoznačný identifikátor `ID` pro každý řádek v tabulce.

`output_table [TEXT]`
Jméno nově vytvořené tabulky, do které se zapíše výsledek predikce. Pokud se již vyskytuje v databázi tabulka se stejným názvem, funkce je ukončena chybovým hlášením.

`type [TEXT]`
Pro regresní stromy je výstup vždy predikován hodnotou závislé proměnné. U klasifikačních stromů může být zvoleno nastavení typu `'response'`, vracející klasifikační predikci, nebo `'prob'` vracející pravděpodobnost dané třídy. Pro každou hodnotu závislé proměnné je ve výstupní tabulce `output_table` vytvořen sloupec s pravděpodobnostmi.

2.7 Příklad použití funkce pro dopředný výběr příznaků

Na následujících dvou příkladech bude předvedeno použití funkce `f_selection` pro dopředný výběr příznaků.

2.7.1 Příklad Golf

Pro názornou ukázku funkce algoritmu byl vybrán data set Golf, který popisuje vliv počasí na možnosti hrát golf (data jsou pouze demonstrační).

ID	Atributy				Klasifikační atribut
	Obloha	Teplota	Vlhkost	Vítr	Hrát?
1	slunečno	85	85	ne	NE
2	slunečno	80	90	ano	NE
3	zataženo	83	78	ne	ANO
4	déšť	70	96	ne	ANO
5	déšť	68	80	ne	ANO
6	déšť	65	70	ano	NE
7	zataženo	64	65	ano	ANO
8	slunečno	72	95	ne	NE
9	slunečno	69	70	ne	ANO
10	déšť	75	80	ne	ANO
11	slunečno	75	70	ano	ANO
12	zataženo	72	90	ano	ANO
13	zataženo	81	75	ne	ANO
14	déšť	71	80	ano	NE

Tabulka 8: Obsah data setu Golf

Data set je poměrně malý, obsahuje čtrnáct záznamů s čtyřmi popisnými a jedním hlavním klasifikačním atributem. Jejich popis je uveden v následující tabulce.

Název sloupce	Datový typ	Popis sloupce
id	`SERIAL`	Primární klíč, jedinečný identifikátor každého řádku tabulky.
obloha	`TEXT`	Hodnoty (slunečno, zataženo, déšť)
teplota	`INTEGER`	Hodnoty v rozmezí (64-85) vyjádřeno v Kelvinech.
vlhkost	`INTEGER`	Hodnoty (65-95) vyjádřeno v procentech.
vítr	`TEXT`	Hodnoty (ano, ne)
hrát	`TEXT`	Hlavní klasifikační parametr, hodnoty (ANO, NE)

Tabulka 9: Popis jednotlivých atributů data setu Golf

Zavoláním funkce `f_selection` s těmito parametry,

```
select f_selection('golf','golf','hrat',
                        '{obloha,teplota,vlhkost,vitr}')
```

dojde ke spuštění algoritmu dopředného výběru, a je vybrána nejvhodnější kombinace atributů pro natrénování modelu.

V první iteraci je vybrán atribut s nejvyšší přesností (v tomto případě se jedná o `teplotu`) a je zapsán do tabulky `output`. Ve druhé iteraci se tento atribut zkombinuje se všemi ostatními, kromě sám sebe, a je znovu vybrán ten s nejvyšší přesností. Následuje porovnání dosažené přesnosti s výsledky v tabulce `output`, jestliže se přesnost v porovnání s předchozím nejlepším výsledkem zvyšuje, celý proces je opakován. Pokud při porovnávání dojde ke zjištění nenavyšující se přesnosti predikce, je proces ukončen. Princip postupu algoritmu je znázorněn v následující tabulce.

ID	Iteration	Features	Tree_result	Tree_ model	Train_ accuracy	Test_ accuracy
1	1	obloha	tm1_summary	tm1	71.43	71.43
2	1	teplota	tm2_summary	tm2	85.71	85.71
3	1	vlhkost	tm3_summary	tm3	78.57	78.57
4	1	vítr	tm4_summary	tm4	64.29	64.29
5	2	teplota, obloha	tm5_summary	tm5	92.86	92.86
6	2	teplota, vlhkost	tm6_summary	tm6	85.71	85.71
7	2	teplota, vítr	tm7_summary	tm7	85.71	85.71
8	3	teplota, obloha, vlhkost	tm8_summary	tm8	92.85	92.85
9	3	teplota, obloha, vítr	tm9_summary	tm9	92.85	92.85

Tabulka 10: Průběh výpočtu funkce po jednotlivých iteracích

Výstup funkce `f_selection` je potom následující:

ID	Iteration	Features	Tree_result	Tree_ model	Train_ accuracy	Test_ accuracy
2	1	teplota	tm2_summary	tm2	85.71	85.71
5	2	teplota, obloha	tm5_summary	tm5	92.86	92.86

Tabulka 11: Tabulka output, obsahující výstup funkce `f_selection`

Ve druhé iteraci došlo k dosažení přesnosti predikce téměř 93%, čímž byl nalezen nejlepší výsledek a algoritmus byl ukončen. Model stromu s názvem `tm5` byl uložen do databáze a informace o tomto modelu jsou obsaženy v tabulce `tm5_summary`.

Z důvodu malé velikosti data setu, byla pro testování použita stejná data jako pro trénink.

2.7.2 Příklad funkce na obsáhlejších datech

Data set obsahuje 20 000 záznamů pro trénink a 40 000 záznamů pro testování. Do tabulky pro trénink bylo přidáno pět, zcela náhodně vygenerovaných sloupců pro ověření možnosti výběru jednotlivých atributů na testování. Konkrétně se jedná o sloupce c11 až c15.

Popis tabulky pro trénink:

Název sloupce	Datový typ	Popis sloupce
id	`SERIAL`	Primární klíč, jedinečný identifikátor každého řádku tabulky.
decision	`BOOLEAN`	Hlavní klasifikační parametr, Hodnoty (0, 1)
c1 - c15	`DOUBLE PRECISION`	Atributy c1 až c15, c11 - c15 obsahují náhodně vygenerovaná data.

Tabulka 12: Popis tabulky `train_data`

Popis tabulky pro test:

Název sloupce	Datový typ	Popis sloupce
id	`SERIAL`	Primární klíč, jedinečný identifikátor každého řádku tabulky.
decision	`BOOLEAN`	Hlavní klasifikační parametr, Hodnoty (0, 1)
c1 - c10	`DOUBLE PRECISION`	Atribut c1 až c10

Tabulka 13: Popis tabulky `test_data`

Spuštění funkce:

```
select f_selection('train_data','test_data',
                'decision','{c1,c2,c3,c4,c5,c6,c7,c8,c9,c10}')
```

Proběhne výpočet a výsledná data jsou zaznamenána do tabulky `output`.

ID	Iteration	Features	Tree_result	Tree_ model	Train_ accuracy	Test_ accuracy
1	1	c1	tm1_summary	tm1	92.90	92.52
11	2	c1,c2	tm11_summary	tm11	96.01	95.93
24	3	c1,c2,c7	tm24_summary	tm24	96.08	96.07

Tabulka 14: Výstup funkce `f_selection`

Nejvyšší přesnosti dosahovala kombinace atributů c1, c2 a c7 sestavená při třetí iteraci. Do databáze byl tento model uložen pod názvem `tm24`, informace o něm pak pod názvem `tm24_summary`.

Celý postup výpočtu napříč jednotlivými iteracemi je znázorněn v následující tabulce.

ID	Iteration	Features	Tree_result	Tree_ model	Train_ accuracy	Test_ accuracy
1	1	c1	tm1_summary	tm1	92.90	92.52
2	1	c2	tm2_summary	tm2	63.24	62.73
3	1	c3	tm3_summary	tm3	63.24	62.73
4	1	c4	tm4_summary	tm4	63.24	62.73
5	1	c5	tm5_summary	tm5	63.24	62.73
6	1	c6	tm6_summary	tm6	63.24	62.73
7	1	c7	tm7_summary	tm7	63.24	62.73
8	1	c8	tm8_summary	tm8	63.24	62.73
9	1	c9	tm9_summary	tm9	63.24	62.73
10	1	c10	tm10_summary	tm10	63.24	62.73
11	2	c1,c2	tm11_summary	tm11	96.01	95.93
12	2	c1,c3	tm12_summary	tm12	93.43	93.02
13	2	c1,c4	tm13_summary	tm13	92.91	92.52
14	2	c1,c5	tm14_summary	tm14	92.77	92.35
15	2	c1,c6	tm15_summary	tm15	92.65	92.22
16	2	c1,c7	tm16_summary	tm16	92.82	92.89
17	2	c1,c8	tm17_summary	tm17	92.85	92.43
18	2	c1,c9	tm18_summary	tm18	92.83	92.41
19	2	c1,c10	tm19_summary	tm19	92.87	92.94
20	3	c1,c2,c3	tm20_summary	tm20	95.95	95.87
21	3	c1,c2,c4	tm21_summary	tm21	96.08	96.03
22	3	c1,c2,c5	tm22_summary	tm22	96.08	96.01
23	3	c1,c2,c6	tm23_summary	tm23	95.91	95.83
24	3	c1,c2,c7	tm24_summary	tm24	96.08	96.07
25	3	c1,c2,c8	tm25_summary	tm25	96.05	95.97
26	3	c1,c2,c9	tm26_summary	tm26	95.92	95.87
27	3	c1,c2,c10	tm27_summary	tm27	95.91	95.85
28	4	c1,c2,c7,c3	tm28_summary	tm28	96.10	96.05
29	4	c1,c2,c7,c4	tm29_summary	tm29	96.08	96.01
30	4	c1,c2,c7,c5	tm30_summary	tm30	96.03	95.93
31	4	c1,c2,c7,c6	tm31_summary	tm31	96.08	96.02
32	4	c1,c2,c7,c8	tm32_summary	tm32	95.96	95.88
33	4	c1,c2,c7,c9	tm33_summary	tm33	95.95	95.88
34	4	c1,c2,c7,c10	tm34_summary	tm34	95.96	95.87

Tabulka 15: Průběh algoritmu po jednotlivých iteracích

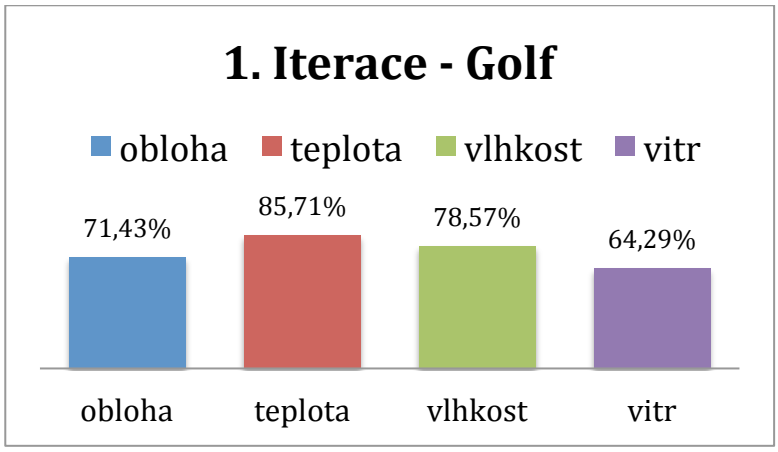
V první iteraci je vybrán atribut s nejvyšší přesností (`c1`) a je zapsán do tabulky `output`. Ve druhé iteraci je (`c1`) zkombinován a otestován se všemi ostatními atributy kromě sám sebe. Z těchto kombinací je vybrána ta, s nejvyšší přesností (`c1`, `c2`) a je porovnána s nejvyšší přesností z první iterace (`c1`). Protože přesnost (`c1`, `c2`) je vyšší než (`c1`), je tento výsledek zapsán do tabulky a algoritmus pokračuje dál. Opět dojde k sestavení kombinací a je vybrána skupina (`c1`, `c2`, `c7`), následuje porovnání s (`c1`, `c2`) a zápisu do tabulky. Ve čtvrté iteraci je vybrána skupina (`c1`, `c2`, `c7`, `c3`), která ale nedosahuje vyšší přesnosti predikce než (`c1`, `c2`, `c7`) a tak je algoritmus ukončen.

2.7.3 Zhodnocení výsledků

Algoritmus pracuje pouze s atributy zadanými při vstupním volání funkce, čímž zcela odpadá problém s nežádoucími přidanými sloupci jako tomu je třeba u druhého příkladu (`c11 - c15`). Ze zadaných atributů jsou sestaveny kombinace a proveden výpočet přesnosti predikce. Kombinace s nejvyšší dosaženou přesností představuje nejvhodnější skupinu atributů pro natrénování modelu stromu.

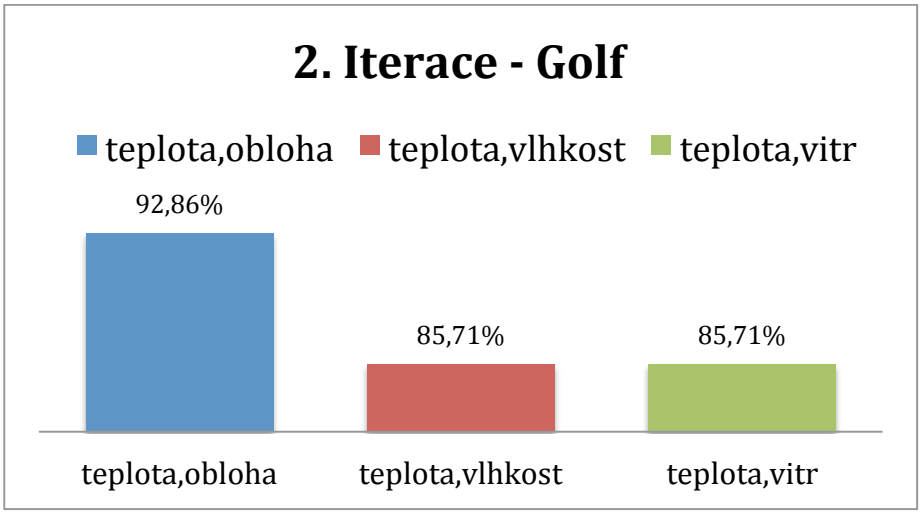
Přiklad Golf

V první iteraci byly otestovány všechny zadané atributy, z nichž nejvyšší procentuální přesnosti (85,71%) dosáhl sloupec `teplota`. Výsledné přesnosti jednotlivých atributů znázorňuje následující graf.



Graf 1: První iterace funkce `f_selection` na data setu Golf

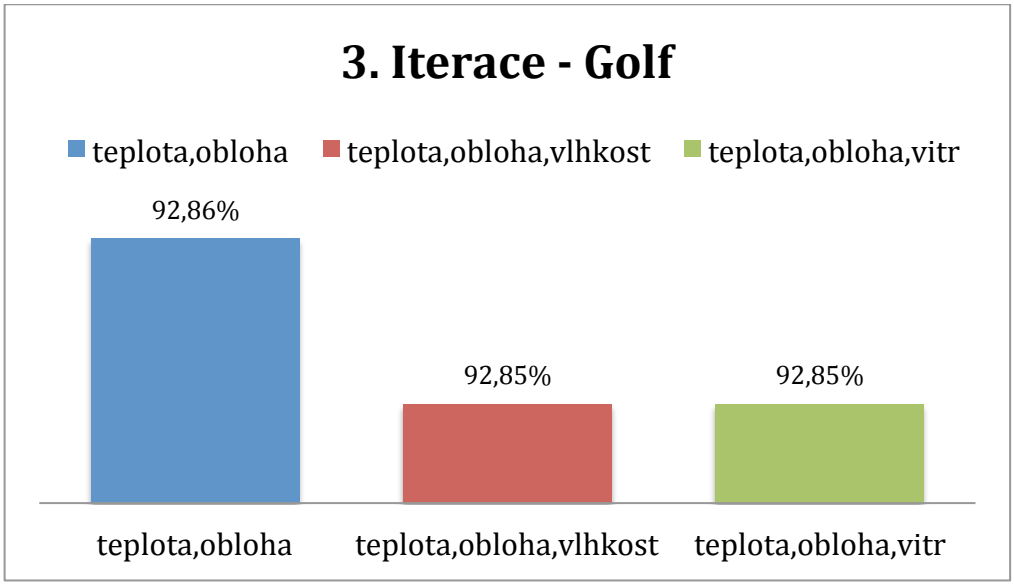
Ve druhé iteraci byl atribut s nejvyšší přesností z prvního kola (`teplota`) vybrán a zkombinován s ostatními. Nejlepšího výsledku dosáhla kombinace `teplota, obloha` (92,86%). Protože přesnost vzrostla, algoritmus pokračoval.



Graf 2: Druhá iterace funkce `f_selection` na data setu Golf

Ve třetí iteraci byl opět vybrán nejlepší výsledek z předchozího kola (`teplota, obloha`), který se zkombinoval se zbylými atributy. Výsledná přesnost predikce byla ovšem nižší, než nejlepší výsledek ze druhé iterace a tak došlo k ukončení algoritmu.

Porovnání výsledků třetí iterace s nejlepším výsledkem druhé iterace je znázorněno v následujícím grafu.

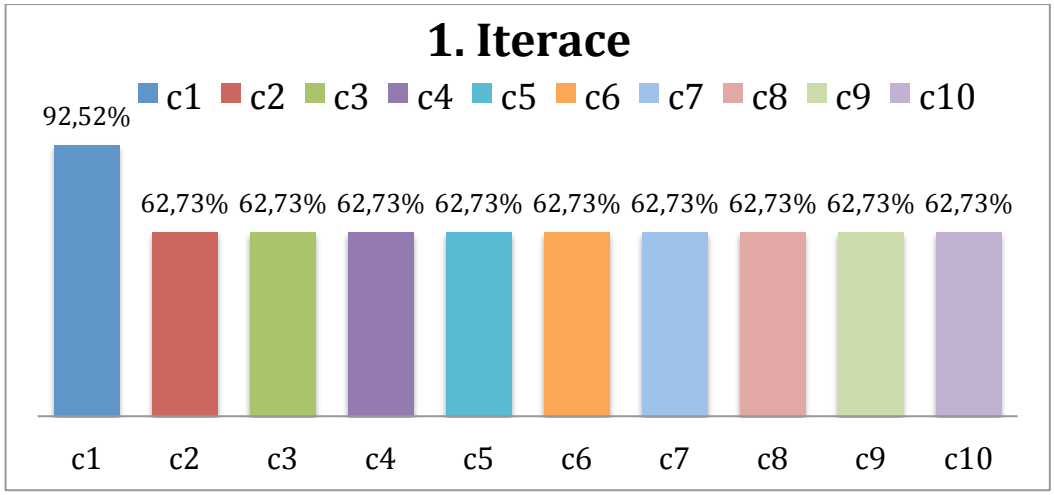


Graf 3: Třetí iterace funkce `f_selection` na data setu Golf

Nejlepšího výsledku (92,86%) tak dosáhla kombinace `teplota, obloha` sestavená při druhé iteraci.

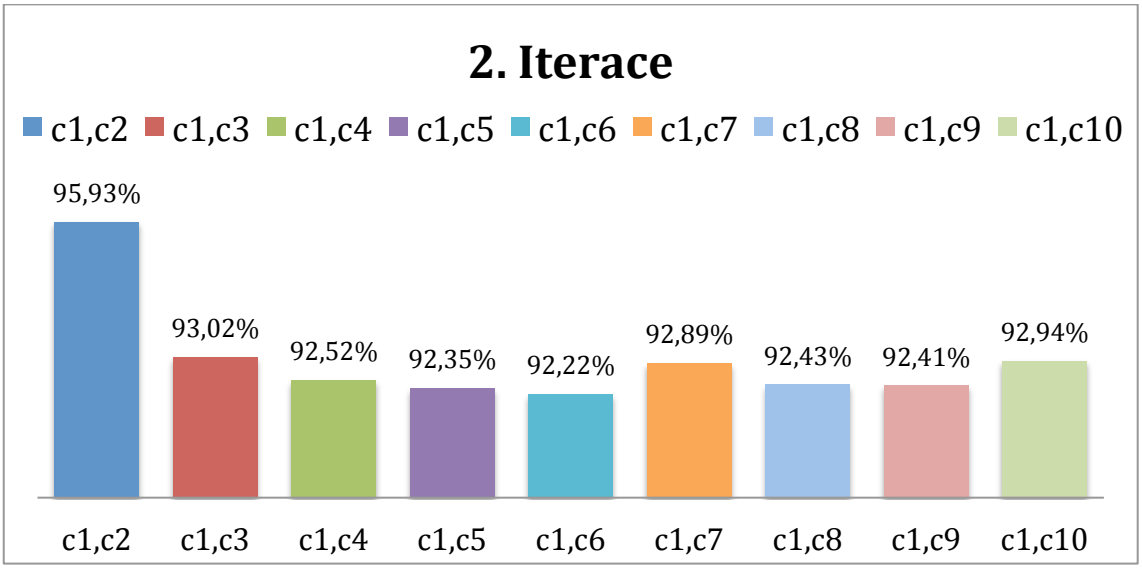
Příklad druhý

Nejvyšší predikční přesnosti (96,07%) dosáhla kombinace atributů `c1,c2` a `c7` vytvořená při třetí iteraci. Průběh výběru nejlepších atributů napříč jednotlivými iteracemi je znázorněn v následujících grafech.



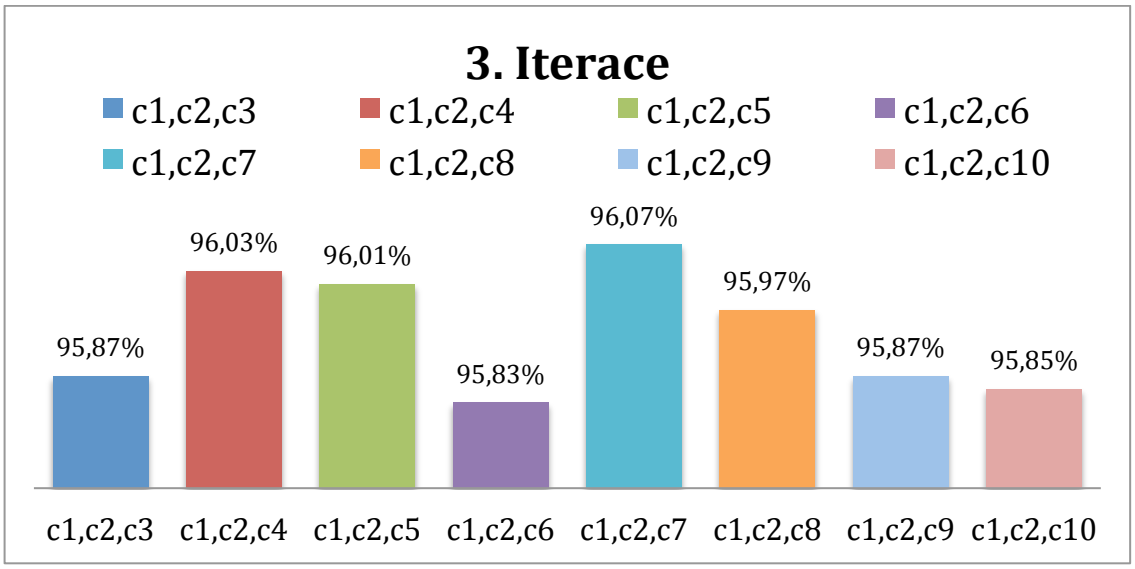
Graf 4: První iterace funkce `f_selection` u druhého příkladu

V první iteraci byl vybrán atribut `c1` s dosaženou přesností 92,90%.



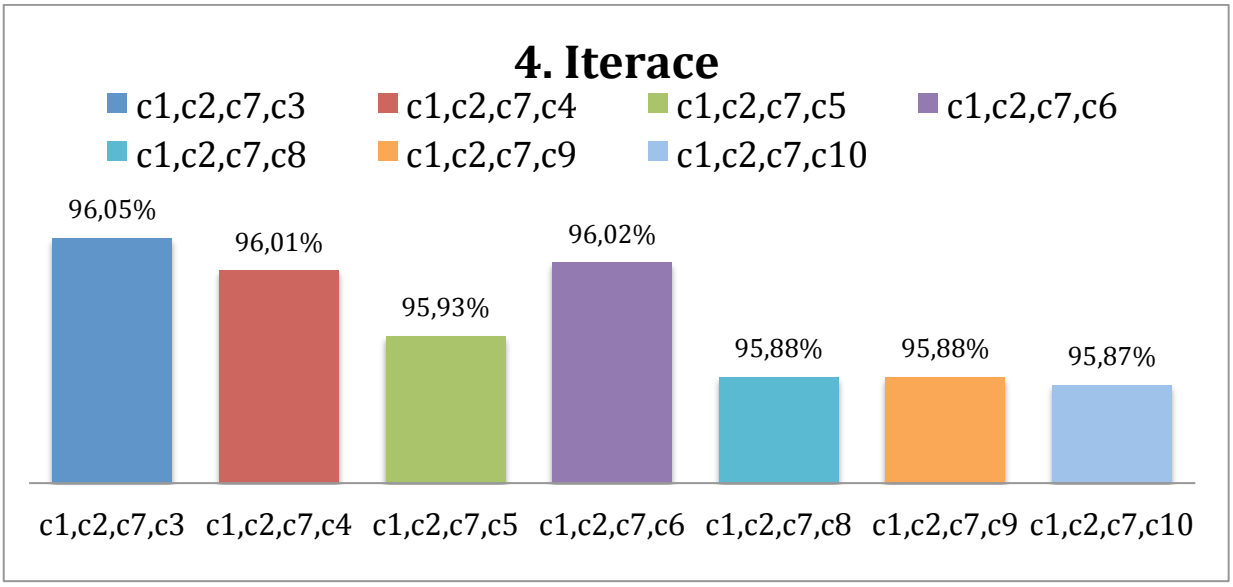
Graf 5: Druhá iterace funkce `f_selection` u druhého příkladu

Ve druhém kole byla vybrána kombinace `c1`, `c2` s výsledkem 95,73%. To znamená, že přesnost oproti prvnímu kolu vzrostla a algoritmus tak pokračoval dál.



Graf 6: Třetí iterace funkce `f_selection` u druhého příkladu

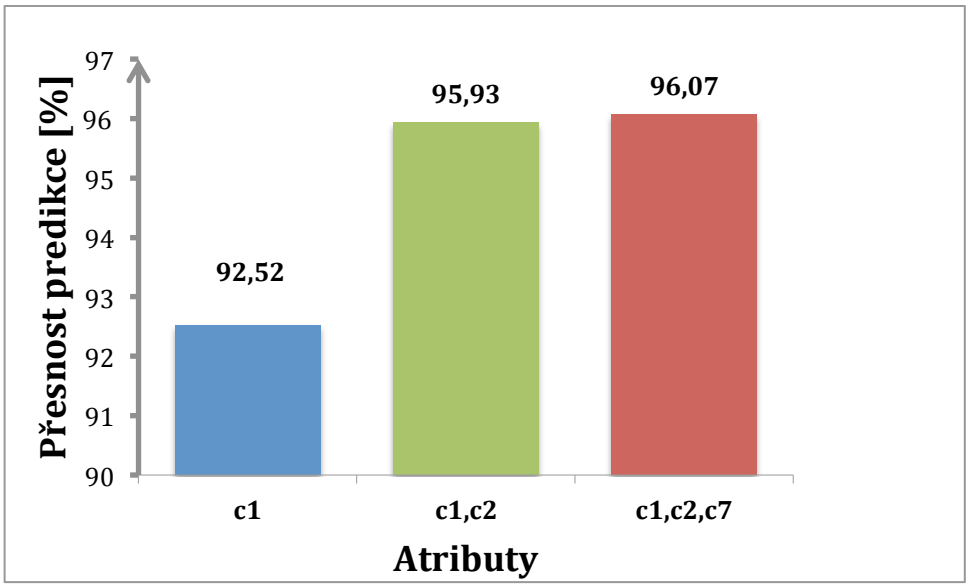
Ve třetím kole byla jako nejlepší vyhodnocena kombinace `c1`, `c2`, `c7` s přesností 96,07%, což je oproti předchozí iteraci opět lepší výsledek.



Graf 7: Čtvrtá iterace funkce `f_selection` u druhého příkladu

Ve čtvrté iteraci už k vylepšení predikční přesnosti nedošlo a tak se algoritmus ukončil. Nejlepší nalezená kombinace tedy zůstala `c1`, `c2` a `c7` ze třetí iterace.

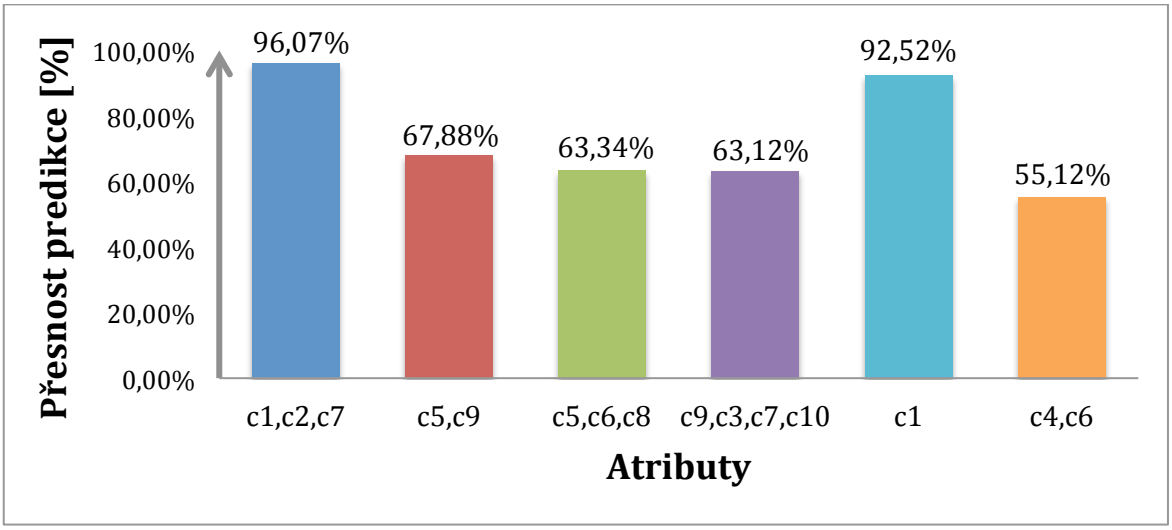
Vylepšení přesnosti predikce modelu v závislosti na počtu atributů, v tomto případě, tedy vypadá takto:



Graf 8: Postupné vylepšování přesnosti s rostoucím počtem atributům

Pro demonstraci a ověření faktu, že algoritmus opravdu vybral tu nejlepší možnou variantu, byly zcela náhodně vybrány a otestovány kombinace některých atributů.

Porovnání výsledné přesnosti algoritmem sestavené nejlepší skupiny (`c1`, `c2`, `c7`) s jinými, zcela náhodně vybranými kombinacemi atributů je znázorněno v následujícím grafu.



Graf 9: Porovnání přesnosti `c1`, `c2`, `c7` s náhodnými kombinacemi atributů

3 ZÁVĚR

Při přímém pokusu o manuální instalaci databáze PostgreSQL a jejího serveru se vyskytly problémy spojené s kompatibilitou jazyka PL/Python, proto byl vybrán nástroj MacPorts, s jehož pomocí byly nainstalovány všechny potřebné součásti projektu. Pro lepší orientaci a grafické rozhraní byl zvolen program pgAdmin3 pomocí kterého byla realizována celá administrace databáze a také provedeny všechny dotazy.

Spuštění vzorového příkladu *k*-means dopadlo dle předpokladu, vyhodnocení přesnosti metody se zde neprovedlo z důvodu velmi malého počtu zdrojových dat.

Při realizaci metody se vstupními body získanými z Iris data setu bylo provedeno několik modifikací oproti vzorovému příkladu. Zdrojová data zde byla rozdělena na trénovací a testovou množinu v poměru 2:1. Obě tyto množiny byly importovány do databáze, ale algoritmus se v tomto případě učil pouze na datech z trénovací množiny, ze které byly také vypočítány středy shluků. Natrénovaný model byl poté aplikován jak na data z trénovací tak i testové množiny a následně bylo provedeno vyhodnocení jeho kvality, která se pohybovala kolem 90% u obou množin. Což je dobrý výsledek a značí, že je model poměrně kvalitní. Pokud by totiž byla přesnost z trénování například 98% a z testování 70%, znamenalo by to, že model byl přetrénován a naučil se pouze konkrétní příklady z trénovací množiny, ale nebyl by obecný a nešel by šel aplikovat na jakákoli data. Celková nepřesnost je způsobena malým počtem dat a podobností záznamů Iris Versicolor a Iris Virginica.

Při návrhu funkce pro dopředný výběr příznaků, byl použit procedurální programovací jazyk PL/pgSQL. Pro výpočty, sestavení modelu rozhodovacího stromu a následné predikci hodnot byly použity funkce strojového učení z knihovny MADlib, konkrétně se jedná o `tree_train` a `tree_predict`.

Algoritmus funkce je naprogramován tak, aby vždy vybral atribut, nebo skupinu atributů, dosahující nejvyšší možné přesnosti predikce. Jeho princip je založen na rozdělení výpočtu do *n* iterací, kdy v každé z nich jsou sestaveny kombinace a je vybrán ten nejlepší výsledek, který je následně porovnán s nejlepším výsledkem předchozí iterace *n*-1. Počítá i s možností, kdy stejných nejlepších výsledků dosáhne více atributů nebo skupin.

Testování funkce proběhlo na dvou příkladech. Prvním z nich byl data set golf, obsahující pouze čtrnáct záznamů a čtyři atributy, s nejlepším výsledkem `obloha`, `teplota` ve druhé iteraci a dosaženou přesností 92,86%. Ve druhém příkladu byla funkce otestována na obsáhlejším data setu, o celkové velikosti 60 000 záznamů a patnácti atributech, s nejlepší kombinací `c1`, `c2`, `c7` při třetí iteraci a dosaženou přesností 96,07%. Na tomto příkladu byla také demonstrována skutečnost, že algoritmus pracuje pouze s atributy zadanými při vstupním volání funkce, čímž zcela odpadl problém s nežádoucími přidanými sloupci (`c11` - `c15`).

Celkově se tedy potvrdila teorie o možnosti provádět výpočty přímo v databázi. Hlavním přínosem této práce je pak rozšíření knihovny o funkci dopředného výběru příznaků, která zpřesňuje modely pro strojového učení.

LITERATURA

[1] http://www.linuxsoft.cz. PostgreSQL - Historie a pohledy jinam [online]. 2004 [cit. 2014-11-24]. Dostupné z: http://www.linuxsoft.cz/article.php?id_article=304

[2] ŠIMUNEK, Milan. SQL: Kompletní kapesní průvodce. 1. vyd. Praha: Grada Publishing, 1999. ISBN 80-7169-692-7.

[3] Oppel, Andrew. Databáze bez předchozích znalostí. [překl.] Computer Press. Vydání první. Brno : Computer Press, 2006. str. 240. ISBN 80-251-1199-7

[4] METODY PŘÍSTUPU K DATABÁZÍM POSTGRESQL V .NET FRAMEWORK [online]. Brno, 2009 [cit. 2014-11-24]. Dostupné z: https://www.vutbr.cz/www_base/zav_prace_soubor_verejne.php?file_id=15888. Diplomová práce. Vysoké učení technické v Brně.

[5] STĚHULE, Pavel. Historie projektu PostgreSQL. Historie projektu PostgreSQL [online]. 2012, č. 1 [cit. 2014-12-16]. Dostupné z: http://www.root.cz/clanky/historie-projektu-postgresql/

[6] STĚHULE, Pavel. PostgreSQL [online]. 2011-11-19 [cit. 2014-11-24]. Dostupné z: http://postgres.cz/wiki/PostgreSQL

[7] PostgreSQL - About. PostgreSQL [online]. 2014 [cit. 2014-11-24]. Dostupné z: http://www.postgresql.org/about/

[8] PostgreSQL 9.3.5 Documentation. PostgreSQL [online]. 2014 [cit. 2014-11-24]. Dostupné z: http://www.postgresql.org/docs/9.3/interactive/index.html

[9] Strojové učení. Gauss Algorithmic [online]. 2014 [cit. 2014-11-24]. Dostupné z: http://www.gaussalgo.com/strojove-uceni/

[10] INTELIGENTNÍ KLASIFIKACE PŘÍZNAKŮ PRO PODPORU DIAGNOSTIKY GLAUKOMU [online]. Brno, 2012 [cit. 2014-11-24]. Dostupné z: http://www.vutbr.cz/www_base/zav_prace_soubor_verejne.php?file_id=53062. Bakalářská práce. Vysoké učení technické v Brně.

[11] Jihočeská univerzita v Českých Budějovicích. *Ekonomická fakulta* [online]. 2006 [cit. 2015-05-23]. Dostupné z: http://www2.ef.jcu.cz/~jfrieb/rmp/data/teorie_oa/STROMY.pdf

[12] Optimics. TREJBAL, Pavel. *Oprimics* [online]. 2014 [cit. 2015-05-23]. Dostupné z: http://www.optimics.cz/c/jak-na-rozhodovaci-stromy

[13] Algoritmus k-means. KUČERA, Jiří. *Shluková analýza* [online]. 2008 [cit. 2015-05-23]. Dostupné z: http://is.muni.cz/th/172767/fi_b/5739129/web/web/kmeans.html

[14] *MADlib* [online]. 2015 [cit. 2015-05-23]. Dostupné z: http://madlib.net

[15] *MacPorts* [online]. 2002 [cit. 2015-05-23]. Dostupné z: https://www.macports.org

[16] Příprava a klasifikace biomedicínských dat. JIŘINA, Marcel. *FBMI ČVUT* [online]. 2014 [cit. 2015-05-25]. Dostupné z: http://www.fbmi.cvut.cz/files/nodes/612/public/prednaska%20jirina.pdf

SEZNAM SYMBOLŮ, VELIČIN A ZKRATEK

ACID	Atomicita, Konzistence, Izolovanost, Trvalost
ANSI	American National Standarts Institute (Signál ve frekvenční oblasti)
BSD	Berkeley Software Distribution (licence pro svobodný software)
DBMS	Database Management Systém
DCL	Data Control Language (Jazyk pro řízení dat)
DDL	Data Definition Language (Jazyk pro definici dat)
DML	Data Manipulation Language (Jazyk pro manipulaci s daty)
DQL	Data Query Language (Jazyk pro dotazování)
RDBMS	Relational DataBase Management System
RDL	Report Definition Language
SE-QUEL	Structured English Query Language (Strukturovaný Angl. Dotazovací Jazyk)
SQL	Structured Query Language (Strukturovaný Dotazovací Jazyk)
SŘBD	Systém řízení báze dat

OBSAH PŘILOŽENÉHO DVD

Na přiloženém DVD se nachází tyto soubory:

- f_selection.txt - Obsahuje zdrojový kód funkce pro dopředný výběr příznaků.
- Tento dokument ve formátu pdf.
- Data set použitý při druhém příkladu.