VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV INFORMAČNÍCH SYSTÉMŮ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF INFORMATION SYSTEMS

ANALÝZA A MONITOROVÁNÍ OBCHODNÍCH PROCESŮ

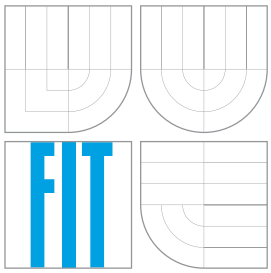
BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE
AUTHOR

MARTINA PROCHÁZKOVÁ

BRNO 2013

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV INFORMAČNÍCH SYSTÉMŮ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF INFORMATION SYSTEMS

ANALÝZA A MONITOROVÁNÍ OBCHODNÍCH PROCESŮ

ANALYSIS AND MONITORING OF BUSINESS PROCESSES

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE
AUTHOR

MARTINA PROCHÁZKOVÁ

VEDOUCÍ PRÁCE
SUPERVISOR

Ing. MILAN POSPÍŠIL

BRNO 2013

Abstrakt

V první části se práce zabývá dolováním z procesů, dolováním z dat obecně, metodami pro klasifikaci a predikci, managmentem podnikových procesů a simulací. V druhé části práce je popsán program na vytváření simulačních dat a testování metody rozhodovacího stromu a k-nejbližších sousedů.

Abstract

First part includes study on Process Mining, Data Mining in general, classification and prediction methods, business process management and simulation. Second part describes programm made for creating simulation data and testing decision tree and k-nearest neighbour method.

Klíčová slova

dolování z dat, dolování z procesů, management podnikových procesů, algoritmus k-nejbližších sousedů, rozhodovací strom

Keywords

data mining, process mining, business process management, k-nearest neighbour algorithm, decision tree

Citace

Martina Procházková: Analýza a monitorování obchodních procesů, bakalářská práce, Brno, FIT VUT v Brně, 2013

Analýza a monitorování obchodních procesů

Prohlášení

Prohlašuji, že jsem tuto bakalářskou práci vypracovala samostatně pod vedením pana Ing. Milana Pospíšila. Uvedla jsem všechny literární prameny a publikace, ze kterých jsem čerpala.

.....
Martina Procházková
15. května 2013

Poděkování

Chtěla bych poděkovat svému vedoucímu práce Ing. Milanovi Pospíšilovi za vedení při zpracovávání bakalářské práce.

Obsah

Kapitola 1

Úvod

Cílem této práce je popsat oblast dolování z dat (*Data Mining*), dolování z procesů (*Process Mining*), modelování a simulace a dalších s tématem souvisejících pojmů, především jednotlivé metody pro klasifikaci a predikci údajů podnikových procesů. Dále vytvořit testovací obchodní procesy fiktivní firmy, ty simulovat a poté na simulací vytvořených datech testovat některé metody určené k dolování z procesů.

V první fázi bude simulován fiktivní proces, který generuje data pomocí předem definovaných závislostí, v těchto datech jsou pro každý výrobek generovány časy výroby atributů na různých strojích. Právě tyto časy výroby je poté možné předpovídat pomocí metod pracujících s dříve vygenerovanými daty. Jejich použití přinese přesnější výsledky než by bylo použití triviální metody (jako například obyčejný výpočet průměru dříve generovaných časů).

V druhé kapitole je popsán pojem management obchodních procesů, několik modelů používaných pro reprezentaci systému procesů a základy simulace. Ve třetí kapitole je vysvětlena problematika dolování z dat a procesů (s tím související předzpracování), jednotlivé metody pro klasifikaci a predikci. Ve čtvrté kapitole je pak popsán samotný program, jeho specifikace, návrh, struktura a funkce. Pátá kapitola se zabývá kokrétně implementací zkoumaných metod. Experimenty provedené na programu a jejich zhodnocení je pak popsáno v kapitole šesté a sedmé.

Kapitola 2

Analýza a monitorování obchodních procesů

Vzhledem k tomu, že počítače nejrůznějších druhů pronikají do všech oblastí výroby a obchodu, hromadí se data. Zpracováním těchto dat můžeme získat informace, kterými lze zlepšit efektivitu práce, detekovat krizová místa provozu nebo třeba lépe zacílit reklamu. Softwarovým programem provedená analýza již existujících dat ani nemusí vyžadovat sbírání dalších informací, než jaké již jsou sbírány a firma tak není nucena narušovat svůj chod.[3]

Obchodním procesem se rozumí řetězec činností, které mají za cíl vytvoření specifického výstupu pro konkrétního zákazníka nebo trh. Optimalizací procesů se zabývá postup označovaný jako *BPR* (*Business Process Re-engineering*), který může mít za cíl nejen změny v procesech samotných, ale také v organizační a kvalifikační struktuře podniku.[2]

2.1 BPM

BPM (*Business Process Management* - management obchodních procesů) je přístup, který se snaží nahlížet na studované obchodní procesy jako na celek, který mění za účelem přizpůsobení se potřebám klientů pomocí průběžného procesu optimalizace. Každý produkt, který firma vyrábí, je výstupem několika provedených činností. Tyto činnosti lze pomocí podnikových procesů pojmenovat a zorganizovat. Jeho účelem je také vytvoření strategie reagování na neočekávané změny.

Pro vytvoření optimálních změn procesů je třeba nahlížet na ně z těchto základních pohledů:

- Design procesu - určuje jaké úkony jsou vykonávány, kým, kdy, kde, za jakých okolností, s jakou přesností a za použití jakých informací. Vytváří tak nutnou specifikaci procesu.
- Metriky procesu - musí být stanoveny tak, aby vycházely z potřeb zákazníků a firemních cílů. Mezi základní metriky patří cena, rychlost a kvalita. Výsledné metriky musí být stanoveny tak, aby úspěch v jedné oblasti nezakrýval propad v jiné.

- Vykonavatelé procesu - lidé, kteří se podílejí na procesu, musí mít dané schopnosti, musí rozumět průběhu procesu a jeho cílům a musí mít schopnost pracovat v týmu i samostatně.
- Infrastruktura procesu - vykonavatelé musí mít vhodnou podporu ze strany informatiků, manažerů a oddělení lidských zdrojů, jinak může být struktura procesu narušena.
- Vlastník procesu - je osoba, která je za daný proces a jeho výstup zodpovědná. Existence vlastníka je základním předpokladem pro správně fungující procesně orientovanou správu chodu firmy.

Management procesů respektuje tyto principy:

- Všechna práce v systému je práce procesů, včetně nejmenších úkonů. To ale nemá vést k automatizaci a rutinizaci práce a omezení individuální kreativní práce lidí. Definováním a zpracováním procesů mají být jednoduché pracovní úlohy umístěny do složitějšího kontextu. Základem práce managementu je pak definování základních procesů (transakčních a výrobních), podpůrných a řídících procesů.
- Je-li v systému nevhodný proces, musí být nahrazen novým, nejlépe v kontextu nového a lepšího celkového návrhu.
- Jedna verze procesu je lepší než více - standardizace procesů má přínos pro mnoho stran - zákazníky, dodavatele, podpůrné služby jako je zaučování a IT podpora. Mimo to je pak možný mnohem snazší přesun pracovníků mezi jednotlivými částmi organizace.
- I dobrý proces musí být prováděn efektivně. Dobře vytvořený proces je nutný, ale musí být dohlédnuto na to, že je návrh procesu prováděn správně i v praxi.
- I dobrý proces může být lepší. Vlastníci procesů musí neustále hledat příležitosti pro vytvoření změn vedoucích k lepšímu výkonu procesů.
- Každý dobrý proces se časem stane špatným, protože se mění potřeby zákazníků, technologie, konkurence atd. Po nějakém čase je proto třeba nahradit původní návrh lepším.

[2]

2.2 Modely procesů

S tím, jak se obchodní procesy stávají složitějšími a více závislými na informačních systémech, se vytváření kvalitních modelů obchodních procesů stává čím dál tím žádanější. Vysoká míra abstrakce umožňuje mnohem lepší pochopení problému pro všechny strany a nabízí se také možnost analyzování modelu a hledání významných faktorů, následnou optimalizaci výrobních procesů a predikci. Ve velkých firmách fungují modely také jako zdroj přesných instrukcí pro koordinaci procesů probíhajících mezi různými částmi firmy.[1]

2.2.1 Petriho sítě

Petriho síť je struktura obsahující místa a přechody, mezi místy se pohybují tokeny. Struktura modelu je statická, tokeny se v síti pohybují dynamicky. Přechody mohou být časované, prioritní nebo pravděpodobnostní, místa mohou mít určenou kapacitu, která je defaultně nekonečná. Za místem se musí v diagramu vždy nacházet přechod a za přechodem zase místo.

Petriho síť je reprezentována trojicí $N = (P,T,F)$, kde P je konečná množina míst, T je konečná množina přechodů takových, že

$$P \cap T = \emptyset \tag{2.1}$$

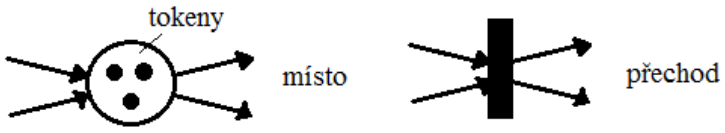
a

$$F \subseteq (P \times T) \cup (T \times P) \tag{2.2}$$

je incidenční relace (relace mezi místem a přechodem). Dále je definován počáteční stav, kapacita míst a váhová funkce incidenčních relací.

Petriho sítě modelují paralelismus, komunikaci a synchronizaci procesů a nedeterminismus. Mimo to mohou obsahovat systém hromadné obsluhy (*SHO*).

Jejich síla spočívá v jednoduchosti a v silných teoretických základech, ze kterých vychází. Problémem těchto sítí je ale fakt, že mají problém se zobrazením aspektů týkajících se dat a času. S těmito problémy se umí vypořádat varianta *CPN* (*Colored Petri Net*), ve které tokeny nesou data s sebou a mají časovou značku (*timestamp*) určující nejmenší jednotku času, po jejímž uběhnutí může být token zpracován - do systému je tak přidán moment čekání tokenů.

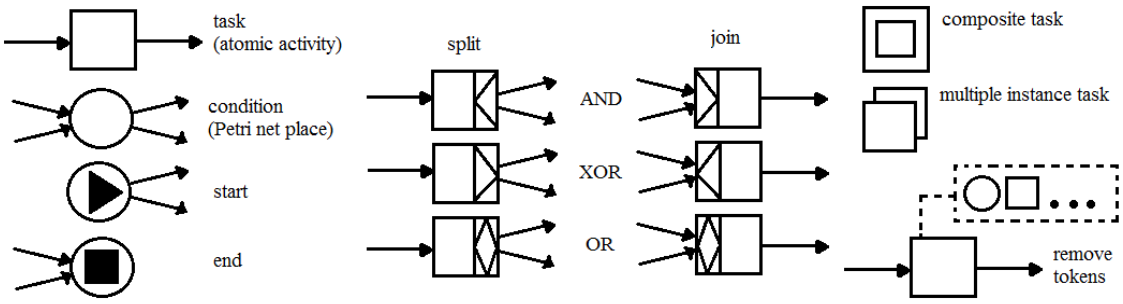


Obrázek 2.1: Symboly Petriho sítí

[1]

2.2.2 YAWL

YAWL (*Yet Another Workflow Language*) je jazyk postavený na workflow - sestává ze sekvence kroků spojených za sebou tak, že v momentě, kdy jeden krok skončí, hned bez prodlevy začíná další. Sekvence kroků je prací lidí nebo strojů a tok bývá vyjádřen jako dokument nebo produkt, který je předáván mezi jednotlivými kroky. Pro definici (a manipulaci) dat, rozhraní a komponent používá popisy založené na *XML*. Tento jazyk je dostupný v open source programu vydaném pod licencí *LGPL*.[1]



Obrázek 2.2: YAWL symboly[1]

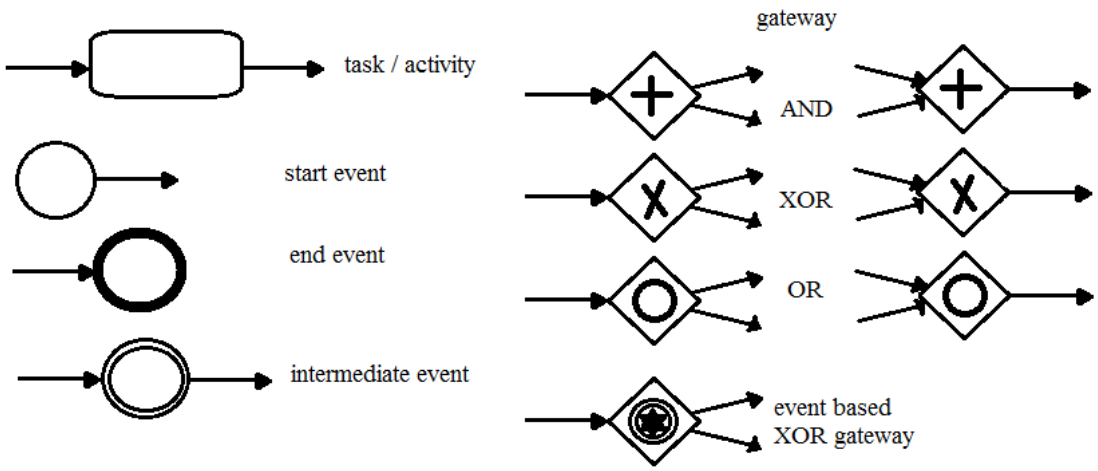
2.2.3 BPMN

Společností *OMG* standardizovaná notace *BPMN* (*Business Process Modeling Notation*) pro modelování podnikových procesů se podobá diagramu aktivit z *UML* a je jazykem, kterému díky velmi jednoduchým diagramům můžou porozumět nejen vývojáři a analytici, ale také lidé, kteří pracují v provozu modelované výroby. Účelem tohoto modelu je také usnadnění komunikace částí systému (umožňuje modelování *end-to-end* procesů, které rozdělují diagram na sekce) a podpora rozhodování založená na technikách jako je analýza ceny, analýza možných scénářů a simulace. Pro činnosti se v modelu používají funkce mající právě jeden vstupní a právě jeden výstupní argument.

Jeho strukturu tvoří:

- Objekty toku (události, aktivity a brány) - události jsou počáteční, průběžná a koncová, aktivity se skládají z podprocesů a úloh, brány tvoří větvení či spojení toku (pomocí logických operací *AND*, *OR* a *XOR*).
- Spojovací objekty (sekvenční tok, tok zpráv, asociace).
- Plavecké dráhy (bazény, dráhy) - slouží k organizaci a určují kategorie činností - bazén je tvořen drahami, které obsahují objekty toku, artefakty a další.
- Artefakty (datové objekty, skupiny, anotace) - přidávají další informace, datové objekty obsahují data nezbytná pro vykonání činnosti, skupiny seskupují aktivity i před hranici bazénu.

[1]



Obrázek 2.3: BPMN symboly

2.3 Simulace

Základem simulace je model, což je zjednodušená reprezentace části reality. Na vytvořeném modelu jsou pak prováděny experimenty, které přinesou informace o realitě a to často za podstatně menší cenu a rychleji než při experimentování přímo s reálným systémem. Někdy je modelování dokonce jedinou možností, například matematické a fyzikální modely popisující přírodní jevy jsou základem vědy a techniky. V simulaci jde také provádět experimenty podstatně větší rychlostí a jejich provádění je zcela bezpečné a nespotřebovávají materiál.

V diskrétním systému je změna stavu popsána jako událost, ta je z hlediska doby trvání atomická operace - proběhne celá v jednom okamžiku modelového času, má tedy nulovou dobu trvání a její implementace je obvykle triviální (jde o jednoduchou funkci). Posloupnost vykonávaných událostí je definována jako proces. Zpracování procesů probíhá v reálném systému paralelně, v simulaci může být zastoupena takzvaným kvaziparalelismem - tj. zpracování procesů paralelně (v jednom časovém okamžiku) na jednoprocesorovém počítači. Je-li použita tato metoda zpracování, je nutné umět se vypořádat s problémy, které mohou vzniknout, pokud záleží na pořadí v jednom okamžiku provedených operací.

Průběh vytváření simulace:

- První etapou v procesu simulace je vytvoření abstraktního modelu, což je zjednodušený popis reality. Míra abstrakce, výběr objektů a jejich atributů musí být přizpůsoben tak, aby přinesl co nejmenší zkreslení klíčových informací. Atributy pak mohou být diskrétní nebo spojité.
- Poté je vytvořen model simulační, který zapisuje abstraktní model formou programu. Atributy jsou po strojové zpracování diskretizovány. Pro získání správných výsledků simulace je zásadní, aby byl model validní a verifikovaný. Hned ze začátku procesu simulace je provedena verifikace, která zajistí, že model odpovídá reálnému systému neboli simulační a abstraktní model mají stejnou strukturu a chování. Validace se pak

pokouší dokázat, že je model adekvátní a produkuje použitelné výsledky. Validace modelu je náročná a průběžná práce provázející celý proces modelování a simulace.

- S použitím simulačního modelu proběhnout experimenty, které jsou adekvátně zaznamenány.
- Nakonec jsou výsledky analyzovány pomocí statistiky, vizualizace a dalších metod, provede se ověření jejich věrohodnosti.

2.3.1 Systémy hromadné obsluhy

Systémy hromadné obsluhy (*SHO*) poskytují obsluhu transakcím. Po příchodu je transakce obsloužena na obslužné lince nebo čeká ve frontě. Pokud je linka obsazena, vytváří se fronta. Čekání transakcí v této frontě je vhodné omezit, dosáhnout ideálního zatížení linky a optimalizovat tak systém.

Na vstup SHO přichází požadavky (zákazníci, objednávky, výrobky), popis procesu příchodů je obvykle stochastický s definovanou střední dobou mezi příchody (např. exponenciální rozložení) nebo je určen počet příchodů za jednotku času (např. poissonovo rozložení).

Fronty požadavků mohou být typu *FIFO* (první příchozí je první obsloužen), *LIFO* (poslední příchozí je první obsloužen), *SIRO* (prvky z fronty se vybírají náhodně) nebo může být fronta prioritní, čekající prvky mohou po určitém čase frontu samy opustit (takzvaná fronta s netrpělivými požadavky) nebo mohou být v případě omezeně dlouhé fronty zahazovány. Do fronty příchozí transakce pak mohou mít prioritu procesu (příchozí transakce předbíhají ostatní ve frontě) nebo prioritu obsluhy (příchozí transakce přeruší a nahradí transakci probíhající na lince).

[6]

Kapitola 3

Metody analýzy

3.1 Data Mining a Process Mining

Dolování z dat neboli *Data Mining* je způsob, kterým lze získat informace z velkého množství dat a to takové informace, které nelze zjistit žádnou jednoduchou metodou, která se obvykle používá k přístupu do databáze (vyhledávací dotaz). Jde tedy o informaci, která není zřejmá, nelze ji získat jiným způsobem a její hledání se vyplatí, protože přinese užitek - výzkumný nebo komerční.[3]

Pojem *Process Mining* v sobě zahrnuje *Data Mining* a modelování podnikových procesů, snaží se z dat vydolovat informace o procesech. *Process Mining* ke svojí práci potřebuje takzvaný *event log*, který obsahuje seznam údajů o úlohách, které v rámci každého zkoumaného procesu proběhly. Na základě takového souboru dat je možné sestavit model systému, který může být použit k analýze nebo k simulaci, simulovat jej lze i v případě, že model není zcela kompletní. Procesu sestavování modelu na základě event logu se říká *Process Discovery.*[4]

Pro zjištění, co ovlivňuje tok objektů systémem, se používá dolování rozhodnutí (*Decision Mining*). Informace o průběhu toku objektů je možné získat i z reálného systému, takové rozhodnutí je ovšem často příliš zjednodušené, postavené na malém počtu atributů známých v momentě hodnocení.[4]

3.2 Předzpracování dat

Data, která obsahují mylné informace, mohou výsledek analýzy zcela znehodnotit, předzpracování dat tak často tvoří větší část práce. Nejčastější závady dat:

- Data mohou být neúplná, informace mohou chybět z důvodů technických výpadků, lidských chyb nebo odlišných požadavků při zaznamenání dat a při jejich analýze.
- Může dojít k tomu, že data obsahují evidentně chybné hodnoty (šum), k jejichž zápisu došlo při poruše, v momentě zaznamenávání nebo zpracování.
- Pokud se pokoušíme spojit dohromady záznamy z více různých zdrojů, snadno můžeme nalézt rozpor v hodnotách, což přináší nutnost vytvoření mechanismu pro výběr vý-

sledné hodnoty. Pokud jsou hodnoty záznamu mezi sebou logicky propojeny, může k takovému problému (nekonzistence dat) docházet i pouze v jednom zdroji.

- Data jsou příliš rozsáhlá a/nebo obsahují duplicitní záznamy.

Kvalitní data jsou:

- Přesná (vztah vzhledem k realitě)
- Věrohodná
- Aktuální
- Konzistentní (neexistují záznamy s navzájem si odporujícími informacemi)
- Úplná (obsahují všechny záznamy a u každého všechny atributy)
- S přidanou hodnotou (jejich sbírání zapříčiní vznik nových znalostí)
- Relevantní
- Srozumitelná

Předzpracování dat probíhá v tomto pořadí:

1. Čištění dat
2. Integrace dat z více zdrojů
3. Redukce dat
4. Transformace dat
5. Dolování z dat
6. Hodnocení modelů a vzorů
7. Prezentace znalostí

3.2.1 Odstranění šumu

První možností je použití plnění (*binning*) - data jsou uspořádána a poté rozdělena do tzv. košů (s přibližně stejnou četností hodnot), poté jsou hodnoty všech prvků v koši nahrazeny průměrem, mediánem atp. Podobné je použití shlukování - shluky mají na rozdíl od košů různou četnost a vytváří se jiným způsobem. Vyhladit šum lze také za pomoci regresní funkce (tj. funkce, která se snaží co nejlépe vystihnout rozložení hodnot v datech).

Často se stane, že vytvoření správných hodnot pro nahrazení šumu pomocí programu je problematické, nevytváří vhodné hodnoty nebo označuje za šum i správné hodnoty. V takovém případě je vhodné použít kombinovanou kontrolu, při které program vyhledá potenciálně chybné hodnoty a člověk poté rozhoduje, zda jde o chybu a čím ji nahradí.

3.2.2 Redukce dat

Cílem redukce je vytvořit takovou reprezentaci dat, jejíž analýza vrací stejné výsledky jako analýza původních dat, ale zabírá mnohem méně paměti a analýza probíhá rychleji.

Redukce dimenzionality

Pro redukci dimenzionality můžeme použít dva postupy - selekce (*Feature selection*), při které vybereme relevantní podmnožinu dat/rysů a ostatní zanedbáme, a extrakce (*Feature extraction*), kdy se pokusíme vybrat pro popis dat/rysů záznamů deskriptory, které popisují data jiným, efektivnějším způsobem.

Pro extrakci se používá například metoda *PCA* (*Principal Component Analysis*), která používá statistické charakteristiky dat reprezentované kovarianční maticí, odpovídající vlastní vektory a další hodnoty.

Vzorkování

Cílem této metody je získání menšího vzorku dat, který vhodně reprezentuje všechna zpracovávaná data. Na to lze použít tzv. jednoduchý náhodný vzorek bez návratu (*SRSWOR*), který vznikne náhodným výběrem a přesunem n-tic, každé se stejnou pravděpodobností. Pokud použijeme metodu s návratem (*SRSWR*), vybrané n-tice se zkopírují, takže zůstávají ve zdrojovém souboru i pro další kroky výběru.

Shlukování

Metoda používající vzorek shluků rozděluje data na vzájemně disjunktní shluky (podle vlastností prvků nebo umístění v paměti), z nichž jsou některé náhodně vybrány. Shluky jsou vytvářeny tak, aby data ve shlucích byla co nejpodobnější a data mezi jednotlivými shluky co nejvíce odlišná. Kvalita shluku je definována jako průměr kružnice reprezentující maximální vzdálenost dvou záznamů ve shluku nebo jako průměrná vzdálenost prvku od centra shluku (průměrný objekt nebo průměrná pozice v prostoru).

Aby byla shluková metoda použitelná, musí umět zpracovat velké množství dat, což je problém u mnoha metod, které pracují správně jen na malém objemu dat (vzorkování dat není vždy dobrým řešením). Dále musí umět zpracovat různé datové typy atributů, vytvářet shluky různých tvarů a různorodé hustoty rozmístění v prostoru. Pro vhodné zpracování metodou shlukování je také důležité z dat odstranit šum nebo použít takovou metodu, která se dovede s šumem vyrovnat. Metoda musí být schopna zpracovávat záznamy bez ohledu na jejich pořadí uložení v databázi a i v případě, že obsahují mnoho dimenzí. Výsledkem analýzy musí být interpretovatelné, srozumitelné a použitelné shluky.

[3]

3.3 Asociační analýza

Pomocí různých metod lze v datech vyhledat důležitá logická spojení (asociační pravidla) mezi na první pohled nesouvisejícími proměnnými (např. analýza potvrdí, že zákazníci, kteří si koupili noviny, si koupí koblihy o 80% častěji než ostatní). Takové informace hledá takzvaná analýza nákupního košíku, jejím cílem je nalezení znalosti o chování zákazníka, výskytech častých kombinací zboží v jeho nákupním košíku. Tuto znalost lze později využít například pro uspořádání zboží v prodejně, nabízení speciálních akcí nebo pro specifikování zákaznických skupin.

Asociační pravidlo má formu:

$$if X then Y \tag{3.1}$$

nebo

$$X \rightarrow Y \tag{3.2}$$

Podpora a spolehlivost

Podpora (*support*) je pravděpodobnost určující, s jakou četností se pravidlo vyskytuje v databázi. Spolehlivost (*confidence*) je podmíněná pravděpodobnost určující, kolik řádků v tabulce obsahujících položky z X obsahuje také položky z Y - jak silná je implikace v asociačním pravidle.

Zdvih

Kromě základních charakteristik se používají další hodnoty pro určení skutečné užitečnosti pravidla, jako například zdvih (*lift*), který určuje poměr pozorované podpory pravidla vůči k hodnotě, kterou bychom pozorovali, kdyby X a Y byly nezávislé. Pokud jsou nezávislé, je hodnota *lift* blízko k 1, čím je pak větší, tím užitečnější pravidlo je (například hodnota 3 znamená, že se pravidla vyskytují třikrát častěji společně).

Další charakteristiky

Na základě počtu zpracovávaných položek (atributů) může jít o pravidlo booleovské, které určuje pouze přítomnost dané položky, nebo o pravidlo kvantitativní, které používá kombinace více hodnot. Všechny tyto hodnoty včetně výsledné v součtu určují počet dimenzí pravidla. Úroveň abstrakce pak určuje, kolik informací pravidlo používá a kolik je zanedbáno.

Práce programu

Do zpracovávajícího programu uživatel zadá minimální podporu a minimální spolehlivost pravidel. Množina položek s podporou vyšší než je zadaná minimální hodnota se nazývá frekventovaná množina, pravidlo, které má navíc i spolehlivost vyšší než je zadaná hodnota, se nazývá silné asociační pravidlo.

Prvním krokem při hledání pravidel je generování frekventovaných vzorů (nalezení pravidel s podporou vyšší nebo stejnou jako je zadaná minimální podpora) a nalezení silných množin. Pomocí těchto znalostí generujeme asociační pravidla, ze kterých poté odstraníme pravidla s nízkou spolehlivostí.

Algoritmus Apriori

Algoritmus hledá taková pravidla, která splňují takzvanou *Apriori vlastnost* - ta určuje, že každá neprázdná podmnožina frekventované množiny je také frekventovaná. V prvním (spojovacím) kroku výpočtu prochází databázi a postupně počítá podporu jednotlivých kandidátů (potenciální pravidla), v druhém (vylučovacím) kroku se z množiny odstraní pravidla, která nesplňují *pravidlo Apriori*, změní se minimální podpora a zvětší se množina kandidátů. V každém cyklu tento postup prochází celou databázi, jde proto o velmi pomalý algoritmus.

[1]

3.4 Metody pro klasifikaci a predikci

Získání informace pomocí dolování z dat vyžaduje složité postupy a/nebo výkonné výpočetní nástroje. Pokud jde navíc o obrovské množství dat, nevhodně předzpracovaná, složitá data nebo o data s vysokou dimenzionalitou (každý záznam obsahuje mnoho různých položek), může být analýza zdlouhavá. Výběr správné metody je proto klíčový.

Proces získávání znalostí lze popsat těmito kroky:

1. Stanovení cílů
2. Předzpracování dat
3. Výběr prostředků
4. Získání znalostí
5. Interpretace výsledků

[3]

3.4.1 Rozhodovací strom

Pro data mining se rozhodovací stromy používají často především díky přehlednosti a srozumitelnosti. Strom používá metodologii rozděl a panuj - rozděluje problém na menší úlohy, které dovede vyřešit. Objekty rozděluje do skupin a to tak, že uzel, který reprezentuje jeden z atributů objektu, rozdělí dle hodnot, kterých objekty u daného atributu nabývají. Dále se strom dělí znova podle jiného atributu. Protože počet podstromů je konečný, je třeba mít v objektu pouze diskrétní atributy nebo je diskretizovat. Pouze terminální listy takového stromu pak obsahují třídy podobných objektů.

Pro vytváření stromu je třeba použít mechanismu, který rozhoduje, jaké atributy se použijí k rozdělení jednotlivých uzlů. Ideálně jde o ten atribut, který rozděluje objekty na

nejvíce odlišné skupiny (tzv. *nejsilnější atribut*). Strom je vytvořen na trénovacích datech - ty umožní vytvoření stromu, který poté umí předpovědět hodnoty u nových dat tak, že vyhledá ve stromu dříve generované objekty, které jsou momentálnímu podobné.

Rozhodovací strom podobný tomu na obrázku různými způsoby a většina z nich používá algoritmus postupující rekurzivně shora dolů shrnutý v těchto bodech:

1. Vytvoří se uzel a všechny instance (trénovacích dat) se do něj přiřadí.
2. Pokud je množina instancí prázdná, skončí.
3. Odebere všechny instance z množiny.
4. Určí, zda je možné a nutné další rozdělení uzlu. Pokud ne, pokračuje krokem 2.
5. Pro všechny atributy určí, jaký efekt by mělo použití atributu pro rozdělení uzlu. Vybere takový atribut, který přinese nejlepší zlepšení stromu. Stejný atribut se nesmí na jedné cestě stromem objevovat víckrát. Pro numerické hodnoty atributů musí být v tomto místě definovány krajní meze výsledných podmnožin.
6. Na základě vybraného atributu vytvoří příslušné synovské uzly.
7. Do každého uzlu přiřadí příslušné instance a pokračuje krokem 2.

Při vytváření konkrétního algoritmu je nutné určit moment, ve kterém se přestanou přidávat uzly. To může být při dosažení určité výšky stromu. U numerických hodnot musí být uzel rozdělen na přiměřený počet poduzlů (nevytváří se podstrom pro každou hodnotu).

[1]

3.4.2 Algoritmus k-nejbližších sousedů (k-NN nebo KNN)

V tomto algoritmu lze rozpoznat frekventované vzory, využívá metodu učení s učitelem. Prvky jsou definovány jako vícedimenzionální vektory a jsou rozdělovány do tříd. V principu lze tímto postupem predikovat vlastnosti prvku podle toho, jaké vlastnosti mají jemu podobné dříve zpracované prvky.

Ve fázi učení se trénovací množina od učitele zpracuje tak, že každý prvek je umístěn do n-rozměrného prostoru. Pokud jsou hodnoty v prvku diskrétní, použije se pro ohodnocení jejich vzájemné vzdálenosti *Hammingova klasifikace*. Hammingova vzdálenost je definována jako nejmenší počet pozic, ve kterých se řetězce stejné délky (binární slova, vektory) liší neboli počet záměn, které je třeba provést pro změnu z jednoho řetězce na druhý (například binární slova 01101010 a 00001010 mají Hammingovu vzdálenost 2).

Ve fázi klasifikace umístí algoritmus prvek určený ke zpracování do prostoru a vyhledá k nejbližších sousedů, prvek pak spadá do stejné třídy jako většina jeho nejbližších sousedů. Jednotliví sousedé mohou mít různý vliv na výsledek dle toho, jak blízko ke zkoumanému prvku se vyskytují. Má-li k hodnotu 1, vyhledá se pouze jeden nejbližší soused.

[3]

3.4.3 Regresní analýza

Pokouší se vytvořit odhad hodnoty pomocí statistických metod, používá pojmy závisle proměnná (kterou se pokoušíme odhadnout) a nezávisle proměnné (jejich znalost používáme pro odhadování). Pokud je závisle proměnná skalár nebo vektor z lineárního prostoru, používá se pro takový odhad výpočet střední hodnoty nebo regresní funkce. Pokud je závisle proměnná diskrétní, používá se výpočet podmíněné pravděpodobnosti (příslušnosti do třídy) a analýza se označuje jako diskriminační.

[1]

3.4.4 Regresní strom a les

Regresní strom je podobný jako rozhodovací strom, ale pro rozhodování uzlů používá místo kategorických atributů numerické hodnoty. Nedosahuje takové přesnosti jako algoritmus k-nejbližších sousedů, ale jeho výpočetní rychlost je mnohonásobně větší. Na jeho stavění se používá varianta algoritmu *ID3*.

Spojením více regresních stromů vzniká les regresních stromů, jehož extrémní formou je náhodný les, což je velmi precizní a rychlá struktura, sémanticky podobná algoritmu k-nejbližších sousedů. Stavění lesa je velmi zdlouhavý výpočet, protože metoda vyžaduje sestavení více než stovky stromů. Při stavění náhodného lesa je pro každý strom vybrána náhodná podmnožina záznamů z trénovacích dat (asi 2/3) a náhodná podmnožina atributů (asi 2/3). Strom není prořezáván a je sestaven celý.

[5]

3.4.5 Neuronové sítě

Neuronové sítě jsou struktury umělé inteligence používané pro distribuované paralelní zpracování dat. Ukládání, zpracování a předávání informace probíhá prostřednictvím celé neuronové sítě spíše než pomocí konkrétních paměťových míst. Použitím neuronových sítí lze předvídat vývoje časových řad, zpracovávat obraz a zvuk, nemalý význam mají i pro výzkum nervových soustav živých organismů.

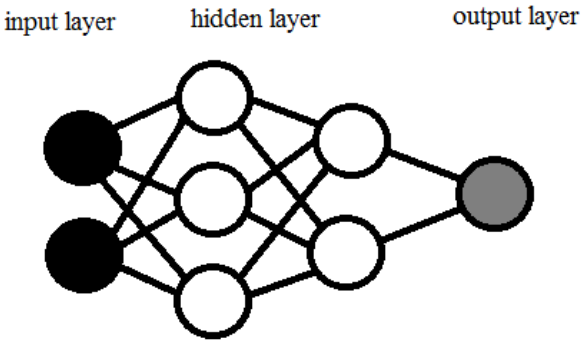
Jednotlivé umělé neurony systému jsou vytvořené podle skutečných neuronů mozku - mají libovolný počet vstupů (ve skutečném neuronu takzvané *dendrity* - na obrázku 3.1 nalevo) a jediný výstup (*axon* - na obrázku napravo). Mimo to má neuron synaptické váhy, které určují důležitost vstupu, práh, který je nutno překročit pro aktivaci neuronu a přenosovou funkci, která určuje, co proběhne, pokud je neuron aktivován. Znalosti jsou uloženy pomocí síly vazeb mezi jednotlivými neurony a ty vazby, které vedou ke správné odpovědi, jsou průběžně zesilovány a naopak - ostatní jsou zeslabovány.[7]

Výhodou neuronových sítí je, že mají schopnost učení - pouze na základě dříve zpracovaných vstupů modifikují svoji strukturu. Také umějí problém zobecnit, takže si poradí i se vstupy, které ještě nikdy dříve nebyly zadány. Pro počáteční vytvoření struktury neuronové sítě dostane program sadu vstupů a výstupů, které mají žádoucí výsledky. Tyto informace vytvoří člověk (programátor) - jde o takzvané učení s učitelem. Síť se potom sama přizpůsobí a všechny svoje výsledky se snaží tvořit tak, aby odpovídaly principům v učitelem zadaných

Obrázek 3.1: Schéma skutečného neuronu

případech. Neuronová síť je také schopná se relativně dobře vyrovnat se ztrátou některé své části, protože funkce zničené části prostě převezme zbytek sítě (podobně jako reálný mozek).

Nejrozšířenějším použití je propojení jednotlivých bodů do vícevrstvé sítě (na obrázku 3.2 jsou zobrazeny čtyři vrstvy - vstupní, výstupní a dvě skryté). Síť musí mít minimálně tři vrstvy a vždy mezi dvěmi sousedními vrstvami je tzv. úplné propojení neuronů (každý neuron nižší vrstvy je spojen se všemi neurony vyšší vrstvy).[7]



Obrázek 3.2: Neuronové sítě

3.4.6 Naive Bayesian klasifikace

Tato metoda klasifikace vytváří statistické klasifikátory, které predikují pravděpodobnost příslušnosti k dané třídě. Naivita přístupu spočívá v tom, že předpokládá, že hodnota atributu je nezávislá na hodnotách ostatních atributů. Tento předpoklad je použit ke zjednodušení výpočtů, ale může také vést k přílišnému zjednodušení. Rychlost Bayeské klasifikace je srovnatelná s rychlostí rozhodovacího stromu a neuronových sítí, je možné ji použít pro databáze obsahující velké množství dat.

Metoda se opírá o *Bayesův teorém*, který zní takto:

$$P(H|X) = \frac{P(X|H)P(H)}{P(X)} \tag{3.3}$$

X je seznam dat považovaných za důkazy nebo také trénovací hodnoty, H je hypotéza (například, že X náleží do třídy C) a $P(H|X)$ je pravděpodobnost platnosti této hypotézy postavená na X, kromě toho existuje ještě $P(H)$, která určuje pravděpodobnost hypotézy nezávislé na datech. Pravděpodobnost $P(X|H)$ určuje pravděpodobnost výskytu dat v případě splnění hypotézy a $P(X)$ pravděpodobnost výskytu dat bez ohledu na hypotézu. Všechny hodnoty na pravé straně rovnice lze získat ze zpracovávaných dat.

Pokud bychom jako příklad použili obchod s elektronikou, mohla by $P(H|X)$ reprezentovat pravděpodobnost toho, že si zákazník daného věku a pohlaví koupí počítač, $P(H)$ pravděpodobnost, že si jakýkoli zákazník koupí počítač, $P(X|H)$ pravděpodobnost toho, pokud si zákazník koupil počítač, že má daný věk a pohlaví a $P(X)$ pravděpodobnost výskytu zákazníků daného věku a pohlaví bez ohledu na jejich nákupy.

Samotný Bayeský klasifikátor pak pracuje takto:

1. Každá položka ze seznamu trénovacích dat je reprezentována jako n-dimenzionální vektor reprezentující n hodnot atributů.
2. Pravděpodobnost je pomocí *Bayesova teorému* (H je v něm nahrazeno označením třídy C) spočítána pro m tříd a X je přiřazeno do té třídy, která má nejvyšší hodnotu pravděpodobnosti.
3. Protože $P(X)$ je pro všechny třídy stejná hodnota, pokoušíme se maximalizovat $P(X|C)$ a $P(C)$. Pokud není pravděpodobnost některých tříd známá, považuje se za rovnoměrně rozloženou mezi třídami a konstatní je pak také $P(C)$, pokoušíme se maximalizovat pouze $P(X|C)$. Právě kvůli tomu, že data s velkým množstvím atributů by byly extrémě výpočetně náročné, je pro výpočet $P(X|C)$ použit naivní předpoklad nezávislosti atributů.

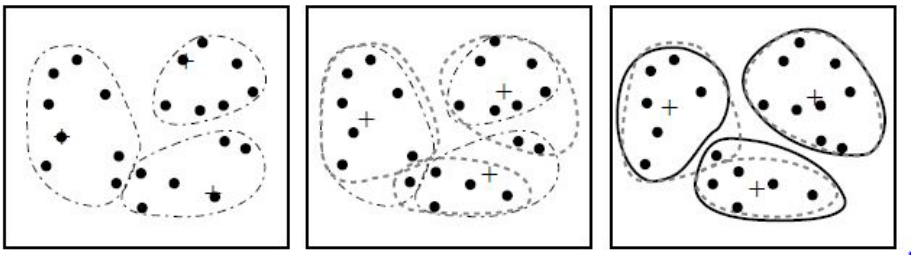
[3]

3.5 Shlukování

3.5.1 Algoritmus k-means

Tento nehierarchický algoritmus využívá shlukovou analýzu - třídí data do shluků na základě jejich vlastností. Každý objekt přiřadí do toho shluku, jehož středu je nejblíže. Po každém kroku zpřesňuje středy shluků tím, že spočítá aritmetický průměr všech bodů ve shluku.

Objekty jsou reprezentovány číselnou hodnotou, nejčastěji jako vektor v n-dimenzionálním prostoru. Na začátku se vytvoří k náhodných bodů, které jsou označeny jako středy shluků, poté se postupně přidávají další objekty a středy se přepočítávají, dokud nezačnou hodnoty konvergovat.[3]



Obrázek 3.3: Algoritmus k-means [3]

Kapitola 4

Návrh a implementace simulace systému

4.1 Specifikace

Cílem je vytvoření simulace výrobny, která umí simulovat průběh výroby (zabírat a uvolňovat stroje, vyrábět atributy výrobků atd.), zobrazit stav jednotlivých výrobků, výrobních strojů a dalších podstatných informací. Každý výrobek má mít seznam atributů, které je třeba vyrobit a každý stroj má mít seznam atributů, které dovede vyrobit. Délka výroby atributu na příslušném stroji je určena hodnotou atributu, ostatními hodnotami atributů výrobku a výkonem stroje. Výrobky se mají generovat automaticky dle vhodného rozložení pravděpodobnosti.

Výstupem simulace bude kolekce výrobků s údaji o trvání času výroby všech atributů a označení stroje, na kterém byl daný atribut zhotoven. Tyto výrobky má být možné uložit do souboru pro potřeby analýzy a predikce, která data ze souboru použije jako trénovací. Systém pak má pomocí několika různých metod Process miningu vydolovat z dat znalosti a použít je pro predikci.

Konkrétní výchozí výrobna bude obsahovat v procesu výrobku výrobní atribut, který musí být vytvořen jako první, poté sekvenci několika atributů, které je možné splnit v libovolném pořadí a poté poslední atribut, které je možný splnit až poté, co jsou splněny všechny předchozí. Pro výrobní atribut budou existovat dva stroje, pro ostatní po jednom.

4.2 Návrh

Systém bude pracovat v objektovém návrhu systému hromadné obsluhy výrobků stroji. Fronty budou dvě a to typu *FIFO*, jedna fronta v sobě bude držet výrobky, které ještě nemají zpracovaný žádný atribut a druhá ty, které už jsou rozpracované. Výrobky patřící jednotlivým skupinám budou zobrazeny na příslušném místě včetně informací o svých atributech a jejich zpracovanosti.

Stejnou simulaci bude možné použít pro generování trénovacích dat i pro vytváření dat s predikcí založenou na dříve vytvořených datech. Trénovací data bude možné uložit do

souboru a opět je načíst. Bude-li program umět predikovat hodnoty, bude u výrobků zobrazena nejen skutečná hodnota trvání výroby, ale také predikované hodnoty. Program bude fungovat jako okno s tlačítky a tabulkami na zobrazení údajů.

4.3 Implementace systému

Program byl napsán v programovacím jazyce C# jako aplikace Windows Forms v prostředí Microsoft Visual Studio. Kromě předdefinovaných tříd *Program* a *Form1* obsahuje 12 dalších tříd:

- **Vyrobek** - obsahuje informace o výrobku, který je vytvořen jako objednávka, poté změní svůj stav na rozpracovaný výrobek (takový je ve výrobně skladován poblíž výrobních strojů) a nakonec na hotový výrobek. Výrobek má jednoznačný identifikátor a atribut určující, zda je výrobek volný nebo je právě zpracováván na stroji. Dále obsahuje seznam atributů (viz třída *Atribut* - atribut jako vlastnost výrobku, nikoli jako proměnná třídy) a další proměnné určené k výpisům atp.
  Třída obsahuje metody pro generování atributů a hodnot výrobku (soukromé) a metody *Zaber*, *Vyrob*, *Hotovo* a další metody pro vyhledávání atributů.

- **ManagerVyrobku** - tato třída v sobě drží seznam všech výrobků, označení výrobny, které tyto výrobky přísluší a seznamy a informace o počtu jednotlivých skupin (objednané, rozpracované, hotové) výrobků. Proto v momentě, kdy se změní stav výrobku, zavolá výrobek metodu svého manažera *Vyrobek_Rozpracovan* nebo *Vyrobek_Hotov* a hodnota příslušící správě se změní. Informace pro správu výrobků a pomocné specializované seznamy byly použity pro zrychlení výpočtů.
  Dále tato třída obsahuje metody pro vyhledání nějakého výrobku (s nezpracovaným zadaným atributem nebo s daným id) a metody pro vytvoření nového výrobku a jemu příslušícího identifikátoru.

- **Stroj** - je definován čtyřmi základními atributy - identifikátorem, názvem, výkonem (výkon určuje, jak rychle na daném stroji trvá vytvoření atributu vzhledem k ostatním strojům) a seznamem atributů, které umí zpracovat. Dále obsahuje označení toho, zda je obsazen výrobkem, a další pomocné proměnné pro zaznamenání trvání výroby atributu a pro výpis.
  Metody stroje jsou *Zaber_Stroj*, *Uvolni_Stroj*, *Pracuj* (při opakovaném volání postupně odečítá z počtu jednotek času potřebných pro vyrobení atributu) a metoda *Loading*, která slouží pro výpis načítacího proužku stroje.

- **ManagerStroju** - podobně jako *ManagerVyrobku* obsahuje seznam všech strojů, metody pro vytvoření stroje a jeho vyhledání.
  Při každém časovém cyklu (tiknutí časovače nebo cyklu v hlavním programu) se zavolá metoda *Pracuj*, která nejdříve zavolá metodu *Pracuj* pro každý stroj, poté se pokusí všechny stroje uvolnit a nakonec se je pokusí opět obsadit vhodnými výrobky.

- **Vyrobna** - v sobě nese informace o strojích a atributech výrobků, kromě toho obsahuje ještě seznam speciálních pravidel, které mění čas vypracování některého atributu výrobku v závislosti na hodnotách některých atributů (například pokud mají být dveře vypracovány v luxusní kvalitě a z dubového dřeva, trvá výroba atributu materiál o 30% více času než obvykle). Výrobna má také svůj identifikátor a název.

- **ManagerVyroben** - opět obsahuje seznam všech možných výroben, jejich počet a jednu výrobnu, která je označená jako aktuálně používaná v simulaci. Metoda *Vytvor_Defaultni_Vyrobnu* vytvoří výrobnu dveří (popsáno v části Vytvoření výrobny dveří níže) a označí ji jako aktuální.

- **Atribut** - základní informace o atributu jsou identifikátor, název, hodnoty, kterých může nabývat a prioritu.
  Platí, že čím větší je priorita, tím menší je číslo v proměnné, a atributy s vyšší prioritou musí být vytvořeny před atributy s nižší prioritou (například nejdříve musí být vytvořen atribut materiál a až poté může být zpracován atribut broušení hran). Má-li atribut prioritu 0, znamená to, že jeho zhotovení nezávisí na stroji a je zhotoven už při vytvoření objednávky (například atributy typ, šířka a výška, které se nezpracovávají žádným strojem, a zastupuje je atribut materiál, který je zpracován na výrobním stroji). Pro konkrétní výrobek jsou ještě definovány atributy *hodnota*, *hotovo*, *vyroben_strojem* a *trvani*, které zůstávají prázdné, pokud je struktura použita jinou třídou než třídou *Výrobek*.

- **Hodnota** - základní identifikace je název hodnoty, dále obsahuje pravděpodobnost, která určuje, jak často se hodnota objevuje vzhledem k ostatním hodnotám (součet těchto hodnot u všech hodnot každého atributu je 1). Koeficient trvání určuje, jak dlouho trvá výroba atributu s danou hodnotou vzhledem k výrobě atributu s běžnou hodnotou - jako běžná hodnota byla určena nejčastější nebo nejmenší hodnota.

- **Pravidlo** - je speciální kombinací atributu pro danou výrobnu. Skládá se z identifikace dvou atributů, jejichž přítomnost ve výrobku vyvolá použití pravidla a identifikace atributu, jehož hodnota trvání se má změnit. Koeficient je číslo, kterým se hodnota trvání cílového atributu vynásobí.

- **Serializer** - tato třída byla vytvořena pro potřeby ukládání objektů do souboru a obsahuje metody *SerializeObject* a *DeSerializeObject*, každý objekt (*Vyrobek*), který je do souboru ukládán, včetně všech objektů, které obsahuje v sobě (*Atribut*, *Hodnota...*), pak obsahuje speciální konstruktor pro načtení atributů objektu ze souboru a metodu *GetObjectData* pro uložení atributů objektu do souboru.

- **Strom** - reprezentuje rozhodovací strom a je určen svým identifikátorem, atributem výrobku a jeho hodnotou, dále obsahuje informace o předcích (identifikátor otce a seznam atributů, které byly použity předky a nemohou být použity tímto stromem a potomky), informace o potomcích (atribut podstromu a samotné podstromy) a pokud je strom listem, pak i informace o výrobcích v listu (seznam výrobků a průměrné trvání jejich výroby).
  Strom obsahuje dva konstruktory, jeden pro kořen stromu a další pro podstromy (ty obsahují navíc ještě dříve zmíněné atributy *id_otce*, označení atributu, hodnoty a seznam atributů použitých předky). Pro stavění stromu jsou vytvořené dvě soukromé metody - *Vyber_Nejsilnejsi_Atribut*, která vyhledá mezi použitelnými atributy takový, u kterého mají zpracovávané výrobky nejmenší rozptyl hodnot trvání, a metoda *Vytvor_Podstromy*, která je zavolána pouze v případě, že strom je třeba rozdělit na podstromy (existuje atribut, podle kterého lze dělit a počet výrobků je větší než minimální počet pro rozdělení).

- **Analyzator** - drží v sobě všechny vytvořené stromy (pro každý stroj jedna rekurzivní struktura stromu), infomaci o tom, jestli je predikce možná (program má k dispozici

trénovací data) a jeho metoda *Vytvor_Stromy_Stroju* je zavolána právě v momentě úspěšného načtení trénovacích dat. Analyzátor pak umí prohledat celý strom a vyhledat list, který nejvíce danému výrobku odpovídá a vrátit průměrnou hodnotu výrobků v tomto listu. Zavoláním metody Predikuj jsou vytvořeny dvě predikované hodnoty - jedna metodou rozhodovacícho stromu, druhá metodou kNN.

4.3.1 Uživatelské rozhraní

Celá plocha okna je rozdělena na dvě části - první má zelené pozadí a zobrazuje objednávky výrobků a záznamy hotových výrobků, v podstatě tak reprezentuje sklad; druhá s tmavě šedým pozadím reprezentuje výrobnu, obsahuje tedy stroje, rozpracované výrobky, stavový řádek výroby a další doplňkové informace.
Program lze spustit jediným tlačítkem spustit simulaci, které se nachází v levém horním rohu, takže je v místě, na které se pravděpodobně uživatel podívá hned po spuštění programu. Nadpisy jednotlivých logických celků jsou napsány výrazně větším písmem, aby pomohly uživateli rychle se zorientovat. Celá plocha okna programu je navržena tak, aby působila na uživatele příjemně a jasně zobrazovala všechny potřebné informace.

Vzhled celého okna programu během simulace si můžete prohlédnout v příloze.

4.3.2 Režimy programu

Režim simulace

Tento režim je vytvořen s ohledem na uživatele - při jeho použití probíhá simulace sledovatelnou rychlostí a v seznamech výrobků a na strojích se zobrazují obrázky výrobků a informace o nich. Tlačítko načíst data pro predikci je povoleno, a pokud uživatel před spuštěním simulace pomocí tohoto tlačítka načte trénovací data ze souboru, u hotových výrobků se zobrazují skutečné hodnoty trvání a predikované hodnoty, které byly vytvořeny za běhu simulace.

Režim pro generování dat

Speciálně pro rychlé generování velkého množství výrobků (trénovacích dat) byl vytvořen tento režim. V něm se nezobrazují žádné obrázky ani informace o výrobcích, pouze počty výrobků jednotlivých skupin, čas a stavový řádek. Program zde nepoužívá timer jako u předchozího režimu, ale provádí výpočty bez jakéhokoli čekání.

4.3.3 Ukládání do souboru

Do souboru se vždy uloží objekt třídy ManagerVyrobku, který obsahuje všechny výrobky a informace o výrobně (tedy o atributech výrobků a strojích). Uživateli se jako základní název souboru nabídne formát *vyrobky_datum.sv*, kde datum je ve formátu *yyyy_MM_dd_HH_mm*, a jako cílová složka se otevře složka saves. Pro ukládání objektu se používá třída Serializer.

4.4 Vytvoření výrobny dveří

Pro potřeby simulace v případě, že uživatel nezadá vlastní výrobnu, byla vytvořena výrobna dveří s názvem Výrobna dveří a následujícími atributy (v závorce je priorita atributu a za dvojtečkou hodnoty atributu):

- typ (0): SA - Standard A, SB - Standard B, E - Exclusive, L - Luxury, Ch - Cheap
- šířka (0): 60, 70, 80, 90, 100
- výška (0): 197, 210, 220, 230, 240
- materiál (1): borovice, smrk, olše, buk, dub, javor
- hrana (2): normal, soft2, soft3, karnis
- sklo (2): Ku - kůra, Ko - konfeta, C - činčila, M - matelux
- kování (2): I1 - interiérové 1, I2 - interiérové 2, B - bezpečnostní
- balení (3): igelit

Výrobna dále obsahuje tyto stroje (v závorce je koeficient stroje, který určuje jeho rychlost vzhledem ke stroji A a za dvojtečkou seznam atributů, které umí zpracovat):

- Výrobní stroj A (1.0): materiál
- Výrobní stroj B (1.16754): materiál
- Stroj na broušení hran (0.27895): hrana
- Stroj na vysklení (0.19229): sklo
- Stroj na kování (0.50071): kování
- Stroj na balení (0.10543): balení

Speciální pravidla výrobny jsou definována takto (hodnotou za **then** se vynásobí trvání výroby atributu za **then**):

- **if** typ L **and** materiál dub **then** materiál *1.34687
- **if** materiál borovice **and** hrana karnis **then** hrana *1.42487
- **if** šířka 60 **and** výška 197 **then** materiál *0.96602

4.4.1 Průběh simulace

Načtení trénovacích dat ze souboru

V případě, že uživatel ještě před spuštěním simulace načte do programu trénovací data ze souboru, generují se během simulace i predikované hodnoty. To může udělat stisknutím tlačítka načíst data ze souboru. Tento krok může ale klidně vynechat, pokud nemá zájem o analýzu nebo zatím nemá k dispozici žádná uložená trénovací data.

Spuštění simulace

Před spuštěním je možné nastavit režim simulace - režim pro rychlé generování výrobků nebo režim pro zobrazení simulace, který zobrazí detailní informace a obrázky výrobků a je po spuštění programu nastaven jako výchozí. Pokud uživatel nastaví rychlý režim, není možné načíst trénovací data a v případě, že byly načteny dříve, nepoužije je, protože predikované hodnoty se nezobrazí. Po stisknutí tlačítka pro spuštění simulace se spustí časovač (v případě rychlejšího režimu se zahájí nekonečná smyčka výpočtu), který s každým tiknutím zkouší vygenerovat výrobek, obsadit a uvolnit stroje a pokročit v práci strojů na výrobcích.

Četnost generování výrobků

Výrobek se vygeneruje na základě konstanty určující kolik časových jednotek průměrně uběhne mezi vygenerováním dvou výrobků. Konkrétně pro ilustrační výrobnu byla tato hodnota stanovena na 70 (tj. vytvoří se v 1.43% případů), protože při použití této hodnoty bylo vytvořeno nejvíce objednávek výrobků a přitom linka stíhala výrobky zpracovávat a objednávky se nehromadily.

Vytvoření výrobku

Pro každý atribut výrobku definovaný v seznamu atributů výrobny se vygeneruje hodnota. Každá hodnota atributu (například hodnota borovice atributu materiál) má svoji hodnotu pravděpodobnosti, která určuje, jak často se tato hodnota vygeneruje vzhledem k ostatním hodnotám příslušného atributu. Dále má každá hodnota atributu svůj koeficient trvání vzhledem k základnímu atributu (popsáno v dalším bodě), koeficient se načte z výrobny a pokud výrobek splňuje některé speciální pravidlo, je podle tohoto pravidla ještě příslušný koeficient upraven.

Obsazení strojů

Pro každý vhodný stroj (není obsazen, existuje pro ně nějaká neprázdná množina výrobků) se program pokusí vyhledat vhodný výrobek (má atribut, který stroj umí zpracovat; všechny předcházející kroky byly splněny a není zpracováván na jiném stroji). Pokud se podaří najít, vloží se výrobek na stroj a do stroje se načte, jak dlouho má trvat výroba atributu. Výrobek se přesouvá z objednávek do rozpracovaných výrobků.

Trvání základního atributu v ilustrační výrobě trvá 100 časových jednotek a jde o výrobu dveří typu *Standard A*, rozměrů 60 x 197, materiálu borovice na výrobním stroji A. Trvání konkrétního atributu se pak počítá jako trvání výroby standardního atributu * koeficient trvání dané hodnoty (například dub trvá o 37% déle než borovice) * koeficient daného stroje (například obroušení hrany dveří trvá o 72% kratší dobu než výroba).

Zároveň s touto generovanou hodnotou trvání výroby atributu mohou být, v případě, že simulace umí predikovat, vytvořeny také predikované hodnoty trvání.

Práce a uvolnění strojů

Pokud je stroj zabrán výrobkem, s každým tiknutím časovače se odečte jednotka od celkového počtu, které zbývají pro zpracování příslušného atributu. Pokud dojde k tomu, že se tato hodnota vyčerpá, je výrobek i stroj uvolněn a atribut výrobku označen jako hotový. Pokud je v tomto momentě zpracován poslední atribut výrobku, přesouvá se výrobek do hotových.

Pozn.: V kódu se akce vyvolají v pořadí: práce strojů, uvolnění strojů, obsazení strojů; ale pro lepší návaznost textu byla sekvence akcí v kapitole přehozena.

Konec simulace

Simulace končí při opětovném stisknutí spouštěcího tlačítka (nyní s nápisem zastavit simulaci) nebo v momentě, kdy je zhotoven zadaný počet výrobků. Hotové výrobky zobrazují pro každý atribut, který byl pro daný výrobek zpracován na nějakém stroji, spočítanou hodnotu trvání a v případě, že je spuštěná analýza, zobrazí také příslušné predikované hodnoty trvání v pořadí: predikce pomocí rozhodovacího stromu, predikce pomocí algoritmu kNN.

Kapitola 5

Implementace metod

Všechny struktury a funkce reprezentující použití metod analýzy jsou umístěny ve třídě Analyzator, která je statická a existuje od spuštění programu. Poté, co jsou načtena trénovací data, sestaví analyzátor strom a vytvoří pomocné proměnné pro algoritmus kNN. Od tohoto momentu lze od analyzátoru získat predikované hodnoty, které se tvoří v momentě přiřazení výrobku ke stroji.

5.1 Rozhodovací stromy

Pro každý stroj je vytvořen vlastní strom, což je rekurzí stavěná struktura uzlů (větví se) a listů (terminální uzly, obsahují výrobky) stromu. Strom stroje je sestaven pouze z výrobků, které byly zpracovány na tomto stroji. Pro sestavování stromu byla použita varianta algoritmu *ID3*.

Uzel obsahuje údaje o své hodnotě atributu, o svých předcích a o svých potomcích, pokud jde o list, pak také informace o příslušných výrobcích. Každý neterminální uzel rozděluje svoje výrobky do podstromů dle hodnoty nejsilnějšího atributu, který je vybrán jako atribut s nejmenším průměrem směrodatných odchylek trvání jednotlivých hodnot (například je jako nejsilnější vybrán atribut materiál a podstromy mají výrobky s hodnotami *borovice, buk, dub* atd.).

Při prohledávání stromu je procházen strom od kořenového uzlu podle hodnot vyhledávaného výrobku, pro který chceme predikovat. Poté, co je nalezen příslušný list, spočítá se aritmetický průměr trvání všech výrobků v něm a je předána predikovaná hodnota.

5.2 Algoritmus k-nejbližších sousedů

Dokud nenajde program k výrobků, prohledává všechny výrobky v seznamu trénovacích dat. Na začátku se snaží nalézt takové výrobky, které mají Hammingovu vzdálenost od hledaného 0 a pokud se mu nepodaří nalézt dost takových výrobků, vyhledává dále pro stále větší vzdálenost. Ze všech vyhledaných nejbližších sousedů je opět spočítán aritmetický průměr trvání.

Z důvodu rychlejšího výpočtu algoritmu je pro každý výrobek vytvořen řetězec reprezentující všechny hodnoty atributů, každá hodnota je reprezentována jednou cifrou, počet hodnot atributu je proto omezen na 10. Funkce pro porovnávání Hammingovy vzdálenosti pak porovnává pouze dva textové řetězce (například 20150010 a 00140010 mají vzdálenost 2).

Kapitola 6

Testování a analýza výsledků

6.1 Simulace

Rychlost generování výrobků je navrženo tak, aby se fronta objednávek nezahlcovala. Pro otestování náhodného generátoru byly provedeny následující statistické testy (pro nastavení na cílovou průměrnou hodnotu 70):

pokus	počet výrobků	suma mezer	průměrná mezera	chyba
1	10004	686417	68.61425	-1.98%
2	10010	687528	68.68412	-1.88%
3	10008	684056	68.35092	-2.36%
4	10004	694755	69.44772	-0.79%
5	10002	706812	70.66707	+0.95%
6	10002	689784	68.96461	-1.48%
7	10000	694748	69.47480	-0.75%
8	10004	691557	69.12805	-1.25%
9	10001	689838	68.97690	-1.46%
10	10002	684495	68.43581	-2.23%

Průměrná chyba generátoru je tedy 1.513%. Tato přesnost byla pro potřeby vytvořené simulace vyhodnocena jako dostačující.

Následující test zkouší přesnost, s jakou se jednotlivé hodnoty atributů ve vzorku výrobků vyskytují v zadaném poměru. Pro tento test byl vybrán atribut typ dveří se zadanými hodnotami SA 31%, SB 31%, E 10%, L 16% a Ch 12%.

pokus	počet výrobků	SA	SB	E	L	Ch
1	10007	31.40%	30.55%	9.92%	16.04%	12.09%
2	10009	31.07%	31.05%	10.19%	16.20%	11.49%
3	10000	30.93%	31.63%	10.34%	15.21%	11.89%
4	10003	30.63%	31.99%	9.77%	15.81%	11.80%
5	10003	30.70%	31.07%	9.92%	15.90%	12.41%
6	10004	31.38%	30.62%	10.19%	15.81%	12.00%
7	10002	30.95%	31.89%	9.87%	15.63%	11.66%
8	10011	32.25%	31.32%	9.39%	15.19%	11.85%
9	10000	31.43%	31.37%	10.08%	15.89%	11.23%
10	10008	30.16%	31.41%	10.40%	15.90%	12.13%

Průměrná chyba tedy je: SA -0.29%, SB -0.94%, E -0.07%, L 1.51% a Ch 1.21%.

Protože jde o demonstrativní ukázku a hodnoty jsou vygenerované simulací (není zpracováván reálný záznam firemních procesů, protože takové informace jsou citlivé a firmy je neposkytují) neobsahují data šum ani redukovatelné dimenze a předzpracování proto není nutné.

Pro testování metod byl použit soubor trénovacích dat obsahující 10 000 položek. Tyto položky umí program vytvořit v rychlém režimu přibližně za dvě minuty a načtení ze souboru včetně sestavení stromu a připravení hodnot pro analýzu trvá pod 30 vteřin. Vyšší počty jsou v některých případech v předchozích testech proto, že v momentě dokončení nastavených deseti tisíc položek bylo ještě několik rozpracovaných výrobků (nebo objednávek) a tyto byly pro statistické výpočty použity, protože na zkoumané hodnoty stav vyrobení výrobku nemá vliv.

6.2 Metody pro predikci

Trénovací data obsahovala 100 000 výrobků, vzorek pro predikci pak 1000 výrobků. Rozdíl je rozdílem mezi skutečnou trvání výroby atributu a příslušnou metodou predikovanou hodnotou.

6.2.1 Rozhodovací strom

Stavění stromu

Strom byl postaven s tím, že se větev dělila pouze v případě, že vyčerpala všechny použitelné atributy pro větvení nebo dosáhla počtu prvků menšího než 1000 na vzorku trénovacích dat o počtu 100 000 výrobků.

Predikce stromem

Bylo provedeno 10 experimentů (jejich kompletní výsledky jsou v tomto dokumentu na konci jako příloha) a jejich hodnoty byly zprůměrovány.

stroj	maximální rozdíl	průměrný rozdíl	průměrný rozdíl v %
A	50.23930375	3.85788625	3.39048125%
B	57.84724125	4.62867625	3.489485%
hrana	14.96843625	1.34458875	3.8058725%
sklo	12.9881475	1.855765	6.7104725%
kování	8.8056725	1.542003	2.87050125%

6.2.2 Algoritmus k-nejbližších sousedů

- Pro k = 1:

stroj	maximální rozdíl	průměrný rozdíl	průměrný rozdíl v %
A	40.430745	0.1110575	0.0976525%
B	50.8738575	0.134665	0.10124%
hrana	13.417855	0.017265	0.04926%
sklo	6.1526575	0.0095125	0.0342375%
kování	28.8906075	0.0469975	0.0871625%

- Pro k = 2:

stroj	maximální rozdíl	průměrný rozdíl	průměrný rozdíl v %
A	22.12297	0.126565	0.1112475%
B	26.470365	0.1518325	0.114425%
hrana	9.83422	0.0233575	0.0665525%
sklo	5.273935	0.0157375	0.0571%
kování	15.250715	0.036025	0.0667475%

- Pro k = 3:

atribut	maximální rozdíl	průměrný rozdíl	průměrný rozdíl v %
A	33.1137175	0.205155	0.1717425%
B	38.769425	0.23591	0.1782425%
hrana	14.1514225	0.03914	0.110335%
sklo	10.83953	0.0338	0.1216375%
kování	15.3622075	0.0720825	0.1344825%

- Pro k = 4:

stroj	maximální rozdíl	průměrný rozdíl	průměrný rozdíl v %
A	25.2330675	0.257885	0.22618%
B	38.1068225	0.308775	0.2327075%
hrana	8.4957125	0.047675	0.13493%
sklo	5.3316225	0.0361775	0.13156%
kování	12.185005	0.0752625	0.1402225%

Jak je vidět v záznamu experimentů, s navyšováním hodnoty k se průměrná chyba zvyšuje. Chyba predikce trvání výroby na stroji A dosahuje pro postupně zvyšované k hodnot 0.0976%, 0.1112%, 0.1717% a 0.2262% Toto pravidlo má jedinou výjimku a tou je stroj na

kování jehož první hodnota je vyšší než předchozí (0.0872%, 0.0667%, 0.1345%, 0.1402%).

V následující tabulce je zobrazen test na zpomalení simulace v rychlém režimu při výrobě tisíce výrobků a predikování metodou k-NN s různým k. Pro srovnání při generování výrobků zcela bez predikce uběhne cca 22 vteřin.

k	**vteřiny**	**nárust oproti předchozí hodnotě**
1	28	+27%
2	38	+36%
3	67	+76%
4	73	+9%
5	75	+3%
6	83	+11%
7	105	+27%
8	111	+6%
9	120	+8%
10	130	+8%

Kapitola 7

Závěr

Metoda k-nejbližších sousedů prokázala největší přenost predikce pro $k = 1$ a i pro vyšší k byla přesnější než rozhodovací strom. Ve fázi učení vytváření stromů v programu zabere desítky vteřin (navíc pro každý stroj musí být vytvořen strom), naproti tomu příprava dat pro algoritmus k-nejbližších sousedů probíhá průběžně během simulace a nemá na ni výrazný zpomalující vliv. Fáze predikce hodnoty pak probíhá podstatně rychleji pro rozhodovací strom, pro algoritmus k-NN je simulace pomalejší - trojnásobného zpomalení dosáhne již při $k = 3$.

Neuronová síť nebyla pro tuto práci vybrána z toho důvodu, že se těžko vypořádává s takovými datovými soubory, které obsahují mnoho kategorií (atributů) a velké rozptyly hodnot. Ukázková výrobna obsahuje 7 atributů (s třemi až šesti hodnotami). Regresní stromy nemohly být použity z toho důvodu, že testovací provozovna měla pouze kategorické atributy.

7.1 Možná rozšíření

- Maximální počet hodnot atributů by mohl být zvýšen použitím písmen namísto čísel v řetězci pro Hammingovu klasifikaci vzdálenosti.
- Fronta výrobků je nyní definována jako *FIFO*, možným vylepšením by bylo zavedení možnosti prioritního zpracovávání výrobku s nějakou hodnotou atributu.
- Díky objektovému návrhu programu by bylo možné přidat vložení vlastní výrobny (definování vlastních atributů výrobků, strojů a speciálních pravidel).

Literatura

[1] AALST, W. v. d. *Process mining: discovery, conformance and enhancement of business processesu.* [b.m.]: New York: Springer, 2011. ISBN 978-3-642-19344-6.

[2] AUTORŮ, K. *Handbook on business process management 1.* [b.m.]: New York: Springer, 2010. ISBN 978-364-2004-155.

[3] HAN, J. a. M. K. *Data mining: concepts and techniques.* [b.m.]: San Francisco: Morgan Kaufmann Publishersy, 2006. ISBN 15-586-0901-6.

[4] MILAN POSPÍŠIL, T. H. p. I. C. *Business Process Simulation for Predictions.* [b.m.]: FIT VUT Brno, 2012. ISBN 978-1-61208-223-3.

[5] MILAN POSPÍŠIL, V. M. I. T. H. p. I. C. *Process Mining in Manufacturing Company.* [b.m.]: FIT VUT Brno, 2013. ISBN 978-1-61208-254-7.

[6] PETR PERINGER, D. I. *Modelování a simulace (IMS), Studijní opora.* [b.m.]: FIT VUT Brno, 2008.

[7] VONDRÁK, I. *Umělá inteligence a neuronové sítě.*

Příloha A

Návod na použití

Simulaci je možné po otevření programu spustit stisknutím tlačítka *spustit simulaci*. Programem uživatele provází černá oblast s výrazným červeným písmem. Volitelně lze:

- Nastavit režim tlačítkem *přepnout* - režim simulace zobrazuje grafický výstup simulace a detailní informace o výrobcích, režim pro generování dat naproti tomu pracuje mnohem rychleji.
- Nastavit počet výrobků, které má simulátor vyrobit, dokud se neukončí (v režimu pro generování dat je nutné tuto hodnotu zadat).
- Načíst data tlačítkem *načíst data pro predikci* a umožnit tak programu vytváření předpovězených hodnot.
- Uložit vygenerovaná data tlačítkem *uložit výsledek*.

Stisknutí tlačítka *resetovat vše* smaže všechny výrobky ze systému (ale nenaruší případné načtené struktury pro analýzu dat).

Příloha B

Obsah CD

Přiložené CD obsahuje adresáře: *text* (se všemi potřebnými soubory pro vytvoření pdf souboru v programu LaTeX), *program* (se soubory pro prostředí Microsoft Visual Studio), *spustit* (se spustitelným programem v souboru s koncovkou exe a složkami potřebnými pro program) a přímo v kořenovém adresáři uložený soubor pdf s výslednou prací. Pro vyzkoušení analýzy jsou v adresáři spustit v podadresáři saves připraveny soubory trénovacích dat.

Příloha C

Kompletní testovací výstupy

C.0.1 Rozhodovací strom

stroj	max. rozdíl	prům. rozdíl	prům. rozdíl v %
A	49.57399	3.66435	3.24863%
B	57.63662	4.38916	3.31028%
hrana	14.98752	1.35741	3.83847%
sklo	13.08399	1.66968	6.08204%
kování	8.77288	1.51126	2.82369%
.	.	.	.
A	50.02127	4.14677	3.61105%
B	57.63662	4.82892	3.67821%
hrana	14.98752	1.30977	3.73184%
sklo	13.06292	1.95976	7.02988%
kování	8.90762	1.55511	2.89107%
.	.	.	.
A	51.19064	3.82098	3.34991%
B	57.9721	4.63251	3.47373%
hrana	14.98752	1.41869	4.00269%
sklo	12.74276	1.84401	6.73197%
kování	8.90762	1.65298	3.07625%
.	.	.	.
A	51.19064	3.82098	3.34991%
B	57.9721	4.63251	3.47373%
hrana	14.98752	1.41869	4.00269%
sklo	12.74276	1.84401	6.73197%
kování	8.90762	1.65298	3.07625%

stroj	**max. rozdíl**	**prům. rozdíl**	**prům. rozdíl v %**
A	49.60237	3.98341	3.53454%
B	57.72122	4.90812	3.7054%
hrana	14.98752	1.27217	3.62069%
sklo	13.06292	1.88348	6.80339%
kování	8.70194	1.45997	2.7275%
.	.	.	.
A	50.02127	4.40069	3.86739%
B	57.63662	5.29245	4.00267%
hrana	14.71769	1.23736	3.51453%
sklo	13.06292	1.81703	6.59643%
kování	8.77288	1.4991	2.80069%
.	.	.	.
A	50.29298	3.26179	2.87101%
B	58.5655	3.89126	2.91901%
hrana	14.98752	1.41026	3.97965%
sklo	13.08399	2.04422	7.23242%
kování	8.77288	1.58051	2.90084%
.	.	.	.
A	50.02127	3.76412	3.29141%
B	57.63715	4.45448	3.35285%
hrana	15.10468	1.33236	3.75642%
sklo	13.06292	1.78393	6.47568%
kování	8.70194	1.42411	2.66772%

C.0.2 Algoritmus k-nejbližších sousedů

- Pro k = 1:

stroj	**max. rozdíl**	**prům. rozdíl**	**prům. rozdíl v %**
A	40.74178	0.13005	0.115%
B	18.08403	0.15831	0.12051%
hrana	4.91789	0.00492	0.01423%
sklo	8.24713	0.0161	0.058%
kování	38.04668	0.10575	0.19559%
.	.	.	.
A	26.223	0.04768	0.04222%
B	73.10421	0.05827	0.04382%
hrana	39.97553	0.05536	0.15787%
sklo	5.4545	0.00825	0.02933%
kování	12.32221	0.01232	0.02313%
.	.	.	.
A	54.01642	0.11981	0.10403%
B	63.06633	0.14351	0.10705%
hrana	4.389	0.00439	0.01252%
sklo	5.4545	0.00825	0.03011%
kování	20.98976	0.02572	0.04759%
.	.	.	.
A	40.74178	0.14669	0.12936%
B	49.24086	0.17857	0.13358%
hrana	4.389	0.00439	0.01242%
sklo	5.4545	0.00545	0.01951%
kování	44.20378	0.0442	0.08234%

- Pro k = 2:

stroj	**max. rozdíl**	**prům. rozdíl**	**prům. rozdíl v %**
A	24.13644	0.21855	0.1911%
B	21.4553	0.25863	0.19467%
hrana	9.62252	0.01428	0.04025%
sklo	3.92434	0.00805	0.02921%
kování	20.98976	0.02715	0.05086%
.	.	.	.
A	18.9505	0.06811	0.05952%
B	38.35043	0.0806	0.06091%
hrana	17.15462	0.0511	0.14578%
sklo	5.25836	0.01324	0.04816%
kování	10.49488	0.04072	0.07481%
.	.	.	.
A	21.08744	0.12491	0.11016%
B	21.4553	0.15143	0.11447%
hrana	10.36524	0.02559	0.07323%
sklo	5.25836	0.02263	0.08153%
kování	19.02334	0.04475	0.08261%
.	.	.	.
A	24.3175	0.09469	0.08421%
B	24.62043	0.11667	0.08765%
hrana	2.1945	0.00246	0.00695%
sklo	6.65468	0.01903	0.0695%
kování	10.49488	0.03148	0.05871%

- Pro k = 3:

stroj	**max. rozdíl**	**prům. rozdíl**	**prům. rozdíl v %**
A	28.11659	0.18215	0.16232%
B	31.71177	0.22353	0.16993%
hrana	13.32518	0.03826	0.10545%
sklo	15.67935	0.04125	0.14964%
kování	14.73459	0.08571	0.16078%
.	.	.	.
A	38.23259	0.20674	0.18169%
B	36.68363	0.24863	0.18804%
hrana	12.83003	0.06195	0.17597%
sklo	5.99971	0.03164	0.11488%
kování	15.57141	0.0454	0.08482%
.	.	.	.
A	21.89814	0.16284	0.14333%
B	42.04422	0.19426	0.14572%
hrana	9.79416	0.01963	0.0554%
sklo	15.67935	0.04132	0.1462%
kování	13.99318	0.04225	0.07754%
.	.	.	.
A	44.20755	0.22499	0.19963%
B	44.63808	0.27722	0.20928%
hrana	20.65632	0.03672	0.10452%
sklo	5.99971	0.02099	0.07583%
kování	17.14965	0.11497	0.21479%

- Pro k = 4:

stroj	**max. rozdíl**	**prům. rozdíl**	**prům. rozdíl v %**
A	26.48419	0.19421	0.17218%
B	30.42079	0.23263	0.17545%
hrana	8.89663	0.05541	0.15668%
sklo	7.82712	0.03226	0.11752%
kování	11.67856	0.10662	0.19921%
.	.	.	.
A	42.17488	0.26541	0.23112%
B	27.40702	0.30907	0.23542%
hrana	10.1261	0.04711	0.13424%
sklo	4.49979	0.02797	0.10034%
kování	16.926	0.08337	0.155%
.	.	.	.
A	16.1366	0.28596	0.25071%
B	47.29974	0.3467	0.25998%
hrana	7.48006	0.04409	0.1244%
sklo	4.49979	0.04224	0.15419%
kování	10.06773	0.05553	0.10334%
.	.	.	.
A	16.1366	0.28596	0.25071%
B	47.29974	0.3467	0.25998%
hrana	7.48006	0.04409	0.1244%
sklo	4.49979	0.04224	0.15419%
kování	10.06773	0.05553	0.10334%

Příloha D

Vzhled okna programu

Obrázek D.1: Průběh simulace

Obrázek D.2: Výsledky predikce