

Slovenská technická univerzita v Bratislave  
Fakulta informatiky a informačných technológií

Bc. Matúš Cimerman

**Analýza prúdu prichádzajúcich udalostí  
použitím rôznych metód pre analýzu údajov**

*Diplomová práca*

Vedúci práce: Ing. Jakub Ševcech

máj, 2016



Slovenská technická univerzita v Bratislave  
Fakulta informatiky a informačných technológií

Bc. Matúš Cimerman

# Analýza prúdu prichádzajúcich udalostí použitím rôznych metód pre analýzu údajov

*Diplomová práca*

Študijný program: Informačné systémy

Študijný odbor: 9.2.6 Informačné systémy

Miesto vypracovania: Ústav informatiky, informačných systémov a softvérového inžinierstva,  
FIIT STU v Bratislave

Vedúci práce: Ing. Jakub Ševcech

máj, 2016



## Návrh zadania diplomovej práce

Finálna verzia do diplomovej práce<sup>1</sup>

### Študent:

**Meno, priezvisko, tituly:** Matúš Cimerman, Bc.  
**Študijný program:** Informačné systémy  
**Kontakt:** matus.cimerman@gmail.com

### Výskumník:

**Meno, priezvisko, tituly:** Jakub Ševcech, Ing.

### Projekt:

**Názov:** Analýza prúdu prichádzajúcich udalostí použitím rôznych metód pre analýzu údajov  
**Názov v angličtine:** Stream analysis of incoming events using different data analysis methods  
**Miesto vypracovania:** Ústav informatiky a softvérového inžinierstva, FIIT STU, Bratislava  
**Oblast problematiky:** analýza dát, prúd udalostí

### Text návrhu zadania<sup>2</sup>

Analýza údajov v sebe spája rôzne techniky z rôznych oblastí štatistiky, strojového učenia a dolovania v údajoch v spojení so znalosťou domény. Každá z týchto domén vyžaduje netriviálne znalosti potrebné pre preloženie otázky do analytickej úlohy, výberu analytickej metódy, vykonanie analýzy a interpretovanie výsledkov. Častokrát je veľmi náročné nájsť experta, ktorý by vedel prepojiť všetky tieto oblasti a s ďalším rozvojom analytickej metód bude tento problém ďalej rásť.

Priestor na zmiernenie tohto problému je napríklad v návrhu nástrojov, ktoré pomáhajú v tomto procese a umožňujú používanie a interpretovanie pokročilých modelov doménovým expertom bez potreby detailných znalostí o fungovaní modelu. Podobné prístupy sme mohli vidieť v podobe rôznych populárnych nástrojov na spracovanie statických kolekcií údajov pomocou metód ako sú lieviková analýza (angl. funnel analysis) alebo vnáranie sa (angl. drill down). V súčasnosti sa však do pozornosti dostáva analýza údajov v čase ich vzniku, kde hovoríme o analýze prúdu prichádzajúcich udalostí.

Analyzujte možnosti použitia známych metód na analýzu statických kolekcií údajov a existujúcich metód na analýzu prúdov údajov. Vyberte a aplikujte metódu na analýzu údajov v doméne spracovania prúdov údajov. Sústredte sa pritom na použiteľnosť metódy, jej jednoduchosť a interpretateľnosť poskytnutých výsledkov používateľom, ktorí nemajú detailné znalosti o fungovaní modelu. Navrhnuté riešenie overte pomocou softvérovej súčiastky implementovaním vybranej metódy vhodnej pre analýzu prúdu udalostí vo zvolenej doméne.

<sup>1</sup> Vytlačiť obojstranne na jeden list papiera

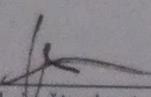
<sup>2</sup> 150-200 slov (1200-1700 znakov), ktoré opisujú výskumný problém v kontexte súčasného stavu vrátane motivácie a smerov riešenia

### Literatúra<sup>3</sup>

- KREMPL, Georg, et al. Open challenges for data stream mining research. ACM SIGKDD Explorations Newsletter, 2014, 16.1: 1-10.
- GABER, Mohamed Medhat; ZASLAVSKY, Arkady; KRISHNASWAMY, Shonali. Mining data streams: a review. ACM Sigmod Record, 2005, 34.2: 18-26.

Vyššie je uvedený návrh diplomového projektu, ktorý vypracoval(a) Bc. Matúš Cimerman, konzultoval(a) a osvojil(a) si ho Ing. Jakub Ševcech a súhlasí, že bude takýto projekt viesť v prípade, že bude pridelený tomuto študentovi.

V Bratislave dňa 12.1.2016



---

Podpis študenta



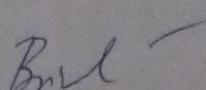
---

Podpis výskumníka

### Vyjadrenie garanta predmetov Diplomový projekt I, II, III

Návrh zadania schválený: áno / nie<sup>4</sup>

Dňa: ..... 15.2.2016 .....



---

Podpis garanta predmetov

<sup>3</sup> 2 vedecké zdroje, každý v samostatnej rubrike a s údajmi zodpovedajúcimi bibliografickým odkazom podľa normy STN ISO 690, ktoré sa viažu k téme zadania a preukazujú výskumnú povahu problému a jeho aktuálnosť (uvedte všetky potrebné údaje na identifikáciu zdroja, pričom uprednostnite vedecké príspevky v časopisoch a medzinárodných konferenciách)

<sup>4</sup> Nehodiace sa prečiarknite

## Zadanie diplomovej práce

**Meno študenta:** Bc. Matúš Cimerman

**Študijný program:** Informačné systémy

**Študijný odbor:** Informačné systémy

**Názov práce:** **Analýza prúdu prichádzajúcich udalostí použitím rôznych metód pre analýzu údajov**

Samostatnou výskumnou a vývojovou činnosťou v rámci predmetov Diplomový projekt I, II, III vypracujte diplomovú prácu na tému, vyjadrenú vyššie uvedeným názvom tak, aby ste dosiahli tieto ciele:

**Všeobecný cieľ:**

Vypracovaním diplomovej práce preukážte, ako ste si osvojili metódy a postupy riešenia relativne rozsiahlych projektov, schopnosť samostatne a tvorivo riešiť zložité úlohy aj výskumného charakteru v súlade so súčasnými metódami a postupmi študovaného odboru využívanými v príslušnej oblasti a schopnosť samostatne, tvorivo a kriticky pristupovať k analýze možných riešení a k tvorbe modelov.

**Specifický cieľ:**

Vytvorte riešenie zodpovedajúce návrhu textu zadania, ktorý je prílohou tohto zadania. Návrh bližšie opisuje tému vyjadrenú názvom. Tento opis je záväzný, má však rámcový charakter, aby vznikol dostatočný priestor pre Vašu tvorivosť.

Riadťe sa pokynmi Vášho vedúceho.

Pokial' v priebehu riešenia, opierajúc sa o hlbšie poznanie súčasného stavu v príslušnej oblasti, alebo o priebežné výsledky Vášho riešenia, alebo o iné závažné skutočnosti, dospejete spoločne s Vašim vedúcim k presvedčeniu, že niečo v texte zadania a/alebo v názve by sa malo zmeniť, navrhnite zmenu. Zmena je spravidla možná len pri dosiahnutí kontrolného bodu.

**Miesto vypracovania:** Ústav informatiky a softvérového inžinierstva, FIIT STU Bratislava

**Vedúci práce:** Ing. Jakub Ševcech

**Termíny odovzdania:**

Podľa harmonogramu štúdia platného pre semester, v ktorom máte príslušný predmet (Diplomový projekt I, II, III) absolvovať podľa Vášho študijného plánu

**Predmety odovzdania:**

V každom predmete dokument podľa pokynov na [www.fiit.stuba.sk](http://www.fiit.stuba.sk) v časti:  
home > Informácie o > štúdiu > organizácia štúdia > diplomový projekt.

V Bratislave dňa 15. 2. 2016



prof. Ing. Pavol Návrat, PhD.  
riaditeľ Ústavu informatiky a softvérového  
inžinierstva



# Anotácia

**Fakulta Informatiky a Informačných Technológií  
Slovenská Technická Univerzita**

Meno:

Bc. Matúš Cimerman

Vedúci diplomovej práce:

Ing. Jakub Ševcech

Diplomová práca:

Analýza prúdu prichádzajúcich  
udalostí použitím rôznych metód  
pre analýzu údajov

Študijný program:

Informačné systémy

Máj 2016

Dnes môžeme pozorovať narastujúcu potrebu analyzovať dátá počas ich vzniku. Spracovanie a analýza prúdov dát predstavuje komplexnú úlohu, pričom je dôležité poskytnúť riešenie s nízkou odozvou, ktoré je odolné voči chybám.

V našej práci sa sústredíme na návrh súboru nástrojov, ktoré pomôžu doménovému expertovi počas analýzy dát. Doménový expert nepotrebuje mať detailné znalosti o fungovaní analytického modelu. Podobný prístup je podobný, ak chceme analyzovať statické kolekcia dát napríklad lievikovou analýzou. Študujeme možnosti použitia tradičných metód pre statické údaje v doméne analýzy prúdov dát. Našim cieľom je aplikovať metódu pre analýzu v doméne prúdu dát. Navrhujeme použitie rozhodovacieho stromu v kontexte klasifikácie prúdu dát, ktorý používa Hoeffdingovu mieru pre výber najlepšieho rozhodovacieho atribútua so stanovenou istotou. Pre vysporiadanie so zmenami aplikujeme algoritmus ADWIN, ktorý adaptívne dokáže detektovať zmeny v prúde. Zameriavame sa pritom na jednoduchosť vybranej metódy a interpretatívnosť výsledkov. Pre doménových expertov je nevyhnutné aby boli tieto požiadavky splnené, pretože nebudú potrebovať detailné znalosti z domén ako strojové učenie sa alebo štatistika. Naše riešenie vychodnocujeme implementovaním softvérovej súčiastky a vybranej metódy.



# Annotation

**Faculty of Informatics and Information Technologies  
Slovak University of Technology**

Name: Bc. Matúš Cimerman  
Supervisor: Ing. Jakub Ševcech  
Diploma thesis: Stream analysis of incoming events  
using different data analysis methods  
Course: Information systems  
2016, May

Nowadays we can see emerging need for data analysis as data occur. Processing and analysis of data streams is a complex task, first, we particularly need to provide low latency and fault-tolerant solution.

In our work we focus on proposal a set of tools which will help domain expert in process of data analysis. Domain expert do not need to have detailed knowledge of analytics models. Similar approach is popular when we want analyse static collections, eg. funnel analysis. We study possibilities of usage well known methods for static data analysis in domain data streams analysis. Our goal is to apply method for data analysis in domain of data streams. We propose usage of decision tree in context of data stream classification, which uses Hoeffding bound to select best splitting attribute with desired confidence. To adaptively deal with concept drift we are using algorithm ADWIN to detect drifts in stream. This approach is focused on simplicity in use of selected method and interpretability of results. It is essential for domain experts to meet these requirements because they will not need to have detailed knowledge from such a domains as machine learning or statistics. We evaluate our solution using software component implementing chosen method.



# Pod'akovanie

Na prvom mieste vyslovujem pod'akovanie vedúcemu mojej diplomovej práce, Ing. Jakubovi Ševcechovi, za všetky jeho odborné rady, odovzdané skúsenosti a usmernenie pri tvorení práce.

Tento cestou taktiež vyslovujem pod'akovanie všetkým výskumníkom zo skupiny PeWe, za prínosné diskusie a ich spätnú väzbu týkajúcu sa mojej práce. V neposlednom rade ďakujem celej mojej rodine a priateľom.

Matúš Cimerman



# Obsah

<b>1</b>	<b>Úvod</b>	<b>1</b>
<b>2</b>	<b>Analytické úlohy nad prúdom dát</b>	<b>3</b>
2.1	Dopyty nad prúdom dát . . . . .	8
2.2	Detekcia zmien . . . . .	10
2.3	Detekcia anomálií . . . . .	13
2.4	Zhlukovanie . . . . .	16
2.5	Klasifikácia . . . . .	19
2.6	Zhodnotenie . . . . .	24
<b>3</b>	<b>Interpretácia a vysvetlenie výsledkov analytických úloh</b>	<b>25</b>
<b>4</b>	<b>Existujúce nástroje pre analýzu prúdu udalostí</b>	<b>29</b>
4.1	MOA . . . . .	29
4.2	WEKA . . . . .	31
4.3	RapidMiner Streams-Plugin . . . . .	34
4.4	StreamBase . . . . .	34
4.5	Spark . . . . .	35
<b>5</b>	<b>Klasifikácia prúdu dát použitím rozhodovacích stromov</b>	<b>39</b>
5.1	Spracovanie prúdu dát . . . . .	42

5.2	Metóda klasifikácie prúdu dát . . . . .	43
5.3	Prezentácia výsledkov používateľovi . . . . .	46
5.4	Vyhodnotenie a experimenty . . . . .	47
<b>6</b>	<b>Zhodnotenie a budúca práca</b>	<b>49</b>
	<b>Literatúra</b>	<b>51</b>
	<b>Prílohy</b>	<b>55</b>
A	Plán na letný semester 2016/2017 . . . . .	55
B	Plán na zimný semester 2016/2017 . . . . .	56

# 1. Úvod

V súčasnosti pozorujeme zvýšený záujem o oblast analýzy a dolovania dát. Vhodné použitie a výber metód pre spracovanie dát prináša hodnotné výstupy a náhľady pre používateľa. Výstupy môžu byť použité pre strategické rozhodnutia v podnikoch. Najčastejší postup je aplikovaním metód ako napríklad lieviková analýza alebo rozhodovacie stromy nad statickou kolekcii dát. Tento prístup má niekoľko problémov a to najmä: všetky trénovacie dáta musia byť uložené v pamäti alebo na disku, spracovanie a výpočtová náročnosť a vysporiadanie sa s trendami a zmenami v dátach. Nutnosť dátaj nájskôr zozbierať a uložiť, čo je dnes, kedy vznikajú milióny záznamov za deň, či hodinu, predstavuje rovnako veľký problém.

Pod pojmom spracovanie v reálnom čase myslíme spracovanie v takmer reálnom čase, tzv. jemné (angl. soft) spracovanie v reálnom čase. Jemné spracovanie v reálnom čase znamená, že systém negarantuje spracovanie a odpoveď v stanovenom časovom limite, pričom niektoré vzorky sa môžu omeškať alebo úplne vynechať (Stankovic and Zhao, 1988). Presné limity, do kedy sa spracovanie považuje za reálny čas závisí od problému. Niekde to môže predstavovať rádovo stotiny sekundy, v inej úlohe rádovo sekundy. V tejto práci budeme pracovať s pojmom spracovanie v reálnom čase chápajúcich ako jemné spracovanie v reálnom čase.

Pri dolovaní v prúde dát čelíme niekoľkým výzvam: objem, rýchlosť (frekvencia) a rozmanitosť. Veľký objem dát, ktoré vznikajú veľmi rýchlo je potrebné spracovať v ohraničenom časovom intervale, často v reálnom čase. Pričom sa objem dát neustále zväčšuje, potenciálne narastá až do nekonečna. Identifikujeme niekoľko najviac zasiahnutých oblastí, ktoré sú zdrojmi týchto dát: počítačové siete, sociálne siete, Webové stránky (sledovanie správania používateľa na stránke) a Internet Vecí (angl. Internet of Things). Na informácie generované z takýchto zdrojov sa často pozerať ako na neohraničené a potenciálne nekonečné prúdy údajov.

---

Spracovanie, analýza a dolovanie v týchto prúdoch je komplexná úloha. Pre aplikácie je kritické spracovať údaje s nízkou odozvou, pričom riešenie musí byť presné, škálovateľné a odolné voči chybám. Nakoľko sú prúdy neohraničené vo veľkosti a potenciálne nekonečné, môžeme spracovať len ohraničený interval prúdu. Potom hovoríme, že dátá musia byť spracované tak ako vznikajú. Tradičné metódy a princípy pre spracovanie statickej kolekcie údajov nie sú postačujúce na takéto úlohy (Krempl et al., 2014; Han et al., 2011).

## 2. Analytické úlohy nad prúdom dát

Spracovanie, analýza a dolovanie dát predstavuje vo všeobecnosti výzvu. Zvláštnu pozornosť si tieto úlohy vyžadujú pri spracovaní, analýze a dolovaní z prúdu prúdu udalostí. Prúd udalostí je často nazývaný *prúd dát* alebo *údajov*, či len skrátene *prúd*. V tomto texte budeme pre jednoduchosť používať najmä termín *prúd* a *prúd dát* (Tran et al., 2014). Avšak môžu sa vyskytnúť aj terminy ako *prúd udalostí*, *sekvencia udalostí*, či *elementov*, pričom všetky termíny majú v tomto teste rovnaký význam.

**Definícia 2.0.1** *Prúd je potenciálne nekonečná sekvencia elementov (Tran et al., 2014).*

$$S = \{(X_1, T_1), \dots, (X_j, T_j), \dots\}$$

*Kde každý element je pári  $(X_j, T_j)$  kde  $X_j$  je  $d$ -dimenzionálny vektor  $X_j = (x_1, x_2, \dots, x_d)$  prichádzajúci v čase  $T_j$ .  $T_j$  je často nazývaný aj časová pečiatka, existujú dva typy časovej pečiatky: explicitná je generovaná ked' dát dorazia, implicitná je priradená vektoru v čase ich vzniku.*

Takmer každé odvetvie dnes generuje masívne množstvo dát. Vzhľadom na ich veľký objem analytici a doménový experti často strácajú schopnosť dolovať v celej sade dát. Stáva sa preto častým zvykom, že sa vyberie reprezentujúca vzorka, ktorej spracovanie predstavuje menšiu časovú a pamäťovú výzvu. Pri pamäťovej náročnosti hovoríme o limitoch počítača, pričom ak hovoríme o časovej náročnosti hovoríme o limitovanom čase doménového experta (čakanie na výsledok analýzy) (Hulten et al., 2001). Predpokladajme, že bude pre doménového experta vysokým prínosom možnosť vykonávať analýzy nad prúdom v reálnom čase. Výstupy z takejto analýzy sú na rôznej granularite a úrovni, pričom môžu byť neskôr použité na ďalšie spracovanie alebo na priame

prezentovanie výsledkov.

Analýza a spracovanie prúdov dát pridáva viaceré otvorené výzvy a možnosti pre výskum (Krempl et al., 2014):

- *Ochrana súkromia a dôvernosti* pri analýze a dolovaní v prúde dát. Hlavným cieľom je vyvinúť metódy a techniky, ktoré neodhalia informácie a vzory, ktoré by kompromitovali potreby dôvernosti a ochrany súkromia. Dve hlavné výzvy pri analýze a dolovaní v prúdoch dát sú: *vysporiadanie sa s neúplnými dátami* a *uchovanie zmien* (angl. *concept drift*) v prúde dát.
- *Predspracovanie* dát je dôležitou súčasťou každej reálnej aplikácie, najmä tých pre analýzu dát. Zatiaľ čo pri tradičnej analýze dát je predspracovanie vykonané jednorázovo, zvyčajne doménovým expertom, ktorý rozumie dátam. Pri prúde dát toto nieje prijateľné, pretože dáta nepretržite prichádzajú. Okrem niekoľkých štúdií (Zliobaite and Gabrys, 2014; Anagnostopoulos et al., 2008) tejto problematike nebola venovaná dostatočná pozornosť ako pri tradičnom spracovaní dát. Hlavné výzvy, ktorým treba čeliť pri predspracovaní prúdu dát sú: *hluk v dátach*, *outliers* a *adaptívny výber vzorky*.
- *Načasovanie a dostupnosť informácie*, väčšina algoritmov robí jednoduchý predpoklad, že prijatá informácia je kompletná, ihneď dostupná, prijatá pasívne a zadarmo. Viaceré výzvy spojené s načasovaním a dostupnosťou informácie sú formulované a nepreskúmané: *spracovanie nekompletívnych dát*, *vysporiadanie sa so skreslenou* (angl. *skewed*) *distribúciou dát* a *spracovanie oneskorených dát*.
- *Dolovanie entít a udalostí* kde entity predstavujúce prúd sú spojené do viacerých inštancií resp. štruktúrovaných informácií (napr. agregácie). Tieto entity môžu byť niekedy spojené s výskyтом udalostí resp. v prúde dát.
- *Evaluácia algoritmov pre prúdy dát* predstavuje úplne novú výzvu v porovnaní s tradičnými metódami. Pri evaluácii v prúde dát sa musíme vysporiadať s problémami ako: *zmeny* (angl. *concept drift*, *limitovaný čas pre spracovanie vzorky*, *vyvíjajúce sa skreslenie tried dát*, či *oneskorenie*

---

*overenia.* Tejto problematike sa v poslednej dobe venuje vyššia pozornosť, ako napríklad pre evaluáciu klasifikátorov nad prúdmi dát (Bifet et al., 2015).

- *Špecializované, reaktívne a jednoduché modely* na pochopenie pre doménového experta. Tieto tri výzvy v sebe ukrývajú potrebu pre minimálizáciu závislosti na nastavení parametrov metódy, kombinácia online a offline modelov a riešenie správneho problému (zmeny v prúdoch).

*Model prúdu dát* môže byť jeden z nasledujúcich: model časových radov, pokladničný model a model turniketu. Podľa modelu prúdu dát existujú príslušné algoritmy, ktoré boli vytvorené pre daný model (Tran et al., 2014). Majme prúd dát  $a_1, a_2, \dots$ , ktorý prichádza sekvenčne za sebou a popisuje podstaný signál  $A$ . V modeli časových radov každá vzorka  $a_i$  sa rovná  $A[i]$  pričom vzorky prichádzajú v vzostupnom poradí. Tento model je vhodný pre prúdy dát, ktoré nesú v sebe časovú postupnosť alebo je ich poradie určované časovou pečiatkou (Muthukrishnan, 2005). Pri pokladničnom modeli môžeme považovať množinu  $U = 1, 2, \dots, n$  za element z prúdu dát. Ak uvažujeme sekvenčiu 2, 1, 2, 5 ako príklad, potom hovoríme o pokladničnom modeli. Tento model je často používaný v praxi, napríklad v prípadoch kde sled IP adres pristupuje na Web server (Ikonomovska and Zelke, 2013; Muthukrishnan, 2005). Model turniketu je veľmi podobný pokladničnému modelu. Rozdiel je v tom, že vzorka môže predstavovať aj zápornú hodnotu - analógia z reálneho sveta kedy niektorí ľudia prichádzajú a vychádzajú turniketom, počet ľudi sa mení (napr. na zjazdovke) (Ikonomovska and Zelke, 2013; Muthukrishnan, 2005).

*Predspracovanie prúdu* je azda najdôležitejším krokom v aplikáciach reálneho sveta a časovo najnáročnejšou úlohou pre každého analyтика. Nakoľko dáta prichádzajú z nehomogénneho sveta, môžu byť zašumené, nekompletné, duplicitné alebo často obsahovať hodnoty, ktoré sa značne líšia od ostatných. Predspracovanie prúdiach údajov je potrebné čo najviac automatizovať. Existuje niekoľko známych metód a techník, ktoré sú používané pri predspracovaní prúdov dát (Krempl et al., 2014; Nguyen et al., 2015):

- *Vzorkovanie*, napríklad podľa pravdepodobnostného modelu.
- *Zahadzovania potenciálne nepotrebných vzoriek*, ak je spracujúci proces

príliš zaťažený. Tu môže nastať problém, že práve zahodená vzorka bola dôležitá (zmena v dátach).

- *Agregácia* údajov môže značne znížiť objem dát, ale môže spôsobiť problém pri potrebe pohľadu do minulosti.
- *Approximačné algoritmy* a ich použitie má za následok podstatné zrýchlenie spracovania a analýzy prúdov za predpokladu istej chybovosti. Chybovosť je zvačsa ohraničená.
- *Posuvné okno*, tento prístup vznikol s potrebou analýzy definovaného časového okna z prúdiacich údajov. Výstup je teda závislý na zvolenej veľkosti okna. Problém pri tomto prístupe je práve správne nastavenie veľkosti okna tak aby sme vedeli zohľadniť zmeny v prúde dát.

Pri dolovaní z prúdov dát (angl. data stream mining) je potrebné aby algoritmy splňovali nasledujúce obmedzenia (Nguyen et al., 2015; Wadewale and Desai, 2015):

- Jeden priechod cez dátu (angl. single-pass). Narozenie od tradičných metód kde je možné dátu prečítať viac krát, pretože sú dátu dostupné na disku. Pri dolovaní v prúde dát nové vzorky prichádzajú kontinálne a musíme preto každú vzorku spracovať práve raz v momente keď príde.
- Odpoveď v reálnom čase v zmysle, že vytvorený model je pripravený kekykoľvek (angl. anytime real-time response) napríklad predikovať triedy nových vzoriek.
- K dispozícii máme len ohraničenú pamäť. Toto obmedzenie súvisí s povahou prúdu dát a to, že prúdy predstavujú potenciálne nekonečné zdroje dát.
- Detekcia zmien (angl. concept-drift detection) je nevyhnutná v situácii keď sa v dátach objavia nové vzory, ktoré sa menia v čase.

*Spracovanie, dolovanie a analýza prúdu dát* v statickej kolekcii dát bola študovaná niekoľko dekád. Zvýšenú pozornosť začala odborná verejnosť venovať pri aplikovaní týchto úloh na prúdy dát. Niektorým z týchto úloh sa venujeme podrobne v nasledujúcich podkapitolách:

- *Zhlukovanie*, existuje niekoľko výskumov, ktoré sa venovali špeciálne klastrovaniu implementovaním napríklad k-mediánu a inkrementálnych algoritmov. Zhlukovanie je úloha učenia bez učiteľa.
- *Klasifikácia*, táto úloha je dlho skúmaná s použitím rôznych metód rozhodovacích stromov. Klasifikácia predstavuje úlohu učenia s učiteľom, znamená to, že na vstupe očakávame označkovaný prúd dát.
- *Počítanie frekvencie a opakovania*, použitím posuvných okien a inkrementálnych algoritmov na detekciu vzorov v prúde. Príkladom tejto úlohy môže byť napríklad hľadanie sezónnych javov v prúde dát.
- *Analýza časových radov použitím symbolickej reprezentácie* časových radov v prúde dát. Takáto reprezentácia nám umožňuje redukciu veľkosti prenášaných dát. Táto technika pozostáva z dvoch hlavných krokov, approximácia po častiach a následná transformácia výsledku do diskrétnych veličín. Táto úloha resp. problém je skôr podpornou metódou pre vyššie spomenuté úlohy, pretože ju radíme medzi úlohy predspracovania dát s cieľom redukcie objemu prenášaných dát.

**Evaluácia** modelov vytvorených nad prúdmi dát je základná a dôležitá úloha pre meranie kvality modelu. Toto so sebou prináša výzvy, pretože prúdy dát sú potenciálne nekončné, evaluáciu modelov je potrebné vykonávať online zatiaľ čo dátá často vytvárajú problémy, napríklad nerovnomerné rozdelenie tried v prúde dát. Potom, tradičné techniky evaluácie známe z dávkových algoritmov pre analýzu dát, nie sú použiteľné pre evaluáciu prúdov dát. Bifet et al. (2015) hovoria o troch hlavných mylných prístupoch k evaluácii prúdu dát:

- *McNemarov test* a jeho použitie na pre štatistické rozlíšenie kvality dvoch klasifikátorov. Aj napriek štatisticky signifikantnému rozdielu medzi klasifikátormi tento test nie je vhodný pre prúdy dát, pretože aj keď je model vytvorený rovnakým algoritmom, väčšina algoritmov je inicializovaná alebo používa nejakú náhodnú zložku. To môže viest k zavádzajúcim výsledkom pri použití tohto testu.
- *Rozdelovanie množiny dát* do trénovacej a testovacej množiny viedie k nemožnosti rozoznať výkonnosť rôznych klasifikátorov. Je to z dôvodu,

že v dátach sa môžu objavovať napríklad zmeny, ktoré budu skreslené rozdeľovaním alebo vzorkovaním dát.

- *Väčšina tried v posuvnom okne* môže spôsobiť pozitívne k štatistiky a tiež pozitívne výsledky harmonického priemeru pre niektoré periódy prúdu dát.

Bifet et al. (2015) odporúčajú použitie nasledujúcich metód pre evaluáciu v kontexte prúdov dát: *Najprv testuj-potom-trénuj* (angl. Test-Then-Train alebo tiež Prequential) spočíva v myšlienke, že každá vzorka z prúdu dát je použitá najprv na testovanie a potom na trénovanie modelu. Keďže modely vytvorené nad prúdmi dát by mali byť schopné poskytnúť predikcie okamžite a v reálnom čase, tento prístup by nemal výrazne ovplyvniť výkon metódy. Obmenou tohto prístupu môže byť evaluácia na nejakom posuvnom okne, ktoré môže byť vyberané napríklad algoritmom *ADWIN*.

## 2.1 Dopyty nad prúdom dát

Vyhodnocovaniu dopytov nad statickou kolekciou dát bola venovaná značná pozornosť, ak však hovoríme o prúdoch dát dopyty musia byť vyhodnocované kontinuálne (Babu and Widom, 2001; Babcock et al., 2002). Vzniká teda nová paradigma pre interakciu s dynamicky sa meniacimi dátami, ktorú nazývame kontinuálne dopyty (angl. continuous queries) (Babu and Widom, 2001). Výsledky kontinuálnych dopytov sú produkované dynamicky v čase vzniku nových dát. Príkladom použitia takýchto dopytov je napríklad sledovanie vývoja akcií burzy. Problém môže nastať pri jednorázových dopytoch, ktoré obsahujú agregačné funkcie. Pri tradičnom spracovaní dát kde sú všetky dáta uložené ako statická kolekcia, je dopyt vykonaný nad celou kolekciou. V prípade kontinuálneho dopytu je problém získať predchádzajúce dáta za predpokladu, že dáta niesú ukladané. Môžu potom nastať dva scenáre:

1. agregačná funkcia je prepočítaná nad kolekciou dát, za predpokladu, že boli historické dáta ukladané.
2. agregačná funkcia je počítaná od momentu zadanie dopytu.

Kontinuálne dopytovanie do prúdu dát nesie so sebou niečo výziev (Babcock et al., 2002):

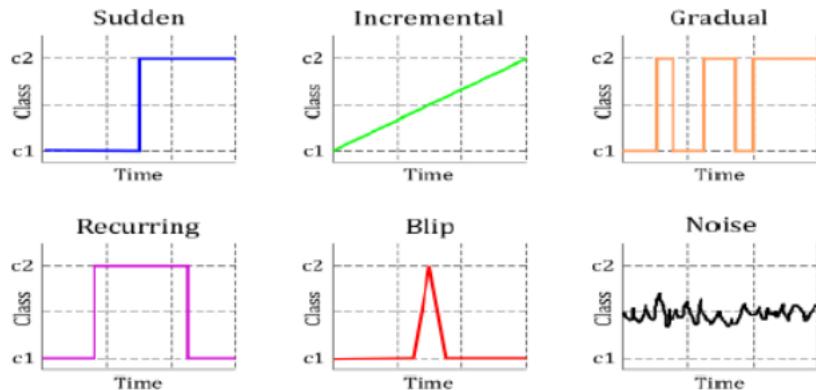
- *Limitované pamäťové požiadavky* na algoritmy spracujúce dopyty, pretože prúd dát predstavuje potenciálne nekonečný prúd udalostí.
- *Približné odpovede na dopyty* sú niekedy postačujúce za predpokladu, že odpoveď je dostatočne rýchla a používateľ rozumie v akej presnosti mu bola odpoveď poskytnutá. Techniky pre redukciu dimenzionality a objemu dát zahŕňajú napríklad: histogramy, náhodné vzorkovanie, symbolické vzorkovanie apod.
- *Dopytovací jazyk* by mal byť podobný štandardu jazyka SQL. Jazyk SQL je známy deklaratívny jazyk, je široko používaný so zavedením štandardom, ktorý poskytuje flexibilitu a optimálnu evaluáciu dopytu a vykonanie nad prúdom, či datasetom.

Výskumné práce sa tiež venovali adaptívnym kontinuálnym dopytom nad prúdmi dát. Bolo ukázané, že takýto prístup môže mať značný prínos v oblasti výkonnosti systému vďaka jeho schopnosti adaptácie na zmeny v prúde dát. Tieto vlastnosti sú dosiahnuté aplikovaním zoskupovania indexov filtrov na priebežný výber predikátov (Madden et al., 2002).

Ďalší priestor na zlepšenie výkonnosti kontinuálnych dopytov nad prúdmi dát predstavujú adaptívne filtre. Pri dopytovaní sa takmer vždy vykonáva filtrovanie dát v nejakej podobe. Tento krok filtrovania je obvykle implementovaný v systéme na spracovanie dopytov. Pre zvýšenie výkonnosti dopytov je preto možné tieto filtre presunúť priamo do zdrojov dát. Ukázalo sa, že takýto prístup môže mať pozitívny dopad na výkonnosť (Olston et al., 2003). Tento prístup prinesie najmä redukciu prenášaných dát výmenou za ich nepresnosť. Problémom tejto techniky je, že je aplikovateľná len v prostredí, ktoré máme plne pod kontrolou a vieme zasahovať do všetkých jeho súčastí.

## 2.2 Detekcia zmien

Detekcia zmien (angl. concept drift) zohráva, v dnešnom rýchlo sa meniacom svete, dôležitú úlohu. Zmeny nastávajú veľmi rýchlo a nečakane. Preto stúpa potreba detektie zmeny a následná správna reakcia, ktorá vyplynie z detektovanej zmeny. Na to aby sme boli schopní na tieto zmeny adekvátnie reagovať je potrebné dátá spracovať tak ako vznikajú a pozerať sa na ne ako na prúd udalostí. Tradičné metódy pre paralelné spracovanie uvažujú len statickú kolekciu dát (Tran et al., 2014). Existuje niekoľko typov zmien, ktoré môžu nastať v prúde dát (Wadewale and Desai, 2015): *náhla* (angl. sudden), *inkrementálna*, *graduálna*, *opakujúca*, *falošná* a *šum*. Spomenuté zmeny sú zobrazené na obrázku 2.2. Detekcia zmeny predstavuje proces identifikácie zmeny aktu-

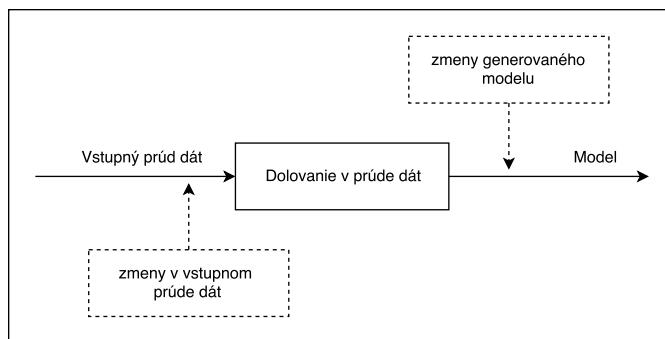


Obrázok 2.1: Typy zmien (angl. concept drift) (?).

álneho stavu modelu voči predchádzajúcemu. Na tento objekt sa pozéráme v rôznom čase. Dôležitý rozdiel medzi zmenou a rozdielom je, že zmena hovorí o prechode modelu do iného stavu, zatiaľ čo rozdiel znamená nepodobnosť v atribútoch dvoch objektov. V kontexte prúdu, detekovanie zmeny je proces segmentácie prúdu udalostí do rôznych segmentov a identifikovanie miest kde sa zmení dynamika prúdu (Ross et al., 2009). Metóda pre detekciu zmien musí riešiť nasledujúce úlohy (Tran et al., 2014): *detekcia zmeny* znamená správnu identifikáciu zmeny a *lokálizácia zmeny* hovorí o identifikovaní momentu kedy zmena nastala. Týmto úloh je potrebné venovať dostatočnú pozornosť, pretože zmeny môžu byť falošné alebo dočasné čo so sebou prináša problém lokálizácie danej zmeny. Ďalší rozdiel, ktorý je potrebné zadefinovať, je medzi rozdiel detekovaním posunu pojmu (angl. concept drift). Pre lepšiu čitateľnosť tohto

textu budeme pod pojmom detekcia zmeny, zmena rozumieť posun pojmu. Detekcia concept drift-u sa sústredí na označkované dátu, zatiaľ čo detekcia zmeny pracuje s označkovanými rovnako ako s neoznačkovanými dátami. Posun pojmu nazývame tiež časté zmeny v účelovej funkcií modelu, ktorý sa učí online.

Metódy pre detekovanie zmien môžme klasifikovať do nasledujúcich prístupov (Liu et al., 2010): *metódy založené na stave*, *metódy sledujúce trend* a *prahové metódy*. Algoritmus pre detekciu zmien by mal spĺňať aspoň nasledovné požiadavky: *presnosť*, *rýchlosť* a *odpoved' v reálnom čase*. Algoritmus by tiež mal detektovať čo najmenej chybných zmien a čo najviac správnych presných miest zmien. Algoritmy by mali byť prispôsobené reálnemu prostrediu a spracovaniu prúdov vysokých objemov a rýchlosťí. Na obrázku 2.2 je zobrazený všeobecný diagram pre detekciu zmeny v prúde udalostí.



**Obrázok 2.2:** Všeobecný diagram zobrazujúci detekciu zmeny v prúde udalostí (Tran et al., 2014).

Pre detekciu zmeny v prúdoch dát bolo vyvinutých niekoľko techník a metód. Niektoré z nich nižšie podrobnejšie popisujeme.

**Charakteristika dát** Metódy pre detekciu zmien môžu byť klasifikované na základe charakteru dát, s ktorými pracujú. Najčastejšie môžme prúdy klasifikovať do kategorických alebo numerických prúdov. Ak hovoríme o kategorických prúdoch, dáta obsiahnuté v prúde majú kategorický charakter, napríklad rôzny výrobcovia áut:  $x \in \{Volvo, Toyota\}$ . Pri numerických prúdoch dáta predstavujú numerické hodnoty  $x \in \mathbb{R}$ . Pre každý takýto prúd boli vyvinuté príslušné algoritmy. Problém nastáva pri aplikáciach s dátami reálneho sveta kde prúdy často obsahujú numerické aj kategorické dátá. V takýchto situáciach má zmysel

dáta rozdeliť rovnomenných skupín obsahujúce dátá rovnakého typu. Na tieto skupiny sú následne použité príslušné algoritmy. Prúdy dát sa ďalej môžu klasifikovať do označkovaných a neoznačkovaných prúdov. Neoznačkované prúdy obsahujú dátá, ktoré niesú zaradené do žiadnej triedy. Naopak označkované prúdy nesú v sebe informáciu o tom, do ktorej triedy patrí vybraný element. Rôzny charakter prúdu predstavuje rôzne zmeny a prístup na ich riešenie pri detekcii zmien v prúde (Tran et al., 2014).

**Metóda pre detekciu zmeny** V skratke DDM z anglického Drift Detection Method. Táto metóda sa zaobrá detekciou zmeny modelu. Majme prúd dát  $(x_i, y_i)$  kde  $x_i$  predstavuje atribúty a  $y_i$  triedu vzorky. Model sa potom snaží predikovať skutočnú triedu  $y_i + 1$  novej vzorky. Gama a spol. založili DDM na fakte, že každá iterácia klasifikátora predikuje triedu vzorky. Klasifikátor je binárny, takže trieda môže byť len *pravda* alebo *nepravda*. Potom, pre množinu vzoriek, chyba predstavuje náhodnú premennú z Bernoulliho pokusov (angl. Bernoulli trials). Vďaka tomu môžeme chybu modelovať s bínomickým rozdeľením. Nech  $p$  je pravdepodobnosť zlej predikcie a  $s_i$  je štandardná odchýlka vypočítaná nasledovne:

$$s_i = \sqrt{\frac{p_i(1-p_i)}{i}}$$

Pre každú vzorku z prúdu sú udržiavané dve premenné,  $p_{min}$  a  $s_{min}$ . Ich hodnoty sú použité na výpočet varovnej hodnoty, ktorá slúži na definovanie optimálnej velkosti kontextového okna. Kontextové okno si udržiava staré vzorky, ktoré obsahujú nový kontext resp. zmenu, či posun pojmu, a minimálny počet elementov zo starého konextu. Ak sa následne zníži množstvo chybne predikovaných vzoriek, okno je zahodené ako zle identifikovaná zmena (false alarm). Naopak, ak je dosiahnutá dostatočná varovná úroveň, predtým naučený model je zahodený a vytvorený nový, ale iba zo vzoriek ktoré boli uložené do kontextového okna (Gama et al., 2004; Brzeziński, 2010).

Existuje tiež rozšírenie EDDM, ktoré je modifikáciou DDM. Tento algoritmus používa rovnakú techniku varovných alarmov, ale namiesto klasifikácie chyby používa metriku množstva rozdielnych chýb. EDDM metóda dosahuje lepšie výsledky pri postupných zmenách, ale je citlivejšia na hluk v dátach (Wadewale and Desai, 2015).

**ADWIN** je skratka pre algoritmus s názvom adaptívne posuvné okno (angl. adapting sliding window). Tento algoritmus je vhodný je prúdy s náhlymi zmenami. Algoritmus si udržiava okno  $W$  s najnovšími vzorkami. Okno  $W$  je automatický zväčšované, ak nieje detekovaná žiadna výrazná zmena v prúde a naopak zmenšované, ak bola zmena detekovaná. Obmedzenie nárastu okna do nekonečna (žiadna zmena v prúde) je možné parametrom algoritmu, ktorý bude limitovať dĺžku okna  $W$ . ADWIN taktiež poskytuje ohraničenie výkonu na základe množstva falošne pozitívne a falošne negatívnych vzoriek (Wadewale and Desai, 2015). Základná verzia algoritmu ADWIN je vhodná pre 1-dimenzionálne dátá. Ak je potrebné detektovať zmeny pre viac-dimenzionálne dátá, potom sa vytvára paralelne niekoľko okien pre každú dimenziu dát (Brzeziński, 2010).

Existuje mnoho ďalších prístupov ako sa vysporiadať so zmenami v prúde, napríklad: exponenciálne váhovaný posuvný priemer, štatistické testovanie rovnomenného podielu, súborové (angl. ensemble) metódy. Popis všetkých metód je nad rámec tejto práce.

## 2.3 Detekcia anomálií

Detekcia anomálií (angl. anomaly detection) predstavuje proces identifikácie dát, ktoré sa význačne odchyľujú (angl. deviate) od historických vzorov (Hodge and Austin, 2004). Anomálie môžu spôsobovať chyby v meraní senzorov, nezvyčajné správanie systému alebo chyba pri prenose dát, či zámerné vytváranie anomálií v používateľmi generovanom obsahu. Takže detekcia anomálií má veľa praktického použitia napríklad v aplikáciach, ktoré dohliadajú na kvalitu a kontrolu dát (Hill et al., 2007) alebo adaptívne monitorovanie sietí (Hill and Minsker, 2010). Tieto aplikácie často kladú požiadavku aby boli anomálie detekované v čase ich vzniku, teda v reálnom čase. Potom metódy pre detekciu anomálií musia byť rýchle vo vykonávaní a mať inkrementálny charakter.

V minulosti sa obvykle anomálie detekovali manuálne s pomocou vizualizačných nástrojov, ktoré doménovým expertom pomáhali v tejto úlohe. Manuálne metódy avšak zlyhávajú pri detekcii anomálií v reálnom čase. Výskumníci navrhli niekoľko metód, ktoré majú myšlienku v prístupoch strojového učenia sa

a automatizovaného štatistického vyhodnocovania (Hill and Minsker, 2010): *minimálny objem elipsoidu, konvexný zvon, najbližší sused, zhľukovanie, klasifikácia neurónovou sieťou, klasifikácia metódou podporných vektorov a rozhodovacie stromy.* Tieto metódy sú pochopiteľne rýchlejšie než manuálna detektia, avšak jeden význačný nedostatok, bez úpravy niesú vhodné pre prúdové spracovanie v reálnom čase. Existujú napríklad rozhodovacie stromy, ktoré si dokážu budovať model inkrementálne, avšak sa líšia od dobre známych algoritmov. Táto metóda je podrobne popísana ďalej v texte.

*Dátovo riadená metóda* (angl. data-driven), ktorú navrhli (Hill and Minsker, 2010), využíva dátovo riadený jednorozmerný autoregresívny model prúdu dát a predikčný interval (ďalej len PI) vypočítaný z posledných historických dát na identifikáciu anomálií v prúde. Dátovo riadený model časového radu je použitý, pretože je jednoduchší na implementáciu a použitie v porovnaní s ostatnými modelmi prúdov dát. Tento model tiež poskytuje rýchle a presné prognózy. Dáta sú potom klasifikované ako anomálie na základe toho, či sú spadnú do zvoleného intervalu PI. Metóda teda poskytuje principiálny rámec pre výber hraničného prahu kedy majú byť anomálie klasifikované. Výhoda metódy je, že nevyžaduje žiadne vzorky dát, ktoré sú vopred označkované alebo klasifikované. Je veľmi dobre škálovateľná na veľké objemy dát a vykonáva inkrementálne počítanie tak ako dáta vznikajú. Metóda pozostáva z nasledujúcich krokov so začiatkom v čase  $t$ :

1. použi model na predikciu o krok vpred (angl. one-step-ahead), ktorý má ako vstup  $D^t = \{x_{t-q+1}, \dots, x_t\}$   $q$  je rôzne meranie  $x$  v čase  $t$  a  $D^t$  je model predikcie. Tento model je použitý na predikovanie hodnoty  $\bar{x}_{t+1}$  ako očakávaná hodnota v čase  $t+1$ .
2. výpočet hornej a spodnej hranice kam by malo spadnúť pozorované meranie s pravdepodobnosťou  $p$ .
3. porovnaj pozorovanie v čase  $t+1$ , či spadá do určeného intervalu. Ak spadne mimo interval, objekt je klasifikovaný ako anomália.
4. (a) pri stratégii metódy detekcie anomálií a zmiernenia (angl. anomalies detection and mitigation) ADAM, ak je pozorovaný objekt klasifikovaný ako anomália, modifikuj  $D^t$  odstránením  $x_{t-q+1}$  z konca

pozorovaného okna a pridaním  $\bar{x}_{t+1}$  na začiatok okna, čím vytvoríme  $D^{t+1}$ .

- (b) pri jednoduchej stratégii detekcie anomálií (angl. anomalies detection) AD, modifikuj  $D^t$  odstránením  $x_{t-q+1}$  z konca okna a pridaj  $x_{t+1}$  na začiatok okna čím vznikne  $D^{t+1}$ .

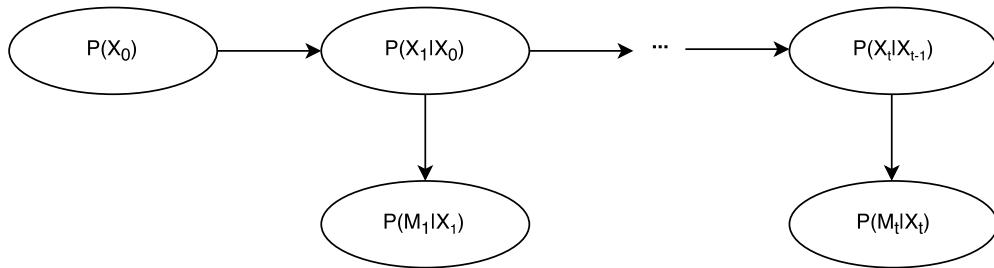
5. opakuj kroky 1-4

**Metóda dynamických bayesových sietí** (angl. Dynamic Bayesian Networks) (Hill et al., 2007) bola vytvorená pre detekciu anomálií v prúdoch zo senzorov, ktoré sú umiestnené v životnom prostredí. Bayesové siete predstavujú acyklický orientovaný graf, zobrazené na obrázku 2.3, v ktorom každý uzol obsahuje pravdepodobnosť informáciu v súvislosti k všetkým možným stavom, v ktorých sa môže premenná nachádzať. Táto informácia spolu s topológiou bayesovej siete, špecifikuje úplné spojenie distribúcie stavu premennej, pričom sada známych premenných môže byť použitá na odvodenie hodnoty neznámych premenných. Dynamické bayesové siete s topológiou, ktorá sa vyvýja v čase, pridáva nové stavové premenné pre lepšiu reprezentáciu stavu systému v aktuálnom čase  $t$ . Stavové premenné môžeme kategorizovať ako *neznáme*, ktoré predstavujú skutočný stav systému a *merané*, ktoré sú nedokonalé merania. Tieto premenné môžu byť naviac diskrétny alebo spojité. Nakoľko sa veľkosť siete zväčšuje s časom, vytváranie záverov použitím celej siete by bolo neefektívne a časovo náročné. Preto boli vyvinuté aproximačné algoritmy ako *Kalmanové filtrovanie* alebo *Rao-Blackwellized časticové filtrovanie*.

Hill et al. navrhli v (Hill et al., 2007) dve stratégie pre detekovanie anomálií v prúde dát:

- *Bayesov dôveryhodný interval* (angl. Bayesian credible interval - BCI), ktorý sleduje viacozmernú gausovskú distribúciu lineárneho stavu premennej, ktorý korešponduje s neznámym stavom systému a jej meraným náprotivkom.
- *Maximálne posteriori meraný status* (angl. Maximum a posteriori measurement status - MAP-ms) používa komplexnejšiu dynamickú bayseovú sieť. Princíp je rovnaký ako pri BCI, pričom MAP-ms metóda je naviac

rozšírená o status (napr. anomália áno/nie), ktorý je reprezentovaný distribúciou diskrétnej premennej každého merania senzoru.



**Obrázok 2.3:** Štruktúra dynamickej bayseovej siete. Vektor  $X$  reprezentuje spojitú zložku, neznáme alebo tiež nazývané skryté premenné systému a vektory  $M$  predstavujú spojité pozorované premenné v čase  $t$  (Hill et al., 2007).

## 2.4 Zhlukovanie

Zhlukovanie (angl. clustering) je proces zoskupovania alebo segmentácie objektov z dátovej množiny do zhlukov (angl. cluster) na základe črt objektov. Cieľom je vytvoriť zhluky, kde vrámci zhluku budú objekty čo najviac podobné a objekty medzi zhlukmi čo najviac odlišné. Podobne ako pri tradičných metódach pre zhlukovanie, aj metódy pre prúdy dát môžu byť rozdelené do piatich kategórií (Nguyen et al., 2015; Aggarwal, 2014): rozdeľovacie (angl. partitioning) metódy, hierarchické (angl. hierarchical) metódy, metódy založené na hustote (angl. density-based), metódy založené na mriežke (angl. grid-based) a metódy založené na modely (angl. model-based). Algoritmus potrebuje naviac kvantifikovať mieru podobnosti, či vzdialenosť zhlukov. Existujú štyri najpoužívanejšie miery pre meranie vzdialenosť: minimálna vzdialosť (angl. single-linkage), maximálna vzdialosť (angl. complete-linkage), priemerná a stredná vzdialosť. Spomenuté miery vzdialenosť sa v niektornej literatúre uvádzajú ako typy zhlukovania, pričom metrika vzdialenosť môže byť napríklad kosínusová podobnosť, v inej literatúre je naopak abstrahované od týchto metrik a miery ako minimálna vzdialosť považujú za miery podobnosti zhlukov. Zhlukovanie je príklad *učenia bez učiteľa* (angl. unsupervised learning) narozené od klasifikácie. Metódy zhlukovania sú často používané počas predspracovania dát napríklad s cieľom redukcie dimenzionality.

*Rozdeľovacie metódy* (angl. partitioning methods) rozdeľujú dátovú množinu o  $n$  objektoch do  $k$  partící kde každá partícia predstavuje zhluk, pričom platí  $k \leq n$ . Parameter  $k$  je obvykle definovaný používateľom vopred. Najznámejšie tradičné metódy sú  $k - means$  a  $k - medians$ . Existujú implementácie, ktoré upravujú  $k - means$  tak aby bola použiteľná na prúdy dát. Všetky tieto implementácie spracujú prúd v malých dávkach, takže nie celkom spôsobom ako je vhodné spracovať prúdy dát (Gaber et al., 2005).

Jeden z prvých algoritmov, ktoré boli navrhnuté pre prúdy dát je *STREAM*, algoritmus je rozšírením algoritmu  $k - medians$ . Algoritmus používa techniku rozdeľuj a panuj (angl. divide-and-conquer) s cieľom vytvárania zhlukov inkrementálne. Účelová funkcia algoritmu *STREAM* je nasledovná:

$$SSQ(M, C) = \sum_{i=1}^k \sum_{x_j \leftarrow c_i} dist(x_j, c_i)$$

kde  $x$  je dátová vzorka a  $c$  reprezentuje zhluk (medián). Funkcia  $dist$  je funkcia na meranie vzdialenosť medzi zhlukmi. Algoritmus avšak tiež spracováva dátu v malých dávkach. Na rozhodnutie o veľkosti dávky používa algoritmus *LOCALSEARCH*.

*Metódy založené na hustote* vytvárajú profil hustoty dátovej množiny. Tento profil je následne použitý na zhlukovanie. Znamená to teda, že za zhluky považujeme miesta v priestore s vysokou hustotou objektov. Výhodou tejto metódy je, že dokáže objaviť v dátach aj neobvyklé tvary zhlukov. Najznámejšie implementácie sú *DBSCAN* a *OPTICS*. Toto je všeobecná výhoda metód založených na hustote v porovnaní s rozdeľovacími metódami (Han et al., 2011).

Algoritmus *DenStream* je rozšírením algoritmu *DBSCAN*, ktorý je vhodný pre zhlukovanie prúdov dát. Tento algoritmus podobne ako *CluStream* algoritmus navrhnutý Aggarwalom vytvára mikro zhluky na zachytenie informácie o prúde dát. Mikro zhluky sú kontinálne aktualizované a udržiavané v kolekcii mikro zhlukov. Algoritmus používa oslabujúci model na zníženie váh elementov v čase. Vytvárané sú tri typy mikro zhlukov: základný, potenciálny a vyčnievajúci (angl. outlier). Algoritmus potom aplikuje známy *DBSCAN* algoritmus na vytvorené mikro zhluky, pričom zhluk vzniká z viacerých mikro zhlukov, ktoré sú pokope (Nguyen et al., 2015).

Algoritmus *OPTICS – stream* je opäť rozšírenie algoritmu *OPTICS*. Podobne ako *DenStream* vytvára mikro zhluky a aplikuje oslabujúci model. Na vytvorenie finálnych zhlukov rovnako používa pôvodný algoritmus *OPTICS* z vytvorených mikro zhlukov.

*Metódy založené na modeloch* sa snažia optimalizovať podobnosť medzi dátami a statickými modelmi. Známe tradičné metódy sú napríklad *Expectation – Maximization(EM)* a *Self – OrganizingMap(SOM)*. EM je jemná (angl. soft) metóda pre zhlukovanie, SOM je metóda neurónových sietí.

*SWEM* je algoritmus rozšírený z EM algoritmu. Tento algoritmus používa posuvné okno. Každý mikro komponent je reprezentovaný n-ticou (váha, priemer, kovariančná matica). Najprv je aplikovaný algoritmus EM na získanie konvergujúcich parametrov, následne používa získané parametre ako inicializačné hodnoty pre vytvorenie modelu. SWEM tiež aplikuje oslabujúci model na expiráciu summarizačných štatistik mikro komponentov (Nguyen et al., 2015).

Ďalším algoritmom, ktorý vytvára statický model je *GCPSON*, ktorý je hybridným algoritmom vytvorený z *GSOM* a *CPSOM*. Algoritmus GSOM je vyvíjajúci sa SOM kde nieje potrebné vopred definovať veľkosť mapy. Mapa GSOM dynamicky rastie podľa hodnoty akumulovaných chýb. CPSOM je bunkový pravdepodobnostný SOM, ktorý používa oslabujúce okno s cieľom redukcie váh neurónov. Teda, GCPSON má schopnosť dynamického rastu mapy čít pre zhlukovanie prúdov dát a udržiavať zhluky tak ako sa prúd vyvíja v čase (Nguyen et al., 2015).

*Hierarchické metódy* Hierarchické metódy zoskupujú dátové objekty do zhlukov hierarchických stromov. Tieto metódy ďalej rozdeľujeme na agglomeratívne a rozdeľovacie kde dekompozícia hierarchií je formovaná zdola nahor spájaním alebo zhora nadol rozdeľovaním. Tradičné metódy sú napríklad BIRCH, CURE alebo ROCK. Metóda CluStream je použiteľná pre prúdu dát pričom rozšíruje tradičnú metódu BIRCH. CluStream používa mikro zhluky pre získanie súmárnych informácií o prúde dát. Mikro zhluky sú definované ako rozšírenie funkčných vektorov metódy BIRCH s pridanou časovou zložkou. CluStream si udržuje množinu mikro zhlukov, ak je vytvorený nový mikro zhluk, iný ktorý je outlier je odstránený alebo sú dva podobné mikro zhluky spojené do jedného. Tento algoritmus tiež analyzuje vývoj mikro zhlukov s cieľom odhaliť zmeny

v prúde dát (Nguyen et al., 2015). Algoritmus HPStream je rozšírením algoritmu CluStream, ktorý adresuje problém zhlukovania vysoko dimenzionálnych dát. Tento algoritmus sa vysporiadáva s takýmito dátami projekčnou technikou pre výber najlepšieho atribútu pre zhluky. SWClustering je algoritmus, ktorý rieši problém postupnej degradácie algoritmu CluStream, ak beží dlhú dobu. SWClustering vytvára dočasný zhluk črt pre posuvné okno. Následne je použitý exponenciálny histogram črt zhlukov pre identifikáciu zmien v prúde dát. Tento algoritmus tiež dosahuje lepšiu časovú a pamäťovú efektivitu ako CluStream (Han et al., 2011).

*Metódy založené na mriežke* rozdeľujú priestor na multi-dimenzionálnu mriežku. Mriežka môže obsahovať veľa buniek, pričom každá môže obsahovať svoj podpriestor a naviac si udržuje sumárne informácie o dátach v podpriestore bunky. Zhluky sú potom identifikované hustými oblastami v okolí buniek. Známy algoritmus pre zhlukovanie prúdov dát podľa mriežky je CellTree (Han et al., 2011). Algoritmus je inicializovaný rozdelením priestoru do množiny rovnako veľkých buniek. CellTree bol rozšírený na lepšiu verziu Cell\*Tree, ktorý používa algoritmus BTree na ukladanie informácií o zhlukoch. Cell\*Tree tiež aplikuje starnúci model na zvýraznenie poslednej zmeny v prúde dát.

## 2.5 Klasifikácia

Klasifikácia je proces hľadania všeobecného modelu, ktorý je vytvorený za základe predchádzajúcich pozorovaní. Tieto pozorovania resp. prúd dát musí obsahovať označované dátá, klasifikácia predstavuje teda úlohu učenia s učiteľom. Vytvorený model je potom použitý na klasifikovanie nových dát. Proces klasifikácie pozostáva tradične z dvoch krokov: *učenie* a *tréновanie*. V kontexte klasifikácie prúdu dát je avšak učenie kontinuálne. Testovanie môže byť tiež kontinuálne, avšak to závisí od zvolenej metódy pre evluáciu klasifikátora. Počas učenia sa snažíme podľa algoritmu vytvoriť klasifikačný model z trénovacích dát (trénovacia množina). Počas testovania je vytvorený model použitý na klasifikovanie neoznačovaných dát z testovej množiny. Existujú rôzne dobre známe metódy pre klasifikáciu: rozhodovacie stromy, naivný Bayes, neurónové siete alebo k-najbližších susedov (Nguyen et al., 2015). Niektoré z týchto metód sú v upravenej podobe vhodné na klasifikáciu prúdov dát, vy-

brané z nich sú detailnejšie popísané v tejto podkapitole. Základné podmienky použiteľnosti algoritmov pre klasifikáciu prúdu dát sú limitované: *ohraničený čas spracovania dát, ohraničená veľkosť použitej pamäte a okamžité použitie modelu.*

Problém klasifikácie je zvyčajne formálne definovaný nasledovne: nech  $A$  je trénovacia množina o  $N$  prvkoch vo forme  $(x, y)$  kde  $y$  predstavuje skutočnú triedu vzorky a  $x$  je vektor s  $d$  atribútmi. Každý atribút môže nadobúdať numerické alebo kategorické hodnoty. Cieľom je vytvoriť na základe trénovacej množiny model resp. funkciu  $y = f(x)$ , ktorá bude predikovať triedu  $y$  pre nové vzorky  $x$  s vysokou presnosťou (Domingos and Hulten, 2000).

Väčšina klasifikačných metód používa vzorkovanie s cieľom zvýšiť presnosť klasifikátora (Aggarwal, 2014; Nguyen et al., 2015). Často je použitá technika zásobníkového vzorkovania (angl. Reservoir sampling), ktorá umožňuje zvýšiť efektivitu klasifikátora. Myšlienka je v udržiavaní malej kontinuálnej trénovacej vzorke dát. Klasifikačný algoritmus je potom kedykoľvek aplikovaný na vzorku s cieľom vytvorenia modelu (Aggarwal, 2014). Klasifikácia je problém *učenia s učiteľom* (angl. supervised learning) čo znamená, že pri trénovaní sú známe skutočné triedy dátových vzoriek.

*Multinomiálny naivný Bayes* je klasifikátor najčastejšie používaný na klasifikáciu dokumentov, ktorý obvykle poskytuje dobré výsledky aj čo sa týka presnosti výsledku aj rýchlosťi. Túto metódu je jednoduché aplikovať v kontexte prúdu dát (Bifet and Frank, 2010). Multinomiálny naivný Bayes sa pozera na dokument ako na zhľuk slov. Pre každú triedu  $c$ ,  $P(w|c)$ , pravdepodobnosť, že slovo  $w$  patrí do tejto triedy je odhadovaná z trénovacích dát jednoducho vypočítaním relatívnej početnosti každého slova v trénovacej sade pre danú triedu. Klasifikátor potrebuje naviac nepodmienenú pravdepodobnosť  $P(c)$ . Za predpokladu, že  $n_{wd}$  je počet výskytov slova  $w$  v dokumente  $d$ , pravdepodobnosť triedy  $c$  z testovacieho dokumentu je nasledovaná:

$$P(c|d) = \frac{P(c) \prod_{w \in d} P(w|c)^{n_{wd}}}{P(d)}$$

Kde  $P(d)$  je normalizačný faktor. Aby sme sa vyhli problému kedy sa trieda nevyskytuje v datasete ani jeden krát, je bežné použitie Laplacovej korekcie a

nahradenie nulových početností jednotkou, resp. inicializovať početnosť každej triedy na 1 namiesto 0.

**Stochastický gradientný zostup a metóda podporných strojov** (angl. Stochastic Gradient Descent, SGD). Bifet a Frank v ich práci použili implementáciu tzv. vanilla stochastický gradientný zostup s pevnou rýchlosťou učenia, optimalizujúc stratu s  $L_2$  penalizáciou.  $L_2$  penalizácia je často používaná pri podporných vektorových strojoch (angl. Support Vector Machines, ďalej len SVM). Lineárny stroj, ktorý je často aplikovaný na problémy klasifikácie dokumentov, optimalizujeme funkciu straty nasledovne:

$$\frac{\lambda}{2} \|w\|^2 + \sum [1 - (yxw + b)]_+$$

kde  $w$  je váhovaný vektor,  $b$  je sklon,  $\lambda$  regulačný parameter a označenie triedy  $y$  je z intervalu  $\{+1, -1\}$ .

SVM ukazujú dobré výsledky v mnohých úlohách a problémoch strojového učenia, ak je táto metóda použitá na statické datasety. Avšak, ich použitie v neupravenej forme je problematické na prúdy dát kvôli ich časovej zložitosti  $O(N^3)$  a pamäťovej zložitosti  $O(N^2)$ , kde  $N$  je počet dátových vzoriek (Nguyen et al., 2015). Tsang a spol. navrhli metódu jadrových vektorových strojov (angl. Core Vector Machine, CVM), ktorá používa uzavretie minimálnou guľou (angl. Minimum Enclosing Ball - MEB, v 2D priestore tiež známe ako problém pokrycia minimálnou kružnicou) na redukciu časovej a pamäťovej zložitosti. Metóda StreamsSVM je rozšírením CVM a bola navrhnutá s ohľadom na spracovanie prúdov dát (Rai et al., 2009). StreamsSVM používa flexibilný rádius MEB, ktorý sa mení podľa nových vzoriek z prúdu dát. Výsledky sa blížia k výsledkom z optimálneho algoritmu, avšak problém tejto metódy je neschopnosť vysporiadať sa so zmenami (angl. concept-drift) v prúde dát.

**Rozhodovacie stromy** (angl. Decision trees) sú častou metódou používanou na klasifikáciu. Modely rozhodovacích stromov dosahujú v praxi vysokú presnosť zatiaľ čo model je jednoduchý na vysvetlenie (Jin and Agrawal, 2003; Hulten et al., 2001; Domingos and Hulten, 2000; Aggarwal, 2014). Existuje niekoľko škálovateľných metód pre rozhodovacie stromy, napríklad SLIQ, Rain-forest alebo BOAT (Aggarwal, 2014). Napriek tomu, že sú tieto metódy škálo-

vateľné, nie sú navrhnuté a ani vhodné na použitie pre prúdy dát. Neskôr boli navrhnuté rodiny algoritmov ako ID3, ktoré boli síce navrhnuté aj s ohľadom na prúdy dát, ale problém je že neboli tak aby zohľadnili zmeny v modeli. Rozhodovacie stromy predikujú resp. klasifikujú novú vzorku do triedy  $y$  podľa výsledkov testov v rozhodovacích uzloch a triedy v liste stromu do ktorého spadne vzorka.

Jedna z prvých state-of-the-art metód, ktorá bola navrhnutá špecificky pre prúdy dát je *Hoeffdingov strom* (angl. Hoeffding tree, ďalej len HT). Je to najznámejšia implementácia rozhodovacích stromov v použití prúdového spracovania (Domingos and Hulten, 2000; Aggarwal, 2014; Nguyen et al., 2015). HT vyžaduje prečítanie každej novej vzorky z prúdu najviac jeden krát. Táto vlastnosť umožňuje použitie HT nad prúdmi dát s akceptovateľnou časovou a pamäťovou zložitosťou. Prečítané vzorky nieje potrebné ukladať na disk. HT využíva fakt, že malá vzorka dát je často postačujúca na výber optimálneho rozdelovacieho atribútu. Toto tvrdenie je matematicky podporené Hoeffdingovou mierou alebo súčtovou Chernoffovou mierou (Domingos and Hulten, 2000; Han et al., 2011). Rutkowski a spol. tvrdia, že stromy, ktoré používajú Hoeffdingovu mieru v skutočnosti používajú McDiarmidovu mieru a mali by sa tieto stromy nazývať McDiarmidove (Rutkowski et al., 2013). HT dosahujú sa vo všeobecnosti asymptoticky približujú kvalitou k tým, ktoré sú vytvorené metódou pre dávkové spracovanie (Hall et al., 2009).

Predpokladajme  $N$  nezávislých pozorovaní náhodnej premennej  $r \in R$  kde  $r$  je metrika výberu atribútu. V prípade HT to môže byť napríklad informačný zisk (angl. information gain). Ak vypočítame priemer vzorky  $\bar{r}$  potom Hoeffdingova miera hovorí, že skutočný priemer  $r$  je aspoň  $\bar{r} - \epsilon$  s pravdepodobnosťou  $1 - \delta$ . Pričom  $\delta$  je parameter definovaný používateľom a

$$\epsilon = \sqrt{\frac{R^2 \ln(1/\delta)}{2n}}$$

HT algoritmus používa Hoeffdingovu mieru na výber najmenšieho čísla  $N$  - počet vzoriek potrebných v uzle na výber rozdelovacieho atribútu. Presnejšie, v každom uzle stromu maximalizujeme  $G(A_j)$  kde funkcia G predstavuje metriku kvality atribútu  $A_j$  vzorky, napríklad informačný zisk. Cieľom je nájsť najmešší počet vzoriek  $N$  tak aby bola splnená Hoeffdingova miera. Nech  $G(A_a)$  predstavuje atribút s najvyššou hodnotou  $G$  a nech  $G(A_b)$  je druhý najlepší

atribút. Potom ak  $G(A_a) - G(A_b) > \epsilon$  môžme s istotou tvrdiť, že rozdiel je väčší ako nula. Následne vyberieme  $A_a$  ako najlepší rozdeľovací atribút v danom uzle s istotu  $1 - \delta$ . Jediné dátá, ktoré si potrebuje HT algoritmus ukladať sú postačujúce štatistiky potrebné pre rozhodovanie a výpočet Hoeffdingovej miery, sú to počítadlá  $n_{ijk}$  pre hodnotu  $v_j$  atribútu  $A_i$  z triedy  $y_k$ . Slabá stránka tohto algoritmu je v tom, že očakáva na vstupe prúd, ktorý neobsahuje zmeny (angl. concept-drift) (Domingos and Hulten, 2000).

Existuje niekoľko modifikácií HT algoritmu. Tá najzákladnejšia je jeho rýchla verzia (angl. Very Fast Decision Trees, VFDT) (Domingos and Hulten, 2000). Modifikácia HT algoritmu, ktorá sa vie vysporiadať so zmenami v prúde sa nazýva *Rýchly algoritmus pre rozhodovacie stromy adaptujúci sa na zmeny* (angl. Concept-adapting Very Fast Decision Tree, CVFDT (Hulten et al., 2001)). CVFDT používa posuvné okno, pričom nevytvára pri detekovanej zmene nový model. Namiesto toho aktualizuje postačujúce štatistiky v uzloch inkrementovaním počítadiel nových vzoriek a dekrementovaním počítadiel vzoriek, ktoré vypadli z posuvného okna. Teda, ak je v prúde dát zmena, niektoré uzly stromu nemusia viac splňať Hoeffdingovu mieru. Keď nastane takáto situácia, alternujúci podstrom začne narastať v uzle, ktorý nesplnil Hoeffdingovu mieru. s novými vzorkami bude alternujúci podstrom rástť, zatiaľ bez toho aby bol použitý v modeli na klasifikáciu. V momente keď sa stane alternujúci podstrom presnejší ako aktuálny, starý podstrom je nahradený alternujúcim podstromom. V algoritme je možné nastaviť hraničnú hodnotu minimálneho počtu vzoriek, ktoré musí alternujúci podstrom spracovať predtým než sa pokusí nahradiť pôvodný (Hulten et al., 2001).

Ďalšou modifikáciou HT je *Adaptívny sa Hoeffdingov strom* (angl. Hoeffding Adaptive Tree) predstavený Bifetom a Gavaladom v 2009. Princíp je veľmi podobný ako CVFDT, ale myšlienka je minimalizovať počet parametrov, ktoré musí používateľ nastaviť (napr. veľkosť okna  $W$  je požadovaný parameter CVFDT). Adaptívny HT používa rôzne kritéria pre odhad potrebnej veľkosti okna automaticky, napríklad algoritmus *ADWIN*. S použitím tohto kritéria používateľ nemusí zadať parameter veľkosti okna čo je obrovský prínos, pretože potrebná veľkosť sa môže meniť spolu so zmenami v prúde dát. Adaptívny HT s kritériom ADWIN dosahuje v niektorých prípadoch lepšie výsledky ako CVFDT (Bifet and Gavaldà, 2009).

Existujú aj ďalšie odlišné metódy rozhodovacích stromov, ktoré aplikujú súborové (angl. ensemble) metódy, rôzne klasifikátory v listoch ako napríklad naivný Bayes, či stromy založené na fuzzy logike (Aggarwal, 2014).

## 2.6 Zhodnotenie

Existuje mnoho problémov analýzy prúdu dát. Väčšina týchto problémov je odvodených od tých z tradičného dávkového spracovania dát. Vďaka tomu tiež väčšina algoritmov na riešenie týchto úloh a problémov rozširuje tradičné algoritmy. Ako napríklad algoritmus rozhodovacích stromov pre klasifikáciu, ktorý používa Hoeffdingovu mieru pre určenie istoty výberu najlepšie atribútu pre vytvorenie rozhodovacieho uzla. Niektoré metódy do istej miery riešia problém zmien v prúde dát, avšak často len pomocou statických nastavených parametrov používateľom. Tento prístup zlyháva, ak sa menia aj samotné zmeny, čo je častý prípad prúdov reálneho sveta.

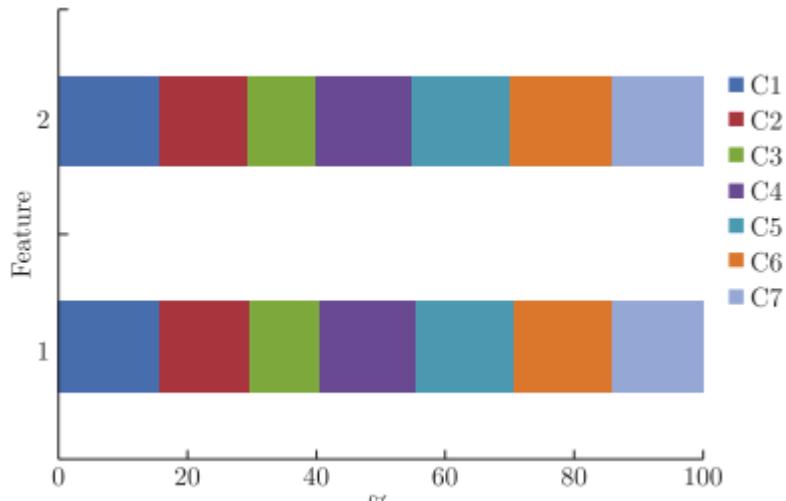
### 3. Interpretácia a vysvetlenie výsledkov analytických úloh

Vysvetlenie a prezentacia analytických modelov je kritickým komponentom pri procese analýzy dát. Je dôležité aby používateľ nevidel výsledný model len ako čiernu skrinku, ale aby bol schopný pochopiť čo viedlo k vzniku modelu. Lepšie povedané, cieľom je čo možno najjednoduchšie vysvetlenie fungovania modelu bez nutnosti všetkých detailov o fungovaní algoritmu, ktorý tvorí model. Dobrým príkladom môže byť model rozhodovacích stromov. Tento model je jednoduchý a intuitívne interpretovateľný, zatiaľ čo nie je potrebná detailná znalosť rôznych algoritmov rozhodovacích stromov.

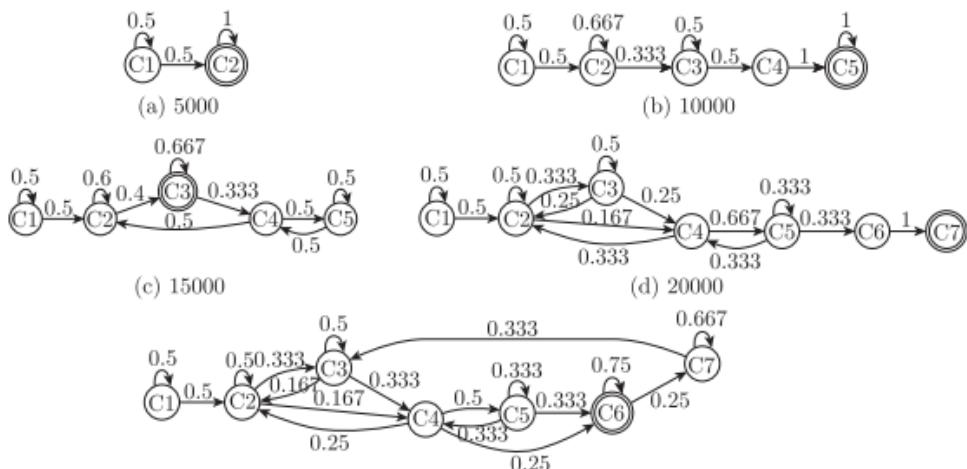
Ďalšia neoddeliteľná súčasť prezentácie modelu je jeho vizuálna reprezentácia relevantná pre pochopenia používateľom. cieľom podobných vizualizácií je nájsť rovnováhu medzi vnímaním a poznávaním a vyžiť tak všetky možnosti ľudského mozgu. Správne vysvetlenie modelu tiež zvyšuje používateľovú dôveryhodnosť vo vytvorený model (Demšar and Bosni, 2014; Barlow and Neville, 2001).

Prúd dát môže byť chápáný ako sekvencia pozorovaní v čase. Výskumy zamerané na techniky vzorkovania zobrazujú dátu ako prechodný alebo dočasný komponent (angl. temporal component). V tomto kontexte vizualizácia predstavuje *sumarizáciu* (Demšar and Bosni, 2014). Výzvou je zobrazovať prechodný komponent v 2D vizualizácií bez animácie. Zobrazovanie zmien (angl. concept drift) je ďalší parameter, ktorý robí vizualizáciu a teda aj prezentáciu a vysvetlenie celého modelu náročnou úlohou. Aj v prípade, ak dokážeme efektívne summarizovať dátu a zobraziť v objavené vzory, na konci stále máme problém vizualizovať zmeny. Možnosťou je použiť viac paralelných vizualizácií pre každú zmenu. Problém tohto prístupu je, že s narastajúcim počtom zmien začne byť vizualizácia neprehľadná a vedieť k fenoménu známeho pod pojmom *neviditeľnosť zmeny* (angl. change blindness) (Demšar and Bosni, 2014).

Výzvou preto ostáva jednoduchá interpretácia modelu a vizualizácia jeho zmeny v čase tak aby táto celková prezentácia ostala jednoduchá na pochopenie pre používateľa (Demšar and Bosni, 2014; Yao, 2013). Na nasledujúcich obrázkoch sú znázornené vizualizácie, ktoré majú snahu vizualizovať zmenu v prúde dát a modeloch vytvorených nad prúdom dát.

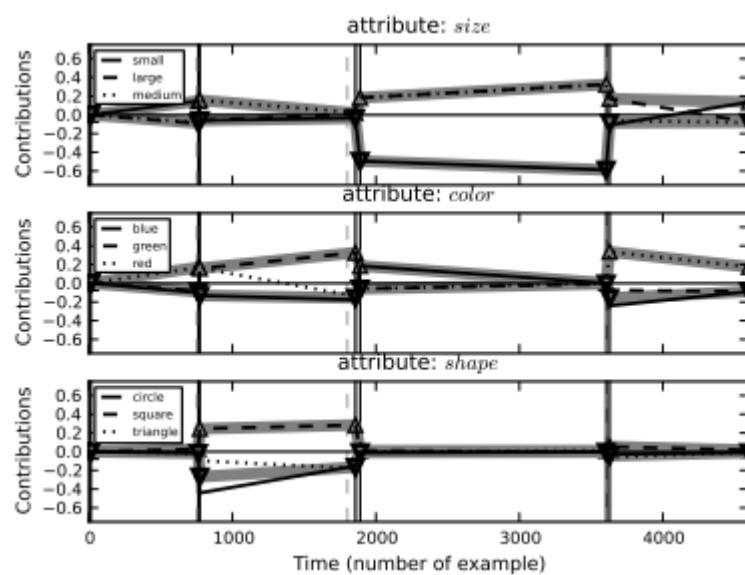


**Obrázok 3.1:** Distribúcia zmien v prúde dát (Yao, 2013).



**Obrázok 3.2:** Mapa transformácií zmien vizualizovaná ako stavový automat (Yao, 2013).

Síce sa tieto práce venujú vizualizácií zmien v prúde dát a ich modelov, problémom je, že nevenujú takmer žiadnu pozornosť kvantitatívnému, či expertnému vyhodnoteniu vizualizácií. Vo väčšine prác konštrujú autori vlastné závery



**Obrázok 3.3:** Vizualizácia detekovaných zmien v čase ich vzniku (Demšar and Bosni, 2014).

bez používateľskej štúdie, či expertného posúdenia celej aplikácie ako celku. V tomto identifikujeme nedostatok a preto sa aj v našej práci zameriavame na interpretáciu výsledkov spolu s vizualizáciou modelu, pričom kladieme dôraz na zobrazenie zmien. Rovnako budeme venovať značnú pozornosť na vyhodnotenie našich výsledkov aby sme mohli rozumne posúdiť použiteľnosť navrhovanej metódy.



## 4. Existujúce nástroje pre analýzu prúdu udalostí

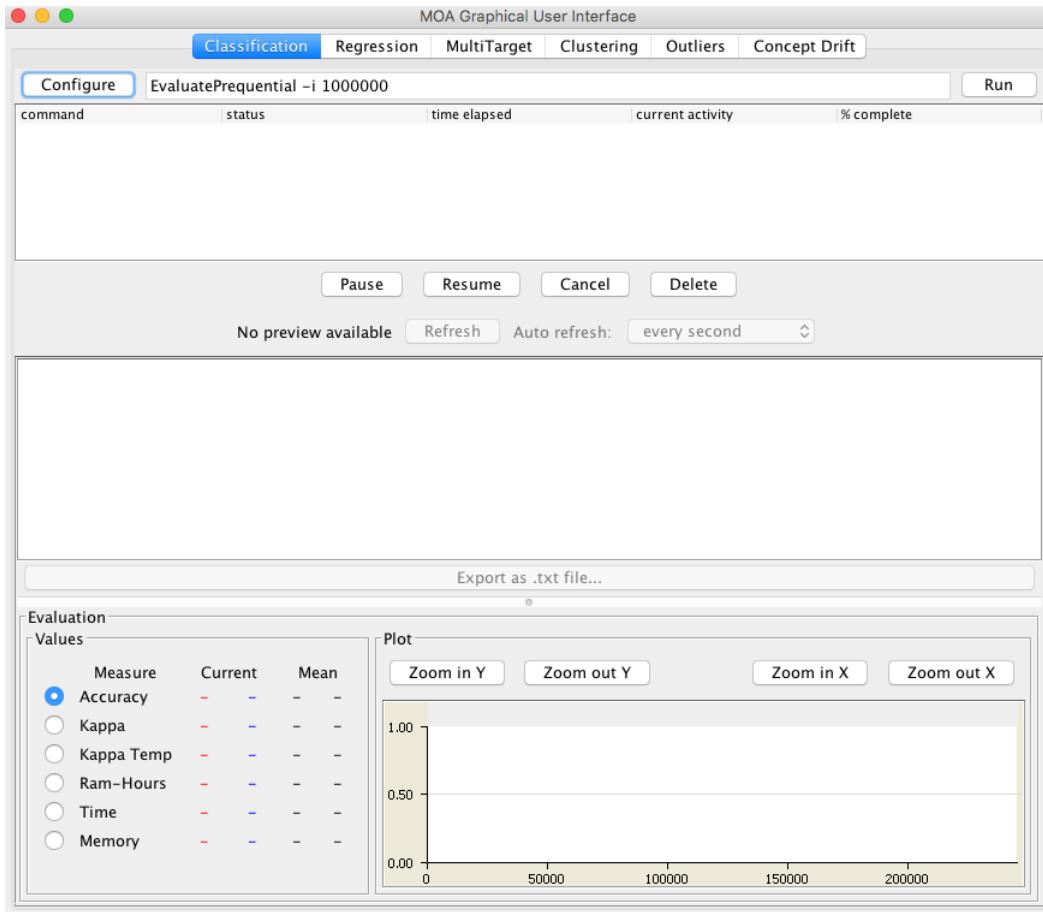
Existuje niekoľko nástrojov, ktoré boli vytvorené s cieľom uľahčiť analýzu a spracovanie prúdov dát. Niektoré z nich sú integrované do existujúcich riešení ako napríklad RapidMiner. Ďalšie nástroje naopak vznikli na zelenej lúke, príkladom takýchto nástrojov môže byť MOA, či WEKA. Špeciálnu pozornosť si zaslúži nástroj MOA, pretože poskytuje širokú bázu algoritmov a API rozhranie pre programovanie a rozširovanie týchto algoritmov. Algoritmy v tomto nástroji sú zamerané na vytvorenie modelov prúdu dát pre napríklad klasifikáciu, nástroj tiež obsahuje príslušné metódy pre evaluáciu kvality vzniknutých modelov. Ďalšími nástrojmi sú Spark, či Storm, ktoré sú orientované viac na samotné spracovanie prúdov dát a poskytujú priestor pre spoľahlivé a škálovateľné riešenia, ktoré môžu byť napríklad klasifikačné algoritmy.

### 4.1 MOA

Masívny online analyzátor (z angl. Massive Online Analysis - MOA, ďalej len MOA) je softvérové prostredie pre implementáciu algoritmov a vykonávanie experimentov pre online učenie sa z vyvýjajúcich sa prúdov dát (Bifet et al., 2010). MOA pozostáva z kolekcie offline a online metód a tiež nástrojov pre evaluáciu týchto metód. MOA implementuje metódy a algoritmy pre klasifikáciu, zhľukovanie prúdu, detekciu inštancií, ktoré sa vymykajú prahovým hodnotám a tiež odporúčacie systémy. Presnejšie MOA implementuje napríklad nasledujúce: stupňovanie (angl. boosting), vrecovanie (angl. bagging) a Hoeffdingove stromy, všetky metódy s a bez Naive Bayes klasifikátorom na listoch. MOA podporuje obojsmernú interakciu s nástrojom WEKA, ktorý je detailne opísaný v nasledujúcej kapitole. Na 4.1 je možné vidieť grafické používateľské rozhranie nástroja (GUI) MOA. Okrem GUI je možné použiť konzolové a API rozhranie prostredníctvom, ktorého sa dá programovať nad

rámcom/nástrojom MOA.

MOA je implementovaná v programovacom jazyku Java. Za hlavnú výhodu implementácie v Java považujú autori jej platformovú nezávislosť. MOA obsahuje tiež syntetický generátor prúdu dát, ktorým je možné modelovať tiež concept drift. V aplikácii je možné definovať pravdepodobnosť, že inštancia prúdu patrí do nového concept drift-u. Možnosti, ktoré MOA poskytuje sú: generovanie prúdov dát, klasifikátory (napr. HoeffdingTree), metódy pre zhlukovanie spolu s ich vizuálizáciu a rozhranie do nástroja WEKA. Sú dostupné nasledovné generátory prúdu (Bifet et al., 2010): *Random Tree Generator, SEA Concepts Generator, STAGGER Concepts Generator, Rotating Hyperplane, Random RBF Generator, LED Generator, Waveform Generator, and Function Generator*.



Obrázok 4.1: Hlavná obrazovka GUI nástroja MOA.

Evaluácia modelu je dôležitou súčasťou dolovania a analýzy dát. MOA implementuje niekoľko state-of-the-art metód pre evaluáciu modelov vytvorených

príslušným algoritmom. Je možné argumentovať, že nasledujúce metódy nie sú najvhodnejšie pre evaluáciu. Toto tvrdenie je akceptovateľné, pretože evaluácia algoritmov pre dolovanie a analýzu prúdov dát je samostatná výskumna oblasť, ktorá sa ešte len vyvíja. Implementované metódy v MOA pre evaluáciu sú:

- *Holdout* je vhodné použiť, pretože križová validácia môže byť často príliš časovo náročná kvôli objemu dát. Namiesto toho je použitá jediná holdout množina na vyhodnotenie kvality modelu.
- *Prerušované testovanie-potom-trénovanie* (angl. Interleaved Test-Then-Train alebo tiež Prequential) funguje tak, že každá nová vzorka sa najprv použije na testovanie a následne na trénovanie. Vďaka tomu môže byť presnosť modelu inkrementálne aktualizovaná. Výhoda je, že nie je potrebné udržiavať holdout množinu v pamäti.

Pre spustenie grafického používateľského rozhrania nástroja MOA je potrebné stiahnuť nástroj<sup>1</sup> a v príkazovom riadku spustiť nasledujúci príkaz:

```
java -Xmx4G -cp moa.jar -javaagent:sizeofag.jar moa.gui.GUI
```

Používanie MOA z príkazového riadku:

```
java -cp moa.jar -javaagent:sizeofag.jar moa.DoTask \
"EvaluatePeriodicHeldOutTest -l \
(OzaBag -l trees.HoeffdingTree -s 10) \
-s generators.WaveformGenerator \
-n 100000 -i 100000000 -f 1000000" > htresult.csv
```

## 4.2 WEKA

The Wakaito Enviroment for Knowledge Analysis (ďalej len Weka) vznikol s jednoduchým cieľom poskytnúť výskumníkom unifikovanú platformu pre prístup k state-of-the-art technikám strojového učenia sa (Hall et al., 2009).

---

<sup>1</sup><http://moa.cms.waikato.ac.nz/downloads/>

Weka vznikla na University of Waikato na Novom Zélande v roku 1992, pričom je aktívne využívaná posledných 16 rokov. Weka poskytuje kolekciu algoritmov strojového učenia sa pre úlohy dolovania v dátach. Algoritmy môžu byť priamo aplikované na datasety prostredníctvom aplikácie alebo použité vo vlastných aplikáciach volaním Java kódu. Weka obsahuje tiež nástroje na predspracovanie dát, klasifikáciu, regresiu, zhlukovanie, asociačné pravidlá a vizualizáciu. Nástroj je tiež vhodný pre navrhovanie a vývoj nových schém pre strojové učenia sa v kontexte dolovania dát. Zaujímavosťou je tiež, že Weka je nelietajúci vták, ktorý žije len na ostrove Nového Zélandu.

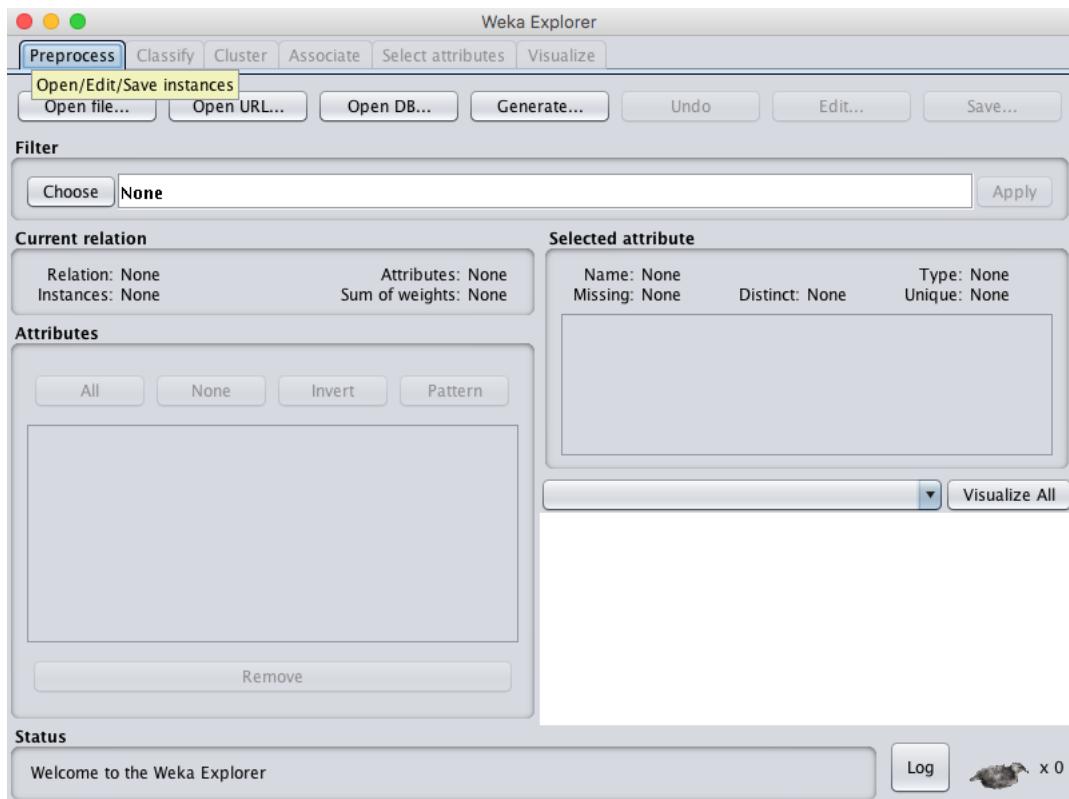
Cieľom nástroja je poskytnúť pracovný nástroj pre výskumníkov. Poskytuje napríklad (nástroj ich obsahuje omnoho viac, vymenované sú len vybrané) tieto algoritmy určené pre klasifikáciu dát:

- *Bayesová logistická regresia* (angl. Bayesian logistic regression), pre kategorizáciu textu s Gausovským a Laplacovým apriori.
- *Najlepší prvý rozhodovací strom* (angl. Best-first decision tree), konštrukcia rozhodovacieho stromu so stratégiou najlepší prvý.
- *Hybridná rozhodovacia tabuľka a naivný Bayes* (angl. Decision table naïve Bayes hybrid) hybridný klasifikátor, ktorý kombinuje rozhodovacie tabuľky a metódu Naivný Bayes.
- *Funkčné stromy* sú rozhodovacie stromy s lomeným rozdelením a lineárnymi funkciami v listoch.

Weka poskytuje tiež nástroje pre predspracovanie dát, zoznam niektorých filtrov (vymenované sú len vybrané základné filtre):

- *Pridanie klasifikátora*, pridá predikcie klasifikátora do datasetu.
- *Pridanie ID* ako nového atribútu pre každý záznam datasetu.
- *Pridanie hodnoty* chýbajúcim hodnotám z poskytnutého zoznamu.
- *Preskupenie atribútov* preusporiadanie poradia atribútov.

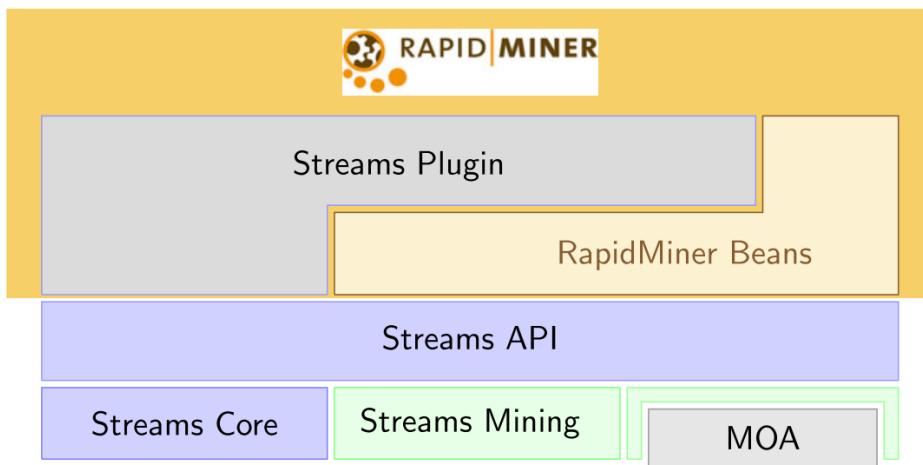
- Numerické hodnoty na nominálne, konverzia numerických hodnôt na nominálne.



Obrázok 4.2: Hlavná obrazovka GUI nástroja WEKA.

### 4.3 RapidMiner Streams-Plugin

Streams plugin poskytuje operátory RapidMiner-u pre základné budovanie blokov Streams API použitím obaľovača (angl. wrapper) pre priame použitie implementácie, ktorú poskytuje Streams balík. Operátory Streams Plugin-u sú automaticky vytvorené pomocou procesora a použitím knižnice RapidMiner Beans (Bockermann and Blom, 2012). Architektúra Streams Plugin-u je postavená na Streams API, ktoré bolo navrhnuté v práci Bockermannu a Bloma. Na obrázku 4.3 je možné vidieť, že RapidMiner Streams-Plugin je tiež možné integrovať s nástrojom MOA, ktorý je detailne popísaný vyššie. Spracova-



**Obrázok 4.3:** Architektúra RapidMiner Stream Plugin-u a ďalších potrebných častí.

nie prúdu v tomto nástroji pozostáva z dvoch elementov: *prúdy* a *procesory*. Tieto elementy sú reprezentovaná operátormi RapidMiner-u. StreamsPlugin tiež poskytuje webové rozhranie pre získanie okamžitých on-line výsledkov cez JSON-RPC protokol.

### 4.4 StreamBase

StreamBase<sup>2</sup> je platforma pre spracovanie udalostí, ktorá poskytuje vysoko-výkonný softvér pre budovanie a nasadanie systémov, ktoré analyzujú a rea-

<sup>2</sup><http://www.streambase.com/>

gujú (napr. akciami) na prúdiace dátá v reálnom čase. StreamBase poskytuje prostredie pre svižný vývoj, server pre spracovanie udalostí s nízkou odozvou a vysokou prieplustnosťou a zároveň integráciu do podnikových nástrojov, napríklad pre spracovanie historických údajov. Server analyzuje prúdiace dátá a poskytuje výsledky a odpovede v reálnom čase s extrémne nízkou odozvou. Toto je dosiahnuté maximalizáciou využitia hlavnej pamäte a ostatných prostriedkov servera, zatiaľ čo sa eliminujú závislosti na ostatné aplikácie. Integrované vývojové prostredie - StreamBase Studio umožňuje programátorom jednoducho a rýchlo vytvoriť, testovať a debugovať StreamSQL aplikácie použitím grafického modelu toku vykonávania. StreamBase aplikácie sú potom skompilované a nasadené za behu servera.

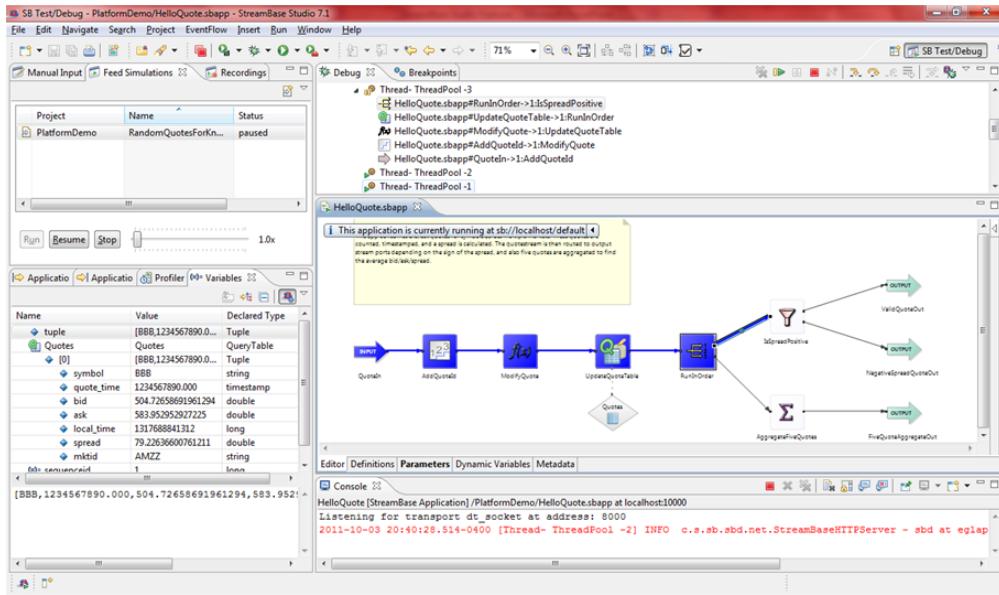
StreamSQL je dopytovací jazyk, ktorý rozširuje štandard SQL. StreamSQL umožňuje spracovanie prúdov v reálnom čase a dopytovanie sa do nich. Základná myšlienka jazyka SQL je možnosť dopytovať sa do uložených statických kolekcií dát, StreamSQL umožňuje to isté, ale do prúdov dát. Teda, StreamSQL musí zvládnuť spracovať kontinuálny prúd udalostí a časovo orientované záznamy. StreamSQL zachováva schopnosti jazyka SQL zatiaľ čo pridáva nové možnosti ako napríklad: bohatý systém posuvných okien, možnosť miešania prúdiacich dát a statických dát a tiež možnosť pridať vlastnú logiku vo forme analytických funkcií.

StreamBase EventFlow je jazyk pre prúdove spracovanie vo forme tokov a operátorov ako grafických elementov. Používateľ má možnosť spájať tieto grafické elementy a vytvárať tak jednoducho topológiu pre prúdové spracovanie bez nutnosti programovania. EventFlow integruje všetky možnosti StreamSQL.

Použitie StreamBase sa výborne hodí pre štrukturované aplikácie "reálneho času", ktoré majú za cieľ rýchle spracovanie spolu s rýchlym prototypovaním a nasadením nových funkcionalít.

## 4.5 Spark

Spark je klastrový výpočtový systém. Kombinuje spracovanie uložených dát v dávkovom móde so spracovaním prúdu údajov v reálnom čase (Cimerman



Obrázok 4.4: Vývojové prostredie nástroja StreamBase.

and Ševcech, 2015). Cieľom Spark-u je poskytnúť rýchlu výpočtovú platformu pre analýzu dát. Spark poskytuje všeobecný model výkonávania ľubovoľných dopytov, ktoré sú výkonávané v hlavnej pamäti (pokiaľ ide o prúdové spracovanie). Tento model je nazvaný *Pružný distribuovaný dataset*, skr. RDD (angl. Resilient Distributed Dataset), čo je dátova abstrakcia distribuovanej pamäti. Keďže výpočet beží v hlavnej pamäti (pri prúdovom spracovaní), nie je potrebné vykonávať zápis na disk, vďaka čomu môže byť dosiahnuté spracovanie v reálnom čase. Výpočet prebieha vo veľkom klastri uzlov s dosiahnutím odolnosti voči chybám za použitia RDD. RDD sídli v hlavnej pamäti, ale môže byť periodicky ukladaný na disk. Vďaka distribuovanej povahe RDD môže byť stratená časť RDD obnovená z pamäti iného uzla. Samotné prúdové spracovanie nie je vykonávané správa po správe (angl. message by message), ale v mikro dávkach, ktoré môžu byť automaticky paralelne distribuované v strapci. Spark Streaming<sup>3</sup> alebo tiež, prúdové spracovanie, si v poslednej dobe vyžiadalo špeciálnu pozornosť od tvorcom programovacieho rámca. Reagujú tým na vysoký dopyt odbornej verejnosti po prúdovom spracovaní dát, ktoré chýbal v Spark-u. Spark Streaming poskytuje integrované rozhranie API pre rôzne programovacie jazyky, pričom v budúcnosti je snaha toto rozhranie úplne integrovať s dávavkovým spracovaním, aby mohli vývojári používať rovnaké dátové typy pre rôzne typy úloh. Poskytuje tiež aspoň raz (angl. at least once) schému do-

<sup>3</sup><http://spark.apache.org/streaming/>

ručenia správ a zaručuje tak odolnosť voči chybám a prípadnej strate správy. Prúdové spracovanie v Spark-u je jednoduché integrovať spolu s dávkovým spracovaním, ktoré poskytuje, či použiť spolu s knižnicou pre strojové učenie sa.



## 5. Klasifikácia prúdu dát použitím rozhodovacích stromov

Klasifikácia dát je dobre známa úloha a problém dolovania, analýzy a spracovania dát. Tento problém bol veľmi dobre a podrobne študovaný pri spracovaní statickej kolekcie údajov. Pri tomto prístupe sú všetky dáta v pamäti počítača. Vybraný algoritmus potom môže pomerne "lacno" prečítať celú množinu niekoľko krát s cieľom zvýšenia presnosti a kvality výsledného modelu. Tento prístup nie je aplikovateľný pre klasifikáciu prúdov dát z nasledujúcich dôvodov:

- *Prúd dát je potenciálne nekonečná sekvencia udalostí*, ktoré môžu byť správne alebo chybne usporiadane v závislosti od spoľahlivosti zdroja dát. Hlavný problém tu predstavuje to, že prúd je nekonečná sekvencia udalostí. Závažnosť problému usporiadania daných udalostí, či vozrieck závisí od konkrétnej úlohy analýzy prúdu dát.
- *Obmedzená pamäť*, nie je možné všetky dáta zbierať a ukladať do pamäti. Toto obmedzenie vyplýva z prvej vlastnosti prúdov dát.
- *Model pre klasifikáciu prúdov musí byť ihneď pripravený k použitiu*. Znamená to, že hneď po tom ako sú spracované prvé dáta z prúdu, model je pripravený na použitie, napríklad klasifikovať iný prúd dát.
- *Prúdy dát takmer vždy v sebe nesú zmeny* (angl. concept drift), na ktoré sa musí vedieť klasifikátor adaptovať. Vlastnosť klasifikačných modelov vysporiadať sa so zmenami považujeme za rozhodujúcu pri hodnotení ich kvality a použiteľnosti v praxi. Preto sa aj nami navrhovaná metóda sústredí na vytvorenie metódy, ktorá je schopná adaptácie na zmeny a ich adekvátné interpretovanie používateľovi. Zmeny v dátach môžu byť náhle, postupné ale môžu predstavovať aj očakávané sezónne vplyvy (napr. obdobie Vianoc z pohľadu počtu nákupov internetového obchodu).

Problém klasifikácie a jej definícia je podrobne opísaný v kapitole 2.5. V skratke, cieľom je nájsť funkciu  $y = f(x)$ , kde  $y$  je skutočná trieda objektu/vzorky z prúdu dát a  $x$  sú atribúty danej vzorky. Potom vieme pomocou funkcie  $f(x)$  klasifikovať nové vzorky do triedy  $y'$  s istou pravdepodobnosťou.

Klasifikácia prúdov dát má zmysel často pre doménových expertov, ktorí potrebujú vytvárať detailné analýzy, či predikčné a klasifikačné modely. Pod pojmom doménový expert rozumieme človeka, ktorý rozumie analyzovaným dátam a pracuje s bežnými analytickými nástrojmi ako napríklad Google Analytics<sup>1</sup> alebo IBM SPSS<sup>2</sup>. Použitie klasifikácie prúdov dát má zmysel v mnohých oblastiach a prípadoch použitia:

- Detekcia podvodov pri finačných prevodoch. Je dôležité detektovať fašošnú, či podvodnú platbu platobnou kartou takmer v reálnom čase pre minimalizáciu nákladov vzniknutých s jej neskorím riešením. Vytvorenie klasifikátora nad prúdmi dát ma zmysel práve preto, že transakcie predstavujú prúd dát, ktorý v sebe často nesie sezónne vzory a zmeny, na ktoré sa nevedia dobre adaptovať tradičné metódy.
- Klasifikácia zákaznika na webe. Toto má zmysel napríklad pre obchody ako Amazon.com. Pre takéto stránky je prínosné vedieť klasifikovať, či je navštievnik webu potenciálny zakazník alebo má tendenciu odísť. Na základe týchto zistení môže majiteľ stránky vytvoriť ponuku pre zákazníka s cieľom udržať ho na stránke.
- Klasifikácia sietovej prevádzky s cieľom klasifikovať potenciálne pokusy o útoky na sieť. Cieľom takejto úlohy je eliminácia útoku a stým spojená minimalizácia prestoja (angl. downtime) siete a nákladov spôsobených škodami z úspešného útoku.

Pre všetky vyššie opísané prípady použitia má zmysel zohľadniť zmeny v prúde dát v modeli. Pretože, ak sa napríklad mení správanie používateľa na webe v závislosti od zmien na stránke, napríklad v podobe zmeny dizajnu, chceme tieto zmeny odzrkadliť aj vo výslednom modeli. Rovnako ma zmysel tieto zmeny aj interpretovať prostredníctvom vizualizácie používateľovi.

---

<sup>1</sup><https://www.google.com/analytics/>

<sup>2</sup><https://www-01.ibm.com/software/sk/analytics/spss/>

---

Existuje niekoľko dobre známych a používaných metód, niektoré z nich sú podrobne opísané v 2.5, pre klasifikáciu prúdov dát:

- *Hoeffdingove stromy* a ich rozšírenia, ktoré schopné adaptovať sa na zmeny (angl. concept drift) v dátach (Hulten et al., 2001; Bifet and Gavaldà, 2009).
- *Bayesová klasifikácia* a jej rozšírenia v podobe Bayesových stromov ukázali použitie najmä pri detekcii anomálií v dátach (Hill et al., 2007).
- *Neurónové siete a evolučné metódy.* Evolučné programovanie našlo uplatnenie v stochastických optimalizačných problémoch, vlastnosti evolučných algoritmov môžu byť tiež aplikované na spracovanie prúdu dát s cieľom vysporiadať sa so zmenami v dátach. Experimentálne použitie neurónových sietí ukázalo porovnatelné výsledky s rozhodovacími stromami.
- *Súborové metódy* (angl. ensemble), ktoré aplikujú vrecovanie (angl. bagging) a zvyšovanie (angl. boosting) s cieľom zvýšenia presnosti modelu pomocou nájdenia optimálneho nastavenia a kombinácie viacerých klasifikátorov. Náhodné lesy sú typickým príkladom súborových metód, dokážú sa vysporiadať so zmeny v dátach, pričom časová náročnosť spracovania vzorky je  $O(1)$  (Abdulsalam et al., 2011).
- Ďalšie metódy sú napríklad: *k-najbližších susedov* a *metóda podporných strojov*.

V tejto práci preto navrhujeme metódu pre klasifikáciu prúdu dát. Navrhnutá metóda môže byť použitá na akúkoľvek úlohu klasifikácie. Navrhovaná metóda používa techniku rozhodovacích stromov, je aplikovateľná na prúdy dát a model je takmer okamžite pripravený na použitie (záasadný rozdiel oproti tradičným metódam). Kladieme dôraz na spracovanie v reálnom čase, ktoré je najzákladnejšie pri spracovaní prúdov dát. Veľkú pozornosť pritom mierime na schopnosť adaptácie metódy na zmeny v dátach. Výsledný model aj s príslušnými zmenami, ktoré v dátach a modely nastali, prezentujeme používateľovi vo výslednej webovej aplikácii prostredníctvom vizualizácie. Okrem toho je našim cieľom navrhnúť metódu tak aby používateľ, ktorý je najčastejšie doménový expert, nemusel mať detailné znalosti o vnútornom fungovaní metódy.

## 5.1 Spracovanie prúdu dát

Spracovaniu prúdu dát venujeme samostatnú kapitolu, pretože si zaslúži špeciálnu pozornosť a rozdielny prístup v porovnaní so spracovaním statickej kolekcie dát. Navrhovaná metóda je všeobecne použiteľná na problémy klasifikácie pre prúdy dát. Znamená to, že spracuje dáta v takmer reálnom čase, poskytne odpoveď a teda aj vytvorený model okamžite a je schopná adaptávacie na zmeny. Pre splnenie týchto požiadaviek je potrebné venovať samostatnú pozornosť spracovaniu prúdu dát, teda požadujeme aby navrhovaná metóda spĺňala nasledujúce kritéria (Cimerman and Ševcech, 2015):

- *Odolnosť voči chybám* z pohľadu architektúry spracujúcej dát. Chybné alebo chýbajúce dátá môžu mať kritický dopad na správne fungovanie a kvalitu klasifikačného modelu.
- *Spracovanie v reálnom čase* je opäť dôležité pre správne fungovanie výsledného modelu, pretože model je aktualizovaný a prispôsobovaný zmenám v dátach kontinuálne. Oneskorenie niektorých správ, napríklad o 24 hodín čo je bežná prax pri ETL<sup>3</sup> procesoch, by mohlo mať nežiadúce následky vo forme skresleného modelu.
- *Horizontálna škálovateľnosť* komponentu, ktorý spracuje prúd dát. Táto vlastnosť podporuje splnenie predchádzajúcich požiadaviek. Pod horizontálnou škálovateľnosťou chápeme to, že je možné zvýšiť výkonnosť celého systému pridaním fyzického uzla bez akýchkoľvek výpadkov. Táto požiadavka implikuje podmienku distribuovanej povahy riešenia.

S cieľom splniť tieto požiadavky sme sa rozhodli použiť nasledujúce programovacie rámce a systémy:

- *Storm*<sup>4</sup> je programovací rámec vytvorený pre spracovanie dát v reálnom čase. Storm poskytuje možnosti škálovateľnej architektúry, ktorá je naviac odolná voči chybám na úrovni kvality dát. Programovanie nad

<sup>3</sup>ETL je proces, či architektonický vzor prenosu dát medzi viacerými časťami databázových systémov a aplikáciami, tento vzor je často používaný pre dátové sklady, skratka znamená Extrahuj, Transformuj a Načítaj (angl. Extract, Transform, Load)

<sup>4</sup><http://storm.apache.org/>

týmto rámcom je možné v každom programocom jazyku, ktorý je možné skompliovať do Java bajtkódu a vykonávať v JVM<sup>5</sup>. Storm poskytuje aplikovať akýkoľvek programovací vzor, model ktorý poskytuje je vyjadrený, resp. vytvára acyklický orientovaný graf zostrojený z tzv. prameňov a skrutiek.

- *Kafka*<sup>6</sup> je distribuovaná platforma pre spracovanie prúdov dát. Kafka je vhodná na budovanie aplikačí, ktoré potrebujú spracovať zdroje dát v reálnom čase a vymieňať tieto dáta medzi aplikáciami. Poskytuje možnosť publikovať (angl. publish) a predplatiť (angl. subscribe) prúdy dát. Kafka je postavená na modely fronty správ, pričom si tieto správy udržiava v pamäti a na disk ich replikuje pre prípad zlyhania.

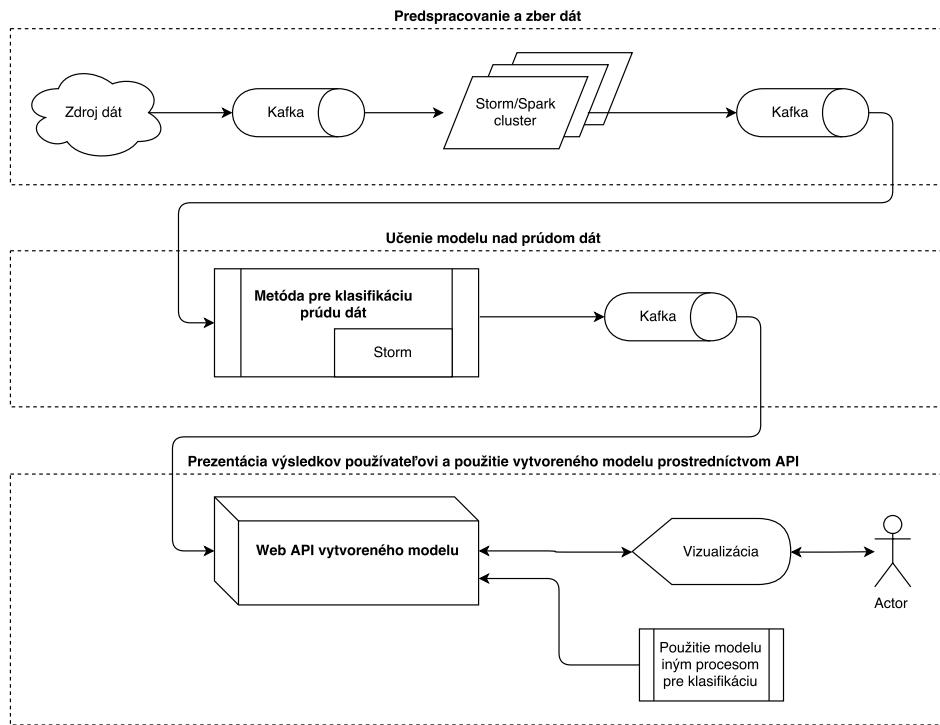
Nasledujúci obrázok schematicky popisuje architektúru spracovania prúdu dát potrebnú pre správne fungovanie metódy pre klasifikáciu prúdu dát s použitím rozhodovacích stromov.

## 5.2 Metóda klasifikácie prúdu dát

Cieľom je klasifikácia prúdu dát, pričom vytvorený model je pripravený na použitie takmer okamžite po prečítaní prvých vzoriek dát. Model sa tiež prispôsobuje zmenám a do istej miery sezónnym efektom v dátach. Základ metódy pre klasifikáciu sme zvolili state-of-the-art algoritmus rozhodovacích stromov, ktorý používa Hoeffdingovu mieru (Domingos and Hulten, 2000; Gaber et al., 2005; Krempl et al., 2014). Hoeffdingova miera je použitá na rozhodnutie, či bol prečítaný dostatočný počet vzoriek na to aby sa mohol uzol v strome zmeniť na rozhodovací uzol. Táto miera zabezpečuje to, že sa výsledný model asymptoticky blíži svojou kvalitou k tomu, ktorý by vznikol podobnou metódou pre statické dátá. Zároveň má táto miera vlastnosť, že dôvera v presnosť modelu exponenciálne rastie s lineárny nárastom počtu prečítaných vzoriek. Hoeffdingova miera je definovaná používateľom parametrom spoľahlivosti  $\delta$ , kde spoľahlivosť je  $(1 - \delta) \in <0, 1>$ . Metrika kvality vzorky  $G$  môže byť použitá ľubovoľná, napríklad informačný zisk (angl. information gain).

<sup>5</sup>Virtuálny stroj Java (angl. Java virtual machine)

<sup>6</sup><https://kafka.apache.org/>



**Obrázok 5.1:** Architektúra potrebná pre klasifikáciu prúdu dát v takmer reálnom čase. Architektúra pozostáva z troch úrovní. V časti predspracovania dát sú dáta zbierané zo zdroja prúdu dát a transformované do potrebnej podoby vhodnej pre ďalší krok. V kroku učenia modelu pre klasifikáciu prúdu dát je semi-automaticky vybraný vhodný algoritmus a atribúty a vytvorený klasifikačný model. Posledný krok obsahuje webovú službu, ktorá poskytuje Web API pre dotazovanie modelu. V tomto kroku tiež prezentujeme výsledky modelu v podobe vizualizácie používateľovi. Kafka je použitá na prenos správ medzi jednotlivými časťami aplikácie, správy sú rozdelené do rôznych tém podľa typu správy.

Metóda potrebuje na trénovanie označkované numerické alebo kategorické dátá do viacerých tried. Dáta musia byť vo forme n-tíc  $(x_1, x_2, \dots, x_n | y)$  kde  $x$  sú atribúty vzorky a  $y$  skutočná trieda vzorky. Nami vybraný potrebuje len minimum parametrov, ktoré je potrebné nastaviť pre správne fungovanie. Jedným z nich metódy je minimálny počet spracovaných vzoriek pred vytvorením prvého modelu. Týmto je možné minimalizovať prvotnú nepresnosť počiatocného modelu. Výsledný model je jednoduchý na reprezentáciu vďaka možnosti jeho intuitívnej interpretácií rozhodovacím stromom. Rozhodovací strom pozostáva z rozdeľovacích uzlov (angl. split node), tiež niekedy nazývané testovacie uzly, a listov (angl. leaf). V rozhodovacích uzloch sa vykonáva testovanie vzorky a jej posunutie do jednej z nasledujúcich vetiev alebo listu stromu. Ak vzorka narazí

na list znamená to, že bola klasifikovaná do istej triedy, ktorú opisuje daný list. Takto vytvorený model je použiteľný na klasifikáciu v rôznych aplikáciách.

Problémom rozhodovacích stromov je najmä ich šírka. Klasifikátory, ktoré používajú modely a algoritmy rozhodovacích stromov môžu podľa dát byť preveľmi široké. Tento problém môže mať za následok preučenie (angl. overfitting) modelu, ktorý bude vedieť klasifikovať veľmi dobre trénovacie dáta, resp. dátu zo začiatku prúdu, ale na nových dátach bude veľmi nepresný. Tento problém nastáva najmä pri spojитých číselných atribútoch a ich nerovnomernej distribúcii. Existuje niekoľko známych spôsobov ako sa stýmto nežiadúcim javom vysporiadať, jedným z nich je pre-prerezávanie (angl. pre-pruning) stromu. Tento spôsob aplikujeme aj v našej metóde priadním nulového atribútu  $X_0$  do každého uzla, ktorý spočíva v nerozdeľovaní daného uzla. Takže uzol sa stane rozhodovacím iba, ak je metrika  $G$ , so spoľahlivosťou  $1 - \delta$ , lepšia ako keby sa uzol nezmenil na rozhodovací.

V našich experimentoch sa, ale ukázalo, že pre-prerezávanie nieje dostatočný spôsob pre redukciu šírky stromu. Navrhujeme preto použiť metódy pre výber významných atribútov. Znamená to teda, že nebudú použité všetky atribúty pre učenie modelu a jeho následné použitie. Na takúto významovú analýzu je možné použiť napríklad náhodné lesy. Náhodné lesy vytvárajú veľa jednoduchých stromov, kde každý strom obsahuje práve jeden rozhodovací uzol. Následne sa zvolenou metrikou vyberú najvýznamnejšie atribúty a tie budú použité pre trénovanie a použitie klasifikátora. Okrem použitia náhodných lesov navrhujeme použiť histogrammi pre spojité numerické atribúty s cieľom redukovať šírku stromu.

Nami navrhovaná metóda sa musí vysporiadať so zmenami v dátach, pretože tie nesú v sebe takmer všetky prúdy dát. Zmeny môžu mať rôzny charakter, napríklad náhly kedy zmena nastane nečakane alebo postupný kedy sa zmena deje dlhú dobu a pomaly. Výsledný model musí pre udržanie svojej presnosti zohľadniť tieto zmeny. Metóda používa algoritmus *ADWIN* z anglického Adaptive Windowing (Hutchison and Mitchell, 2009). Tento algoritmus nepožaduje žiadne nastavenia parametrov používateľom ako napríklad veľkosť posuvného okna. Jediným parametrom je hodnota istoty  $\delta$  s akou bude algoritmus detektovať zmeny v prúde dát. Myšlienka ADWIN spočíva v tom, že nenenašla žiadna zmena v priemernej hodnote vybranej metriky v okne. Ak

je detekovaná zmena, v uzle začne narastať alternujúci podstrom. Tento podstrom musí spracovať definovaný minimálny počet vzoriek. Potom, ak je kvalita podstromu vyššia ako kvalita podstromu, z ktorého začal narastať, starý podstrom je nahradený alternujúcim podstromom. Naraz môže existovať niekoľko alternujúcich podstromov, pričom môže nastať situácia kedy ani jeden z nich nebude mať vyššiu kvalitu a nesplní Hoeffdingovu mieru preto aby nahradil starý podstrom. Výsledok tohto algoritmu chceme detailne prezentovať používateľovi vo forme vizualizácie, ktorá je detailnejšie popísaná v nasledujúcej podkapitole.

### 5.3 Prezentácia výsledkov používateľovi

V situácii keď potrebuje doménovy expert vytvoriť klasifikačný model s použitím dát reálneho sveta, je často potrebná najprv detailná znalosť dát, ktorú doménový expert, predpokladáme má. Následne preto, aby vedel vytvoriť správny model potrebuje mať detailné znalosti o fungovaní klasifikačných metód a algoritmov. Cieľom našej metódy je odbremeníť experta od nutnosti mať detailné znalosti o fungovaní modelu a algoritmov. Zameriavame sa teda na prezentáciu dôležitých informácií, ktoré potrebuje pre správne pochopenie modelu a následné rozhodnutia. Výber atribútov a algoritmov, ktoré budú použité na trénovanie modelu je bez nutnosti interakcie používateľa v zmysle nastavovania parametrov a výberu metódy.

Pretože používateľ nemusí mať detailné znalosti o fungovaní algoritmov a klasifikačných metód, je dôležité vysvetlenie výsledného modelu. Znamená to, že je dôležité aby pre používateľa nebol vzniknutý model len čierna skrinka (angl. black-box), ktorá s nejakou úspešnosťou dokáže klasifikovať prúdy dát. Navrhujeme preto vizualizáciu výsledného modelu. Vizualizácia je vo forme rozhodovacieho stromu, ktorý je jednoduchý na pochopenie aj bez predchádzajúcich znalostí o rozhodovacích stromov (Nguyen et al., 2015). Sústredíme sa tiež na zobrazenie zmien (angl. concept drift) v dátach resp. zmenách, ktoré sa odzrkadlia aj v modely. Zmeny samotného modelu vizualizujeme ako animáciu, ktorú je možné spustiť a pozorovať vývoj stromu. Ďalej poskytujeme náhľad vo forme čiarového grafu na kvalitu alternujúcich podstromov, ktoré boli vytvorené pri detekovaní zmeny.

Pomocou týchto vizualizácií chceme prezentovať výsledky našej klasifikačnej metódy používateľovi. Prezentované výsledky by mali byť jednoduché na pochopenie aj bez detailných znalostí o fungovaní metódy a algoritmov. Overenie použiteľnosti a miery pochopenia prezentovaných výsledkov overujeme detailnými používateľskými štúdiami a vyhodnotením tromi doménovými expertami (majú znalosti o fungovaní metódy a algoritmov).

## 5.4 Vyhodnotenie a experimenty

Pre kvantifikovanie správneho fungovania nami navrhovanej metódy navrhujeme viaceré experimenty. Prvým z experimentov je vyhodnotenie integrácie časti spracovania prúdu dát s našou metódou. Pri tomto vyhodnocovaní sa budeme pozerať na výkonnostné metriky, ktoré hovoria o prieplustnosti a výkone tejto časti aplikácie. Zaujíma nás hlavne odolnosť voči chýbam, spracovanie v reálnom čase a zaťaženie procesoru a pamäte v závislosti na objeme dát.

Vyhodnotenie samotnej metódy pre klasifikáciu prúdu dát nás zaujíma jú bežné metriky používané pri vyhodnocovaní modelov strojového učenia, ako napríklad presnosť (angl. precision) a pokrytie (angl. recall). Keďže ide o klasifikovanie prúdu dát, zameriavame sa tiež na nasledujúce metriky:

- *Kappa štatistiky*, ktoré dobre vyjadrujú presnosť klasifikátora nestabilné prúdy dát.
- *Najprv test-potom-trénovanie* (angl. Test-Then-Train alebo Prequential) je metrika používaná pre meranie výkonnosti klasifikátorov, ktoré sa výjajú v čase.

Pri vyhodnocovaní klasifikačnej metódy sa tiež pozéráme na vhodnosť výberu rôznych algoritmov pre výber atribútov, detekcie zmien (angl. concept drift) a samotného klasifikačného algoritmu.

Vysokú pozornosť venujeme vyhodnoteniu prezentovaných výsledkov používateľovi. Najprv robíme vyhodnotenie fungovania celej aplikácie ako celku s tromi expertami, ktorí majú detailné znalosti o fungovaní metód a algoritmov strojového učenia. Ďalej navrhujeme experiment vo forme používateľskej

štúdie. Používateľská štúdia môže prebiehať v kontrolovanom, ale aj v "domácom" prostredí. Počas tejto štúdie budú účastníci vykonávať definované úlohy s cieľom zmerať a kvantifikovať ich výkonnosť a vôbec schopnosť splniť stanovené úlohy. Tieto úlohy môžu byť od jednoduchých ako odčítanie hodnotu z grafu, až po komplexné ako zistiť počet signifikantných zmien modelu a ich čas kedy nastali, či krátke slovná reprezentácia fungovania modelu.

V tejto kapitole navrhujeme metódu pre klasifikáciu prúdu dát. Metóda poskytuje iba jeden voliteľný parameter, ktorý musí nastaviť používateľ, hodnotu istoty  $\delta$ . Parameter reprezentuje istotu  $1 - \delta$ , že bude výsledný model identický stým, ktorý by vznikol použitím tradičnej metódy. Cieľom metódy je, že používateľ nemusí mať znalosti o fungovaní klasifikačných algoritmov, ale je schopný použiť v praxi nami navrhovanú metódu. Naviac, výsledná model reprezentujeme vo forme vizualizácie, ktorá má pomôcť vysvetliť fungovanie modelu.

Kladieme si teda nasledujúce hypotézy:

**Hypotéza 5.4.1** *Naša metóda je schopná so stanovenou istotou klasifikovať prúdy dát a zároveň poskytuje výsledky v reálnom čase.*

**Hypotéza 5.4.2** *Metóda je ľahká na použitie a interpretované výsledky sú jednoduché na pochopenie pre doménového experta bez detailnej znalosti o fungovaní modelu.*

**Hypotéza 5.4.3** *Dokážeme zmysluplne a pochopiteľne, pre doménového experta, interpretovať blížiacu sa zmenu v uzle stromu a tiež zobraziť históriu zmien modelu.*

## 6. Zhodnotenie a budúca práca



# Literatúra

- Abdulsalam, H., Skillicorn, D. B., and Martin, P. (2011). Classification using streaming random forests. *IEEE Transactions on Knowledge and Data Engineering*, 23(1):22–36.
- Aggarwal, C. C. (2014). A survey of stream classification algorithms.
- Anagnostopoulos, C., Adams, N. M., and Hand, D. J. (2008). Deciding what to observe next: adaptive variable selection for regression in multivariate data streams. In *Proceedings of the 2008 ACM symposium on Applied computing*, pages 961–965. ACM.
- Babcock, B., Babu, S., Datar, M., Motwani, R., and Widom, J. (2002). Models and issues in data stream systems. In *Proceedings of the twenty-first ACM SIGMOD-SIGACT-SIGART symposium on Principles of database systems*, pages 1–16. ACM.
- Babu, S. and Widom, J. (2001). Continuous queries over data streams. *ACM Sigmod Record*, 30(3):109–120.
- Barlow, T. and Neville, P. (2001). Case study: visualization for decision tree analysis in data mining. *IEEE Symposium on Information Visualization, 2001. INFOVIS 2001.*, 2001:149–152.
- Bifet, A., de Francisci Morales, G., Read, J., Holmes, G., and Pfahringer, B. (2015). Efficient online evaluation of big data stream classifiers. In *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 59–68. ACM.
- Bifet, A. and Frank, E. (2010). Sentiment knowledge discovery in twitter streaming data. In *Discovery Science*, pages 1–15. Springer.
- Bifet, A. and Gavaldà, R. (2009). Adaptive learning from evolving data streams. In *International Symposium on Intelligent Data Analysis*, pages 249–260. Springer.

- Bifet, A., Holmes, G., Kirkby, R., and Pfahringer, B. (2010). MOA: massive online analysis. *Journal of Machine Learning Research*, 11:1601–1604.
- Bockermann, C. and Blom, H. (2012). Processing data streams with the rapidminer streams-plugin. In *Proceedings of the 3rd RapidMiner Community Meeting and Conference*.
- Brzeziński, D. (2010). *Mining data streams with concept drift*. PhD thesis, Master’s thesis, Poznan University of Technology.
- Cimerman, M. and Ševcech, J. (2015). *Analýza prúdu údajov*.
- Demšar, J. and Bosni, Z. (2014). Visualization and Concept Drift Detection Using Explanations of Incremental Models Related work. 38:321–327.
- Domingos, P. and Hulten, G. (2000). Mining high-speed data streams. In *Proceedings of the sixth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 71–80. ACM.
- Gaber, M. M., Zaslavsky, A., and Krishnaswamy, S. (2005). Mining data streams: a review. *ACM Sigmod Record*, 34(2):18–26.
- Gama, J., Medas, P., Castillo, G., and Rodrigues, P. (2004). Learning with drift detection. In *Brazilian Symposium on Artificial Intelligence*, pages 286–295. Springer.
- Hall, M., Frank, E., Holmes, G., Pfahringer, B., Reutemann, P., and Witten, I. H. (2009). The weka data mining software: an update. *ACM SIGKDD explorations newsletter*, 11(1):10–18.
- Han, J., Pei, J., and Kamber, M. (2011). *Data mining: concepts and techniques*. Elsevier.
- Hill, D. J. and Minsker, B. S. (2010). Anomaly detection in streaming environmental sensor data: A data-driven modeling approach. *Environmental Modelling & Software*, 25(9):1014–1022.
- Hill, D. J., Minsker, B. S., and Amir, E. (2007). Real-time bayesian anomaly detection for environmental sensor data. In *Proceedings of the Congress-International Association for Hydraulic Research*, volume 32, page 503. Citeseer.

- Hodge, V. J. and Austin, J. (2004). A survey of outlier detection methodologies. *Artificial Intelligence Review*, 22(2):85–126.
- Hulten, G., Spencer, L., and Domingos, P. (2001). Mining time-changing data streams. In *Proceedings of the seventh ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 97–106. ACM.
- Hutchison, D. and Mitchell, J. C. (2009). *Advances in Intelligent Data Analysis VIII*.
- Ikonomovska, E. and Zelke, M. (2013). Algorithmic techniques for processing data streams. *Dagstuhl Follow-Ups*, 5.
- Jin, R. and Agrawal, G. (2003). Efficient decision tree construction on streaming data. In *Proceedings of the ninth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 571–576. ACM.
- Krempl, G., Žliobaite, I., Brzeziński, D., Hüllermeier, E., Last, M., Lemaire, V., Noack, T., Shaker, A., Sievi, S., Spiliopoulou, M., et al. (2014). Open challenges for data stream mining research. *ACM SIGKDD Explorations Newsletter*, 16(1):1–10.
- Liu, X., Wu, X., Wang, H., Zhang, R., Bailey, J., and Ramamohanarao, K. (2010). Mining distribution change in stock order streams.
- Madden, S., Shah, M., Hellerstein, J. M., and Raman, V. (2002). Continuously adaptive continuous queries over streams. In *Proceedings of the 2002 ACM SIGMOD international conference on Management of data*, pages 49–60. ACM.
- Muthukrishnan, S. (2005). *Data streams: Algorithms and applications*. Now Publishers Inc.
- Nguyen, H.-L., Woon, Y.-K., and Ng, W.-K. (2015). A survey on data stream clustering and classification. *Knowledge and information systems*, 45(3):535–569.
- Olston, C., Jiang, J., and Widom, J. (2003). Adaptive filters for continuous queries over distributed data streams. In *Proceedings of the 2003 ACM SIGMOD international conference on Management of data*, pages 563–574. ACM.

- Rai, P., Daumé III, H., and Venkatasubramanian, S. (2009). Streamed learning: one-pass svms. *arXiv preprint arXiv:0908.0572*.
- Ross, G. J., Tasoulis, D. K., and Adams, N. M. (2009). Online annotation and prediction for regime switching data streams. In *Proceedings of the 2009 ACM symposium on applied computing*, pages 1501–1505. ACM.
- Rutkowski, L., Pietruczuk, L., Duda, P., and Jaworski, M. (2013). Decision trees for mining data streams based on the mcdiarmid’s bound. *IEEE Transactions on Knowledge and Data Engineering*, 25(6):1272–1279.
- Stankovic, J. A. and Zhao, W. (1988). On real-time transactions. *ACM Sigmod Record*, 17(1):4–18.
- Tran, D.-H., Gaber, M. M., and Sattler, K.-U. (2014). Change detection in streaming data in the era of big data: models and issues. *ACM SIGKDD Explorations Newsletter*, 16(1):30–38.
- Wadewale, K. and Desai, S. (2015). Survey on method of drift detection and classification for time varying data set.
- Yao, Y. (2013). Concept Drift Visualization. *Journal of Information and Computational Science*, 10(10):3021–3029.
- Zliobaite, I. and Gabrys, B. (2014). Adaptive preprocessing for streaming data. *IEEE transactions on knowledge and data Engineering*, 26(2):309–321.

# Prílohy

## A Plán na letný semester 2016/2017

Tento plán popisuje náš plán ďalšieho vývoja diplomovej práce v nasledujúcom letnom semestri na týždennej granularite. Pričom predpokladáme, že semester má 12 týždňov.

- *1-2 týžden:* Príprava článku na študentskú vedeckú konferenciu IIT.SRC.
- *3 týždeň:* Vyhodnotenie prezentovaných výsledkov na IIT.SRC, určenie ďalšieho smeru a priestoru na zlepšenie aktuálneho stavu implementovanej metódy.
- *4. týždeň:* Implementácia návrhov na zlepšenie metódy pre klasifikáciu a ich vyhodnotenie.
- *5. týždeň:* Implementácia návrhov na zlepšenie metódy pre vizualizáciu a ich vyhodnotenie.
- *6. týždeň:* Vyhodnotenie kvality a výkonnosti implementovanej metódy  
- kvantitatívne vyhodnotenie stanovených metrík kvality.
- *7. týždeň:* Návrh ďalších experimentov vo forme používateľskej a expertnej štúdie v použiteľnosti navrhovanej metódy.
- *8. týždeň:* Používateľská štúdia a spisanie výsledkov kvantitatívnych metrík kvality metódy z 5-6. týždňa.
- *9. týždeň:* Vyhodnotenie používateľskej štúdie.
- *10. týždeň:* Analýza priestoru na zlepšenie navrhovanej a implementovanej metódy podľa výsledkov používateľskej štúdie.
- *11. týždeň:* Spisanie výsledkov používateľskej štúdie a finalizácia diplomovej práce, príprava na odovzdanie.

- 12. týždeň: Odovzdanie diplomovej práce.

## B Plán na zimný semester 2016/2017

- Rozšírenie analýzy o ďalšie metódy a celkovo zpresnenie, zprehľadnenie a orezanie analýzy.
- Dokončenie návhu vlastnej metódy.
- Implementácia metódy.
- Evaluácia metódy - toto je priamo súčasťou navrhovanej metódy.
- Príprava článku na nejakú konferenciu, napr. IIT.SRC.
- Prvý experiment s použitím Eye Tracker-a na evaluáciu vizualizácie výsledkov.

